Приложение 3.1.1 к ОПОП по специальности 09.02.07 Информационные системы и программирование

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Магнитогорский государственный технический университет им. Г.И. Носова»

Многопрофильный колледж

МЕТОДИЧЕСКИЕ УКАЗАНИЯ ДЛЯ ЛАБОРАТОРНЫХ ЗАНЯТИЙ МЕЖДИСЦИПЛИНАРНОГО КУРСА

МДК.01.01 Разработка программных модулей

МДК.01.02 Поддержка и тестирование программных модулей

МДК.01.03 Разработка мобильных приложений

МДК.01.04 Системное программирование

для обучающихся специальности

09.02.07 Информационные системы и программирование

Магнитогорск, 2024

ОДОБРЕНО

Предметно-цикловой комиссией «Информатики и вычислительной техники»
Председатель Т.Б. Ремез
Протокол № 5 от «31»января 2024

Методической комиссией МпК

Протокол № 3 от «21»февраля 2024

Разработчики:
преподаватель отделения №2 "Информационных технологий и транспорта" Многопрофильного колледжа ФГБОУ ВО «МГТУ им. Г.И. Носова»
В.Д. Тутарова

Методические указания по выполнению лабораторных работ разработаны на основе рабочей программы профессионального модуля ПМ 01. «РАЗРАБОТКА МОДУЛЕЙ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ДЛЯ КОМПЬЮТЕРНЫХ СИСТЕМ».
Содержание лабораторных работ ориентировано на подготовку обучающихся к освоению вида деятельности **ВД 1. Разработка модулей программного обеспечения для компьютерных систем** программы подготовки специалистов среднего звена по специальности 09.02.07 Информационные системы и программирование (квалификация Программист) и овладению профессиональными компетенциями.

Содержание

1 ВВЕДЕНИЕ

Важную часть теоретической и профессиональной практической подготовки обучающихся составляют лабораторные занятия.

Состав и содержание лабораторных занятий направлены на реализацию действующих федеральных государственных образовательных стандартов среднего профессионального образования.

Ведущей дидактической целью лабораторных занятий является формирование профессиональных практических умений (умений выполнять определенные действия, операции, необходимые в последующем в профессиональной деятельности) или учебных практических умений (умений решать задачи по математике, физике, информатике и др.), необходимых в последующей учебной деятельности.

Ведущей дидактической целью лабораторных занятий является экспериментальное подтверждение и проверка существенных теоретических положений (законов, зависимостей).

В соответствии с рабочей программой профессионального модуля ПМ.01 «Разработка модулей программного обеспечения для компьютерных систем», МДК 01.01 Разработка программных модулей; МДК.01.02 Поддержка и тестирование программных модулей; МДК. 01.03 Разработка мобильных приложений и МДК. 01.04 Системное программирование предусмотрено проведение лабораторных занятий. В рамках лабораторного занятия обучающиеся могут выполнять одну или несколько лабораторных работ.

В результате их выполнения, обучающийся должен:

уметь:

- У1. осуществлять разработку кода программного модуля на языках низкого и высокого уровней;
- У2. создавать программу по разработанному алгоритму как отдельный модуль;
- У3. выполнять отладку и тестирование программы на уровне модуля;
- У4. осуществлять разработку кода программного модуля на современных языках программирования;
- У5. уметь выполнять оптимизацию и рефакторинг программного кода;
- У6. оформлять документацию на программные средства;
- У7. формировать алгоритмы разработки программных модулей в соответствии с техническим заданием;
- У8. применять инструментальные средства отладки программного обеспечения;
- У9. работать с системой контроля версий.

Содержание лабораторных работ ориентировано на подготовку студентов к освоению профессионального модуля основной профессиональной образовательной программы по специальности и овладению профессиональными компетенциями:

ПК 1.1. Формировать алгоритмы разработки программных модулей в соответствии с техническим заданием;

ПК 1.2. Разрабатывать программные модули в соответствии с техническим заданием;

ПК.1.3. Выполнять отладку программных модулей с использованием специализированных программных средств;

ПК 1.4. Выполнять тестирование программных модулей;

ПК 1.5. Осуществлять рефакторинг и оптимизацию программного кода;

ПК 1.6. Разрабатывать модули программного обеспечения для мобильных платформ.

А также формированию общих компетенций:

ОК 01 Выбирать способы решения задач профессиональной деятельности применительно к различным контекстам.

ОК 02 Использовать современные средства поиска, анализа и интерпретации информации и информационные технологии для выполнения задач профессиональной деятельности

ОК 03 Планировать и реализовывать собственное профессиональное и личностное развитие, предпринимательскую деятельность в профессиональной сфере, использовать знания по финансовой грамотности в различных жизненных ситуациях

ОК 04 Эффективно взаимодействовать и работать в коллективе и команде

ОК 05 Осуществлять устную и письменную коммуникацию на государственном языке Российской Федерации с учетом особенностей социального и культурного контекста.

ОК 06 Проявлять гражданско-патриотическую позицию, демонстрировать осознанное поведение на основе традиционных общечеловеческих ценностей, в том числе с учетом гармонизации межнациональных и межрелигиозных отношений, применять стандарты антикоррупционного поведения.

ОК 07 Содействовать сохранению окружающей среды, ресурсосбережению, применять знания об изменении климата, принципы бережливого производства, эффективно действовать в чрезвычайных ситуациях.

ОК 09 Пользоваться профессиональной документацией на государственном и иностранном языках.

Выполнение обучающихся практических лабораторных работ по профессиональному модулю ПМ.01 «Разработка модулей программного обеспечения для компьютерных систем», МДК 01.01 Разработка программных модулей; МДК.01.02 Поддержка и тестирование программных модулей; МДК. 01.03 Разработка мобильных приложений и МДК. 01.04 Системное программирование направлено на:

- обобщение, систематизацию, углубление, закрепление, развитие и детализацию полученных теоретических знаний по конкретным темам учебной дисциплины;
- формирование умений применять полученные знания на практике, реализацию единства интеллектуальной и практической деятельности;
- развитие интеллектуальных умений у будущих специалистов: аналитических, проектировочных, конструктивных и др.;
- выработку при решении поставленных задач профессионально значимых качеств, таких как самостоятельность, ответственность, точность, творческая инициатива.

Лабораторные занятия проводятся в рамках соответствующей темы, после освоения дидактических единиц, которые обеспечивают наличие знаний, необходимых для ее выполнения.

2 МЕТОДИЧЕСКИЕ УКАЗАНИЯ

МДК.01.01 Разработка программных модулей
Тема:1.1.2 Структурное программирование
Лабораторное занятие № 1 Оценка сложности алгоритмов сортировки.

Цель работы:

- научиться формировать алгоритмы разработки программных модулей в соответствии с техническим заданием;
- разрабатывать программные модули в соответствии с техническим заданием;

Выполнив работу, Вы будете:

уметь:

- У1. Осуществлять разработку кода программного модуля на языках низкого и высокого уровней;
- У7. Формировать алгоритмы разработки программных модулей в соответствии с техническим заданием.
- У6. Оформлять документацию на программные средства;
- У2. Создавать программу по разработанному алгоритму как отдельный модуль.

Материальное обеспечение:

- мультимедийные средства хранения, передачи и представления информации.

Задание:

1. Подсчет операций. Классы входных данных

Одним из способов оценки трудоемкости ($T\,n$) является *подсчет количества выполняемых операций*. Рассмотрим в качестве примера алгоритм поиска минимального элемента массива.

1. начало; поиск минимального элемента массива array из N элементов
2. min := array[1]
3. для i от 2 до N выполнять:
4. если array[i] < min
5. min := array[i]
6. конец; вернуть min

При выполнении этого алгоритма будет выполнена:

1. N — 1 операция присваивания счетчику цикла i нового значения;
2. N — 1 операция сравнения счетчика со значением N;
3. N — 1 операция сравнения элемента массива со значением min;
4. от 1 до N операций присваивания значения переменной min.

Точное количество операций будет зависеть от обрабатываемых данных, поэтому имеет смысл говорить о наилучшем, наихудшем и среднем случаях. При этом худшему случаю всегда уделяется особое внимание, в том числе потому, что «плохие» данные могут быть намеренно поданы на вход злоумышленником.

Понятие *среднего случая* используется для оценки поведения алгоритма с расчетом на то, что наборы данных равновероятны. Однако, такая оценка достаточно сложна:

1. исходные данные разбиваются на группы так, что трудоемкость алгоритма ($t\,i$) для любого набора данных одной группы одинакова;
2. исходя из доли наборов данных группы в общем числе наборов, рассчитывается вероятность для каждой группы ($p\,i$);
3. оценка среднего случая вычисляется по формуле: $\sum i{=}1\ m\ p\,i \cdot t\,i$.

2. **Алгоритм пузырьковой сортировки (bubble sort)** использует два вложенных цикла. Во внутреннем последовательно сравниваются пары элементов и если оказывается, что элементы стоят в неправильном порядке — выполняется перестановка. Внешний цикл выполняется до тех пор, пока в массиве найдется хоть одна пара элементов, нарушающих требуемый порядок.

1. начало; пузырьковая сортировка массива array из N элементов
2. nPairs := N; количество пар элементов
3. hasSwapped := false; пока что ни одна пара не нарушила порядок
4. для всех i от 1 до nPairs-1 выполнять:
5. если array[i] > array[i+1] то:
6. swap(array[i], array[i+1]); обменять элементы местами
7. hasSwapped := true; найдена перестановка
8. nPairs := nPairs - 1; наибольший элемент гарантированно помещен в конец
9. если hasSwapped = true - то перейти на п.3
10. конец; массив array отсортирован

Трудоемкость функции swap не зависит от количества элементов в массиве, поэтому оценивается как $T\,swap = \Theta(1)$. В результате выполнения внутреннего цикла, наибольший элемент смещается в конец массива неупорядоченной части, поэтому через N таких вызовов массив в любом случае окажется отсортирован. Если же массив отсортирован, то внутренний цикл будет выполнен лишь один раз.

Порядок выполнения работы

- Составить математическую модель задачи.
- Выбрать и обосновать наиболее рациональный метод решения задачи;
- Разработать алгоритм для решения задачи.
- Написать и отладить программу.

Форма предоставления результата

- блок – схема.
- код программы.

Критерии оценки:

«Отлично» - теоретическое содержание курса освоено полностью, без пробелов, умения сформированы, все предусмотренные программой учебные задания выполнены, качество их выполнения оценено высоко.

–«Хорошо» - теоретическое содержание курса освоено полностью, без пробелов, некоторые умения сформированы недостаточно, все предусмотренные программой учебные задания выполнены, некоторые виды заданий выполнены с ошибками.

–«Удовлетворительно» - теоретическое содержание курса освоено частично, но пробелы не носят существенного характера, необходимые умения работы с освоенным материалом в основном сформированы, большинство предусмотренных программой обучения учебных заданий выполнено, некоторые из выполненных заданий содержат ошибки.

–«Неудовлетворительно» - теоретическое содержание курса не освоено, необходимые умения не сформированы, выполненные учебные задания содержат грубые ошибки.

Лабораторное занятие № 2 Оценка сложности алгоритмов поиска.

Цель работы:

- Научиться формировать алгоритмы разработки программных модулей в соответствии с техническим заданием;
- Выполнять отладку программных модулей с использованием специализированных программных средств;
- Выполнять тестирование программных модулей;

– Осуществлять рефакторинг и оптимизацию программного кода.

Выполнив работу, Вы будете:

уметь:

- У1. Осуществлять разработку кода программного модуля на языках низкого и высокого уровней;
- У7. Формировать алгоритмы разработки программных модулей в соответствии с техническим заданием.
- У6. Оформлять документацию на программные средства;
- У2. Создавать программу по разработанному алгоритму как отдельный модуль.

Материальное обеспечение:

- Мультимедийные средства хранения, передачи и представления информации.

Задание:

1. **Алгоритм линейного поиска**

Линейный поиск (Linear search) (Последовательный поиск, полный перебор, брутфорс)
Последовательный просмотр каждого элемента последовательности, пока не найден нужный. Выполняется за один проход цикла, сложность, соответственно – O(n).
Двоичный поиск (Binary search) (Бинарный поиск, метод деления пополам)
Поиск элемента в отсортированном массиве.
Суть алгоритма – последовательные запросы с предлагаемым вероятным ответом. Ответ на запрос может быть «нужный элемент меньше», «нужный элемент больше» или «это ответ!». Первым предлагается серединный элемент массива. После получения ответа алгоритм отбрасывает половину массива, содержащую неподходящие значения, и начинает поиск на оставшейся половине.
Троичный поиск (Ternary search) (Тернарный поиск)
Используется для поиска максимума функции, которая строго возрастает, а потом убывает, либо наоборот.
Если известно, что ответ лежит между точками A и B, на этом отрезке выбираются некоторые точки a и b (обычно делящие отрезок на три равные части). Проверяют, на какой из этих двух точек функция принимает меньшее значение. Крайняя треть, отделяемая той точкой, отбрасывается, и поиск продолжается на двух оставшихся третях.
В соответствии с этим алгоритмом эталонный массив просматривается последовательно от первого до последнего элемента. Наиболее сложным, как уже отмечалось, является случай, когда аргумента (ключа) нет в таблице (не найден).
Уточненный алгоритм будет таким.
1. Ввести исходный массив (ключей).
2. Выполнять
2. 1. Ввести аргумент поиска (целое число);
2. 2. Если аргумент поиска больше или равен нулю,
2.2.1. Результат (номер в массиве, num) = - 1 (не найден, такого номера нет);
2.2.2. Для индекса массива (i) от 0 до Длина.массива Если аргумент поиска = ключ[i], номер в массиве (num) = i;
2.2.3. Если номер num = - 1, вывести: «Такого ключа в массиве нет», Иначе вывести: «Ключ найден под номером num». Пока будет аргумент поиска больше или равен нулю.
3. Закончить.

Основной недостаток алгоритма линейного поиска – большое время. Он предполагает использование оператора For в пункте 2.2.2.

При этом ВСЕГДА выполняется ровно n операций сравнения, не зависимо от того, найден ключ или нет. Программа, обнаружив аргумент в начале массива, продолжает его просмотр до конца, т.е. выполняет бесполезную работу. Время поиска может быть существенно сокращено, если обеспечить его прекращение, когда ключ найден. При равномерном распределении элементов в таблице эталонов (ключей) среднее время поиска может стать пропорциональным величине n / 2.
Этого можно достичь, изменив пункт 2.2 в рассмотренном выше алгоритме следующим образом.
Ускорение линейного поиска
2. Выполнять
2. 1. Ввести аргумент поиска (целое число);
2. 2. Если аргумент поиска больше или равен нулю,
2.2.1. Результат (num) = - 1 (не найден);
2.2.2. Начальный индекс, i=0;
2.2.3. Пока (num=-1) и (i<=n – Длины.массива)
a) Если аргумент поиска = ключ[i], номер в массиве (num) = i;
b) i=i + 1
2.2.4. Если номер num = - 1,
вывести: «Такого ключа в массиве нет», Иначе вывести: «Ключ num найден». Пока будет аргумент поиска больше или равен нулю.

Трудоемкость (временная сложность) алгоритма линейного поиска определяется числом операций сравнения, выполняемых при просмотре таблицы эталонов. В лучшем случае количество таких операций равно 1, в худшем – n, а в среднем, если возможные значения ключей равновероятны, - n / 2. Таким образом, асимптотическая оценка O(n) = n.

Порядок выполнения работы

- Составить математическую модель задачи.
- Выбрать и обосновать наиболее рациональный метод решения задачи;
- Разработать алгоритм для решения задачи.
- Написать и отладить программу.

Форма предоставления результата

- Блок – схема.
- Код программы.

Критерии оценки:

«Отлично» - теоретическое содержание курса освоено полностью, без пробелов, умения сформированы, все предусмотренные программой учебные задания выполнены, качество их выполнения оценено высоко.

–«Хорошо» - теоретическое содержание курса освоено полностью, без пробелов, некоторые умения сформированы недостаточно, все предусмотренные программой учебные задания выполнены, некоторые виды заданий выполнены с ошибками.

–«Удовлетворительно» - теоретическое содержание курса освоено частично, но пробелы не носят существенного характера, необходимые умения работы с освоенным материалом в основном сформированы, большинство предусмотренных программой обучения учебных заданий выполнено, некоторые из выполненных заданий содержат ошибки.

–«Неудовлетворительно» - теоретическое содержание курса не освоено, необходимые умения не сформированы, выполненные учебные задания содержат грубые ошибки.

Лабораторное занятие № 3 Оценка сложности рекурсивных алгоритмов.

Цель работы:

- Научиться формировать алгоритмы разработки программных модулей в соответствии с техническим заданием;
- Выполнять отладку программных модулей с использованием специализированных программных средств;
- Выполнять тестирование программных модулей;
- Осуществлять рефакторинг и оптимизацию программного кода.

Выполнив работу, Вы будете:

уметь:

- У1. Осуществлять разработку кода программного модуля на языках низкого и высокого уровней;
- У7. Формировать алгоритмы разработки программных модулей в соответствии с техническим заданием.
- У6. Оформлять документацию на программные средства;
- У2. Создавать программу по разработанному алгоритму как отдельный модуль.

Материальное обеспечение:

- Мультимедийные средства хранения, передачи и представления информации.

Задание:

- Создайте программы, реализующие рекурсивный и итеративный алгоритмы вычисления n-го числа Фибоначчи. Сравните время их работы для n = 5, 25, 45, 65, 85, 100. Чтобы избежать переполнения, для n < 100 используйте беззнаковое представления чисел размером в 8 байт (тип unsigned long в языке Си). Результаты сравнения оформите в виде таблицы и в виде графика.
- Проанализируйте вычислительную сложность алгоритмов умножения чисел. На основе анализа поведите сравнения данных алгоритмов.
- Разработать два алгоритма возведения числа в целую неотрицательную степень, различающиеся по сложности. Для разработанных алгоритмов определите вычислительную сложность алгоритма.
- Разработать два алгоритма возведения числа в целую неотрицательную степень, различающиеся по сложности. Для разработанных алгоритмов проведите сравнительный анализ.
- Проанализируйте пространственную сложность алгоритмов умножения чисел. На основе анализа поведите сравнения данных алгоритмов.
- Создайте программы, реализующие рекурсивный и итеративный алгоритмы вычисления n-го числа Фибоначчи. Сравните время их работы для n = 5, 25, 45, 65, 85, 100. Чтобы избежать переполнения, для n < 100 используйте беззнаковое представления чисел размером в 8 байт(тип unsigned long в языке Си). Результаты сравнения оформите в виде таблицы и в виде графика.
- Разработать два алгоритма возведения числа в целую неотрицательную степень, различающиеся по сложности. Для разработанных алгоритмов определите вычислительную сложность алгоритма.
- Проанализируйте вычислительную сложность алгоритмов умножения чисел. На основе анализа поведите сравнения данных алгоритмов.
- Разработать два алгоритма возведения числа в целую неотрицательную степень, различающиеся по сложности. Для разработанных алгоритмов определите вычислительную сложность алгоритма.
- Разработать два алгоритма возведения числа в целую неотрицательную степень, различающиеся по сложности. Для разработанных алгоритмов проведите сравнительный анализ.

В целях дальнейшего анализа примем следующие допущения: каждая команда выполняется не более чем за фиксированное время; исходные данные алгоритма представляются машинными словами по битов каждое. Конкретная проблема задается N словами памяти, таким образом, на входе алгоритма – N = N* бит информации.
Программа, реализующая алгоритм для решения общей проблемы состоит из M машинных инструкций по м битов – M = M* м бит информации. Кроме того, алгоритм может требовать следующих дополнительных ресурсов абстрактной машины:

- Sd – память для хранения промежуточных результатов;
- Sr – память для организации вычислительного процесса (память, необходимая для реализации рекурсивных вызовов и возвратов).

При решении конкретной проблемы, заданной N словами памяти алгоритм выполняет не более, чем конечное количество «элементарных» операций абстрактной машины в силу условия рассмотрения только финитных алгоритмов. В связи с этим введем следующее определение:
Под трудоёмкостью алгоритма для данного конкретного входа – Fa(N), будем понимать количество «элементарных» операций совершаемых алгоритмом для решения конкретной проблемы в данной формальной системе. Комплексный анализ алгоритма может быть выполнен на основе комплексной оценки ресурсов формальной системы, требуемых алгоритмом для решения конкретных проблем. Очевидно, что для различных областей применения веса ресурсов будут различны, что приводит к следующей комплексной оценке алгоритма: c1 * Fa(N) + c2 * + c3 * Sd + c4 * Sr, где ci – веса ресурсов.
При более детальном анализе трудоемкости алгоритма оказывается, что не всегда количество элементарных операций, выполняемых алгоритмом на одном входе длины N, совпадает с количеством операций на другом входе такой же длины. Это приводит к необходимости введения специальных обозначений, отражающих поведение функции трудоемкости данного алгоритма на входных данных фиксированной длины. Пусть DA – множество конкретных проблем данной задачи, заданное в формальной системе. Пусть D DA – задание конкретной проблемы и |D| = N.
В общем случае существует собственное подмножество множества DA, включающее все конкретные проблемы, имеющие мощность N:

- обозначим это подмножество через DN: DN = {D DN,: |D| = N};
- обозначим мощность множества DN через MDN → MDN = |DN |. Тогда содержательно данный алгоритм, решая различные задачи размерности N, будет выполнять в каком-то случае наибольшее количество операций, а в каком-то случае наименьшее количество операций. Ведем следующие обозначения:

1. Fa(N) – худший случай – наибольшее количество операций, совершаемых алгоритмом A для решения конкретных проблем размерностью N: DDN Fa(N) = max {Fa (D)} – худший случай на DN
2. Fa(N) – лучший случай – наименьшее количество операций, совершаемых алгоритмом A для решения конкретных проблем размерностью N: DDN Fa(N) = min {Fa (D)} – лучший случай на DN
3. Fa(N) – средний случай – среднее количество операций, совершаемых алгоритмом A для решения конкретных проблем размерностью N: DDNFa(N) = (1 / MDN)* {Fa (D)} – средний случай на DN.

В зависимости от влияния исходных данных на функцию трудоемкости алгоритма может быть предложена следующая классификация, имеющая практическое значение для анализа алгоритмов: Количественно - зависимые по трудоемкости алгоритмы Это алгоритмы, функция трудоемкости которых зависит только от размерности конкретного входа, и не зависит от конкретных значений:

Fa (D) = Fa (|D|) = Fa (N). Примерами алгоритмов с количественно-зависимой функцией трудоемкости могут служить алгоритмы для стандартных операций с массивами и матрицами – умножение матриц, умножение матрицы на вектор и т.д. Параметрически - зависимые по трудоемкости алгоритмы
Это алгоритмы, трудоемкость которых определяется не размерностью входа (как правило, для этой группы размерность входа обычно фиксирована), а конкретными значениями обрабатываемых слов памяти:
Fa (D) = Fa (d1,…,dn) = Fa (P1,…,Pm), m n
Краткие теоретические сведения:
Примерами алгоритмов с параметрически-зависимой трудоемкостью являются алгоритмы вычисления стандартных функций с заданной точностью путем вычисления соответствующих степенных рядов. Очевидно, что такие алгоритмы, имея на входе два числовых значения – аргумент функции и точность выполняют существенно зависящее от значений количество операций.
а) Вычисление xk последовательным умножением Fa(x, k) = Fa (k).
б) Вычисление ex=(xn/n!), с точностью до Fa = Fa (x,)
Количественно-параметрические по трудоемкости алгоритмы
Однако в большинстве практических случаев функция трудоемкости зависит как от количества данных на входе, так и от значений входных данных, в этом случае:
Fa (D) = Fa (||D||, P1,…,Pm) = Fa (N, P1,…,Pm)
В качестве примера можно привести алгоритмы численных методов, в которых параметрически-зависимый внешний цикл по точности включает в себя количественно-зависимый фрагмент по размерности.
Порядково - зависимые по трудоемкости алгоритмы
Среди разнообразия параметрически - зависимых алгоритмов выделим еще оду группу, для которой количество операций зависит от порядка расположения исходных объектов.
Пусть множество D состоит из элементов (d1,…,dn), и ||D||=N,
Определим Dp = {(d1,…,dn)}-множество всех упорядоченных N-ок из d1,…,dn, отметим, что |Dp|=n!.
Если Fa (iDp) Fa (jDp), гдеiDp, jDp Dp, то алгоритм будем называть порядково-зависимым по трудоемкости. Примерами таких алгоритмов могут служить ряд алгоритмов сортировки, алгоритмы поиска минимума и максимума в массиве.
Рассмотрим более подробно алгоритм поиска максимума в массиве S, содержащим n элементов:
MaxS (S,n; Max)
Max S1
For i2 to n
if Max < Si then Max Si (количество выполненных операций присваивания зависит от порядка следования элементов массива)
При анализе поведения функции трудоемкости алгоритма часто используют принятые в математике асимптотические обозначения, позволяющие показать скорость роста функции, маскируя при этом конкретные коэффициенты.
Такая оценка функции трудоемкости алгоритма называется сложностью алгоритма и позволяет определить предпочтения в использовании того или иного алгоритма для больших значений размерности исходных данных.
В асимптотическом анализе приняты следующие обозначения [6]:
Пусть f(n) и g(n) – положительные функции положительного аргумента, n ≥ 1 (количество объектов на входе и количество операций – положительные числа), тогда:
n0
f,gnc2g(n)
f(n)
c1g(n)

f(n) = (g(n)),
если существуют положительные
c1, c2, n0, такие, что:
c1 * g(n) f(n) c2 * g(n),
при n > n0
Обычно говорят, что при этом функция g(n) является асимптотически точной оценкой функции f(n), т.к. по определению функция f(n) не отличается от функции g(n) с точностью до постоянного множителя.
Отметим, что из f(n) = (g(n)) следует, что g(n) = (f(n)).
Примеры:
1) f(n)=4n2+nlnN+174 – f(n)= (n2);
2) f(n)=(1) – запись означает, что f(n) или равна константе, не равной нулю, или f(n) ограничена константой на : f(n) = 7+1/n = (1).

В отличие от оценки , оценка O требует только, что бы функция f(n) не превышала g(n) начиная с n > n0, с точностью до постоянного множителя:
cg(n)
f(n)
c > 0, n0 > 0 :
0 f(n) c * g(n), n n0
Вообще, запись O(g(n)) обозначает класс функций, таких, что все они растут не быстрее, чем функция g(n) с точностью до постоянного множителя, поэтому иногда говорят, что g(n) мажорирует функцию f(n).
Например, для всех функций:
f(n)=1/n, f(n)= 12, f(n)=3*n+17, f(n)=n*Ln(n), f(n)=6*n2 +24*n+77
будет справедлива оценка O(n2)Указывая оценку O есть смысл указывать наиболее «близкую» мажорирующую функцию, поскольку например для f(n)= n2 справедлива оценка O(2n), однако она не имеет практического смысла.
В отличие от оценки O, оценка является оценкой снизу – т.е. определяет класс функций, которые растут не медленнее, чем g(n) с точностью до постоянного множителя:
F(n)
cg(n)
c > 0, n0 >0 :
0 c * g(n) f(n)
Например, запись (n*Ln(n)) обозначает класс функций, которые растут не медленнее, чем g(n) = n*Ln(n), в этот класс попадают все полиномы со степенью большей единицы, равно как и все степенные функции с основанием большим единицы.
Отметим, что не всегда для пары функций справедливо одно из асимптотических соотношений, например для f(n)=n1+sin(n) и g(n)=n не выполняется ни одно из асимптотических соотношений.
В асимптотическом анализе алгоритмов разработаны специальные методы получения асимптотических оценок, особенно для класса рекурсивных алгоритмов. Очевидно, что оценка является более предпочтительной, чем оценка O. Знание асимптотики поведения функции трудоемкости алгоритма - его сложности, дает возможность делать прогнозы по выбору более рационального с точки зрения трудоемкости алгоритма для больших размерностей исходных данных.

Порядок выполнения работы

- Составить математическую модель задачи.
- Выбрать и обосновать наиболее рациональный метод решения задачи;
- Разработать алгоритм для решения задачи.
- Написать и отладить программу.

Форма предоставления результата

- Блок – схема.
- Код программы.

Критерии оценки:

«Отлично» - теоретическое содержание курса освоено полностью, без пробелов, умения сформированы, все предусмотренные программой учебные задания выполнены, качество их выполнения оценено высоко.

–«Хорошо» - теоретическое содержание курса освоено полностью, без пробелов, некоторые умения сформированы недостаточно, все предусмотренные программой учебные задания выполнены, некоторые виды заданий выполнены с ошибками.

–«Удовлетворительно» - теоретическое содержание курса освоено частично, но пробелы не носят существенного характера, необходимые умения работы с освоенным материалом в основном сформированы, большинство предусмотренных программой обучения учебных заданий выполнено, некоторые из выполненных заданий содержат ошибки.

–«Неудовлетворительно» - теоретическое содержание курса не освоено, необходимые умения не сформированы, выполненные учебные задания содержат грубые ошибки.

Лабораторное занятие № 4,5 Оценка сложности эвристических алгоритмов.

Цель работы:

- Научиться формировать алгоритмы разработки программных модулей в соответствии с техническим заданием;
- Выполнять отладку программных модулей с использованием специализированных программных средств;
- Выполнять тестирование программных модулей;
- Осуществлять рефакторинг и оптимизацию программного кода.

Выполнив работу, Вы будете:

уметь:

- У1. Осуществлять разработку кода программного модуля на языках низкого и высокого уровней;
- У7. Формировать алгоритмы разработки программных модулей в соответствии с техническим заданием.
- У6. Оформлять документацию на программные средства;
- У2. Создавать программу по разработанному алгоритму как отдельный модуль.

Материальное обеспечение:

- Мультимедийные средства хранения, передачи и представления информации.

Краткие теоретические сведения:

Эвристический алгоритм — это алгоритм решения задачи, правильность которого для всех возможных случаев не доказана, но про который известно, что он даёт достаточно хорошее решение в большинстве случаев.

Наиболее частые эвристики:

- Жадный алгоритм
- Ограниченный перебор только перспективных вариантов
- Последовательное улучшение («локальный поиск»)

Жадный алгоритм — однопроходный итерационный алгоритм. Строит решение, добавляя на каждом шаге к текущему *частичному* решению новый элемент. Добавляемый элемент выбирается на основе локального оптимума («наилучший на текущем шаге»).

Жадные алгоритмы:

- Дают точное решение для задач на матроидах (с аддитивной целевой функцией)
- некоторых задач дают приближённое решение с гарантированной (не обязательно константной) оценкой приближения
- В общем случае используются как эвристики

Задание:

1. Некий путешественник задумал повторить путь А.Н. Радищева "Из Петербурга в Москву", для чего решил воспользоваться личным автомобилем. Путешественник решил так спланировать свой маршрут, чтобы минимизировать затраты. Для этого ему нужно было знать, сколько на его пути встретится заправочных станций и на каком расстоянии друг от друга они находятся. Помимо всего прочего приходилось учитывать, что емкость бензобака машины ограничена. Требуется определить, какое минимальное количество заправок ему нужно посетить.

Считать, что первая АЗС находится в Петербурге, а последняя в Москве.
Di - расстояние от i-ой до (i+1)-ой заправки;
S - расстояние, которое машина может проехать с полным баком;
L - расстояние от Петербурга до Москвы.
В ходе решения задачи нужно получить номера заправок, на которых придётся заправляться.
Решение:
В данной задаче жадный алгоритм будет работать таким образом:

1. Путешественник заправляет полный бак в исходной точке маршрута(Петербурге);
2. Если от данной заправки до Москвы бензина достаточно, то едем в Москву; иначе находим самую удалённую заправку, до которой можно доехать и едем туда, заправляем полные баки и повторяем шаг 2.

2. Перед праздниками шеф получает очень много приглашений на торжественные заседания. Чтобы лучше планировать свое время, шеф ввел правило, чтобы в каждом из приглашений четко указывался отрезок времени заседаний. Шеф не любит половинчатых решений, поэтому или находится на заседании все указанное время, или не приходит на него. Между посещениями заседаний должен быть хотя бы минимальный перерыв, то есть шеф может успеть на j-е заседание по списку приглашений, если. Напишите программу, позволяющую шефу посетить как можно больше заседаний.

Решение:
В этой задаче правильным оказывается неожиданно простой жадный алгоритм: на первом шаге выбрать заседание с наименьшим значением b, на каждом следующем шаге- с наименьшим значением b, но только среди тех заседаний, которые начинаются после конца предыдущего выбранного.
Для реализации представленного алгоритма сначала отсортируем массив интервалов времени по неубыванию значений b. Поскольку для окончательного ответа нужны исходные номера заседаний, целесообразно организовать данные в записи с полями a, b, idx (границы интервала заседания и его номер в начальном списке) и сортировать записи по значениям в поле b.

Порядок выполнения работы

- Составить математическую модель задачи.
- Выбрать и обосновать наиболее рациональный метод решения задачи;
- Разработать алгоритм для решения задачи.
- Написать и отладить программу.

Форма предоставления результата

- Блок – схема.
- Код программы.

Критерии оценки:

«Отлично» - теоретическое содержание курса освоено полностью, без пробелов, умения сформированы, все предусмотренные программой учебные задания выполнены, качество их выполнения оценено высоко.

–«Хорошо» - теоретическое содержание курса освоено полностью, без пробелов, некоторые умения сформированы недостаточно, все предусмотренные программой учебные задания выполнены, некоторые виды заданий выполнены с ошибками.

–«Удовлетворительно» - теоретическое содержание курса освоено частично, но пробелы не носят существенного характера, необходимые умения работы с освоенным материалом в основном сформированы, большинство предусмотренных программой обучения учебных заданий выполнено, некоторые из выполненных заданий содержат ошибки.

–«Неудовлетворительно» - теоретическое содержание курса не освоено, необходимые умения не сформированы, выполненные учебные задания содержат грубые ошибки.

Тема 1.1.3 Объектно-ориентированное программирование
Лабораторное занятие № 6 Работа с классами. Перегрузка методов.

Цель работы:

- Научиться формировать алгоритмы разработки программных модулей в соответствии с техническим заданием;
- Выполнять отладку программных модулей с использованием специализированных программных средств;
- Выполнять тестирование программных модулей;
- Осуществлять рефакторинг и оптимизацию программного кода.

Выполнив работу, Вы будете:

уметь:

- У1. Осуществлять разработку кода программного модуля на языках низкого и высокого уровней;
- У7. Формировать алгоритмы разработки программных модулей в соответствии с техническим заданием.
- У6. Оформлять документацию на программные средства;
- У2. Создавать программу по разработанному алгоритму как отдельный модуль.

Материальное обеспечение:

- Мультимедийные средства хранения, передачи и представления информации.

Задание:

1. Определить класс «Вектор в трехмерном пространстве».

Реализовать в виде класса методы для выполнения следующих операций над векторами:

- вывод координат вектора;
- получение длины вектора;
- сложение векторов;
- скалярное произведение векторов;
- определение угла между векторами;

Сложение векторов реализовать в виде перегрузки операции «+», взятие модуля вектора реализовать как результат приведения объекта класса к типу double.

В программе продемонстрировать использование объектов класса «Вектор в трехмерном пространстве»

Перегрузка операции присваивания реализована в классе vector3d для того, чтобы в выражении `c=a+b` вектор *c* не потерял свое имя. Если не определять перегрузку операции присваивания, то результат сложения в виде временно созданного в функции *operator+* объекта с именем вектора "t_copy" будет присвоен вектору *c* и поэлементно изменит значение всех его компонентных данных, в том числе и имя вектора. Чтобы вектор не сменил имя, а только получил новые координаты, перегружена операция присваивания.

2. Создать внешний вид формы программы
3. Написать программные коды для созданной формы
4. Запустить и отладить программу

Порядок выполнения работы

- Составить математическую модель задачи.
- Выбрать и обосновать наиболее рациональный метод решения задачи;
- Разработать алгоритм для решения задачи.
- Написать и отладить программу.

Форма предоставления результата

- Блок – схема.
- Код программы.

Критерии оценки:

«Отлично» - теоретическое содержание курса освоено полностью, без пробелов, умения сформированы, все предусмотренные программой учебные задания выполнены, качество их выполнения оценено высоко.

–«Хорошо» - теоретическое содержание курса освоено полностью, без пробелов, некоторые умения сформированы недостаточно, все предусмотренные программой учебные задания выполнены, некоторые виды заданий выполнены с ошибками.

–«Удовлетворительно» - теоретическое содержание курса освоено частично, но пробелы не носят существенного характера, необходимые умения работы с освоенным материалом в основном сформированы, большинство предусмотренных программой обучения учебных заданий выполнено, некоторые из выполненных заданий содержат ошибки.

–«Неудовлетворительно» - теоретическое содержание курса не освоено, необходимые умения не сформированы, выполненные учебные задания содержат грубые ошибки.

Лабораторное занятие № 7 Определение операций в классе. Создание наследованных классов.

Цель работы:

- Научиться формировать алгоритмы разработки программных модулей в соответствии с техническим заданием;
- Выполнять отладку программных модулей с использованием специализированных программных средств;
- Выполнять тестирование программных модулей;
- Осуществлять рефакторинг и оптимизацию программного кода.

Выполнив работу, Вы будете:

уметь:

- У1. Осуществлять разработку кода программного модуля на языках низкого и высокого уровней;
- У7. Формировать алгоритмы разработки программных модулей в соответствии с техническим заданием.

- У6. Оформлять документацию на программные средства;
- У2. Создавать программу по разработанному алгоритму как отдельный модуль.

Материальное обеспечение:

- Мультимедийные средства хранения, передачи и представления информации.

Задание:

1. Разработать перечисленные ниже классы.

При разработке каждого класса возможны два варианта решения:

а) данные-члены класса представляют собой переменные и массивы фиксированной размерности;

б) память для данных-членов класса выделяется динамически.

- «**Комплексное число**» – **Complex**. Класс должен содержать несколько конструкторов и операции для сложения, вычитания, умножения, деления, присваивания. Создать два вектора размерности *n* из комплексных координат. Передать их в функцию, которая выполняет сложение комплексных векторов.

- Разработать класс **«Многочлен» – Polynom** степени *n* . Написать несколько конструкторов, в том числе конструктор копирования. Реализовать методы для вычисления значения полинома; сложения,

вычитания и умножения полиномов. Перегрузить операции сложения, вычитания, умножения, инкремента, декремента, индексирования, присваивания. Создать массив объектов класса. Передать его в функцию, вычисляющую сумму полиномов массива и возвращающую полином-результат, который выводится на экран в головной программе.

Порядок выполнения работы

- Составить математическую модель задачи.
- Выбрать и обосновать наиболее рациональный метод решения задачи;
- Разработать алгоритм для решения задачи.
- Создать внешний вид формы программы
- Написать программные коды для созданной формы
- Запустить и отладить программу
- Написать и отладить программу.

Форма предоставления результата

- Блок – схема.
- Код программы.

Критерии оценки:

«Отлично» - теоретическое содержание курса освоено полностью, без пробелов, умения сформированы, все предусмотренные программой учебные задания выполнены, качество их выполнения оценено высоко.

–«Хорошо» - теоретическое содержание курса освоено полностью, без пробелов, некоторые умения сформированы недостаточно, все предусмотренные программой учебные задания выполнены, некоторые виды заданий выполнены с ошибками.

–«Удовлетворительно» - теоретическое содержание курса освоено частично, но пробелы не носят существенного характера, необходимые умения работы с освоенным материалом в основном сформированы, большинство предусмотренных программой обучения учебных заданий выполнено, некоторые из выполненных заданий содержат ошибки.

–«Неудовлетворительно» - теоретическое содержание курса не освоено, необходимые умения не сформированы, выполненные учебные задания содержат грубые ошибки.

Лабораторное занятие № 8 Работа с объектами через интерфейсы. Использование стандартных интерфейсов.

Цель работы:

- Научиться формировать алгоритмы разработки программных модулей в соответствии с техническим заданием;
- Выполнять отладку программных модулей с использованием специализированных программных средств;
- Выполнять тестирование программных модулей;
- Осуществлять рефакторинг и оптимизацию программного кода.

Выполнив работу, Вы будете:

уметь:

- У1. Осуществлять разработку кода программного модуля на языках низкого и высокого уровней;
- У7. Формировать алгоритмы разработки программных модулей в соответствии с техническим заданием.
- У6. Оформлять документацию на программные средства;
- У2. Создавать программу по разработанному алгоритму как отдельный модуль.

Материальное обеспечение:

- Мультимедийные средства хранения, передачи и представления информации.

Задание:

1. Разработать приложение исходя их следующих требований:

1.1. Приложение должно состоять минимум из двух форм. Первая – для работы со списком объектов, вторая – для работы с содержимым самого объекта

1.2. На форме должны быть предусмотрены операции “Добавления”, “Удаления”, “Изменения”, “Сортировки” и “Поиска” элементов списка.

1.3. Операции сортировки должны выполняться с использованием метода Sort() шаблонного класса List<T>. Количество вариантов сортировки – не менее 3.

1.4. Операция поиска выполняется только по ключевому атрибуту, с использование возможностей класса Dictionary<T, T>.

Любой класс или структура, реализующие интерфейс IEquatable<T>, должны содержать определение метода Equals, который соответствует сигнатуре, определяемой интерфейсом. В результате можно рассчитывать, что любой класс, реализующий интерфейс IEquatable<T>, будет содержать метод Equals, с которым экземпляр класса может определить, равен ли он другому экземпляру класса.

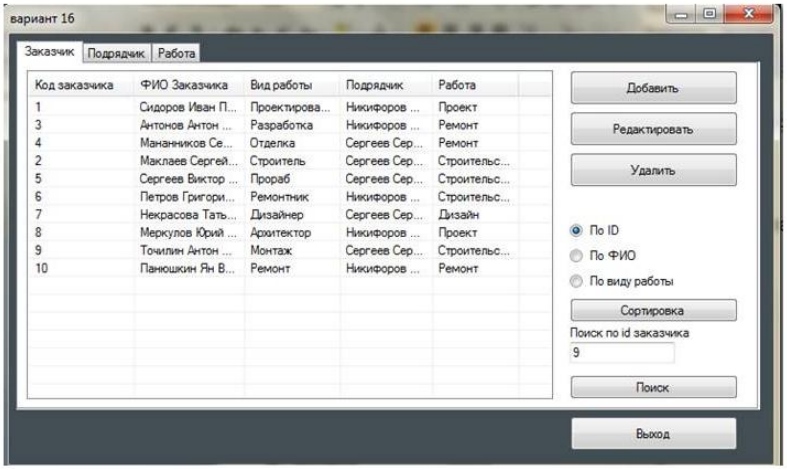
Для реализации члена интерфейса соответствующий член класса должен быть открытым, нестатическим, и иметь то же имя и сигнатуру, что и член интерфейса.

Когда класс или структура реализуют интерфейс, класс или структура должны обеспечивать реализацию всех членов которые определяет интерфейс. Сам интерфейс не предоставляет никакой функциональности, класс или структура может наследовать таким способом, она может наследовать функциональность базового класса. Однако если базовый класс реализует интерфейс, то любой класс, производный от базового класса наследует ее реализацию.

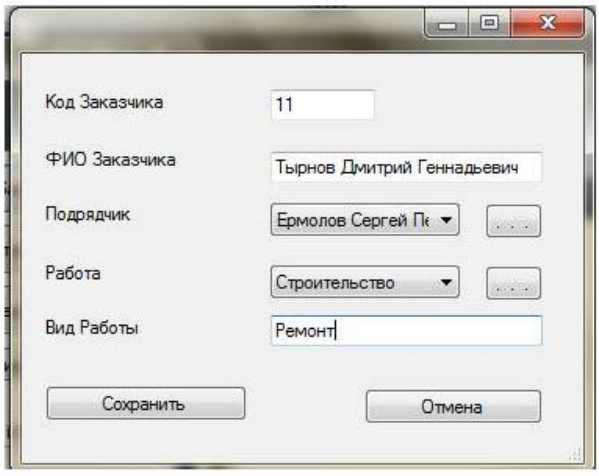
Работа с коллекциями объектов, класс List<T>.

Шаблонный класс List<T> является удобным механизмом для работы со списком однотипных объектов и представляет строго типизированный список объектов,

доступных по индексу. Поддерживает методы для поиска по списку, выполнения сортировки и других операций со списками.
Сортировка элементов списка осуществляется с помощью метода Sort(). Если тип элементов списка является системным, то это означает в частности, что данные типы уже реализуют интерфейс IEquatable<T> (метод Equals()) и интерфейс IComparable<T> (метод CompareTo(T)). Таким образом, сортировка выполняется методом Sort(), без написания дополнительного кода.
Если тип элементов списка – пользовательский, то это означает, что методы проверки равенства (Equals) и сравнения элементов (CompareTo) должны быть переопределены.
Форма для работы со списком объектов показана ниже:
На форме предусмотрены операции “Добавления”, “Удаления”, “Редактирования”, “Сортировки” и “Поиска” элементов списка.

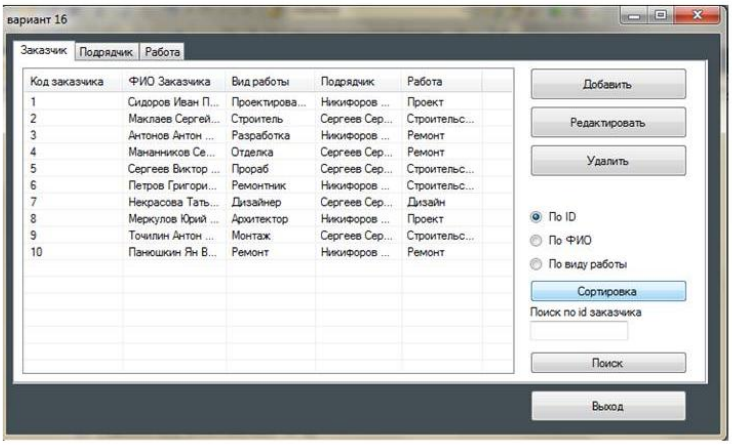


Форма для работы с содержимым самого объекта выглядит так:

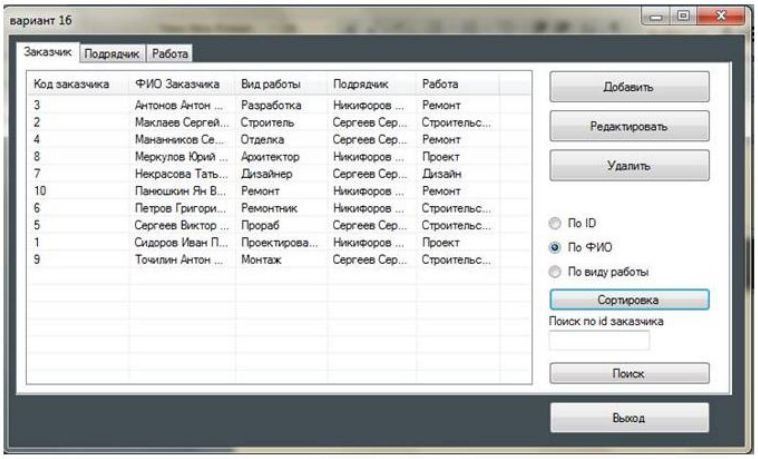


Сортировка может осуществляться 3 способами

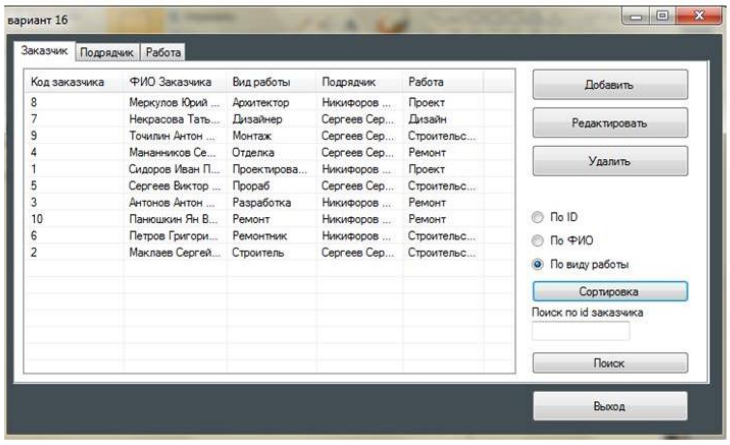
По ID заказчика:



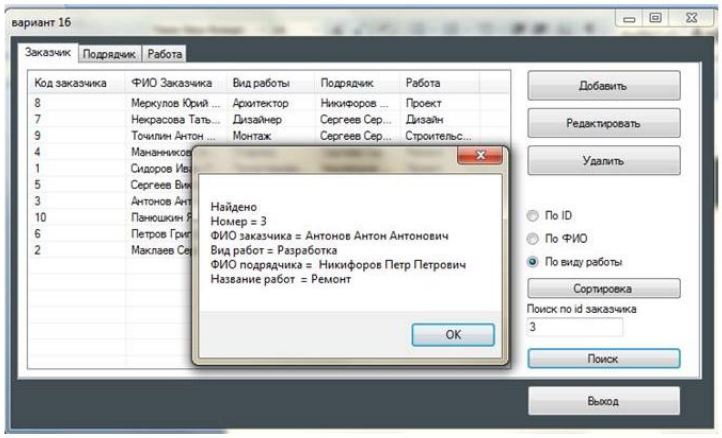
По ФИО заказчика (в алфавитном порядке):

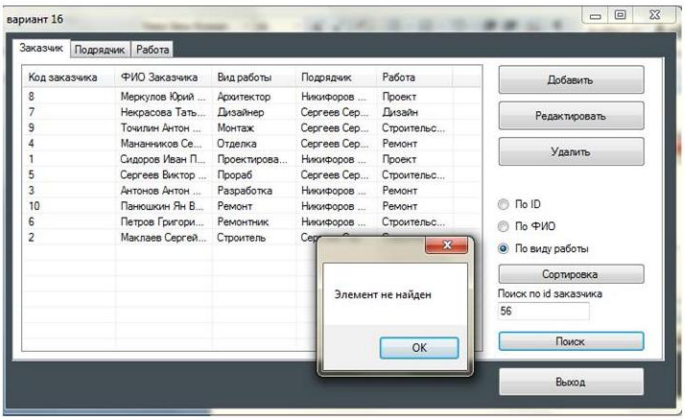


По виду работы (в алфавитном порядке):



Поиск происходит по ключевому полю:





Операция поиска выполняется только по ключевому атрибуту, с использованием возможностей класса Dictionary<T, T>.

Порядок выполнения работы

- Составить математическую модель задачи.
- Выбрать и обосновать наиболее рациональный метод решения задачи;
- Разработать алгоритм для решения задачи.
- Создать внешний вид формы программы
- Написать программные коды для созданной формы
- Запустить и отладить программу

Форма предоставления результата

- Блок – схема.
- Код программы.

Критерии оценки:

«Отлично» - теоретическое содержание курса освоено полностью, без пробелов, умения сформированы, все предусмотренные программой учебные задания выполнены, качество их выполнения оценено высоко.

–«Хорошо» - теоретическое содержание курса освоено полностью, без пробелов, некоторые умения сформированы недостаточно, все предусмотренные программой учебные задания выполнены, некоторые виды заданий выполнены с ошибками.

–«Удовлетворительно» - теоретическое содержание курса освоено частично, но пробелы не носят существенного характера, необходимые умения работы с освоенным материалом в основном сформированы, большинство предусмотренных программой обучения учебных заданий выполнено, некоторые из выполненных заданий содержат ошибки.

–«Неудовлетворительно» - теоретическое содержание курса не освоено, необходимые умения не сформированы, выполненные учебные задания содержат грубые ошибки.

Лабораторное занятие№ 9 Работа с типом данных структура. Коллекции. Параметризованные классы.

Цель работы:

- Научиться формировать алгоритмы разработки программных модулей в соответствии с техническим заданием;
- Выполнять отладку программных модулей с использованием специализированных программных средств;
- Выполнять тестирование программных модулей;
- Осуществлять рефакторинг и оптимизацию программного кода.

Выполнив работу, Вы будете:

уметь:

- У1. Осуществлять разработку кода программного модуля на языках низкого и высокого уровней;
- У7. Формировать алгоритмы разработки программных модулей в соответствии с техническим заданием.
- У6. Оформлять документацию на программные средства;
- У2. Создавать программу по разработанному алгоритму как отдельный модуль.

Материальное обеспечение:

- Мультимедийные средства хранения, передачи и представления информации.

Задание:

1. Создать шаблон заданного класса. Определить конструкторы, деструктор, перегруженную операцию присваивания (“=”) и операции, заданные в варианте задания.
2. Написать программу тестирования, в которой проверяется использование шаблона для стандартных типов данных.
3. Выполнить тестирование.
4. Определить пользовательский класс, который будет использоваться в качестве параметра шаблона. Определить в классе необходимые функции и перегруженные операции.
5. Написать программу тестирования, в которой проверяется использование шаблона для пользовательского типа.
6. Выполнить тестирование.

Методические указания

1. Класс АТД реализовать как динамический массив. Для этого определение класса должно иметь следующие поля:

– указатель на начало массива;
– максимальный размер массива;
– текущий размер массива.

2. Для вода и вывода определить в классе функции inputи print.3
3. Чтобы у вас не возникало проблем, аккуратно работайте с константными объектами.

   Например:

   *конструктор копирования следует определить так: MyTmp (const MyTmp& ob);

   *операцию присваивания перегрузить так: MyTmp& operator= (const MyTmp& ob);4.

Для шаблонов множеств, списков, стеков и очередей в качестве стандартных типов использовать символьные, целые и вещественные ти-пы. Для пользовательского типа взять класс из лабораторной работы No 1.5. Для шаблонов массивов в качестве стандартных типов использовать целые и вещественные типы. Для пользовательского типа взять класс “комплексное число”complex.classcomplex{intre; // действительная часть intim; // мнимая часть public;// необходимые функции и перегруженные операции};

6. Реализацию шаблона следует разместить вместе с определением в заголовочном файле.

7. Программа создается как EasyWin-приложение в BorlandC++5.02.

Постановка задачи.

- Следует дать конкретную постановку, т.е указать шаблон какого класса должен быть создан, какие должны быть в нем конструкторы, компоненты-функции, перегруженные операции и т.д. То же самое следует указать для пользовательского класса.
- Определение шаблона класса с комментариями.
- Определение пользовательского класса с комментариями.
- Реализация конструкторов, деструктора, операции присваивания и операций, которые заданы в варианте задания
- То же самое для пользовательского класса.
- Результаты тестирования. Следует указать для каких типов и какие операции проверены и какие выявлены ошибки (или не выявлены)

Варианты заданий

1.Класс –одномерный массив.

Дополнительно перегрузить следующие операции: * – умножение массивов; [] – доступ по индексу.

2. Класс –одномерный массив. Дополнительно перегрузить следующие операции:int() – размер массива;[] – доступ по индексу.

Порядок выполнения работы

- Составить математическую модель задачи.
- Выбрать и обосновать наиболее рациональный метод решения задачи;
- Разработать алгоритм для решения задачи.
- Создать внешний вид формы программы
- Написать программные коды для созданной формы
- Запустить и отладить программу

Форма предоставления результата

- Блок – схема.
- Код программы.

Критерии оценки:

«Отлично» - теоретическое содержание курса освоено полностью, без пробелов, умения сформированы, все предусмотренные программой учебные задания выполнены, качество их выполнения оценено высоко.

–«Хорошо» - теоретическое содержание курса освоено полностью, без пробелов, некоторые умения сформированы недостаточно, все предусмотренные программой учебные задания выполнены, некоторые виды заданий выполнены с ошибками.

–«Удовлетворительно» - теоретическое содержание курса освоено частично, но пробелы не носят существенного характера, необходимые умения работы с освоенным материалом в основном сформированы, большинство предусмотренных программой обучения учебных заданий выполнено, некоторые из выполненных заданий содержат ошибки.

–«Неудовлетворительно» - теоретическое содержание курса не освоено, необходимые умения не сформированы, выполненные учебные задания содержат грубые ошибки.

Лабораторное занятие№ 10 Использование регулярных выражений. Операции со списками.

Цель работы:

- Научиться формировать алгоритмы разработки программных модулей в соответствии с техническим заданием;
- Выполнять отладку программных модулей с использованием специализированных программных средств;
- Выполнять тестирование программных модулей;
- Осуществлять рефакторинг и оптимизацию программного кода.

Выполнив работу, Вы будете:

уметь:

- У1. Осуществлять разработку кода программного модуля на языках низкого и высокого уровней;
- У7. Формировать алгоритмы разработки программных модулей в соответствии с техническим заданием.
- У6. Оформлять документацию на программные средства;
- У2. Создавать программу по разработанному алгоритму как отдельный модуль.

Материальное обеспечение:

- Мультимедийные средства хранения, передачи и представления информации.

Задание:

1. Определить класс-строку. В класс включить два конструктора: для определения класса строки строкой символов и путем копирования другой строки (объекта класса строки). Предусмотреть функции поиска слова в строке и добавления другой строки, начиная с позиции N.
2. Создать внешний вид формы программы
3. Написать программные коды для созданной формы
4. Запустить и отладить программу

2. Определить класс-строку. В класс включить два конструктора: для определения класса строки строкой символов и путем копирования другой строки (объекта класса строки). Предусмотреть функции слияния двух строк и функцию подсчёта предложений в строке.

Порядок выполнения работы

- Составить математическую модель задачи.
- Выбрать и обосновать наиболее рациональный метод решения задачи;
- Разработать алгоритм для решения задачи.
- Создать внешний вид формы программы
- Написать программные коды для созданной формы
- Запустить и отладить программу

Форма предоставления результата

- Блок – схема.
- Код программы.

Критерии оценки:

«Отлично» - теоретическое содержание курса освоено полностью, без пробелов, умения сформированы, все предусмотренные программой учебные задания выполнены, качество их выполнения оценено высоко.

–«Хорошо» - теоретическое содержание курса освоено полностью, без пробелов, некоторые умения сформированы недостаточно, все предусмотренные программой учебные задания выполнены, некоторые виды заданий выполнены с ошибками.

–«Удовлетворительно» - теоретическое содержание курса освоено частично, но пробелы не носят существенного характера, необходимые умения работы с освоенным материалом в основном сформированы, большинство предусмотренных программой обучения учебных заданий выполнено, некоторые из выполненных заданий содержат ошибки.

–«Неудовлетворительно» - теоретическое содержание курса не освоено, необходимые умения не сформированы, выполненные учебные задания содержат грубые ошибки.

Тема 1.1.4 Паттерны проектирования

Лабораторное занятие№ 11,12 Использование основных шаблонов

Цель работы:

- Научиться формировать алгоритмы разработки программных модулей в соответствии с техническим заданием;
- Выполнять отладку программных модулей с использованием специализированных программных средств;
- Выполнять тестирование программных модулей;
- Осуществлять рефакторинг и оптимизацию программного кода.

Выполнив работу, Вы будете:

уметь:

- У1. Осуществлять разработку кода программного модуля на языках низкого и высокого уровней;
- У7. Формировать алгоритмы разработки программных модулей в соответствии с техническим заданием.
- У6. Оформлять документацию на программные средства;
- У2. Создавать программу по разработанному алгоритму как отдельный модуль.

Материальное обеспечение:

- Мультимедийные средства хранения, передачи и представления информации.

Задание:

1. В каждом из вариантов указан шаблон для реализации и проект, использующий этот шаблон.

Необходимо сделать следующее:

- Нарисовать в UML диаграмму классов реализуемой программы. (проектирование)
- Реализовать программу на C++. (реализация)

Для каждого из шаблонов, предложенных в вариантах можно найти пример реализации UML и кода в приложенной книге “Паттерны проектирования”.

2. Шаблон “Стратегия”. Проект “Принтеры”. В проекте должны быть реализованы разные модели принтеров, которые выполняют разные виды печати.

3. Шаблон “Наблюдатель”. Проект “Оповещение постов ГАИ”. В проекте должна быть реализована отправка сообщений всем постам ГАИ.

Порядок выполнения работы

- Составить математическую модель задачи.
- Выбрать и обосновать наиболее рациональный метод решения задачи;
- Разработать алгоритм для решения задачи.
- Создать внешний вид формы программы
- Написать программные коды для созданной формы
- Запустить и отладить программу

Форма предоставления результата

- Блок – схема.
- Код программы.

Критерии оценки:

«Отлично» - теоретическое содержание курса освоено полностью, без пробелов, умения сформированы, все предусмотренные программой учебные задания выполнены, качество их выполнения оценено высоко.

–«Хорошо» - теоретическое содержание курса освоено полностью, без пробелов, некоторые умения сформированы недостаточно, все предусмотренные программой учебные задания выполнены, некоторые виды заданий выполнены с ошибками.

–«Удовлетворительно» - теоретическое содержание курса освоено частично, но пробелы не носят существенного характера, необходимые умения работы с освоенным материалом в основном сформированы, большинство предусмотренных программой обучения учебных заданий выполнено, некоторые из выполненных заданий содержат ошибки.

–«Неудовлетворительно» - теоретическое содержание курса не освоено, необходимые умения не сформированы, выполненные учебные задания содержат грубые ошибки.

Лабораторное занятие № 13. Использование порождающих шаблонов.

Цель работы:

- Научиться формировать алгоритмы разработки программных модулей в соответствии с техническим заданием;
- Выполнять отладку программных модулей с использованием специализированных программных средств;
- Выполнять тестирование программных модулей;
- Осуществлять рефакторинг и оптимизацию программного кода.

Выполнив работу, Вы будете:

уметь:

- У1. Осуществлять разработку кода программного модуля на языках низкого и высокого уровней;
- У7. Формировать алгоритмы разработки программных модулей в соответствии с техническим заданием.
- У6. Оформлять документацию на программные средства;
- У2. Создавать программу по разработанному алгоритму как отдельный модуль.

Материальное обеспечение:

- Мультимедийные средства хранения, передачи и представления информации.

Задание:

1. В каждом из вариантов указан шаблон для реализации и проект, использующий этот шаблон.

Необходимо сделать следующее:

- Нарисовать в UML диаграмму классов реализуемой программы. (проектирование)
- Реализовать программу на C++. (реализация)

Для каждого из шаблонов, предложенных в вариантах можно найти пример реализации UML и кода в приложенной книге “Паттерны проектирования”.

2. Шаблон “Декоратор”. Проект “Универсальная электронная карта”. В проекте должна быть реализована универсальная электронная карта, в которой есть функции паспорта, страхового полиса, банковской карты и т. д.
3. Шаблон “Фабричный метод”. Проект “Фабрика смартфонов”. В проекте должно быть реализовано создание смартфонов с различными характеристиками.

Порядок выполнения работы

- Составить математическую модель задачи.
- Выбрать и обосновать наиболее рациональный метод решения задачи;
- Разработать алгоритм для решения задачи.
- Создать внешний вид формы программы
- Написать программные коды для созданной формы
- Запустить и отладить программу

Форма предоставления результата

- Блок – схема.
- Код программы.

Критерии оценки:

«Отлично» - теоретическое содержание курса освоено полностью, без пробелов, умения сформированы, все предусмотренные программой учебные задания выполнены, качество их выполнения оценено высоко.

–«Хорошо» - теоретическое содержание курса освоено полностью, без пробелов, некоторые умения сформированы недостаточно, все предусмотренные программой учебные задания выполнены, некоторые виды заданий выполнены с ошибками.

–«Удовлетворительно» - теоретическое содержание курса освоено частично, но пробелы не носят существенного характера, необходимые умения работы с освоенным материалом в основном сформированы, большинство предусмотренных программой обучения учебных заданий выполнено, некоторые из выполненных заданий содержат ошибки.

–«Неудовлетворительно» - теоретическое содержание курса не освоено, необходимые умения не сформированы, выполненные учебные задания содержат грубые ошибки.

Лабораторное занятие № 14. Использование структурных шаблонов.

Цель работы:

- Научиться формировать алгоритмы разработки программных модулей в соответствии с техническим заданием;
- Выполнять отладку программных модулей с использованием специализированных программных средств;
- Выполнять тестирование программных модулей;
- Осуществлять рефакторинг и оптимизацию программного кода.

Выполнив работу, Вы будете:

уметь:

- У1. Осуществлять разработку кода программного модуля на языках низкого и высокого уровней;
- У7. Формировать алгоритмы разработки программных модулей в соответствии с техническим заданием.
- У6. Оформлять документацию на программные средства;
- У2. Создавать программу по разработанному алгоритму как отдельный модуль.

Материальное обеспечение:

- Мультимедийные средства хранения, передачи и представления информации.

Задание:

1. В каждом из вариантов указан шаблон для реализации и проект, использующий этот шаблон.

Необходимо сделать следующее:

- Нарисовать в UML диаграмму классов реализуемой программы. (проектирование)
- Реализовать программу на C++. (реализация)

Для каждого из шаблонов, предложенных в вариантах можно найти пример реализации UML и кода в приложенной книге “Паттерны проектирования”.

1. Постановка задачи

Шаблон “Стратегия”. Проект “Принтеры”. В проекте должны быть реализованы разные модели принтеров, которые выполняют разные виды печати.

2. Шаблон “Наблюдатель”. Проект “Оповещение постов ГАИ”. В проекте должна быть реализована отправка сообщений всем постам ГАИ.

Порядок выполнения работы

- Составить математическую модель задачи.
- Выбрать и обосновать наиболее рациональный метод решения задачи;
- Разработать алгоритм для решения задачи.
- Создать внешний вид формы программы
- Написать программные коды для созданной формы
- Запустить и отладить программу

Форма предоставления результата

- Блок – схема.
- Код программы.

Критерии оценки:

«Отлично» - теоретическое содержание курса освоено полностью, без пробелов, умения сформированы, все предусмотренные программой учебные задания выполнены, качество их выполнения оценено высоко.

–«Хорошо» - теоретическое содержание курса освоено полностью, без пробелов, некоторые умения сформированы недостаточно, все предусмотренные программой учебные задания выполнены, некоторые виды заданий выполнены с ошибками.

–«Удовлетворительно» - теоретическое содержание курса освоено частично, но пробелы не носят существенного характера, необходимые умения работы с освоенным материалом в основном сформированы, большинство предусмотренных программой обучения учебных заданий выполнено, некоторые из выполненных заданий содержат ошибки.

–«Неудовлетворительно» - теоретическое содержание курса не освоено, необходимые умения не сформированы, выполненные учебные задания содержат грубые ошибки.

Лабораторное занятие № 15. Использование поведенческих шаблонов.

Цель работы:

- Научиться формировать алгоритмы разработки программных модулей в соответствии с техническим заданием;
- Выполнять отладку программных модулей с использованием специализированных программных средств;
- Выполнять тестирование программных модулей;
- Осуществлять рефакторинг и оптимизацию программного кода.

Выполнив работу, Вы будете:

уметь:

- У1. Осуществлять разработку кода программного модуля на языках низкого и высокого уровней;
- У7. Формировать алгоритмы разработки программных модулей в соответствии с техническим заданием.
- У6. Оформлять документацию на программные средства;
- У2. Создавать программу по разработанному алгоритму как отдельный модуль.

Материальное обеспечение:

- Мультимедийные средства хранения, передачи и представления информации.

Задание:

В каждом из вариантов указан шаблон для реализации и проект, использующий этот шаблон.

Необходимо сделать следующее:

- Нарисовать в UML диаграмму классов реализуемой программы. (проектирование)
- Реализовать программу на C++. (реализация)

Для каждого из шаблонов, предложенных в вариантах можно найти пример реализации UML и кода в приложенной книге “Паттерны проектирования”.

1. Шаблон “Фабричный метод”. Проект “Фабрика смартфонов”. В проекте должно быть реализовано создание смартфонов с различными характеристиками. Пример использования шаблона в главе 4.
2. Шаблон “Абстрактная фабрика”. Проект “Заводы по производству автомобилей”. В проекте должно быть реализована возможность создавать автомобили различных типов на разных заводах.

Порядок выполнения работы

- Составить математическую модель задачи.
- Выбрать и обосновать наиболее рациональный метод решения задачи;
- Разработать алгоритм для решения задачи.
- Создать внешний вид формы программы
- Написать программные коды для созданной формы
- Запустить и отладить программу
-

Форма предоставления результата

- Блок – схема.
- Код программы.

Критерии оценки:

«Отлично» - теоретическое содержание курса освоено полностью, без пробелов, умения сформированы, все предусмотренные программой учебные задания выполнены, качество их выполнения оценено высоко.

–«Хорошо» - теоретическое содержание курса освоено полностью, без пробелов, некоторые умения сформированы недостаточно, все предусмотренные программой учебные задания выполнены, некоторые виды заданий выполнены с ошибками.

–«Удовлетворительно» - теоретическое содержание курса освоено частично, но пробелы не носят существенного характера, необходимые умения работы с освоенным материалом в основном сформированы, большинство предусмотренных программой обучения учебных заданий выполнено, некоторые из выполненных заданий содержат ошибки.

–«Неудовлетворительно» - теоретическое содержание курса не освоено, необходимые умения не сформированы, выполненные учебные задания содержат грубые ошибки.

–

Тема 1.1.5. Событийно-управляемое программирование

Лабораторное занятие № 16,17. Разработка приложения с использованием текстовых компонентов

Цель работы:

- Научиться формировать алгоритмы разработки программных модулей в соответствии с техническим заданием;
- Выполнять отладку программных модулей с использованием специализированных программных средств;
- Выполнять тестирование программных модулей;
- Осуществлять рефакторинг и оптимизацию программного кода.

Выполнив работу, Вы будете:

уметь:

- У1. Осуществлять разработку кода программного модуля на языках низкого и высокого уровней;
- У7. Формировать алгоритмы разработки программных модулей в соответствии с техническим заданием.
- У6. Оформлять документацию на программные средства;
- У2. Создавать программу по разработанному алгоритму как отдельный модуль.

Материальное обеспечение:

- Мультимедийные средства хранения, передачи и представления информации.

Задание:

Для решения задачи использовать простые текстовые компоненты и кнопки:

- TEdit - для ввода скалярных исходных данных,
- TLabel – для задания подписей элементов формы,
- TButton – для активизации действий («Вычислить», «Выполнить новый расчет»),
- TMemo – для вывода условия задачи, исходных массивов и вывода результата.

Все компоненты расположить на панелях (TPanel). При разработке программного интерфейса выровнять панели относительно границ формы, а компоненты относительно границ панели с использованием свойств выравнивания и фиксации компонент относительно контейнера (Align, Anchor). Компоненты, в которых выводятся массивы и другие результаты, должны быть недоступными для изменения.

Примеры внешнего вида форм приведены ниже на рисунках.

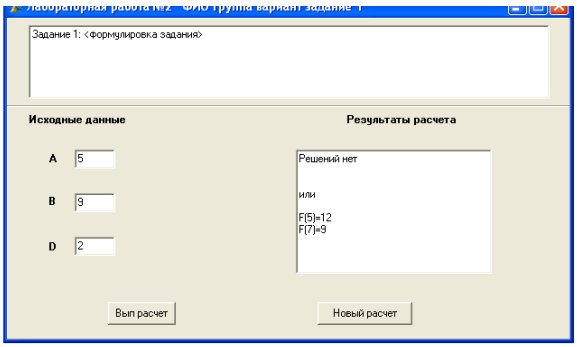


Рис. 2. Пример расположения компонентов на форме для задания 1

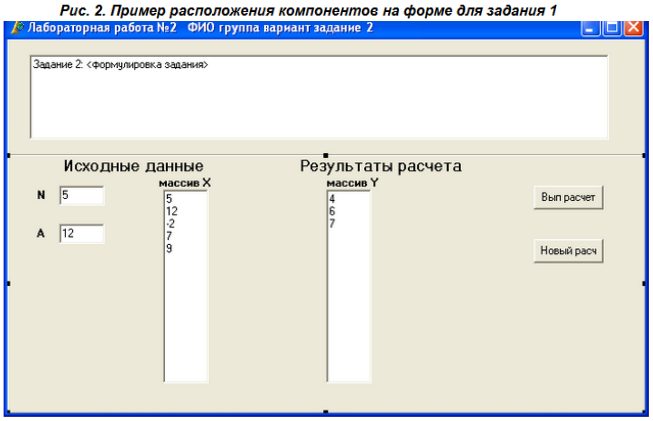


Рис. 3. Пример расположения компонентов на форме для задания 2

Программирование событий

Управление работой оконного приложения выполняется по событиям (открытие формы, щелчок мышью по кнопке и т.д.).

Действия (выполняемые операторами программы) при открытии формы необходимо добавить в обработчик события открытия формы OnShow.

Действия (выполняемые операторами программы) при нажатии кнопки (щелчке мышью по кнопке) необходимо добавить в обработчик события «щелчок» по кнопке OnClick. Для этого в инспекторе объектов необходимо в выпадающем списке выбрать объект-кнопку (допустим кнопке «Выполнить расчет» соответствует объект Button1). Перейти на закладку Events, найти свойство OnClick и справа в пустом поле сделать двойной щелчок мышью (можно на форме сделать двойной щелчок по кнопке). В программном модуле будет добавлен шаблон процедуры обработчика:

Разработать оконное приложение для решения задач индивидуального задания. Исходные массивы сгенерировать с использованием датчика случайных чисел и вывести в компоненте формы. В программе использовать динамические массивы.

1. Даны координаты точки (X,Y). Определить, попадает ли точка внутрь кольца с центром в точке (C,D) и радиусами R1, R2.

2. Дан одномерный массив Zm. Сформировать массив Yn, состоящий из элементов массива Zm, значение которых меньше среднего арифметического нечетных элементов исходного массива.

Порядок выполнения работы

- Составить математическую модель задачи.
- Выбрать и обосновать наиболее рациональный метод решения задачи;
- Разработать алгоритм для решения задачи.
- Создать внешний вид формы программы
- Написать программные коды для созданной формы
- Запустить и отладить программу

Форма предоставления результата

- Блок – схема.
- Код программы.

Критерии оценки:

«Отлично» - теоретическое содержание курса освоено полностью, без пробелов, умения сформированы, все предусмотренные программой учебные задания выполнены, качество их выполнения оценено высоко.

–«Хорошо» - теоретическое содержание курса освоено полностью, без пробелов, некоторые умения сформированы недостаточно, все предусмотренные программой учебные задания выполнены, некоторые виды заданий выполнены с ошибками.

–«Удовлетворительно» - теоретическое содержание курса освоено частично, но пробелы не носят существенного характера, необходимые умения работы с освоенным материалом в основном сформированы, большинство предусмотренных программой обучения учебных заданий выполнено, некоторые из выполненных заданий содержат ошибки.

–«Неудовлетворительно» - теоретическое содержание курса не освоено, необходимые умения не сформированы, выполненные учебные задания содержат грубые ошибки.

Лабораторное занятие № 18,19. Разработка приложения с несколькими формами.

Цель работы:

- Научиться формировать алгоритмы разработки программных модулей в соответствии с техническим заданием;
- Выполнять отладку программных модулей с использованием специализированных программных средств;
- Выполнять тестирование программных модулей;
- Осуществлять рефакторинг и оптимизацию программного кода.

Выполнив работу, Вы будете:

уметь:

- У1. Осуществлять разработку кода программного модуля на языках низкого и высокого уровней;
- У7. Формировать алгоритмы разработки программных модулей в соответствии с техническим заданием.
- У6. Оформлять документацию на программные средства;
- У2. Создавать программу по разработанному алгоритму как отдельный модуль.

Материальное обеспечение:

- Мультимедийные средства хранения, передачи и представления информации.

Задание:

1.Разработайте два класса потомка от Animal, которые будут отображать особенности двух пород собак. Разработайте методы для этих классов, позволяющие получить некоторые характеристики породы (рост, длина шерсти, длина ушей и т.д.). Дополните форму компонентами, позволяющими увидеть все характеристики разработанных классов.

Примерная форма проекта представлена на рисунке 4.4.

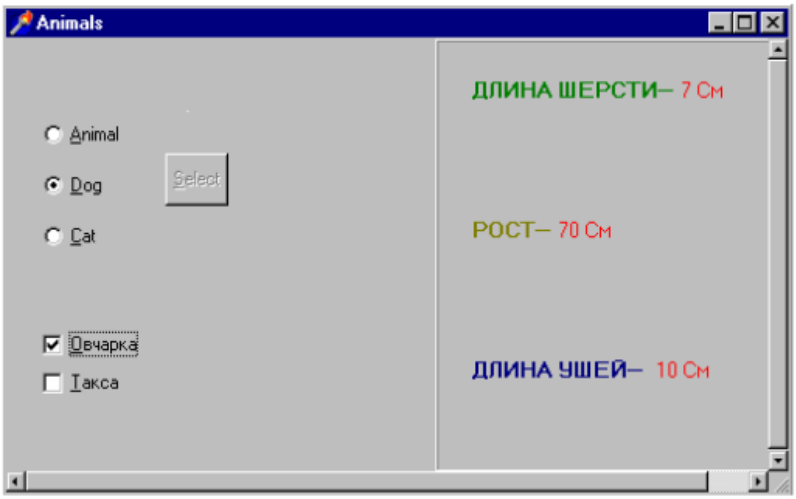


Рис. 4.4. Форма характеристик пород собак

После запуска программы представляется возможность выбора между значениями: Animal, Dog, Cat.
Выбор осуществляется при помощи кнопки SELECT. Далее предоставляется дополнительная возможность для выбора породы собаки. При выделении знаком: 
Интересующей породы можно получить дополнительную информацию.

2. Создать внешний вид формы программы
3. Написать программные коды для созданной формы
4. Запустить и отладить программу

3. Разработайте двакласса потомка от Animal, которые будут отображать особенности двух новых животных Wolf (волк) и Jascal (шакал). Разработайте методы для этихк лассов, позволяющие получить некоторые характеристики этих видов животных (рост по холке, длина клыков, вес и т.д.).

Дополните форму компонентами, позволяющими увидеть все характеристики разработанных классов.

Порядок выполнения работы

- Составить математическую модель задачи.
- Выбрать и обосновать наиболее рациональный метод решения задачи;
- Разработать алгоритм для решения задачи.
- Создать внешний вид формы программы
- Написать программные коды для созданной формы
- Запустить и отладить программу

Форма предоставления результата

- Блок – схема.
- Код программы.

Критерии оценки:

«Отлично» - теоретическое содержание курса освоено полностью, без пробелов, умения сформированы, все предусмотренные программой учебные задания выполнены, качество их выполнения оценено высоко.

–«Хорошо» - теоретическое содержание курса освоено полностью, без пробелов, некоторые умения сформированы недостаточно, все предусмотренные программой учебные задания выполнены, некоторые виды заданий выполнены с ошибками.

–«Удовлетворительно» - теоретическое содержание курса освоено частично, но пробелы не носят существенного характера, необходимые умения работы с освоенным материалом в основном сформированы, большинство предусмотренных программой

обучения учебных заданий выполнено, некоторые из выполненных заданий содержат ошибки.

–«Неудовлетворительно» - теоретическое содержание курса не освоено, необходимые умения не сформированы, выполненные учебные задания содержат грубые ошибки.

Лабораторное занятие№ 20,21. Разработка приложения с не визуальными компонентами.
Лабораторное занятие № 22,23. Разработка игрового приложения.
Лабораторное занятие № 24. Разработка приложения с анимацией.

Цель работы:

- Научиться формировать алгоритмы разработки программных модулей в соответствии с техническим заданием;
- Выполнять отладку программных модулей с использованием специализированных программных средств;
- Выполнять тестирование программных модулей;
- Осуществлять рефакторинг и оптимизацию программного кода.

Выполнив работу, Вы будете:

уметь:

- У1. Осуществлять разработку кода программного модуля на языках низкого и высокого уровней;
- У7. Формировать алгоритмы разработки программных модулей в соответствии с техническим заданием.
- У6. Оформлять документацию на программные средства;
- У2. Создавать программу по разработанному алгоритму как отдельный модуль.

Материальное обеспечение:

- Мультимедийные средства хранения, передачи и представления информации.

Задание:

1. Игра «Шашки». Игра для двух игроков. Шашки передвигаются с помощью мыши. Шашки удаляются с поля двойным щелчком мыши.

2. Игра «Сапер». На поле размером 8х8 случайно размещается N мин. Задача игрока найти все мины не подорвавшись.

Порядок выполнения работы

- Составить математическую модель задачи.
- Выбрать и обосновать наиболее рациональный метод решения задачи;
- Разработать алгоритм для решения задачи.
- Создать внешний вид формы программы
- Написать программные коды для созданной формы
- Запустить и отладить программу

Форма предоставления результата

- Блок – схема.
- Код программы.

Критерии оценки:

«Отлично» - теоретическое содержание курса освоено полностью, без пробелов, умения сформированы, все предусмотренные программой учебные задания выполнены, качество их выполнения оценено высоко.

–«Хорошо» - теоретическое содержание курса освоено полностью, без пробелов, некоторые умения сформированы недостаточно, все предусмотренные программой учебные задания выполнены, некоторые виды заданий выполнены с ошибками.

–«Удовлетворительно» - теоретическое содержание курса освоено частично, но пробелы не носят существенного характера, необходимые умения работы с освоенным материалом в основном сформированы, большинство предусмотренных программой обучения учебных заданий выполнено, некоторые из выполненных заданий содержат ошибки.

–«Неудовлетворительно» - теоретическое содержание курса не освоено, необходимые умения не сформированы, выполненные учебные задания содержат грубые ошибки.

Тема 1.1.6 Оптимизация и рефакторинг кода

Лабораторное занятие№ 25,26,27,28,29,30,31,32. Оптимизация и рефакторинг кода.

Цель работы:

- Научиться формировать алгоритмы разработки программных модулей в соответствии с техническим заданием;
- Выполнять отладку программных модулей с использованием специализированных программных средств;
- Выполнять тестирование программных модулей;
- Осуществлять рефакторинг и оптимизацию программного кода.

Выполнив работу, Вы будете:

уметь:

- У1. Осуществлять разработку кода программного модуля на языках низкого и высокого уровней;
- У7. Формировать алгоритмы разработки программных модулей в соответствии с техническим заданием.
- У6. Оформлять документацию на программные средства;
- У2. Создавать программу по разработанному алгоритму как отдельный модуль.

Материальное обеспечение:

- Мультимедийные средства хранения, передачи и представления информации.

Задание:

Выполните следующие задания:

1. Создайте C# решение в среде Visual Studio. Назовите его в соответствии с вашим вариантом задания. В качестве исходного типа проекта выберите проект динамической библиотеки (.dll). Назовите проект также согласно вашему варианту или просто Model. Данный проект будет содержать в себе бизнес-логику вашей будущей программы, т.е. содержать ключевые сущности, структуры данных и способы их взаимодействия.
2. Создайте сущность-интерфейс согласно вашему варианту. Интерфейс в языке C# описывается с помощью ключевого слова interface. Опишите ключевые свойства и методы интерфейса. Не забудьте о правильном именовании элементов в соответствии с RSDN. Подумайте, какие свойства и методы в дальнейшем будут являться общими (т.е. будут в интерфейсе), а какие должны быть реализованы в конкретных классах.
3. Создайте два класса, реализующих данный интерфейс. Классы ОБЯЗАТЕЛЬНО должны иметь различные реализации методов интерфейса. При этом, дочерние классы

не должны иметь никаких ссылок друг на друга, также, как и интерфейс не должен ничего знать о классах, его реализующих.
4. Реализуйте валидацию свойств с помощью механизмов обработки исключений – если на вход свойству приходит некорректное значение, выходящее за допустимые пределы, свойство должно выбросить исключение соответствующего типа с описанием возникшей ошибки. Например, если в свойство «Возраст» пытаются присвоить отрицательное значение, необходимо выбросить экземпляр IncorrectArgumentException. Внимательно продумайте все возможные некорректные варианты данных, в том числе и ссылки на null. В случае, если механизмы валидации одинаковы у всех свойств, измените архитектуру бизнес-логики: вместо реализации интерфейса двумя классами, сделайте наследование двух классов от абстрактного класса с приобретением механизмов валидации.
5. Добавьте в решение еще один проект. Это можно сделать с помощью контекстного меню панели «Обозреватель решения». Для этого кликните правой кнопкой по названию решения в панели и выберите пункт «Добавить проект». Создайте исполняемый проект консоли (Win32 Console Application) и назовите его «ConsoleLoader». Данный проект является временным и необходим для начального тестирования бизнес-логики.
Примечание: в последующих лабораторных вы избавитесь от этого проекта и создадите вместо него проект графического интерфейса (WinForms Application). Однако если вы уже можете продемонстрировать работу бизнес-логики на оконном пользовательском интерфейсе – можете сразу создать необходимый проект.
6. Продемонстрируйте корректную работу бизнес-логики простыми заданиями в консоли: создайте переменную-ссылку на интерфейс, поочередно присвойте в неё экземпляры реализующих классов и вызовите реализации методов – покажите, что это разные реализации. Вероятно, для демонстрации ваших классов в проекте ConsoleLoader придется добавить методы ввода/вывода классов бизнес-логики через консоль.

Задача

1. Геометрические фигуры с собственными реализациями расчета площади фигуры: круг, прямоугольник, разносторонний треугольник.

Порядок выполнения работы

- Составить математическую модель задачи.
- Выбрать и обосновать наиболее рациональный метод решения задачи;
- Разработать алгоритм для решения задачи.
- Создать внешний вид формы программы
- Написать программные коды для созданной формы
- Запустить и отладить программу

Форма предоставления результата

- Блок – схема.
- Код программы.

Критерии оценки:

«Отлично» - теоретическое содержание курса освоено полностью, без пробелов, умения сформированы, все предусмотренные программой учебные задания выполнены, качество их выполнения оценено высоко.

–«Хорошо» - теоретическое содержание курса освоено полностью, без пробелов, некоторые умения сформированы недостаточно, все предусмотренные программой учебные задания выполнены, некоторые виды заданий выполнены с ошибками.

–«Удовлетворительно» - теоретическое содержание курса освоено частично, но пробелы не носят существенного характера, необходимые умения работы с освоенным материалом в основном сформированы, большинство предусмотренных программой обучения учебных заданий выполнено, некоторые из выполненных заданий содержат ошибки.

–«Неудовлетворительно» - теоретическое содержание курса не освоено, необходимые умения не сформированы, выполненные учебные задания содержат грубые ошибки.

Тема 1.1.7 Разработка пользовательского интерфейса.
Лабораторное занятие 33,34,35,36,37,38,39,40 Разработка интерфейса пользователя

Цель работы:

- Научиться формировать алгоритмы разработки программных модулей в соответствии с техническим заданием;
- Выполнять отладку программных модулей с использованием специализированных программных средств;
- Выполнять тестирование программных модулей;
- Осуществлять рефакторинг и оптимизацию программного кода.

Выполнив работу, Вы будете:

уметь:

- У1. Осуществлять разработку кода программного модуля на языках низкого и высокого уровней;
- У7. Формировать алгоритмы разработки программных модулей в соответствии с техническим заданием.
- У6. Оформлять документацию на программные средства;
- У2. Создавать программу по разработанному алгоритму как отдельный модуль.

Материальное обеспечение:

- Мультимедийные средства хранения, передачи и представления информации.

Краткие теоретические сведения:

- В ходе данной работы мы рассмотрим создание средствами СУБД Access пользовательского интерфейса. Данные можно вводить и прямо в таблицы, но для конечных пользователей удобнее осуществлять ввод в отдельных окнах, предоставляющих дополнительные элементы управления для навигации и работы с записями. В Access такие окна называются формами, они служат для организации пользователю удобного графического интерфейса для работы с данными. При работе с формами есть несколько режимов: • режим Формы;
- режим Конструктора;
- режим Макета.

В режиме Формы пользователь работает с данными; производятся добавление новых записей, просмотр, удаление или редактирование записей таблицы или запроса, являющегося источником данных для формы. В режиме Конструктора осуществляется модификация структуры формы
(изменение внешнего вида, добавление и удаление элементов управления и т.д.). Режим Макета позволяет увидеть данные и в то же время редактировать структуру формы.
Макет формы имеет следующие разделы:

- заголовок формы;
- область данных;
- примечание формы.

Область данных определяет основную часть формы. Этот раздел может содержать элементы управления, отображающие данные из источника данных (таблицы или запроса), а также неизменяемые данные (например, надписи и линии). При печати формы область данных будет повторяться для каждой записи таблицы, тогда как заголовок и примечание – только один раз.
Размещаемые на форме элементы управления могут быть разных типов.
Связанные элементы управления связаны с полями базовой таблицы, которая является источником данных для формы. Если источником данных является запрос, то

присоединенные элементы управления могут связываться с полями в разных таблицах. При изменении значения в элементе управления обновляется и значение поля соответствующей записи в таблице.

Свободные элементы управления не связаны с таблицами, в частности они служат для оформления (линии, надписи и т.д.).

Вычисляемые элементы управления – это элементы, значения которых рассчитываются на основе значений других элементов. В качестве источника данных для этих элементов используются выражения и функции.

Ниже перечислены некоторые элементы управления, которые могут быть размещены на форме (полный перечень можно увидеть, работая в режиме Конструктора).

Надпись (Label) содержит небольшой текст, как правило, пользователем не изменяемый. *Поле* (Text Box) служит для отображения, ввода и изменения данных. *Флажок* (Check box), *Переключатель* (Option button), *Выключатель* (Toggle button) – элементы, определяющие значения логического типа. В Access, в зависимости от настроек поля, это может быть "Истина/ Ложь", "Да/Нсг" или "Вкл./Выкл.".

Элементы *Список* (List Box) и *Поле со списком* (Combo Box) позволяют выбрать значения из определенного списка (это может быть просто заданный набор возможных значений или, например, данные одного из столбцов таблицы). Поле со списком также позволяет ввести в поле новое значение.

Кроме "простых" форм, Access позволяет создавать "сложные" формы, включающие данные, собранные из нескольких связанных таблиц. При этом *подчиненной формой* называется форма, которая встраивается в другую. Форма, содержащая подчиненную форму, называется *главной.*

В папке с файлами к лабораторной работе находятся две базы Access – уже знакомая но прошлому занятию база lib.accdb, в которую добавлены две формы, и база lib_l .accdb, отличающаяся от первой только отсутствием форм.

Начнем с создания формы с информацией об изданиях. Создадим ее с помощью мастера и потом изменим в редакторе. Перейдем на закладку Создание → Мастер форм. В первом окне "мастера" надо выбрать таблицу (или запрос), данные из которой будут отображаться в форме (таблица

Book), и поля из этой таблицы. Надо отметить, что идентификатор книги (типа Счетчик) показывать пользователю на форме не надо (рис. П.2.1): изменяется идентификатор автоматически и задать его значение вручную пользователь не сможет, что может привести к путанице.

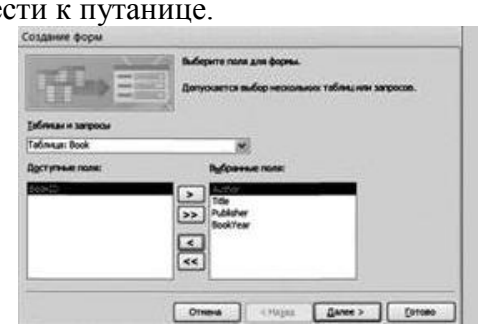


Рис. П.2.1. Выбор полей таблицы

В представленном примере поля располагаются в один столбец, форма названа "Информация об изданиях". В последнем окне мастера выберите опцию "Изменить макет формы", после чего откроется конструктор, в котором нужно изменить размер полей, текст надписей, добавить элементы оформления. В примере в примечание формы добавлена надпись "F1 – справка", а заголовок – дата и время.

Задание:

В базе данных lib_1.acc.db в соответствии с приведенным выше описанием и примером создайте новую форму. Не забудьте в заголовке формы поставить дату и время, а в примечании – надпись про справку. Введите через форму какие-нибудь данные.

Посмотрите, как будет выглядеть форма при печати (Файл → Печать → Предварительный просмотр).
Самостоятельно создайте форму для редактирования информации о статусах книг.
Теперь рассмотрим создание составных форм. Такие формы очень 5'добны при работе со связанными таблицами – в одной форме можно представить данные из нескольких таблиц (рис. П.2.2).
Проще всего создать подобную форму с помощью мастера. В первом окне мастера укажите, что на форме надо представить информацию об авторе, названии, издательстве, годе издания, взятую из таблицы *Book,* и информацию об идентификаторе и статусе книги из таблицы *Book_in_lib.* Мастер предложит создать подчиненные формы. Дальнейшая работа мастера не отличается от предыдущего случая. Обратите внимание, что мастер создал две формы – основную и подчиненную.
После окончания работы мастера можно отредактировать форму в режиме конструктора, например поменять подписи с подставляемых по умолчанию названий столбцов на какие-то более подробные пояснения.

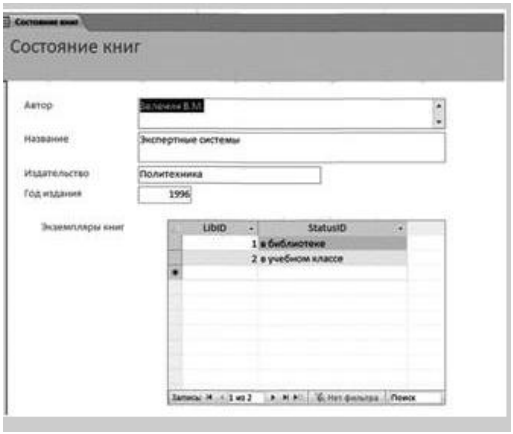


Рис. П.2.2. Составная форма

Задание.
Создайте составную форму для отображения информации об издании и текущем состоянии экземпляров книг в библиотеке.
Нередко для удобства работы пользователя нужно разместить на форме какие-нибудь дополнительные элементы, например кнопки. Внизу каждой формы расположены кнопки, позволяющие передвигаться по записям (следующая, предыдущая, первая, последняя, новая запись). Однако для того чтобы удалить запись, приходится использовать соответствующую кнопку с основной напели инструментов, что не очень удобно.
Добавим кнопку удаления на форму "Информация об изданиях". Для этого в режиме редактирования надо воспользоваться панелью элементов. На панели элементов надо выбрать элемент "кнопка" и мышкой нарисовать его на форме. После этого появится диалоговое окно Создание кнопок, в котором надо указать, что создаваемая кнопка относится к категории "Обработка записей" и требуемое действие – "Удалить запись". Далее для созданной кнопки можно выбрать подходящую пиктограмму или надпись, а также ее имя.

Порядок выполнения работы

- Составить математическую модель задачи.
- Выбрать и обосновать наиболее рациональный метод решения задачи;
- Разработать алгоритм для решения задачи.
- Создать внешний вид формы программы
- Написать программные коды для созданной формы
- Запустить и отладить программу
- .

Форма предоставления результата

- Блок – схема.
- Код программы.

Критерии оценки:

«Отлично» - теоретическое содержание курса освоено полностью, без пробелов, умения сформированы, все предусмотренные программой учебные задания выполнены, качество их выполнения оценено высоко.

–«Хорошо» - теоретическое содержание курса освоено полностью, без пробелов, некоторые умения сформированы недостаточно, все предусмотренные программой учебные задания выполнены, некоторые виды заданий выполнены с ошибками.

–«Удовлетворительно» - теоретическое содержание курса освоено частично, но пробелы не носят существенного характера, необходимые умения работы с освоенным материалом в основном сформированы, большинство предусмотренных программой обучения учебных заданий выполнено, некоторые из выполненных заданий содержат ошибки.

–«Неудовлетворительно» - теоретическое содержание курса не освоено, необходимые умения не сформированы, выполненные учебные задания содержат грубые ошибки.

Тема 1.1.8 Программирование в среде 1С Предприятие
Лабораторное занятие №41,42,43,44,45 Создание приложения с БД

Цель работы:

- Научиться формировать алгоритмы разработки программных модулей в соответствии с техническим заданием;
- Выполнять отладку программных модулей с использованием специализированных программных средств;
- Выполнять тестирование программных модулей;
- Осуществлять рефакторинг и оптимизацию программного кода.

Выполнив работу, Вы будете:

уметь:

- У1. Осуществлять разработку кода программного модуля на языках низкого и высокого уровней;
- У7. Формировать алгоритмы разработки программных модулей в соответствии с техническим заданием.
- У6. Оформлять документацию на программные средства;
- У2. Создавать программу по разработанному алгоритму как отдельный модуль.

Материальное обеспечение:

- Мультимедийные средства хранения, передачи и представления информации.

Задание:

1. Сформировать новую конфигурацию.
2. Создать в конфигурации подсистемы: «Лабораторное занятие№», «Администрирование». Расположить их на панели разделов в указанной последовательности.
3. Создать две роли: Администратор, наделенный всеми правами, в т. числе и административными, и пользователь без административных прав.
4. Создать двух пользователей, один из которых должен играть роль администратора, а второй – пользователя.

5. Создать константу, указанную в задании, включив ее в подсистему «Администрирование». Дополнительно создать форму констант, включив ее в подсистему «Администрирование». В пользовательском режиме задать значение данной константы. В дальнейшем при запуске приложения выводить значение данной константы в окно сообщений. Если для разработки типа константы необходимо разработать дополнительные объекты, включить их в подсистему «Лабораторное занятие№».

6. Разработать справочники, указанные в задании. Включить их в подсистему «Лабораторное занятие№». Основной (первый в каждом задании) справочник должен быть многоуровневым. Количество уровней - произвольно. Название групп в справочнике - произвольно. Решение задачи состоит из следующих этапов:

- Описать структуры справочников средствами конфигуратора;
- Разработать формы диалога ввода данных в справочники (как для ввода групп, так и для ввода элементов). Группы справочников должны состоять только из кодов и наименований;
- В режиме 1С:Предприятие ввести несколько групп, содержащий не менее 2-3 элементов.
- Дополнительно в случае необходимости разработать объекты метаданных, необходимые для работы заданного справочника в Конфигурации.
- Обеспечить проверку заполнения всех реквизитов шапки основного справочника. Разрешать запись элемента справочника только в случае заполненности всех реквизитов шапки.
- Поместить команду создания нового элемента основного справочника в задании на рабочий стол приложения.
- В форму списка основного справочника вставить кнопку с названием «Справка». При нажатии на эту кнопку программа должна вывести в окно сообщений справку. Содержание справки – см. варианты заданий. Использовать команду Сообщить(…)
- Для разработанного основного справочника сформировать подчиненный справочник. Структура справочника – см. варианты.
- Проверку заполненности трех обязательных (произвольных) реквизитов основного справочника перед записью элемента справочника выполнить программным путем, поместив процедуру проверки в модуль менеджера.

7. Создать обработку, которая должна выводить список всех справочников, разработанных в конфигурации. Для вывода информации использовать команду Сообщить(…)

Задача

1. Константа «Главный бухгалтер», тип СправочникСсылка.Сотрудники

2 Справочник сотрудников:

- Табельный номер сотрудника;
- ФИО сотрудника;
- Тип сотрудника (выбирается из списка – перечисления со значениями: штатный, внутренний совместитель, внешний совместитель);
- Дата поступления на работу;
- Оклад;
- Признак членства в профсоюзе.

Табличная часть элементов справочника содержит список детей сотрудника и имеет следующую структуру:

- Пол (выбирается из списка, заданного перечислением);
- ФИО ребенка.
- Дата рождения

3 Справочник стандартных вычетов сотрудника (подчиненный):

- Тип вычета (выбирается из списка, задаваемого перечислением со значениями: Собственный, на ребенка, Герой России);
- Сумма вычета.
- Дата начала действия
- Дата окончания действия

Замечание. При вводе нового элемента справочника обеспечить автоматическое заполнение реквизита "Тип сотрудника" предопределенным значением - "ИТР" (по умолчанию).
Замечание 2. Сформировать предопределенную группу в справочнике «Уволенные»

Получить справку: вывести наименования всех сотрудников – не членов профсоюза
Задача
1. Константа «Главный контрагент», тип СправочникСсылка.Контрагенты
2. Справочник контрагентов:

- Код контрагента;
- Наименование контрагента;
- Адрес контрагента;
- Основной расчетный счет(элемент справочника расчетных счетов контрагента);
- Банк контрагента (элемент справочника банков)
- Телефон ;
- Размер кредита.

Табличная часть элементов справочника содержит список договоров контрагента и имеет следующую структуру:

- Код договора;
- Наименование договора.
- Дата заключения договора
- условия договора (текст)

3. Справочник расчетных счетов контрагента(подчиненный)

- код
- наименование
- банк (элемент справочника банков)

Замечание. При вводе нового элемента справочника обеспечить автоматическое заполнение реквизита "Размер кредита" предопределенным значением - "5 000 руб" (по умолчанию).
Замечание 2. Сформировать предопределенную группу в справочнике «Зарубежные контрагенты»
Получить справку: вывести наименования всех контрагентов с нулевым значением кредита.
Порядок выполнения работы

- Составить математическую модель задачи.
- Выбрать и обосновать наиболее рациональный метод решения задачи;
- Разработать алгоритм для решения задачи.
- Написать и отладить программу.

Форма предоставления результата

- Блок – схема.
- Код программы.

Критерии оценки:

«Отлично» - теоретическое содержание курса освоено полностью, без пробелов, умения сформированы, все предусмотренные программой учебные задания выполнены, качество их выполнения оценено высоко.

–«Хорошо» - теоретическое содержание курса освоено полностью, без пробелов, некоторые умения сформированы недостаточно, все предусмотренные программой учебные задания выполнены, некоторые виды заданий выполнены с ошибками.

–«Удовлетворительно» - теоретическое содержание курса освоено частично, но пробелы не носят существенного характера, необходимые умения работы с освоенным материалом в основном сформированы, большинство предусмотренных программой обучения учебных заданий выполнено, некоторые из выполненных заданий содержат ошибки.

–«Неудовлетворительно» - теоретическое содержание курса не освоено, необходимые умения не сформированы, выполненные учебные задания содержат грубые ошибки.

Лабораторное занятие №46,47,48,49,50 Создание запросов к БД

Цель работы:

- Научиться формировать алгоритмы разработки программных модулей в соответствии с техническим заданием;
- Выполнять отладку программных модулей с использованием специализированных программных средств;
- Выполнять тестирование программных модулей;
- Осуществлять рефакторинг и оптимизацию программного кода.

Выполнив работу, Вы будете:

уметь:

- У1. Осуществлять разработку кода программного модуля на языках низкого и высокого уровней;
- У7. Формировать алгоритмы разработки программных модулей в соответствии с техническим заданием.
- У6. Оформлять документацию на программные средства;
- У2. Создавать программу по разработанному алгоритму как отдельный модуль.

Материальное обеспечение:

- Мультимедийные средства хранения, передачи и представления информации.

Задание:

Разработать отчет с нестандартной расшифровкой вывода списка элементов основного справочника (см. варианты заданий в главе 4). Для этого необходимо:

· Создать новую подсистему «Лабораторное занятие3». Все новые объекты данного задания поместить в данную подсистему.

· Создать в конфигурации новый объект-отчет.

· Создать форму отчета. Для формирования отчета использовать конструктор запроса с обработкой результата.

· Внести корректировку в результаты работы конструктора, обеспечивающие вывод отчета и его нестандартную расшифровку

Порядок выполнения работы

- Составить математическую модель задачи.
- Выбрать и обосновать наиболее рациональный метод решения задачи;
- Разработать алгоритм для решения задачи.
- Написать и отладить программу.

Форма предоставления результата

- Блок – схема.
- Код программы.

Критерии оценки:

«Отлично» - теоретическое содержание курса освоено полностью, без пробелов, умения сформированы, все предусмотренные программой учебные задания выполнены, качество их выполнения оценено высоко.

–«Хорошо» - теоретическое содержание курса освоено полностью, без пробелов, некоторые умения сформированы недостаточно, все предусмотренные программой учебные задания выполнены, некоторые виды заданий выполнены с ошибками.

–«Удовлетворительно» - теоретическое содержание курса освоено частично, но пробелы не носят существенного характера, необходимые умения работы с освоенным материалом в основном сформированы, большинство предусмотренных программой обучения учебных заданий выполнено, некоторые из выполненных заданий содержат ошибки.

–«Неудовлетворительно» - теоретическое содержание курса не освоено, необходимые умения не сформированы, выполненные учебные задания содержат грубые ошибки.

Лабораторное занятие № 51,52,53,54 Создание хранимых процедур.

Цель работы:

- Научиться формировать алгоритмы разработки программных модулей в соответствии с техническим заданием;
- Выполнять отладку программных модулей с использованием специализированных программных средств;
- Выполнять тестирование программных модулей;
- Осуществлять рефакторинг и оптимизацию программного кода.

Выполнив работу, Вы будете:

уметь:

- У1. Осуществлять разработку кода программного модуля на языках низкого и высокого уровней;
- У7. Формировать алгоритмы разработки программных модулей в соответствии с техническим заданием.
- У6. Оформлять документацию на программные средства;
- У2. Создавать программу по разработанному алгоритму как отдельный модуль.

Материальное обеспечение:

- Мультимедийные средства хранения, передачи и представления информации.

Задание:

Выполнение работы состоит из следующих этапов:

- Создать новую подсистему «Лабораторное занятие№». Все новые объекты, созданные в данной работе, поместить в данную подсистему.
- Разработать, если это необходимо, дополнительные документы
- Разработать тип и структуру регистров накопления для хранения движений, формируемых при проведении документов.
- Разработать модули проведения документов.
- Разработать отчет. Средства обращения к источнику данных – запрос. Использовать конструктор запроса с обработкой результатов. Отчет должен выводить общие итоги.
- Отчет должен иметь расшифровку. Тип расшифровки – см. варианты заданий.

– В случае разработки нестандартного отчета вид отчета разработать самостоятельно.

Задача

Ведомость поступления товара _____ на склады от поставщиков
за ___ период с _____ по ___
Склад Документ Дата Поставщик Колич. Единицы Цена Стоимость

Замечание 1. Строки отчета группируются по поставщикам. В конце группы следует включать строку «Итого по поставщику».
Замечание 2. Разработать документ «Поступление товаров от поставщиков» Расшифровка – стандартная.

Задача

Ведомость поступления товара _____ на склады от поставщика_____
за ___ период с _____ по ___
Склад Документ Дата Кол. Цена Стоимость Сумма НДС Всего с НДС

Замечание 1. Строки отчета группируются по складам. В конце группы следует включать строку «Итого по складу».
Замечание 2. Разработать документ «Поступление товаров от поставщиков» Расшифровка – стандартная.

Порядок выполнения работы

- Составить математическую модель задачи.
- Выбрать и обосновать наиболее рациональный метод решения задачи;
- Разработать алгоритм для решения задачи.
- Написать и отладить программу.

Форма предоставления результата

- Блок – схема.
- Код программы.

Критерии оценки:

«Отлично» - теоретическое содержание курса освоено полностью, без пробелов, умения сформированы, все предусмотренные программой учебные задания выполнены, качество их выполнения оценено высоко.

–«Хорошо» - теоретическое содержание курса освоено полностью, без пробелов, некоторые умения сформированы недостаточно, все предусмотренные программой учебные задания выполнены, некоторые виды заданий выполнены с ошибками.

–«Удовлетворительно» - теоретическое содержание курса освоено частично, но пробелы не носят существенного характера, необходимые умения работы с освоенным материалом в основном сформированы, большинство предусмотренных программой обучения учебных заданий выполнено, некоторые из выполненных заданий содержат ошибки.

–«Неудовлетворительно» - теоретическое содержание курса не освоено, необходимые умения не сформированы, выполненные учебные задания содержат грубые ошибки.

МДК.01.02 Поддержка и тестирование программных модулей

Тема 1.2.1 Отладка и тестирование программного обеспечения
Лабораторное занятие № 1,2 Тестирование «белым ящиком».

Цель работы:

- Научиться формировать алгоритмы разработки программных модулей в соответствии с техническим заданием;
- Выполнять отладку программных модулей с использованием специализированных программных средств;
- Выполнять тестирование программных модулей;
- Осуществлять рефакторинг и оптимизацию программного кода.

Выполнив работу, Вы будете:

уметь:

- У3. Выполнять отладку и тестирование программы на уровне модуля;
- У5. Уметь выполнять оптимизацию и рефакторинг программного кода;
- У6. Оформлять документацию на программные средства.
- У8. Применять инструментальные средства отладки программного обеспечения.
- У9. Работать с системой контроля версий.

Материальное обеспечение:

- Мультимедийные средства хранения, передачи и представления информации.

Краткие теоретические сведения:

Изучить методы тестирования логики программы, формализованные описания результатов тестирования и стандарты по составлению схем программ.

Стратегия «белого ящика»

Существуют следующие методы тестирования по принципу «белого ящика»:

- покрытие операторов;
- покрытие решений;
- покрытие условий;
- покрытие решений/условий;
- комбинаторное покрытие условий.

Метод покрытия операторов

Целью этого метода тестирования является выполнение каждого оператора программы хотя бы один раз.

Задание:

Если для тестирования задать значения переменных $A = 2$, $B = 0$, $X = 3$, будет реализован путь *ace,* т. е. каждый оператор программы выполнится один раз (рис. Л5.1, *а).* Но если внести в алгоритм ошибки — заменить в первом условии *and* на *or,* а во втором $X > 1$ на $X < 1$ (рис. Л5.1, *б),* ни одна ошибка не будет обнаружена (табл. Л5.1). Кроме того, путь *abd* вообще не будет охвачен тестом, и если в нем есть ошибка, она также не будет обнаружена. В табл. Л5.1 ожидаемый результат определяется по блок-схеме на рис. Л5.1, *а,* а фактический — по рис. Л5.1, *б.*

Как видно из этой таблицы, ни одна из внесенных в алгоритм ошибок не будет обнаружена.

Таблица Л5.1. Результат тестирования методом покрытия операторов

Тест	Ожидаемый результат	Фактический результат	Результат тестирования
A = 2, *B*= 0, *X*=Ь	*X*= 2,5	*X*= 2,5	Неуспешно

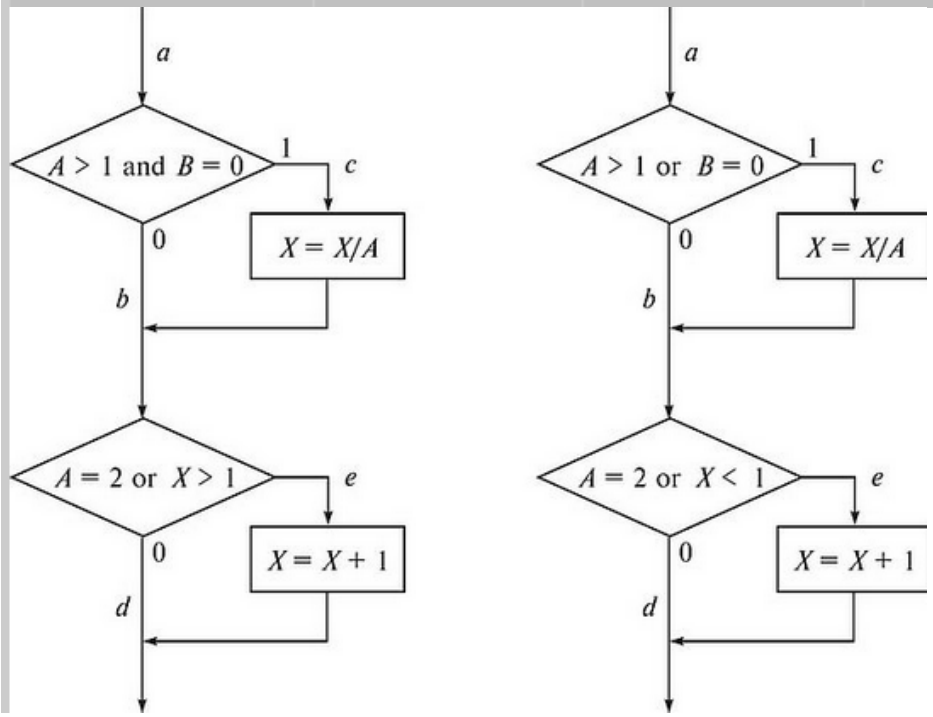


Рис. Л5.1. Пример алгоритма программы: *а* — правильный; *б* — с ошибкой

Метод покрытия решений (покрытия переходов)

Согласно методу покрытия решений каждое направление перехода должно быть реализовано, по крайней мере, один раз. Этот метод включает в себя критерий покрытия операторов, так как при выполнении всех направлений переходов выполнятся все операторы, находящиеся на этих направлениях.

Для программы, приведенной на рис. Л5.1, покрытие решений может быть выполнено двумя тестами, покрывающими пути *{ace, aЬc*!}, либо *{acc1, aЬe*}. Для этого выберем следующие исходные данные; *{A* = 3, *B*= О, *X=3}* — в первом случае и *{A = 2, B*= 1, *X*= 1} — во втором. Однако путь, где *X* не меняется, будет проверен с вероятностью 50 %: если во втором условии вместо условия *X*> 1 записано *X*<, то ошибка не будет обнаружена двумя тестами.

Результаты тестирования приведены в табл. Л5.2.

Таблица Л5.2. Результат тестирования методом покрытия решений

Тест	Ожидаемый результат	Фактический результат	Результат тестирования
/1 = 3, B= 0, X=2	X = 1	X = 1	Неуспешно
A = 2, B = 1, X= 1	X=2	* II	Успешно

Метод покрытия условий

Этот метод может дать лучшие результаты по сравнению с предыдущими. В соответствии с методом покрытия условий записывается число тестов, достаточное для того, чтобы все возможные результаты каждого условия в решении выполнялись, по крайней мере, один раз.

В рассматриваемом примере имеем четыре условия: *{A*> 1, 5=0}, *{A* = 2, *X*> 1}. Следовательно, требуется достаточное число тестов, такое, чтобы реализовать ситуации, где *A* > 1, *A* < 1, 5 = 0 и й#0 в точке а и /1 = 2, *A* *2, *X*> и Т< 1 в точке *Ь.* Тесты, удовлетворяющие критерию покрытия условий (табл. Л5.3), и соответствующие им пути:

а) *A = 2,* 5 = 0, *X=4 ace;*

б) *A* = 1, 5= 1, *X=0 aЬc!.*

Таблица Л5.3. Результаты тестирования методом покрытия условий

Тест	Ожидаемый результат	Фактический результат	Результат тестирования
/1 = 2, B= 0, X= 4	*=з	X= 3	Неуспешно
A= 1, B= 1, X=0	*=0	X= 1	Успешно

Метод покрытия решений/условий

Критерий покрытия решений/условий требует такого достаточного набора тестов, чтобы все возможные результаты каждого условия выполнялись по крайней мере один раз, все результаты каждого решения выполнялись по крайней мере один раз и, кроме того, каждой точке входа передавалось управление по крайней мере один раз.

Недостатки метода:

- • не всегда можно проверить все условия;
- • невозможно проверить условия, которые скрыты другими условиями;
- • недостаточная чувствительность к ошибкам в логических выражениях.

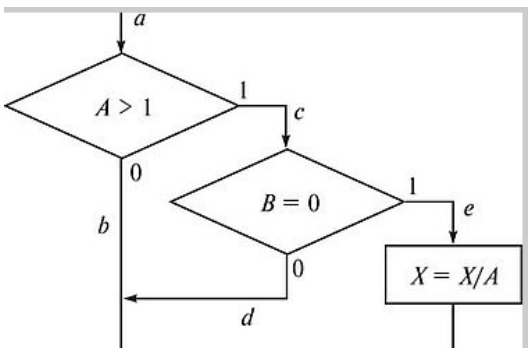
Так, в рассматриваемом примере два теста метода покрытия условий

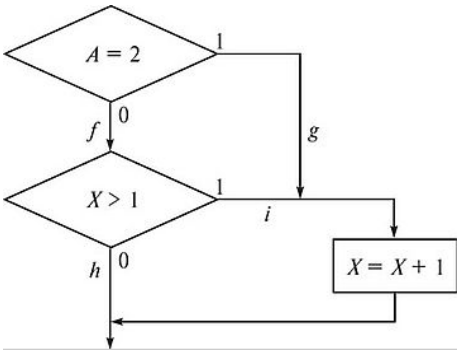
а) *A* = 2, *B* = О, *X= 4 ace;*

б*) A*= 1, *B*= 1, *X=0 aЬc!*

отвечают и критерию покрытия решений/условий. Это является следствием того, что одни условия приведенных решении скрывают другие условия в этих решениях. Так, если условие *A* > 1 будет ложным, транслятор может не проверять условия *B*= О, поскольку при любом результате условия *B*= 0 результат решения *((A >)&(B=0))* примет значение *ложь.* То есть в варианте на рис. Л5.1 не все результаты всех условий выполнятся в процессе тестирования.

Рассмотрим реализацию того же примера на рис. Л5.2. Наиболее полное покрытие тестами в этом случае осуществляется так, чтобы выполнялись все возможные результаты каждого простого решения.





Для этого нужно покрыть пути *aceg* (тест *A = 2, B*= О, *X*= 4), *асё/к* (тест *A* = 3, *B* = 1, *X*= 0), *аЬ/к* (тест *A* = 0, *B*= 0, *X*= 0), *аЬр* (тест Л = 0, *B* = 0, *X*= *2).*

Протестировав алгоритм на рис. Л5.2, нетрудно убедиться в том, что критерии покрытия условий и критерии покрытия решений/условий недостаточно чувствительны к ошибкам в логических выражениях.

Метод комбинаторного покрытия условий

Критерий комбинаторного покрытия условий удовлетворяет также и критериям покрытия решений, покрытия условий и покрытия решений/условий.

Этот метод требует создания такого числа тестов, чтобы все возможные комбинации результатов условия в каждом решении выполнялись по крайней мере один раз. По этому критерию в рассматриваемом примере должны быть покрыты тестами следующие восемь комбинаций:

1. *A*> 1, *B*= 0. *5. A = 2, X*> 1.
2. *A* > 1, *Bф* 0.
3. *A*< 1, *B*= 0.
4. *A*< 1, Вф 0.
6. *A* = 2, 1.
7. Аф2, *X*> 1.
8. Аф2, *X*< 1.

Для того чтобы протестировать эти комбинации, необязательно использовать все 8 тестов. Фактически они могут быть покрыты четырьмя тестами (табл. Л5.4):

- *A* = 2, *B = 0, X=4* (покрывает 1,5);
- *A = 2, B = 1, X= 1* (покрывает 2,6);
- *A* = 0,5, *B*= 0, *X= 2* (покрывает 3, 7);
- *A = I, B = 0, X*= (покрывает 4, 8_.

Таблица Л5.4. Результаты тестирования методом комбинаторного покрытия условий

Тест	Ожидаемый результат	Фактический результат	Результат тестирования
A-2, B=0, X- 4	*=з	*=з	Неуспешно
A = 2, B= 1, *X=* 1	*X= 2*	1 *Г)* СЧ II	Успешно
A = 0,5, *B=* 0, *X=2*	*X=3*	*X= 4*	Успешно
A= 1, ?=0, *= 1	*X=* 1	*X=* 1	Неуспешно

Порядок выполнения работы

1. Спроектировать тесты по принципу «белого ящика» для программы, разработанной в лабораторной работе № 4. Использовать схемы алгоритмов, разработанные и уточненные в лабораторных работах № 2, 3.
2. Выбрать несколько алгоритмов для тестирования и обозначить буквами или цифрами ветви этих алгоритмов.
3. Выписать пути алгоритма, которые должны быть проверены тестами для выбранного метода тестирования.
4. Записать тесты, которые позволят пройти по путям алгоритма.
5. Протестировать разработанную вами программу. Результаты оформить в виде таблиц (см. табл. Л5.1— Л5.4).
6. Проверить все виды тестов и сделать выводы об их эффективности.

Форма предоставления результата

- Блок – схема.
- Код программы.

Критерии оценки:

«Отлично» - теоретическое содержание курса освоено полностью, без пробелов, умения сформированы, все предусмотренные программой учебные задания выполнены, качество их выполнения оценено высоко.

–«Хорошо» - теоретическое содержание курса освоено полностью, без пробелов, некоторые умения сформированы недостаточно, все предусмотренные программой учебные задания выполнены, некоторые виды заданий выполнены с ошибками.

–«Удовлетворительно» - теоретическое содержание курса освоено частично, но пробелы не носят существенного характера, необходимые умения работы с освоенным материалом в основном сформированы, большинство предусмотренных программой обучения учебных заданий выполнено, некоторые из выполненных заданий содержат ошибки.

–«Неудовлетворительно» - теоретическое содержание курса не освоено, необходимые умения не сформированы, выполненные учебные задания содержат грубые ошибки.

Лабораторное занятие № 3,4 Тестирование «черным ящиком»

Цель работы:

- Научиться формировать алгоритмы разработки программных модулей в соответствии с техническим заданием;
- Выполнять отладку программных модулей с использованием специализированных программных средств;
- Выполнять тестирование программных модулей;
- Осуществлять рефакторинг и оптимизацию программного кода.

Выполнив работу, Вы будете:

уметь:

- У3. Выполнять отладку и тестирование программы на уровне модуля;
- У5. Уметь выполнять оптимизацию и рефакторинг программного кода;
- У6. Оформлять документацию на программные средства.
- У8. Применять инструментальные средства отладки программного обеспечения.
- У9. Работать с системой контроля версий.

Материальное обеспечение:

- Мультимедийные средства хранения, передачи и представления информации.

Краткие теоретические сведения:

Тестирование «черного ящика» — это метод тестирования функционального поведения объекта (программной системы) с точки зрения внешнего мира.

Основная задача тестировщика для данного метода тестирования состоит в последовательной проверке соответствия поведения системы требованиям. Кроме того, тестировщик должен проверить работу системы в критических ситуациях — в случае подачи неверных входных значений. В идеальной ситуации все варианты критических ситуаций должны быть описаны в требованиях на систему, и тестировщику остается только придумывать конкретные проверки этих требований. Однако на самом деле в результате тестирования обычно выявляются следующие проблемы системы:

• поведение системы не соответствует требованиям;

• в ситуациях, не предусмотренных требованиями, система ведет себя неадекватно (при этом под неадекватным поведением может пониматься как полный крах системы, так и вообще любое поведение, не описанное в требованиях).

В первом случае обычно требуются изменения программного кода, гораздо реже — изменения требований. Изменение требований может потребоваться из-за их противоречивости (несколько разных требований описывают разные модели поведения системы в одной и той же ситуации) или некорректности (требования не соответствуют действительности).

Во втором случае однозначно требуются изменения требований ввиду их неполноты — в требованиях явно не учтена ситуация, приводящая к неадекватному поведению системы.

Методы «черного ящика» обеспечивают:

• эквивалентное разбиение;
• анализ граничных значений;
• анализ причинно-следственных связей;
• предположение об ошибке.

Эквивалентное разбиение состоит в разбиении входной области данных программы на конечное число классов эквивалентности так, чтобы каждый тест, являющийся

представителем некоторого класса, был эквивалентен любому другому тесту этого класса. Классы эквивалентности выделяются путем перебора входных условий и разбиения их на две (или более) группы. При этом различают два типа классов эквивалентности: правильные, задающие входные данные для программы, и неправильные, основанные на задании ошибочных входных значений. Разработка тестов методом эквивалентного разбиения осуществляется в два этапа: выделение классов эквивалентности и построение тестов. При построении тестов, основанных на выборе входных данных, проводится символическое выполнение программы. Одна из целей, которая стоит перед разработчиком тестов — сокращение количества тестов, чтобы уложиться в адекватные сроки тестирования, но при этом не пропустить серьезных ошибок. Именно для этих целей используется техника разбиения на классы эквивалентности, которая и заключается в разбиении всего набора тестов на классы эквивалентности с последующим сокращением числа тестов.

Анализ граничных значений — это проверка ошибок на границах классов эквивалентности. Если техника анализа классов эквивалентности ориентирована на тестовое покрытие, то эта техника основана на рисках. Она основывается на идее о том, что программа может «дать сбой» в области граничных значений, поскольку при разработке большое число проблем возникает именно на границах входных переменных. Даже если эквивалентные классы найдены правильно, то граничные значения могут быть ошибочно отнесены к другому классу. Эффективное применение этой техники зависит от способности правильно выделить классы эквивалентности и затем выбрать тесты для проверки границ этих классов.

Анализ причинно-следственных связей происходит следующим образом. Спецификация разбивается на рабочие участки. В спецификации определяется множество причин и следствий (под причиной понимается отдельное входное условие или класс эквивалентности, следствие представляет собой выходное условие или преобразование системы). На основе анализа семантического содержания спецификации строится таблица истинности, в которой последовательно перебираются всевозможные комбинации причин и определяются следствия для каждой комбинации причин. Недостатком этого подхода является плохое исследование граничных условий.

Предположение об ошибке в значительной степени основано на интуиции. Основная идея метода состоит в том, чтобы составить список, который перечисляет возможные ошибки и ситуации, в которых эти ошибки могли проявиться. После этого на основе списка составляются тесты.

Во время подготовки к тестированию методом «черного ящика» строятся таблицы условий, причинно-следственные графы и области разбивки. Кроме того, подготавливаются тестовые наборы, учитывающие параметры и условия среды, которые влияют на поведение функций.

Задание:

Пусть необходимо выполнить тестирование программы, определяющей точку пересечения двух прямых на плоскости. Попутно, она должна определять параллельность прямой одной их осей координат.

В основе программы лежит решение системы линейных уравнений:

$Ax + By = C$ *и* $Dx + Ey = F$.

1**.** Используя **метод эквивалентных разбиений,** получаем для всех коэффициентов один правильный класс эквивалентности (коэффициент - вещественное число) и один неправильный (коэффициент - не вещественное число). Откуда можно предложить 7 тестов:

- поочередно каждый из коэффициентов - не вещественное число.

2. По **методу граничных условий**:

можно считать, что для исходных данных граничные условия отсутствуют (коэффициенты - ";любые"; вещественные числа);

для результатов - получаем, что возможны варианты: единственное решение, прямые сливаются (множество решений), прямые параллельны (отсутствие решений). Следовательно, можно предложить тесты, с результатами внутри области:

Порядок выполнения работы

- Составить математическую модель задачи.
- Выбрать и обосновать наиболее рациональный метод решения задачи;
- Разработать алгоритм для решения задачи.
- Написать и отладить программу.

Форма предоставления результата

- Блок – схема.
- Код программы.

Критерии оценки:

«Отлично» - теоретическое содержание курса освоено полностью, без пробелов, умения сформированы, все предусмотренные программой учебные задания выполнены, качество их выполнения оценено высоко.

–«Хорошо» - теоретическое содержание курса освоено полностью, без пробелов, некоторые умения сформированы недостаточно, все предусмотренные программой учебные задания выполнены, некоторые виды заданий выполнены с ошибками.

–«Удовлетворительно» - теоретическое содержание курса освоено частично, но пробелы не носят существенного характера, необходимые умения работы с освоенным материалом в основном сформированы, большинство предусмотренных программой обучения учебных заданий выполнено, некоторые из выполненных заданий содержат ошибки.

–«Неудовлетворительно» - теоретическое содержание курса не освоено, необходимые умения не сформированы, выполненные учебные задания содержат грубые ошибки.

Лабораторное занятие № 5,6,7,8,9,10, 11,12, 13, 14. Модульное тестирование

Цель работы:

- Научиться формировать алгоритмы разработки программных модулей в соответствии с техническим заданием;
- Выполнять отладку программных модулей с использованием специализированных программных средств;
- Выполнять тестирование программных модулей;
- Осуществлять рефакторинг и оптимизацию программного кода.

Выполнив работу, Вы будете:

уметь:

- У3. Выполнять отладку и тестирование программы на уровне модуля;
- У5. Уметь выполнять оптимизацию и рефакторинг программного кода;
- У6. Оформлять документацию на программные средства.
- У8. Применять инструментальные средства отладки программного обеспечения.
- У9. Работать с системой контроля версий.

Материальное обеспечение:

- Мультимедийные средства хранения, передачи и представления информации.

Краткие теоретические сведения:

Задание:

- Изучить основы разработки модульных тестов
- Получить навыки работы со средствами тестирования Unit Testing Frameworkот Microsoft, NUnit.
- Изучить средства тестирования, доступные в VisualStudio–Unit TestingFramework, NUnit. Допускается использование альтернативных средств (необходимо согласовать с преподавателем).
- Разработать набор unit-тестов для алгоритма в соответствии с номером варианта.
- Реализовать алгоритм в соответствии с номером варианта на любом языке, поддерживаемом платформой .NET. Использование других языков необходимо заранее согласовать с преподавателем.
- Обеспечить максимально возможное покрытие кода тестами.
- Продемонстрировать: Работающую функциональность в соответствии с заданием; Проходящие unit-тесты.

При реализации алгоритма необходимо учитывать следующие требования:

- Объем набора данных –не менее 500 миллионов элементов.
- Потребляемые алгоритмом источники данных задаются интерфейсом IEnumerable<T>.
- Результат работы алгоритма представлен интерфейсом IEnumerable<T>.
- Параметризуемые условия для алгоритма задаются делегатами Func<T, bool>.
- При защите необходимо продемонстрировать понимание реализованного алгоритма

В качестве справочных материалов предлагается набор статей: http://msdn.microsoft.com/ru-ru/library/dd264975.aspx

Порядок выполнения работы

- Составить математическую модель задачи.
- Выбрать и обосновать наиболее рациональный метод решения задачи;
- Разработать алгоритм для решения задачи.
- Написать и отладить программу.

Форма предоставления результата

- Блок – схема.
- Код программы.

Критерии оценки:

«Отлично» - теоретическое содержание курса освоено полностью, без пробелов, умения сформированы, все предусмотренные программой учебные задания выполнены, качество их выполнения оценено высоко.

–«Хорошо» - теоретическое содержание курса освоено полностью, без пробелов, некоторые умения сформированы недостаточно, все предусмотренные программой учебные задания выполнены, некоторые виды заданий выполнены с ошибками.

–«Удовлетворительно» - теоретическое содержание курса освоено частично, но пробелы не носят существенного характера, необходимые умения работы с освоенным материалом в основном сформированы, большинство предусмотренных программой обучения учебных заданий выполнено, некоторые из выполненных заданий содержат ошибки.

–«Неудовлетворительно» - теоретическое содержание курса не освоено, необходимые умения не сформированы, выполненные учебные задания содержат грубые ошибки.

Лабораторное занятие №15,16,17,18,19,20,21,22,23. Интеграционное тестирование

Цель работы:

– Научиться формировать алгоритмы разработки программных модулей в соответствии с техническим заданием;
– Выполнять отладку программных модулей с использованием специализированных программных средств;
– Выполнять тестирование программных модулей;
– Осуществлять рефакторинг и оптимизацию программного кода.

Выполнив работу, Вы будете:

уметь:

– У3. Выполнять отладку и тестирование программы на уровне модуля;
– У5. Уметь выполнять оптимизацию и рефакторинг программного кода;
– У6. Оформлять документацию на программные средства.
– У8. Применять инструментальные средства отладки программного обеспечения.
– У9. Работать с системой контроля версий.

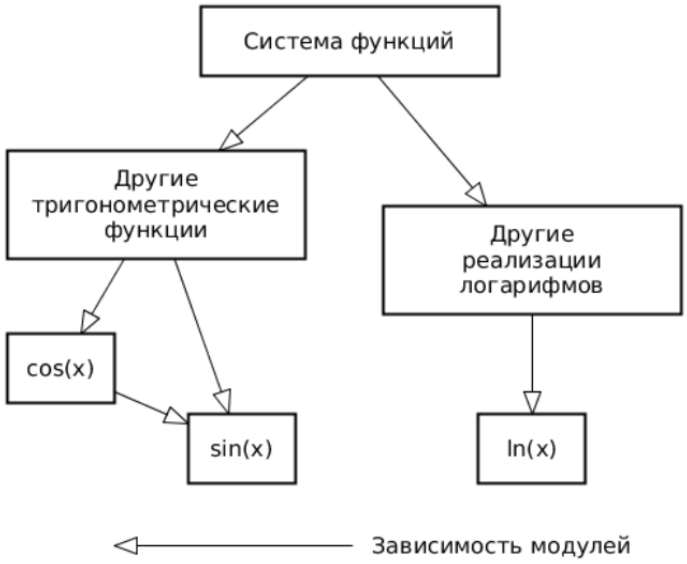
Материальное обеспечение:

– Мультимедийные средства хранения, передачи и представления

Задание:

Провести интеграционное тестирование программы, осуществляющей вычисление системы функций.

1. Все составляющие систему функции, как тригонометрические, так и логарифмические, должны быть выражены через базовые (тригонометрическая зависит от варианта; логарифмическая - натуральный логарифм).
2. Структура приложения, тестируемого в рамках лабораторной работы, должна выглядеть следующим образом (пример приведён для базовой тригонометрической функции sin(x)):



3. Обе ”базовые”функции (в примере выше - sin(x) и ln(x)) должны быть реализованы при помощи разложения в ряд с задаваемой погрешностью. Использовать тригонометрические / логарифмические преобразования для упрощения функций ЗАПРЕЩЕНО. Для КАЖДОГО модуля должны быть реализованы табличные заглушки. При этом, необходимо найти область допустимых значений функций, и, при необходимости, определить взаимозависимые точки в модулях.

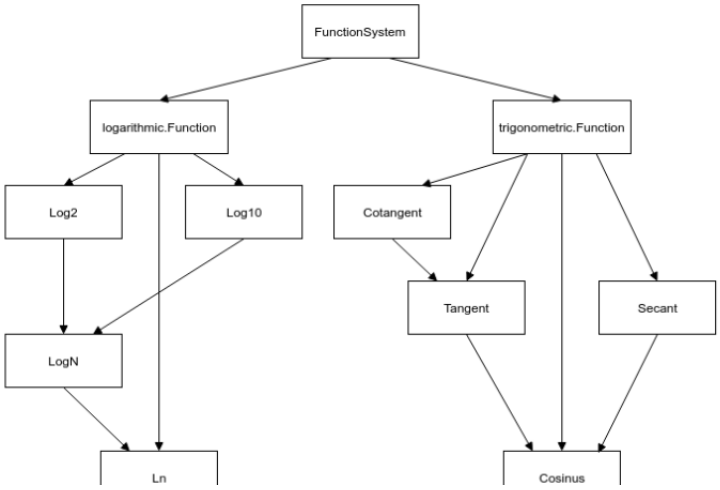
4. Разработанное приложение должно позволять выводить значения, выдаваемое любым модулем системы, в файл вида «X, Результаты модуля (X)», позволяющее произвольно менять шаг наращивания X. Разделитель в файле можно использовать произвольный.
Порядок выполнения работы:
1. Разработать приложение, руководствуясь приведёнными выше правилами.
2.С помощью JUNIT4 разработать тестовое покрытие системы функций, проведя анализ эквивалентности и учитывая особенности системы функций. Для анализа особенностей системы функций и составляющих ее частей можно использовать сайт https://www.wolframalpha.com/.
3. Собрать приложение, состоящее из заглушек. Провести интеграцию приложения по 1 модулю, с обоснованием стратегии интеграции, проведением интеграционных тестов и контролем тестового покрытия системы функций.

1.1 Система уравнений

$$\begin{cases} \left(\frac{\left(\left(\left(\cot(x)^2\right)\cdot\sec(x)\right)^2\right)-(\cot(x)-\cos(x))}{\frac{\tan(x)}{\sec(x)+\tan(x)}}\right) & \text{if } x \leq 0 \\ \left(\left(\left(\frac{(\ln(x)-\log_{10}(x))^2}{\log_2(x)}\right)^3\right)\cdot\log_2(x)\right) & \text{if } x > 0 \end{cases}$$

2 Выполнение

2.1 Диаграмма классов



2.2 Описание тестового покрытия

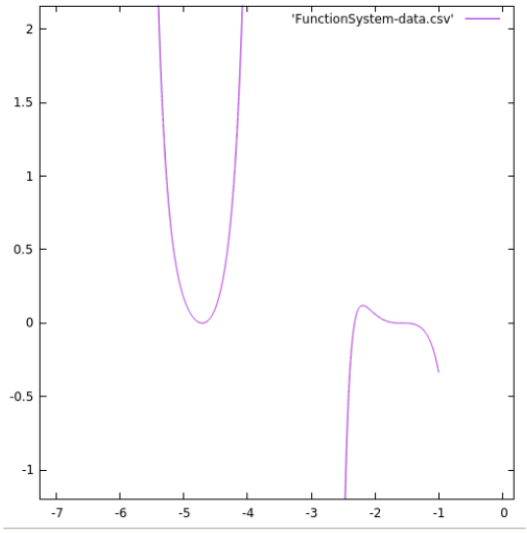
$$\left(\frac{\left(\left(\cot(x)^2\right)\cdot\sec(x)\right)^2-(\cot(x)-\cos(x))}{\frac{\tan(x)}{\sec(x)+\tan(x)}}\right) \quad \text{if} \quad x \leq 0$$

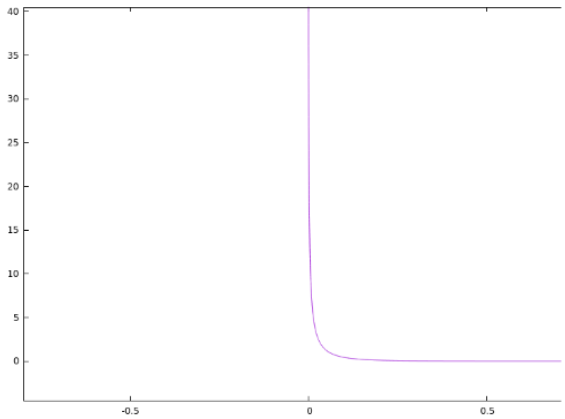
Функция периодичная, определена на $-\frac{\pi}{2} < x - 2\pi m < 0$
Краевые точки:
$\pi < x < 0$
$x = -1,5708, y = 0$
$x = -2,1904, y = 0.1202$
$x = -2,3197, y = 0$
$2\pi < x < \pi$
$x = -4,713, y = 0$

$$\left(\left(\left(\frac{(\ln(x)-\log_{10}(x))^2}{\log_2(x)}\right)^3\right)\cdot\log_2(x)\right) \quad \text{if} \quad x > 0$$

$0 < x < 1$
$x = 0, y = NaN$
$x = 1, y = NaN$
$1 < x < \infty$

2.3 Графики





Выводы
В ходе данной лабораторной работы были получены навыки разработки ПО с использованием интеграционного тестирования

Порядок выполнения работы

- Составить математическую модель задачи.
- Выбрать и обосновать наиболее рациональный метод решения задачи;
- Разработать алгоритм для решения задачи.
- Написать и отладить программу.

Форма предоставления результата

- Блок – схема.
- Код программы.

Критерии оценки:

«Отлично» - теоретическое содержание курса освоено полностью, без пробелов, умения сформированы, все предусмотренные программой учебные задания выполнены, качество их выполнения оценено высоко.

–«Хорошо» - теоретическое содержание курса освоено полностью, без пробелов, некоторые умения сформированы недостаточно, все предусмотренные программой учебные задания выполнены, некоторые виды заданий выполнены с ошибками.

–«Удовлетворительно» - теоретическое содержание курса освоено частично, но пробелы не носят существенного характера, необходимые умения работы с освоенным материалом в основном сформированы, большинство предусмотренных программой обучения учебных заданий выполнено, некоторые из выполненных заданий содержат ошибки.

–«Неудовлетворительно» - теоретическое содержание курса не освоено, необходимые умения не сформированы, выполненные учебные задания содержат грубые ошибки.

Тема 1.2.2 Документирование

Лабораторное занятие №24,25,26,27,28,29,30,31,32,33. Оформление документации на программные средства с использованием инструментальных средств

Цель работы:

- Научиться формировать алгоритмы разработки программных модулей в соответствии с техническим заданием;
- Выполнять отладку программных модулей с использованием специализированных программных средств;
- Выполнять тестирование программных модулей;
- Осуществлять рефакторинг и оптимизацию программного кода.

Выполнив работу, Вы будете:

уметь:

- У3. Выполнять отладку и тестирование программы на уровне модуля;
- У5. Уметь выполнять оптимизацию и рефакторинг программного кода;
- У6. Оформлять документацию на программные средства.
- У8. Применять инструментальные средства отладки программного обеспечения.
- У9. Работать с системой контроля версий.

Материальное обеспечение:

- Мультимедийные средства хранения, передачи и представления информации.

Краткие теоретические сведения:

Виды программных документов

Документирование программного обеспечения осуществляется в соответствии с Единой системой программной документации (ГОСТ 19.XXX). ГОСТ 19.101—77 содержит виды

программных документов для программного обеспечения различных типов. В данном ГОСТе перечислены документы следующих типов:

• *спецификация* должна содержать перечень и краткое описание назначения всех файлов программного обеспечения, в том числе и файлов документации на него, и является обязательной для программных систем, а также их компонентов, имеющих самостоятельное применение;

• *ведомость держателей подлинников* (код вида документа — 05) должна содержать список предприятий, на которых хранятся подлинники программных документов. Необходимость этого документа определяется на этапе разработки и утверждения технического задания только для программного обеспечения со сложной архитектурой;

• *текст программы* (код вида документа — 12) должен содержать текст программы с необходимыми комментариями. Необходимость этого документа определяется на этапе разработки и утверждения технического задания;

• *описание программы* (код вида документа — 13) должно содержать сведения о логической структуре и функционировании программы. Необходимость данного документа также определяется на этапе разработки и утверждения технического задания;

• *ведомость эксплуатационных документов* (код вида документа — 20) должна содержать перечень эксплуатационных документов на программу, к которым относятся документы с кодами 30, 31, 32, 33, 34, 35, 46. Необходимость этого документа также определяется на этапе разработки и утверждения технического задания;

• *формуляр* (код вида документа — 30) должен содержать основные характеристики программного обеспечения, комплектность и сведения об эксплуатации программы;

• *описание применения* (код вида документа — 31) должно содержать сведения о назначении программного обеспечения, области применения, применяемых методах, классе решаемых задач, ограничениях для применения, минимальной конфигурации технических средств;

• *руководство системного программиста* (код вида документа — 32) должно содержать сведения для проверки, обеспечения функционирования и настройки программы на условия конкретного применения;

• *руководство программиста* (код вида документа — 33) должно содержать сведения для эксплуатации программного обеспечения;

• *руководство оператора* (код вида документа — 34) содержит сведения для обеспечения процедуры общения оператора с вычислительной системой в процессе выполнения программы;

• *описание языка* (код вида документа — 35) — описание синтаксиса и семантики языка программы;

• *руководство по техническому обслуживанию* (код вида документа — 46) содержит сведения для применения программы при обслуживании технических средств.

Задание:

1. Написать код программ для решения поставленной задачи на языке программирования, выбранном на этапе эскизного проектирования.
2. Отладить программный модуль.
3. Получить результаты работы.
4. Оформить документацию к разработанному программному обеспечению.

Порядок выполнения работы

- Составить математическую модель задачи.
- Выбрать и обосновать наиболее рациональный метод решения задачи;
- Разработать алгоритм для решения задачи.
- Написать и отладить программу.

Форма предоставления результата

- Блок – схема.
- Код программы.

Критерии оценки:

«Отлично» - теоретическое содержание курса освоено полностью, без пробелов, умения сформированы, все предусмотренные программой учебные задания выполнены, качество их выполнения оценено высоко.

–«Хорошо» - теоретическое содержание курса освоено полностью, без пробелов, некоторые умения сформированы недостаточно, все предусмотренные программой учебные задания выполнены, некоторые виды заданий выполнены с ошибками.

–«Удовлетворительно» - теоретическое содержание курса освоено частично, но пробелы не носят существенного характера, необходимые умения работы с освоенным материалом в основном сформированы, большинство предусмотренных программой обучения учебных заданий выполнено, некоторые из выполненных заданий содержат ошибки.

–«Неудовлетворительно» - теоретическое содержание курса не освоено, необходимые умения не сформированы, выполненные учебные задания содержат грубые ошибки.

МДК.01.03 Разработка мобильных приложений

Тема 1.3.1 Основные платформы и языки разработки мобильных приложений Лабораторное занятие№ 1,2,3,4,5 Установка инструментария и настройка среды для разработки мобильных приложений

Цель работы:

- Научиться формировать алгоритмы разработки программных модулей в соответствии с техническим заданием;
- Разрабатывать программные модули в соответствии с техническим заданием;
- Выполнять отладку программных модулей с использованием специализированных программных средств;
- Выполнять тестирование программных модулей;
- Осуществлять рефакторинг и оптимизацию программного кода;
- Разрабатывать модули программного обеспечения для мобильных платформ.

Выполнив работу, Вы будете:

уметь:

- У2. Создавать программу по разработанному алгоритму как отдельный модуль.
- У1. Осуществлять разработку кода программного модуля на языках низкого уровня и высокого уровней;
- У4. Осуществлять разработку кода программного модуля современных языках программирования;
- У6.Оформлять документацию на программные средства.

Материальное обеспечение:

- Мультимедийные средства хранения, передачи и представления информации.

Задание:

1.Установка и настройка среды разработки

Описание приложения

- Архитектура приложения
- Модель «Дизайн-XML»
- Используемые API
- Установка на устройство
- Отладка

Установить среду разработки Eclipse

- http://www.eclipse.org/downloads/
- Установить JDK — Java Development Kit, http://www.oracle.com/technetwork/java/javase/downloads/index.html
- Для установки ADT откройте панель расширения среды (Install NewSoftware), и затем в появившемся окне добавьте новый путь - https://dl-ssl.google.com/android/eclipse/
- Далее Eclipse выполнит поиск требуемых данных, и в выпадающем списке вы сможете выбрать ADT и установить его. После установки у вас появится AVD Manager. Его ожно запустить прямо из Eclipse (Window-> Android SDK and AVD Manager)
- Настройте Virtual Device
- Установите версии API

Необходимые знания и навыки

- Общие представления об ОС Android
- Знакомство с материалом лекции№ 2

– Базовое знание языка программирования Java

Необходимые программные и аппаратные средства

– ПК под ОС Windows (или любой другой, подходящей для Android SDK)
– Устройство под управлением Android или эмулятор Android SDK

Обзор приложения

– Расположение элементов управления: TabLayout, LinearLayout, RelativeLayout и GridLayout
– Простейшие элементы управления: Button, EditText, TextView
– Список GridView

Архитектура приложения

Обычно, если приложение не имеет сложную структуру, то в одном состоянии приложения используется только один стиль размещения. Мы рассмотрим приложение, которое содержит в себе много различных layouts. Достичь этого, можно путем создания в качестве главного layout, TabLayout, у которого каждый tab имеет свой способ размещения.

l TabLayout →

- Tab1: -> LinearLayout.
- Tab2: -> RelativeLayout.
- ØTab3: -> GridLayout.

Архитектура Tab1

Интерфейс для работы с датой в Андроиде

Компоненты

-TextView - простое текстовое поле, в котором будет отображаться текущая или ручную установленная дата.

- Button - кнопка, которая инициирует появление окошка DitaPicker, в котором отображается текущая дата. В окне стандартно есть контролы позволяющие изменить дату.

Архитектура Tab2

Некоторые виджеты UI

l Компоненты.

-TextView - текстовое поле отображающее статическую информацию.

-EditText - текстовое поле с поддержкой редактирования.

-Button - кнопка OK.

-Button - кнопка Cancel.

- Spinner - выпадающий список с обработкой события выбора.

Архитектура Tab3

Графические объекты (картинки) размещенные в виде таблицы

Компоненты.

-GridView — представление, отображающее элементы в двумерной сетке с поддержкой вертикального или горизонтального скрола. Элементы, ассоциированные с Grid представлением, должны быть представлены с помощью ListAdapter.

Дизайн-XML

-Традиционное разделение кода (Java) и ресурсов (XML) — дизайна, строк и т.п.

-Удобно управлять ресурсами

-Упрощение локализации

-Разделение работы программиста и дизайнера

Используемые API

Date Picker — виджет позволяющий работать

Создать этот виджет можно при помощи конструкора:

-DatePickerDialog(Context, DatePickerDialog.

-OnDateSetListener, Year, Month, Day).

Установить дату можно при помощи объекта слушателя

DatePickerDialog.OnDateSetListener () , который вызывается всякий раз, когда пользователь нажимает на кнопку «Set». Реализуя метод экземпляра слушателя OnDataSet(DataPicker,year, month, day), мы сможем получить данные о дате из виджета и в дальнейшем работать с ними.

Компонеты Tab2

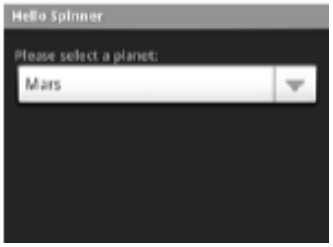
Spinner — это виджет похожий на выпадающий вниз список, отображающий один элемент и позволяющий выбрать любой другой, среди содержимого раскрывающегося списка.

- Spinner в программе можно создать с помощью конструкторов:

-Spinner(Context, int mode) — параметр mode позволяет задать способ отображения элементов списка MODE_DIALOG или MODE_DROPDOWN.

-Spinner(Context context, AttributeSet attrs, int defStyle, int mode).

Элементы спиннера определяются классом Adapter, а затем присоединяются к нему с помощью метода SetAdapter(adapter).



Tab Layout

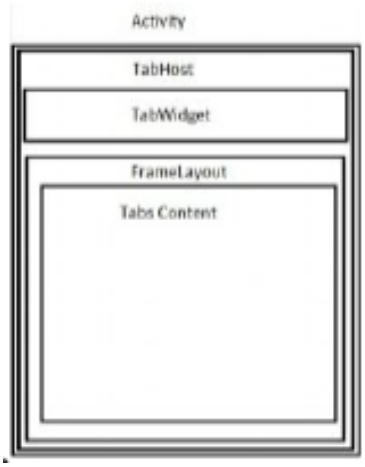
Рассмотрение структуры приложения начнем с графических компонентов. В приложении все компоненты UI определяются в xml файле. Основным менеджером размещения в примере является Tab- layout.

Использовать менеджер закладок можно двумя подходами:

- Можно создать представления для каждой закладки в одной Activity(форма приложения), а затем прицепить их к TabHost.

-Можно описать представления конкретными Activity и через главный Activity связать их с TabHost.

В приложении используется первый подход (см. Рис.).



Установка на устройство и отладка

Проделайте следующие шаги, чтобы посмотреть предварительно подготовленное приложение в действии.

1). Импорт приложения. Загрузите пример из каталога lab01 файла labAtom21.rar.

2). Запуск приложения.

Run as->Androidapplication

В результате проект перестроится и запустится приложение. После загрузке эмулятора можно убедиться в работоспособности программы и проанализировать ее код по описанию работы.

Порядок выполнения работы

- Составить математическую модель задачи.
- Выбрать и обосновать наиболее рациональный метод решения задачи;
- Разработать алгоритм для решения задачи.
- Написать и отладить программу.

Форма предоставления результата

- Блок – схема.
- Код программы.

Критерии оценки:

«Отлично» - теоретическое содержание курса освоено полностью, без пробелов, умения сформированы, все предусмотренные программой учебные задания выполнены, качество их выполнения оценено высоко.

–«Хорошо» - теоретическое содержание курса освоено полностью, без пробелов, некоторые умения сформированы недостаточно, все предусмотренные программой учебные задания выполнены, некоторые виды заданий выполнены с ошибками.

–«Удовлетворительно» - теоретическое содержание курса освоено частично, но пробелы не носят существенного характера, необходимые умения работы с освоенным материалом в основном сформированы, большинство предусмотренных программой обучения учебных заданий выполнено, некоторые из выполненных заданий содержат ошибки.

–«Неудовлетворительно» - теоретическое содержание курса не освоено, необходимые умения не сформированы, выполненные учебные задания содержат грубые ошибки.

Лабораторное занятие№ 6,7,8,9,10,11 Установка среды разработки мобильных приложений с применением виртуальной машины

Цель работы:

- Научиться формировать алгоритмы разработки программных модулей в соответствии с техническим заданием;
- Разрабатывать программные модули в соответствии с техническим заданием;

- Выполнять отладку программных модулей с использованием специализированных программных средств;
- Выполнять тестирование программных модулей;
- Осуществлять рефакторинг и оптимизацию программного кода;
- Разрабатывать модули программного обеспечения для мобильных платформ.

Выполнив работу, Вы будете:

уметь:

- У2. Создавать программу по разработанному алгоритму как отдельный модуль.
- У1. Осуществлять разработку кода программного модуля на языках низкого уровня и высокого уровней;
- У4. Осуществлять разработку кода программного модуля современных языках программирования;
- У6.Оформлять документацию на программные средства.

Материальное обеспечение:

- Мультимедийные средства хранения, передачи и представления информации.

Задание:

1.Установка и запуск гостевой ОС в виртуальной машине VirtualBox. Виртуальные машины поддерживают установку как с ISO-образа, так и установочного диска ОС. Для установки используем ISO – образ операционной системы Ubuntu (Linux) ubuntu-11.10-desktop-i386.iso.

- Укажем папку для виртуальных машин.
- Необходимо создать виртуальную машину, используя VirtualBox (VMware Workstation, Player или другое ПО на выбор). Установить и запустить гостевую ОС Linux (Дистрибутив Linux либо предложенный, либо на выбор любой другой – Linux Mint, Ubuntu, Debian...). В качестве имени пользователя использовать свою фамилию, пароль – ваше имя. Дополнительно реализовать задание по варианту. В отчет включить описание процесса установки, настройки виртуальной машины и ОС, сопровождая скриншотами.
-

1.Используя проводник файлов Nautilus создать в каталоге /home/имяпользователя/ папку с названием предмета, где будут содержаться отчеты по лабораторным работам. Также научиться производить основные операции над файлами, включая создание, копирование, переименование файлов и удаление. Отсортировать файлы по имени.
2.Используя проводник файлов Nautilus создать в каталоге /home/имяпользователя/ папку для хранения изображений. Также научиться производить основные операции над файлами, включая копирование, переименование файлов и удаление. Отсортировать файлы по размеру.

Порядок выполнения работы

- Составить математическую модель задачи.
- Написать код программ для решения поставленной задачи на языке программирования, выбранном на этапе эскизного проектирования
- Отладить программный модуль.
- Получить результаты работы.
- Оформить документацию к разработанному программному обеспечению.

Форма предоставления результата

- Блок – схема.
- Код программы.

Критерии оценки:

«Отлично» - теоретическое содержание курса освоено полностью, без пробелов, умения сформированы, все предусмотренные программой учебные задания выполнены, качество их выполнения оценено высоко.

–«Хорошо» - теоретическое содержание курса освоено полностью, без пробелов, некоторые умения сформированы недостаточно, все предусмотренные программой учебные задания выполнены, некоторые виды заданий выполнены с ошибками.

–«Удовлетворительно» - теоретическое содержание курса освоено частично, но пробелы не носят существенного характера, необходимые умения работы с освоенным материалом в основном сформированы, большинство предусмотренных программой обучения учебных заданий выполнено, некоторые из выполненных заданий содержат ошибки.

–«Неудовлетворительно» - теоретическое содержание курса не освоено, необходимые умения не сформированы, выполненные учебные задания содержат грубые ошибки.

Тема 1.3.2 Создание и тестирование модулей для мобильных приложений Лабораторное занятие №12. Создание эмуляторов и подключение устройств. Лабораторное занятие №13. Настройка режима терминала.

Цель работы:

- Научиться формировать алгоритмы разработки программных модулей в соответствии с техническим заданием;
- Разрабатывать программные модули в соответствии с техническим заданием;
- Выполнять отладку программных модулей с использованием специализированных программных средств;
- Выполнять тестирование программных модулей;
- Осуществлять рефакторинг и оптимизацию программного кода;
- Разрабатывать модули программного обеспечения для мобильных платформ.

Выполнив работу, Вы будете:

уметь:

- У2. Создавать программу по разработанному алгоритму как отдельный модуль.
- У1. Осуществлять разработку кода программного модуля на языках низкого уровня и высокого уровней;
- У4. Осуществлять разработку кода программного модуля современных языках программирования;
- У6.Оформлять документацию на программные средства.

Материальное обеспечение:

- Мультимедийные средства хранения, передачи и представления информации.

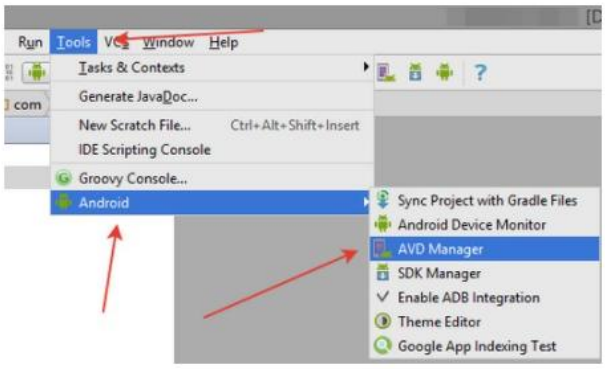
Задание:

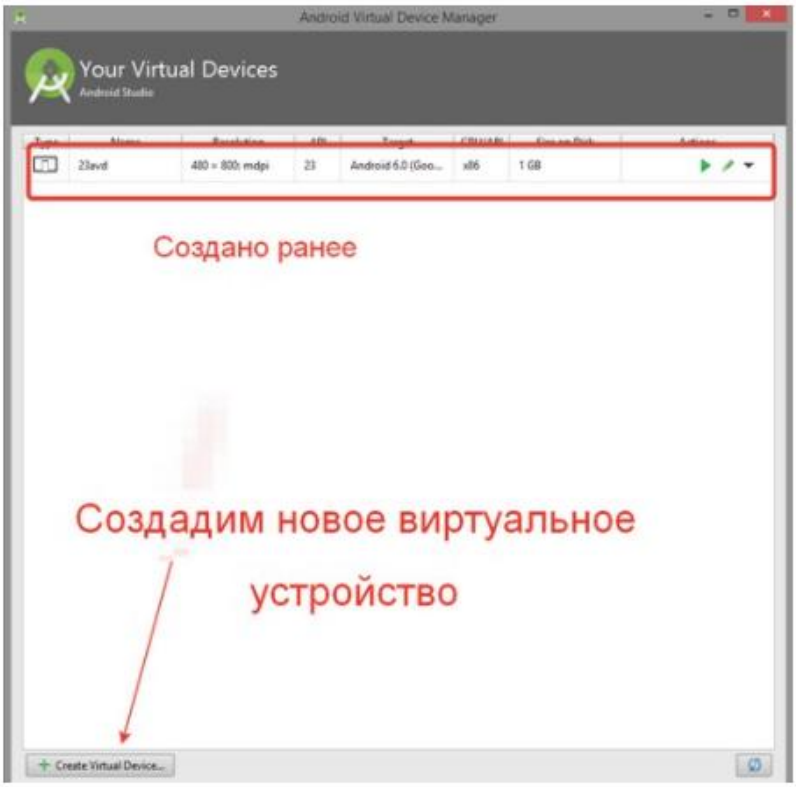
– Создать эмулятор Android.

– Настроить эмулятор.

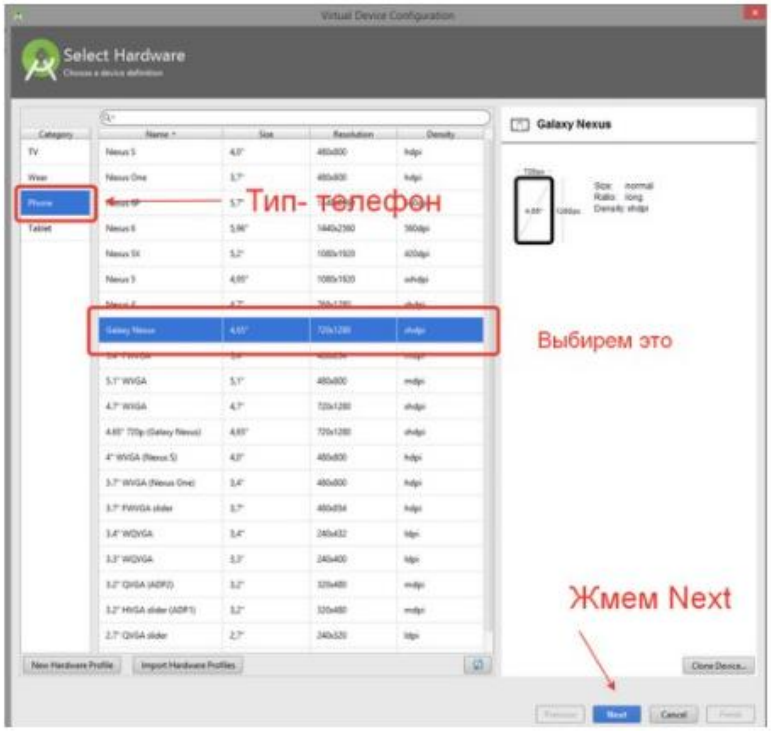
Порядок выполнения работы

1. Запустить Android Studio, в верхнем меню выбрать Tools->Android->AVD Manader. В нем нажать кнопку Create Virtual Device…

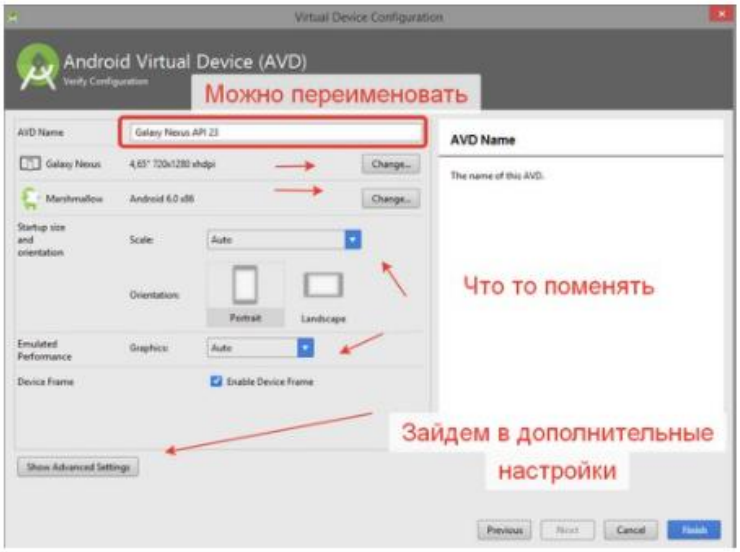




2. В Category выбрать Phone.



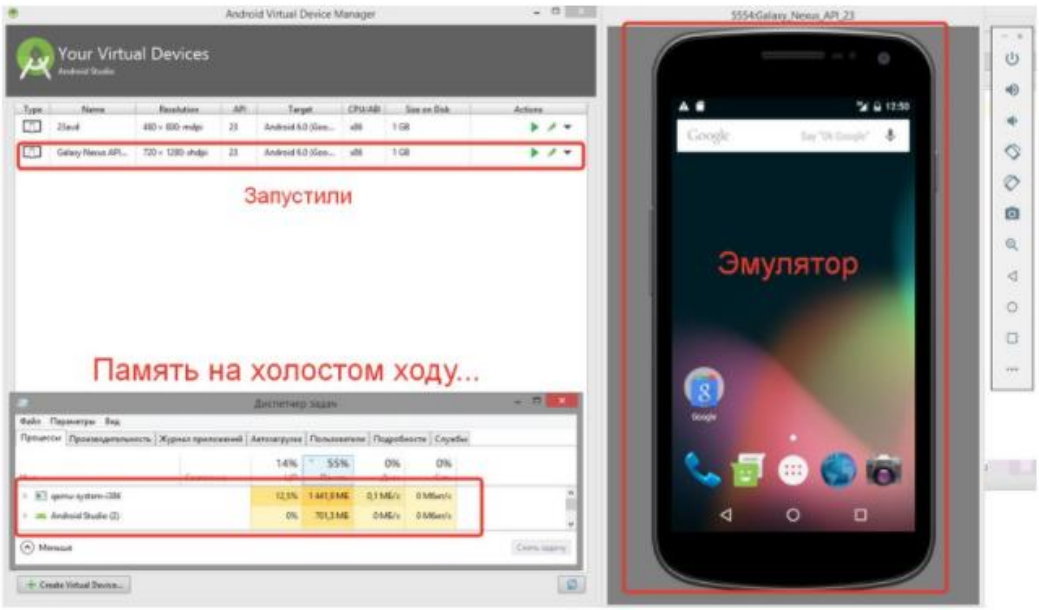
3. Переименовать виртуальное устройств, сменить размер экрана, версию андроида.



4. Настроить дополнительно Show Advances Setting, на которой будут показаны элементы настройки камеры, сети, можно регулировать объем оперативной памяти, размер SD CARS.

5. При создании нового AVD, создается новое виртуальное устройство.

6. Запустить виртуальное устройство.



Форма предоставления результата

Отчет о проделанной работе.

Критерии оценки:

«Отлично» - теоретическое содержание курса освоено полностью, без пробелов, умения сформированы, все предусмотренные программой учебные задания выполнены, качество их выполнения оценено высоко.

–«Хорошо» - теоретическое содержание курса освоено полностью, без пробелов, некоторые умения сформированы недостаточно, все предусмотренные программой учебные задания выполнены, некоторые виды заданий выполнены с ошибками.

–«Удовлетворительно» - теоретическое содержание курса освоено частично, но пробелы не носят существенного характера, необходимые умения работы с освоенным материалом в основном сформированы, большинство предусмотренных программой обучения учебных заданий выполнено, некоторые из выполненных заданий содержат ошибки.

–«Неудовлетворительно» - теоретическое содержание курса не освоено, необходимые умения не сформированы, выполненные учебные задания содержат грубые ошибки.

–

Лабораторное занятие №14. Создание нового проекта.
Лабораторное занятие №15. Изучение и комментирование кода.
Лабораторное занятие №16. Изменение элементов дизайна

Цель работы:

- Научиться формировать алгоритмы разработки программных модулей в соответствии с техническим заданием;
- Разрабатывать программные модули в соответствии с техническим заданием;
- Выполнять отладку программных модулей с использованием специализированных программных средств;
- Выполнять тестирование программных модулей;
- Осуществлять рефакторинг и оптимизацию программного кода;
- Разрабатывать модули программного обеспечения для мобильных платформ.

Выполнив работу, Вы будете:

уметь:

- У2. Создавать программу по разработанному алгоритму как отдельный модуль.
- У1. Осуществлять разработку кода программного модуля на языках низкого уровня и высокого уровней;
- У4. Осуществлять разработку кода программного модуля современных языках программирования;
- У6.Оформлять документацию на программные средства.

Материальное обеспечение:

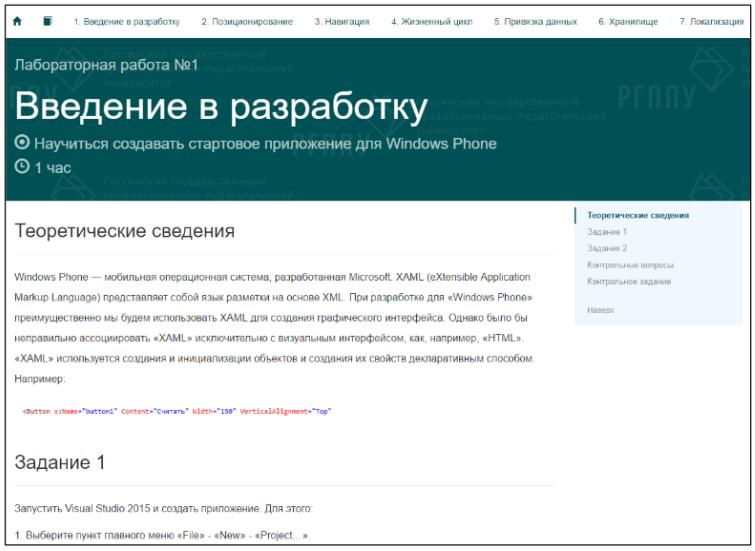
- Мультимедийные средства хранения, передачи и представления информации.

Задание:

1.

- Создать новый проект.
- Создать новое приложение.
- Добавить элементы управления в приложение.
- Запустить приложение.

Лабораторное занятиепредставлена на рисунке.



Пример итогового результата выполнения лабораторной работы изображен на рисунке

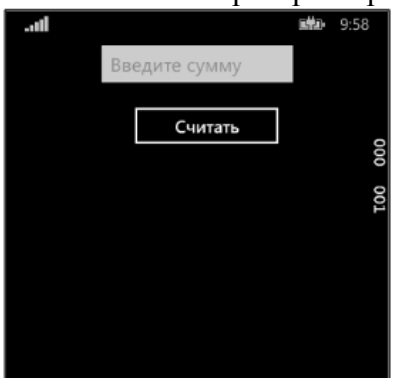
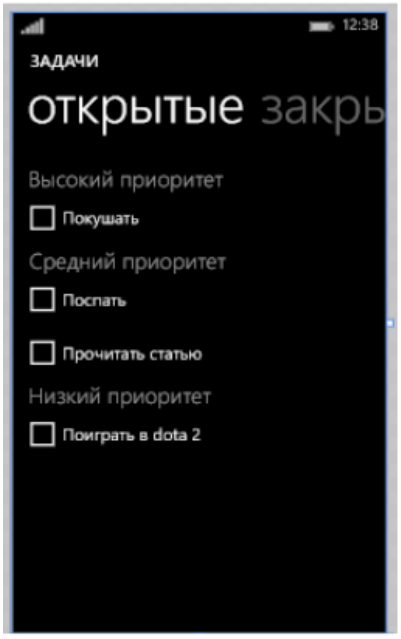


Рисунок – Вид итогового результата выполнения лабораторной работы N1

Задание 2

- Создать новый проект.
- Изучить разметку.
- Изучить расположение элементов на экране.
- Изучить новые элементы управления.
- Наполнить приложение элементами управления с использованием компоновки.

Пример итогового результата выполнения лабораторной работы изображен на рисунке.



Порядок выполнения работы

- Написать код программ для решения поставленной задачи на языке программирования, выбранном на этапе эскизного проектирования.
- Отладить программный модуль.
- Получить результаты работы.
- Оформить документацию к разработанному программному обеспечению.

Форма предоставления результата

- Блок – схема.
- Код программы.

Критерии оценки:

«Отлично» - теоретическое содержание курса освоено полностью, без пробелов, умения сформированы, все предусмотренные программой учебные задания выполнены, качество их выполнения оценено высоко.

–«Хорошо» - теоретическое содержание курса освоено полностью, без пробелов, некоторые умения сформированы недостаточно, все предусмотренные программой учебные задания выполнены, некоторые виды заданий выполнены с ошибками.

–«Удовлетворительно» - теоретическое содержание курса освоено частично, но пробелы не носят существенного характера, необходимые умения работы с освоенным материалом в основном сформированы, большинство предусмотренных программой обучения учебных заданий выполнено, некоторые из выполненных заданий содержат ошибки.

–«Неудовлетворительно» - теоретическое содержание курса не освоено, необходимые умения не сформированы, выполненные учебные задания содержат грубые ошибки.

Лабораторное занятие №17. Обработка событий: подсказки
Лабораторное занятие №18. Обработка событий: цветовая индикация
Лабораторное занятие №19. Подготовка стандартных модулей
Лабораторное занятие № 20. Обработка событий: переключение между экранами

Цель работы:

- Научиться формировать алгоритмы разработки программных модулей в соответствии с техническим заданием;
- Разрабатывать программные модули в соответствии с техническим заданием;
- Выполнять отладку программных модулей с использованием специализированных программных средств;
- Выполнять тестирование программных модулей;
- Осуществлять рефакторинг и оптимизацию программного кода;
- Разрабатывать модули программного обеспечения для мобильных платформ.

Выполнив работу, Вы будете:

уметь:

- У2. Создавать программу по разработанному алгоритму как отдельный модуль.
- У1. Осуществлять разработку кода программного модуля на языках низкого уровня и высокого уровней;
- У4. Осуществлять разработку кода программного модуля современных языках программирования;
- У6.Оформлять документацию на программные средства.

Материальное обеспечение:

- Мультимедийные средства хранения, передачи и представления информации.

Задание:

1.

- Добавить вторую страницу для приложения.
- Изучить новые элементы управления.
- Добавить кнопки, которые будут осуществлять навигацию.
- Создать навигацию между страницами.

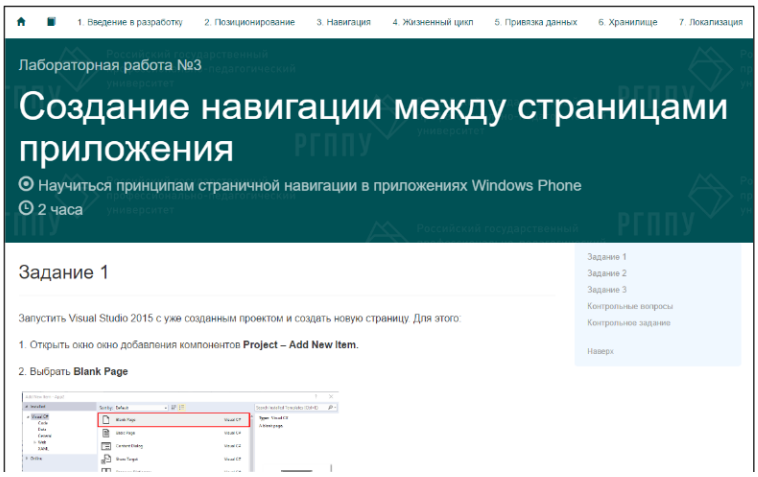
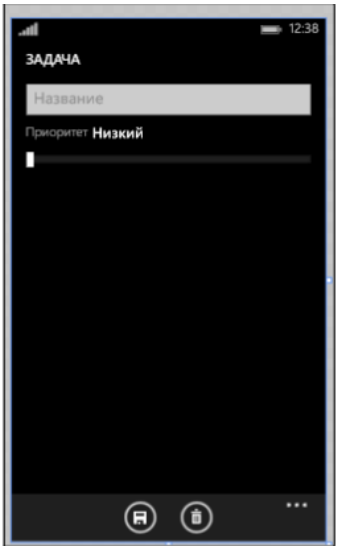


Рисунок – Вид страницы лабораторной работы

Результатом выполнения данной работы является новая страница в приложении с элементами управления и новые кнопки на обеих страницах, которые осуществляют навигацию между этими страницами

.Пример итогового результата выполнения лабораторной работы изображен на рисунке



Порядок выполнения работы

- Написать код программ для решения поставленной задачи на языке программирования, выбранном на этапе эскизного проектирования.
- Отладить программный модуль.
- Получить результаты работы.
- Оформить документацию к разработанному программному обеспечению.

Форма предоставления результата

- Блок – схема.
- Код программы.

Критерии оценки:

«Отлично» - теоретическое содержание курса освоено полностью, без пробелов, умения сформированы, все предусмотренные программой учебные задания выполнены, качество их выполнения оценено высоко.

–«Хорошо» - теоретическое содержание курса освоено полностью, без пробелов, некоторые умения сформированы недостаточно, все предусмотренные программой учебные задания выполнены, некоторые виды заданий выполнены с ошибками.

–«Удовлетворительно» - теоретическое содержание курса освоено частично, но пробелы не носят существенного характера, необходимые умения работы с освоенным материалом в основном сформированы, большинство предусмотренных программой обучения учебных заданий выполнено, некоторые из выполненных заданий содержат ошибки.

–«Неудовлетворительно» - теоретическое содержание курса не освоено, необходимые умения не сформированы, выполненные учебные задания содержат грубые ошибки.

Лабораторное занятие №21. Передача данных между модулями. Лабораторное занятие № 22,23. Тестирование и оптимизация мобильного приложения

Цель работы:

- Научиться формировать алгоритмы разработки программных модулей в соответствии с техническим заданием;
- Разрабатывать программные модули в соответствии с техническим заданием;
- Выполнять отладку программных модулей с использованием специализированных программных средств;

– Выполнять тестирование программных модулей;
– Осуществлять рефакторинг и оптимизацию программного кода;
– Разрабатывать модули программного обеспечения для мобильных платформ.

Выполнив работу, Вы будете:

уметь:

– У2. Создавать программу по разработанному алгоритму как отдельный модуль.
– У1. Осуществлять разработку кода программного модуля на языках низкого уровня и высокого уровней;
– У4. Осуществлять разработку кода программного модуля современных языках программирования;
– У6.Оформлять документацию на программные средства.

Материальное обеспечение:

– Мультимедийные средства хранения, передачи и представления информации.

Задание:

– Настроить сохранение данных.
– Изменить классы в соответствии с заданием.
– Настроить отображение сохраненных данных.
– Настроить чтение данных.
– Протестировать и оптимизировать приложение.

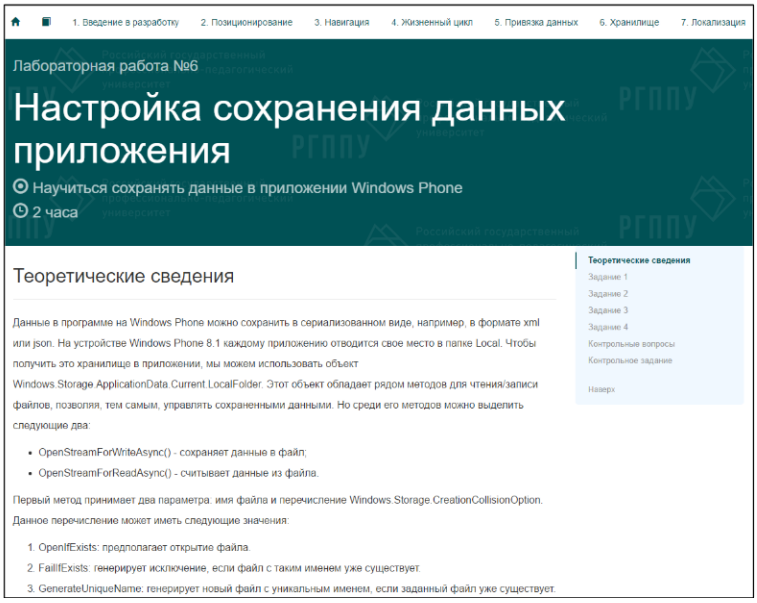
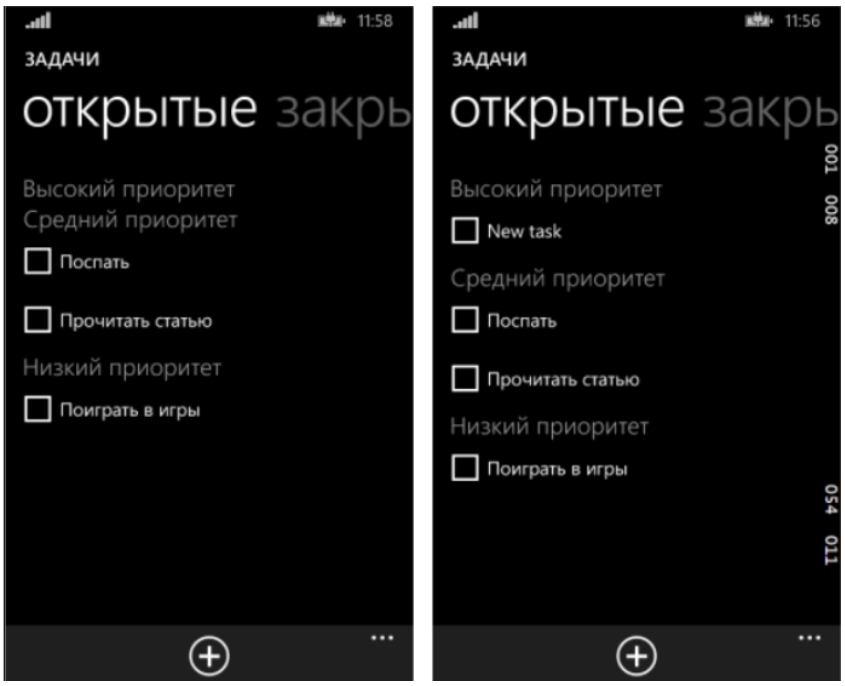


Рисунок – Вид страницы лабораторной работы

Пример итогового результата выполнения лабораторной работы изображен на рисунке.



Порядок выполнения работы

- Написать код программ для решения поставленной задачи на языке программирования, выбранном на этапе эскизного проектирования.
- Отладить программный модуль.
- Получить результаты работы.
- Оформить документацию к разработанному программному обеспечению.

Форма предоставления результата

- Блок – схема.
- Код программы.

Критерии оценки:

«Отлично» - теоретическое содержание курса освоено полностью, без пробелов, умения сформированы, все предусмотренные программой учебные задания выполнены, качество их выполнения оценено высоко.

–«Хорошо» - теоретическое содержание курса освоено полностью, без пробелов, некоторые умения сформированы недостаточно, все предусмотренные программой учебные задания выполнены, некоторые виды заданий выполнены с ошибками.

–«Удовлетворительно» - теоретическое содержание курса освоено частично, но пробелы не носят существенного характера, необходимые умения работы с освоенным материалом в основном сформированы, большинство предусмотренных программой обучения учебных заданий выполнено, некоторые из выполненных заданий содержат ошибки.

–«Неудовлетворительно» - теоретическое содержание курса не освоено, необходимые умения не сформированы, выполненные учебные задания содержат грубые ошибки.

МДК.01.04 Системное программирование

Тема 1.4.1 Программирование на языке низкого уровня

Лабораторное занятие №1. Перевод чисел в различные системы счисления

Цель работы:

- приобретение навыков выполнения операций в различных системах счисления.

Выполнив работу, Вы будете:

уметь:

- У2. Создавать программу по разработанному алгоритму как отдельный модуль.
- У1.Осуществлять разработку кода программного модуля на языках низкого уровня и высокого уровней;
- У3. Выполнять отладку и тестирование программы на уровне модуля.
- У6. Оформлять документацию на программные средства.
- У8. Применять инструментальные средства отладки программного обеспечения.

Материальное обеспечение:

- Мультимедийные средства хранения, передачи и представления информации.

Задание

Выполнить перевод чисел

а) из 10–ой с/с в 2–ую систему счисления: 165; 541; 600; 720; 43,15; 234,99.

б) из 2–ой в 10–ую систему счисления: 1101012; 110111012; 1100010112; 1001001,1112

в) из 2–ой с/с в 8–ую ,16–ую с/с:
1001011102; 1000001112; 1110010112; 10110010112; 1100110010112; 10101,101012; 111,0112

г) из 10–ой с/с в 8–ую, 16–ую с/с: 69; 73; 113; 203; 351; 641; 478,99; 555,555

д) из 8–ой с/с в 10–ую с/с: 358 ; 658 ; 2158 ; 3278 ; 5328 ; 7518; 45,4548

е) из 16–ой с/с в 10–ую с/с: D816 ; 1AE16 ; E5716 ; 8E516 ; FAD16; AFF,6A716

Выпишите целые десятичные числа, принадлежащие следующим числовым промежуткам:

[101012; 1100002]; [148; 208]; [1816; 3016]

Выполнить операции:
а) сложение в двоичной системе счисления

+ 100100112	+ 10111012	+ 101100112	+10111001,12
10110112	111011012	10101012	10001101,12

б) вычитание в 2–ой системе счисления

– 1000010002 – 1101011102 – 111011102 -10111001,12
101100112 101111112 10110112 10001101,12

в) умножение в 2–ой системе счисления
× 1000012 × 1001012 × 1111012 × 11001,012
1111112 1110112 1111012 11,012

г) деление в 2–ой системе счисления
1) 1110100010012 / 1111012
2) 1000110111002 / 1101102
3) 100000011112 / 1111112

д) сложение 8–ых чисел
+ 7158 + 5248 + 7128 + 3218 + 57318 + 63518
738 578 7638 7658 13768 7378

е) вычитание 8–ых чисел
– 1378 – 4368 – 7058 – 5388 – 72138
728 1378 768 578 5378

ж) сложение 16–ых чисел
+ A1316 + F0B16 + 2EA16 + ABC16 + A2B16
16F16 1DA16FCE16 C7C16 7F216

з) вычитание 16–ых чисел
– À1716 – DFA16 – FO516 – DE516 – D3C116
1FC16 1AE16 AD16 AF16 D1F16

Вычислите выражение:
(11111012 + AF16) / 368; 1258 + 111012 × A216 / 14178

Порядок выполнения

Система счисления, или просто счисление, или нумерация, – набор конкретных знаков–цифр вместе с системой приемов записи, которая представляет числа этими цифрами.

Основные понятия систем счисления

Система счисления – это совокупность правил и приемов записи чисел с помощью набора цифровых знаков. Количество цифр, необходимых для записи числа в системе, называют основанием системы счисления. Основание системы записывается в справа числа в нижнем индексе: 5_{10}; 1110110_2; $AF178_{16}$.

Различают два типа систем счисления:

позиционные, когда значение каждой цифры числа определяется ее позицией в записи числа;

непозиционные, когда значение цифры в числе не зависит от ее места в записи числа.

Примером непозиционной системы счисления является римская: числа IX, IV, XV и т.д. Примером позиционной системы счисления является десятичная система, используемая повседневно.

Любое целое число в позиционной системе можно записать в форме многочлена:

$$X_S = \{A_n A_{n-1} \ldots A_2 A_1\} = A_n \cdot S^{n-1} + A_{n-1} \cdot S^{n-2} + \ldots + A_2 \cdot S^1 + A_1 \cdot S^0,$$

где S — основание системы счисления;

A_n — цифры числа, записанного в данной системе счисления;

n — количество разрядов числа.

Пример. Число 6293_{10} запишется в форме многочлена следующим образом:

$6293_{10} = 6 \cdot 10^3 + 2 \cdot 10^2 + 9 \cdot 10^1 + 3 \cdot 10^0$

Десятичная система счисления – в настоящее время наиболее известная и используемая. неправильное название удерживается и поныне.

Десятичная система использует десять цифр —— 0, 1, 2, 3, 4, 5, 6, 7, 8 и 9, а также символы “+” и “–” для обозначения знака числа и запятую или точку для разделения целой и дробной частей числа.

В вычислительных машинах используется двоичная система счисления, её основание — число 2. Для записи чисел в этой системе используют только две цифры — 0 и 1.

Таблица 1

Соответствие чисел, записанных в различных системах счисления

Десятичная	Двоичная	Восьмеричная	Шестнадцатеричная
1	001	1	1
2	010	2	2
3	011	3	3
4	100	4	4
5	101	5	5
6	110	6	6
7	111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F
16	10000	20	10

Правила перевода чисел из одной системы счисления в другую

Перевод чисел из одной системы счисления в другую составляет важную часть машинной арифметики. Рассмотрим основные правила перевода.

Для перевода двоичного числа в десятичное необходимо его записать в виде многочлена, состоящего из произведений цифр числа и соответствующей степени числа 2, и вычислить по правилам десятичной арифметики:

$X_2 = A_n \cdot 2^{n-1} + A_{n-1} \cdot 2^{n-2} + A_{n-2} \cdot 2^{n-3} + ... + A_2 \cdot 2^1 + A_1 \cdot 2^0$ При переводе удобно пользоваться таблицей степеней двойки:

Таблица 2

Степени числа 2

n	0	1	2	3	4	5	6	7	8	9	10
2^n	1	2	4	8	16	32	64	128	256	512	1024

Пример. Число 11101000_2 перевести в десятичную систему счисления.

$11101000_2 = 1 \cdot 2^7 + 1 \cdot 2^6 + 1 \cdot 2^5 + 0 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 0 \cdot 2^0 = 232_{10}$ Для перевода восьмеричного числа в десятичное необходимо его записать в виде многочлена,

состоящего из произведений цифр числа и соответствующей степени числа 8, и вычислить по правилам десятичной арифметики:

$$X_8 = A_n \cdot 8^{n-1} + A_{n-1} \cdot 8^{n-2} + A_{n-2} \cdot 8^{n-3} + \ldots + A_2 \cdot 8^1 + A_1 \cdot 8^0$$

При переводе удобно пользоваться таблицей степеней восьмерки:

Таблица 3

Степени числа 8

n	0	1	2	3	4	5	6
8^n	1	8	64	512	4096	32768	262144

Пример. Число 75013_8 перевести в десятичную систему счисления.

$$75013_8 = 7 \cdot 8^4 + 5 \cdot 8^3 + 0 \cdot 8^2 + 1 \cdot 8^1 + 3 \cdot 8^0 = 31243_{10}$$

Для перевода шестнадцатеричного числа в десятичное необходимо его записать в виде многочлена, состоящего из произведений цифр числа и соответствующей степени числа 16, и вычислить по правилам десятичной арифметики:

$$X_{16} = A_n \cdot 16^{n-1} + A_{n-1} \cdot 16^{n-2} + A_{n-2} \cdot 16^{n-3} + \ldots + A_2 \cdot 16^1 + A_1 \cdot 16^0$$ При переводе удобно пользоваться таблицей степеней числа 16:

Таблица 4

Степени числа 16

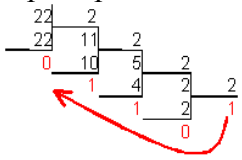
n	0	1	2	3	4	5	6
16^n	1	16	256	4096	65536	1048576	16777216

Пример. Число $FDA1_{16}$ перевести в десятичную систему счисления.

$$FDA1_{16} = 15 \cdot 163 + 13 \cdot 16^2 + 10 \cdot 16^1 + 1 \cdot 16^0 = 64929_{10}$$

Для перевода десятичного числа в двоичную систему его необходимо последовательно делить на 2 до тех пор, пока не останется остаток, меньший или равный 1. Число в двоичной системе записывается как последовательность последнего результата деления и остатков от деления в обратном порядке.

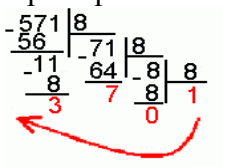
Пример. Число 22_{10} перевести в двоичную систему счисления.



$$22_{10} = 10110_2$$

Для перевода десятичного числа в восьмеричную систему его необходимо последовательно делить на 8 до тех пор, пока не останется остаток, меньший или равный 7. Число в восьмеричной системе записывается как последовательность цифр последнего результата деления и остатков от деления в обратном порядке.

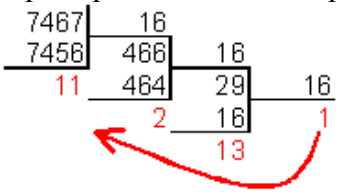
Пример. Число 571_{10} перевести в восьмеричную систему счисления.



$$571_{10} = 1073_8$$

Для перевода десятичного числа в шестнадцатеричную систему его необходимо последовательно делить на 16 до тех пор, пока не останется остаток, меньший или равный 15. Число в шестнадцатеричной системе записывается как последовательность цифр последнего результата деления и остатков от деления в обратном порядке.

Пример. Число 7467_2 перевести в шестнадцатеричную систему счисления.



$7467_{10} = 1D2B_{16}$

Чтобы перевести число из двоичной системы в восьмеричную, его нужно разбить на триады (тройки цифр), начиная с младшего разряда, в случае необходимости дополнив старшую триаду нулями, и каждую триаду заменить соответствующей восьмеричной цифрой (табл. 3).

Пример. Число 1001011_2 перевести в восьмеричную систему счисления.

$001\ 001\ 011_2 = 113_8$

Чтобы перевести число из двоичной системы в шестнадцатеричную, его нужно разбить на тетрады (четверки цифр), начиная с младшего разряда, в случае необходимости дополнив старшую тетраду нулями, и каждую тетраду заменить соответствующей восьмеричной цифрой (табл. 3).

Пример. Число 1011100011_2 перевести в шестнадцатеричную систему счисления.

$0010\ 1110\ 0011_2 = 2E3_{16}$

Для перевода восьмеричного числа в двоичное необходимо каждую цифру заменить эквивалентной ей двоичной триадой.

Пример. Число 531_8 перевести в двоичную систему счисления.

$531_8 = 101011001_2$

Для перевода шестнадцатеричного числа в двоичное необходимо каждую цифру заменить эквивалентной ей двоичной тетрадой.

Пример. Число $EE8_{16}$ перевести в двоичную систему счисления.

$EE8_{16} = 111011101000_2$

При переходе из восьмеричной системы счисления в шестнадцатеричную и обратно, необходим промежуточный перевод чисел в двоичную систему.

Пример 1. Число FEA_{16} перевести в восьмеричную систему счисления.

$FEA_{16} = 111111101010_2$

$111\ 111\ 101\ 010_2 = 7752_8$

Пример 2. Число 6653_8 перевести в шестнадцатеричную систему счисления.

$6653_8 = 110110101011_2$

$1101\ 1010\ 1011_2 = DAB_{16}$

Арифметические действия над целыми числами в 2-ой системе счисления :

1.Операция сложения выполняется с использованием таблицы двоичного сложения в одном разряде:

+	0
0	0
1	1

Пример.

а) +10012 б) +11012 в) +111112

10102 10112 12

100112 110002 1000002

2.Операция вычитания выполняется с использованием таблицы вычитания, в которой 1 обозначается заем в старшем разряде.

Пример.

а) –1011100112 б) –1101011012

1000110112 1010111112

0010110002 0010011102

3.Операция умножения выполняется по обычной схеме, применяемой в десятичной с/с с последовательным умножением множимого на очередную цифру множителя.

×	0
0	0
1	0

Пример.

а) × 110012 б) × 1012

11012 112

11001 101

11001 101

11001 11112

1010001012

4.Операция деления выполняется по алгоритму, подобному алгоритму выполнения операции деления в 10–ой с/с.

Пример.

```
10100010 12  11012    1000110002 11112
 1101      110012     1111       100102
 1110                 10100
 1101                 1111
  001101              10102 –остаток
   1101
     0
```

Сложение и вычитание в восьмеричной системе счисления.

При выполнении сложения и вычитания в 8-ой с/с необходимо соблюдать следующие правила:

в записи результатов сложения и вычитания могут быть использованы только цифры восьмеричного алфавита;

десяток восьмеричной системы счисления равен 8, т.е. переполнение разряда наступает, когда результат сложения больше или равен 8.

В этом случае для записи результата надо вычесть 8, записать остаток, а к старшему разряду прибавить единицу переполнения;

3)если при вычитании приходится занимать единицу в старшем разряде, эта единица переносится в младший разряд в виде восьми единиц.

Пример

+ 7708	+ 7508
2368	2368
12268	5128

Сложение и вычитание в шестнадцатеричной системе счисления.

При выполнении этих действий в 16–ой с/с необходимо соблюдать следующие правила:

1)при записи результатов сложения и вычитания надо использовать цифры шестнадцатеричного алфавита: цифры, обозначающие числа от 10 до 15 записываются латинскими буквами, поэтому, если результат является числом из этого промежутка, его надо записывать соответствующей латинской буквой;

2)десяток шестнадцатеричной системы счисления равен 16, т.е. переполнение разряда поступает, если результат сложения больше или равен 16, и в этом случае для записи результата надо вычесть 16, записать остаток, а к старшему разряду прибавить единицу переполнения;

3)если приходится занимать единицу в старшем разряде, эта единица переносится в младший разряд в виде шестнадцати единиц.

Примеры.

+ B0916	+ B0916
TFA16	7FA16
1A0316	30F16

Форма предоставления результата

- Блок – схема.
- Код программы.

Критерии оценки:

«Отлично» - теоретическое содержание курса освоено полностью, без пробелов, умения сформированы, все предусмотренные программой учебные задания выполнены, качество их выполнения оценено высоко.

–«Хорошо» - теоретическое содержание курса освоено полностью, без пробелов, некоторые умения сформированы недостаточно, все предусмотренные программой учебные задания выполнены, некоторые виды заданий выполнены с ошибками.

–«Удовлетворительно» - теоретическое содержание курса освоено частично, но пробелы не носят существенного характера, необходимые умения работы с освоенным материалом в основном сформированы, большинство предусмотренных программой обучения учебных заданий выполнено, некоторые из выполненных заданий содержат ошибки.

–«Неудовлетворительно» - теоретическое содержание курса не освоено, необходимые умения не сформированы, выполненные учебные задания содержат грубые ошибки.

Лабораторное занятие №2. Работа и использование отладчика AFD (интерфейс, функциональные клавиши, основные команды отладчика)

Цель работы:

- познакомиться с интерфейсом отладчика AFDP и его основными возможностями

Выполнив работу, Вы будете:

уметь:

- У2. Создавать программу по разработанному алгоритму как отдельный модуль.
- У1.Осуществлять разработку кода программного модуля на языках низкого уровня и высокого уровней;
- У3. Выполнять отладку и тестирование программы на уровне модуля.
- У6. Оформлять документацию на программные средства.
- У8. Применять инструментальные средства отладки программного обеспечения.

Материальное обеспечение:

– Мультимедийные средства хранения, передачи и представления информации.

Задание

Пример 1

Создать файл длиной 256 байтов. Половина файла содержит код 37h, остальная – 67h. Первый байт в файле 00H, последний – FFh. Записать файл в корневой каталог на диск C: под именем PRIMER1.dat.

Решение:

Для записи файла необходимо сформировать область памяти согласно условию задачи. Эту область памяти назовем буфером и выберем адрес этого буфера в текущем кодовом сегменте – 1000h.

Для решения задачи необходимо уметь оперировать с числами в шестнадцатиричной форме. Число 256 в шестнадцатеричном виде – 100h, а половина длины файла

256/2=128 = 80h. Для контроля начало первой половины буфера памяти отобразим в окне Memory 1 командой:

M1 1000

Начало второй половины буфера памяти отобразим во 2 окне Memory 2 командой:

M2 1080

Заполним первую и вторую половину буфера:

F 1000,80,37

F 1080,80,67

Записываем байт в начало и конец буфера:

F 1000,1,0

F 10FF,1,FF

Сохраняем буфер в виде файла: W 1000,100,c:\primer2.dat

На диске c:\ создался файл длиной 256 байт, который можно просмотреть в виде шестнадцатеричного дампа памяти, например встроенным редактором в оболочках Norton Commander или Far manager.

Задачи для самостоятельного выполнения

Задача 1

Создать файл длиной 1001 байт, половина которого заполнена строкой «Miha», а вторая половина строкой «Rita». В 501 байте должен находиться символ “+” Записать файл на диск под именем polovina.dat. Просмотрите полученный файл текстовым редактором в символьном и шестнадцатеричном виде.

Рекомендации:

Для создания буфера в памяти можно воспользоваться командами отладчика F, P, W.

Разобраться с решением примера 1 (п.4).

Не используйте область PSP для своего буфера данных. Область PSP – это первые 256 байт текущего сегмента.

Задача 2

Составить файл на диске с именем data.bin. Файл должен содержать 256 байт, байты идут в порядке возрастания:

00 01 02 03 04 05 ... FF.

Данные для файла data.bin готовит ваша программа с именем data.com. Программа пишется в формате COM с адреса 100h.

Рекомендации. Использовать адресацию к памяти внутри одного сегмента. Составить цикл с количеством повторений 256, внутри которого меняется адрес байта буфера (регистр SI) и содержимое байта буфера (регистр AL).

Ход работы

Отладчик – это специальная программа, предназначенная для ввода и пошагового выполнения исполняемых программ. Существует ряд причин для использования данных программ разными категориями пользователей: школьниками, студентами и профессиональными программистами:

- ♦ С помощью отладчика можно по ходу выполнения программы просматривать исполняемый код, наблюдать за изменениями в регистрах процессора, отслеживать значения и изменения отдельных байтов и целых областей памяти, а также портов ввода/вывода и многое другое.
- ♦ Отладчик с учебной точки зрения – это довольно удобная модель процессора, позволяющая понять роль и работу отдельных его элементов «изнутри», а также функционирование и устройство всей компьютерной системы.
- ♦ Отладчик является замечательным инструментом, дающим возможность изучать исполняемый код своих и чужих программ. Самый лучший способ изучить отдельные команды — написать небольшую тестовую программу в мнемокодах, загрузить ее в отладчик и наблюдать за результатами ее работы в пошаговом режиме. Удивляет то количество информации, которое можно получить, просто наблюдая и анализируя работу машинных команд.

Всем, кто хочет понимать свои действия, любит управлять событиями и глубоко вникать в детали, для того отладчик – незаменимый инструмент.

Для реального режима работы процессора наиболее широко используются такие отладчики, как **DEBUG, AFD** и **Turbo Debug**. Эти средства по своим функциональным возможностям во многом схожи, а основные отличия заключаются в некоторых сервисных возможностях и организации интерфейса с пользователем.

Простейший отладчик DEBUG.exe входит в операционную систему MS DOS и Windows 9∗ в качестве внешней команды. У него полностью отсутствует интерфейсная оболочка, и работа осуществляется посредством командной строки в

текстовом режиме. После его запуска появляется характерное приглашение:

– (дефис) и если набрать вопросительный знак (?), то появляется помощь на русском языке в виде списка синтаксиса основных команд. В данных указаниях мы его рассматривать не будем из-за относительной простоты этого отладчика и наличия подробного описания по его использованию во многих книгах, например [1-3].

Отладчик **Turbo Debug** – поставляется с большинством версий Borland C++ и Pascal. Подобно всем другим отладчикам, он работает в режиме супервизора, беря на себя управление программой, и позволяет по шагам исполнять код программы. Можно потребовать, чтобы Turbo Debug исполнял программу до некоторой точки или до появления определенной ситуации. Можно изменять значения в памяти, временно вставлять новые команды, а также менять значения регистров и флагов. Можно использовать Turbo Debug для ввода или редактирования программ в машинных кодах, если количество команд не слишком велико. Из-за наличия хорошего описания по его применению во многих книгах, например, [4-6], мы на его работе также останавливаться не будем.

Простым и довольно удобным отладчиком, имеющим интерфейсную оболочку, является AFD. Известны две его версии:

• **AFD** (Advanced Full screen Debug);

• **AFDP** (Advanced Full screen Debug Professional). Работа с ними во многом схожа, и мы не будем их различать, ориентируясь на более позднюю и удобную версию – AFDP.

1.1. Описание основных полей интерфейса

Программа AFD, предназначена для отладки выполняемых программ из файлов типа EXE или COM. Однако можно загрузить в память, просмотреть и при необходимости отредактировать любой другой файл.

Основное окно программы AFDP состоит из следующих **полей информации** (рис. 1):

- *состояния процессора,* включающее в себя содержимое пользовательских регистров, указатель команд, содержимое четырех слов стека (Staks),

регистра флагов (Flags) и значение восьми отдель-$_4$ ных флагов **(1)**;

- *командная строка* для ввода команд отладчика **(2);**
- размещение области памяти в виде текста программы в мнемониках Ассемблера **(3);**
- *поле* отображения содержимого дампа памяти *(Memory 1)* по восемь байт в строке с указанием полного адреса первого байта строки **(4);**
- *поле* отображения содержимого дампа памяти Рис.1. Основное окно отладчика AFDP

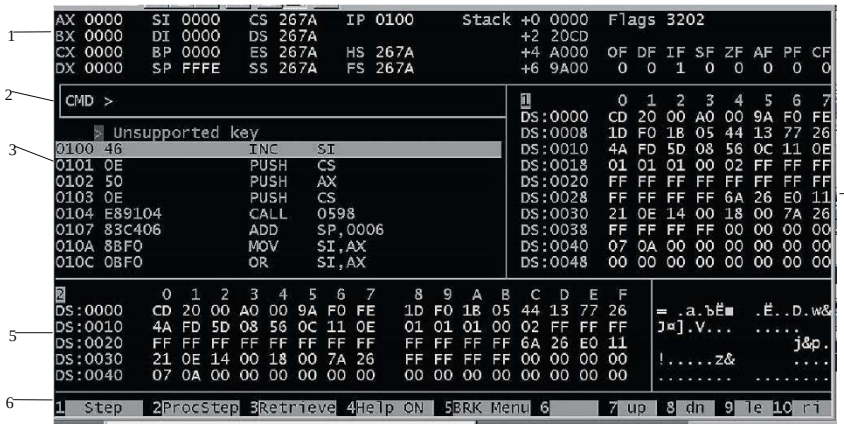
(Memory 2) по 16 байт в строке в шестнадцатеричном, а также в символьном виде с указанием полного адреса первого байта строки **(5);**

- *поле подсказки* по использованию функциональных клавиш F1-F10 **(6).**

Внутри любого поля (1) – (5) можно изменять его содержимое: регистров, памяти, стека и состояние флагов. Для изменения соответствующего поля в него необходимо перевести курсор, который изначально находится в поле ввода команд. Перевод курсора осуществляется функциональными клавишами F7-↑, F8 -↓, F9 - , F10 – →.

Внутри активного поля курсор перемещается с помощью клавиш управления курсором и клавишей табуляции – Tab. Можно изменять любые данные в полях Memory 1 и Memory 2.

Необходимо иметь в виду, что отладчик оперирует только с шестнадцатеричными числами, которые являются компактным представлением двоичных чисел. Допустимыми значениями являются числа от 0 до F (можно писать как строчными, так и заглавными символами). Исключением является правая часть поля 5 символьного представления данных, где допустимы любые ASCII символы. Кроме того, отдельные флаги могут принимать лишь значения 0 или 1. При написании и редактировании программы все адреса, данные и константы записываются только в шестнадцатеричном виде.



1.2. Функциональные клавиши

В поле (6) подсказки приведены данные по использованию функциональных клавиш.

F1 (Step) – *пошаговое* выполнение одной машинной команды. При нажатии на эту клавишу выполняется текущая команда программы, указатель команд переходит на следующую команду и, соответственно, меняется содержимое регистров IP и CS. Вызов подпрограмм, работа циклов и выполнение прерываний обрабатывается по одной команде.

F2 (ProcStep) – *процедурное* выполнение программы. При нажатии на F2 выполняется текущая команда, как и в случае нажатия F1. Однако, если текущей командой является вызов процедуры (CALL), прерывания (INT) или цикла (LOOP), то за

одно нажатие клавиши выполняется целиком процедура, прерывание или весь цикл. Это удобно для достижения различных целей отладки.

F3 (Retrieve) – просмотреть предыдущие выполненные команды. Это довольно удобно для повтора предыдущей команды. **F4 – (Help ON {OFF})** вызвать или убрать контекстную подсказку для текущей команды или просмотреть все доступные команды отладчика листая справочную информацию клавишами

Page Up и Page Dn. При нажатии F4 второе окно памяти (*Memory 2*) заменяется на окно подсказки для текущей команды. **F5 (BRK Menu) –** установка точек прерывания выполнения программы и переход к экрану установки точек прерывания. При этом функциональные клавиши F1, F3, F4, F7, F9, F10 начинают нести другой смысл. Выход из этого режима повторное нажатие **F5**

(CMD line).

F7 (up) – перевод курсора на вышестоящее поле;

F8 (dn) – перевод курсора на поле ниже;

F9 (le) – перевод курсора в левое поле от активного; **F10 (ri) –** перевод курсора в правое поле от активного.

Основные команды отладчика

Команды набираются в поле ввода команд (2) и выполняются нажатием на клавишу Enter. Рассмотрим более подробно работу основных команд. В фигурных скобках указаны не обязательные операнды.

L {/p} {/addr} filename {parameter}

Команда используется для загрузки программы или данных из файла в память.

filename – полное имя файла включая путь к файлу по спецификации DOS; addr – адрес загрузки файла в память является необяза-

тельным параметром и представляет полный адрес либо только смещение относительно сегмента CS. При загрузке выполняемой программы формируется PSP со всеми необходимыми параметрами. В отдельных случаях можно указать адрес памяти, начиная с которого будет загружаться и далее выполняться программа. По умолчанию для ∗.com-программы этот адрес определяется содержимым регистра CS и смещением 100h. Если файл благополучно загружен, то в регистровой паре (BX; CX) находится количество записанных в память байт программы.

/p – параметр для распаковки выполняемых файлов EX-

EPACK.

W { addr, length, filename }

Команда записывает содержимое области памяти в файл. По умолчанию сегментным регистром адреса принимается DS. Длина записывается как количество байт в формате четырехразрядного шестнадцатиричного числа.

{R} Регистр = Значение. Команда служит для установки содержимого регистра или флага, например, AX=11FF; AX=BX; FL=3202. Регистр флагов (FL) определяется как шестнадцатиразрядный. Для установки отдельных флагов используются следующие их названия: OF, DF, IF, SF, ZF, AF, PF и CF, например, CF=1,

ZF=1.

D addr. Команда служит для установки адреса области памяти, которая отображается в поле размещения текста программы.

Сегментный регистр по умолчанию CS.

Mn Адрес/Регистр. Команда определяет адрес памяти, отображаемой в одном из двух полей содержимого памяти (n=1 или n=2). Сегментный регистр указывается в соответствующем поле вывода. Для адресации может использоваться содержимое регистра, например:

M2 DX; M2 ES:125A; M1 DS:SI.

M2 FE00:0 (информация о Bios)

M2 FFFF:5

G {Стартовый адрес} {, Адрес останова}. Команда предназначена для выполнения программы или ее части в автоматическом режиме, начиная со стартового адреса (start addr). По умолчанию используется сегментный регистр CS. Если указан адрес останова (stop addr), то происходит остановка программы при его достижении. Прервать выполнение программы в автоматическом режиме можно, нажимая клавиши Ctrl–Esc. Если при выполнении программы встретилась точка останова break_addr, то автоматическое выполнение программы прекращается. Если стартовый адрес не указан, то программа выполняется с текущего адреса определяемого CS:IP.

QUIT {R{ESIDENT}}. Команда предназначена для завершения работы AFD. Единственный параметр {R} позволяет оставить ее резидентной в памяти. Когда программа находится резидентно в памяти, обращение к ней выполняется нажатием комбинации клавиш Ctrl-Esc.

A {Адрес}. Команда предназначена для перехода в поле программы и позволяет изменять ее текст, набирая команды в мнемониках Ассемблера. Если не указывается адрес, то курсор устанавливается в текущую команду, на которую указывает программный счетчик CS:IP. Нажатие на клавишу Enter вызывает трансляцию набранной или измененной команды. Выход из режима набор/редактирование осуществляется нажатием на клавиши Ctrl– Enter. Раздвижки команд не происходит, т. е. новые команды записываются поверх старых, т.к. каждая команда «привязана» к конкретному месту в памяти. Необходимо также иметь в виду, что разные команды имеют разную длину, поэтому если необходимо зарезервировать место для будущих команд можно воспользоваться однобайтовой командой означающей пустую операцию – NOP (код 90). При этой операции не выполняется никаких действий.

P Адрес, Строка.

Команда записывает в память по адресу первого операнда строку информации. По умолчанию используется сегментный регистр CS. Строку можно задавать в символьном или в цифровом виде, например:

P 1000,”12345678”, P 1000,11 22 33 44 55 66 77 88

По смещению 1000 в оперативной памяти изменяется 8 байт.

F Адрес, Повторение, Строка.

Команда предназначена для записи в память строки информации с определенным коэффициентом повторения. Операнд «Повторение» позволяет указать количество записываемых в память копий строки. По умолчанию при адресации используется сегментный регистр DS. . Например:

♦ F 200,5,1 2 3 4 5

По смещению 200 в память занесется 25 байт:

01 02 03 04 05 01 02 03 04 05 …

♦ F 500,3,“alfa”

По смещению 500 в память заносится 12 байт:

61 6C 66 61 61 6C 66 61 61 6C 66 61

PD Адрес, Длина {, Имя_файла}. Данная команда предназначена для печати машинного кода программы в мнемониках языка Ассемблер. Печать идет на принтер или в указанный файл. По умолчанию используется сегментный регистр CS. Операнд Длина указывает на количество распечатываемых байт, а не инструкций Ассемблера.

PH {/A} Адрес, Длина {, Имя_файла}. Команда аналогична PD и используется для печати машинного кода программы в виде шестнадцатиричного дампа. Параметр /A ограничивает вывод данных ASCII- таблицы диапазоном от 20h до 7Fh. Например, при отработке инструкций:

PD 100,100, progr.cod PH 100,100, progr.hex выводится 256 Байт по адресу CS:100 в файл текущего каталога – progr.cod в дизассемблерном виде и в файл progr.hex – в виде шестнадцатиричного дампа.

? = выражение. Данная команда является встроенным целочисленным калькулятором. Ее удобно использовать для расчета различных констант, адресов и переменных. В выражение справа от знака равенства могут входить как шестнадцатеричные, так и десятичные значения, которые начинаются знаком %. Результатом является шестнадцатеричное или десятичное значение без знака выводимое строкой красного цвета. Например:

? = %256 получим >Result =0100 ? = AA-BB получим >Result =0011 ? = AA*BB получим >Result =7C2E ? %=35*5 - %10 получим >Result =%255.

Порядок выполнения работы

- Составить математическую модель задачи.
- Выбрать и обосновать наиболее рациональный метод решения задачи;
- Разработать алгоритм для решения задачи.
- Написать и отладить программу.

Форма предоставления результата

- Блок – схема.
- Код программы.

Критерии оценки:

«Отлично» - теоретическое содержание курса освоено полностью, без пробелов, умения сформированы, все предусмотренные программой учебные задания выполнены, качество их выполнения оценено высоко.

–«Хорошо» - теоретическое содержание курса освоено полностью, без пробелов, некоторые умения сформированы недостаточно, все предусмотренные программой учебные задания выполнены, некоторые виды заданий выполнены с ошибками.

–«Удовлетворительно» - теоретическое содержание курса освоено частично, но пробелы не носят существенного характера, необходимые умения работы с освоенным материалом в основном сформированы, большинство предусмотренных программой обучения учебных заданий выполнено, некоторые из выполненных заданий содержат ошибки.

«Неудовлетворительно» - теоретическое содержание курса не освоено, необходимые умения не сформированы, выполненные учебные задания содержат грубые ошибки.

Лабораторное занятие №3. Использование потоков. Работа и использование отладчика AFD. Способы задания операндов команды. Адресация к памяти

Цель работы:

- получить навыки работы с основными командами отладчика.

Выполнив работу, Вы будете:

уметь:

- У2. Создавать программу по разработанному алгоритму как отдельный модуль.
- У1.Осуществлять разработку кода программного модуля на языках низкого уровня и высокого уровней;
- У3. Выполнять отладку и тестирование программы на уровне модуля.
- У6. Оформлять документацию на программные средства.
- У8. Применять инструментальные средства отладки программного обеспечения.

Материальное обеспечение:

- Мультимедийные средства хранения, передачи и представления информации.

Задание

Структура машинной команды

Машинная команда представляет собой закодированное по определенным правилам указание микропроцессору на выполнение некоторой операции или действия. Длина машинной команды может составлять целое число байт: 1, 2, 3, 4..., например:

	Команда	*Мнемоника*	*операнд(ы)*
Однобайтовая	50	**PUSH**	AX;
Двухбайтовая	CD 21	**INT**	21;
Трехбайтовая	05 00 10	**ADD**	AX, 1000;
Четырехбайтовая	2B 06 00 10	**SUB**	AX, [1000].

Каждая команда содержит элементы, определяющие: • *код операции*, то есть какое действие необходимо сделать (это обязательный элемент);

• *операнды*, то есть объекты, над которыми необходимо что-то сделать;

• *типы операндов* (обычно задаются неявно исходя из команды или операндов).

Способы задания операндов команды

♦ *Операнды могут задаваться неявно на микропрограммном уровне*. В этом случае команда не содержит явных операндов, и они используются по умолчанию. Например, команды **STC** и **CLC** воздействует на флаг **CF** (устанавливает его в «1» или в «0»).

♦ *Операнд может задаваться непосредственно в самой команде.* Непосредственный операнд может быть только источником. Например,

MOV AX, 1122; **SUB** [BX], 5F.

♦ *Операнд может находиться в одном из* 8/16/32 битных *регистров.* Например: **INC** CH – увеличивает регистр CH на 1; **ADD** BX, AX – суммируется содержимое регистров AX и BX и результат заносится в BX.

♦ *Операнд может находиться в памяти.* Это наиболее сложный и в тоже время наиболее гибкий и чаще всего используемый способ задания операндов. Более подробно способы адресации к памяти рассмотрены в п. 2.2.

♦ *Операндом является порт ввода-вывода.* Источником информации или приемником выступает аккумулятор AL, AX, EAX. Выбор регистра определяется разрядностью порта.

Номер порта задается в регистре DX.

♦ *Операнд находится в стеке.* В этом случае команда содержит один операнд в виде шестнадцатибитного регистра, константы или адреса памяти.

Адресация к памяти

При программировании на языке машинных команд различают следующие основные способы адресации к памяти: – **абсолютная прямая**, например: **MOV [200], AL** записать в ячейку памяти DS:200 содержимое регистра AL;

– **относительная прямая**, например: **JC** 50 – перейти на метку 50, если флаг CF=1. Несмотря на то, что метка указана явно, на самом деле при компиляции в команду входит смещение в формате SHORT (целое однобайтовое со знаком) относительно адреса следующей команды;

– **косвенная адресация.** Различают следующие ее разновидности:

косвенная базовая со смещением, косвенная индексная со смещением, косвенная базовая индексная адресация со смещением и

т.п.

При адресации ячеек памяти передача данных в память и из памяти может происходить с использованием сегментных регистров **SR** (DS, SS, ES, CS, FS, GS), базовых регистров **BR** (BX, BP) и индексных регистров **IR** (SI, DI). Общий вид косвенной адресации к памяти для процессоров в реальном режиме их работы можно записать следующим образом:

SR: [BR+IR+(D8 или D16 или D32)],

где D8, D16; D32 – восьми, шестнадцати или тридцатидвухразрядная константа;

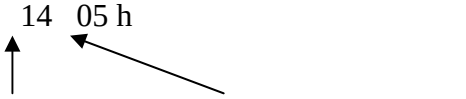
SR – любой из сегментных регистров;

BR – базовый регистр (BP, BX); IR – индексный регистр (SI, DI).

В квадратных скобках указано смещение относительно сегментного регистра SR. Любой из трех компонентов смещения является необязательным, но наличие хотя бы одного из них необходимо. Недопустимо одновременное использование двух базовых или двух индексных регистров.

Сегментный регистр является также необязательным параметром, и если он не указан явно, то действует следующее соглашение: если в качестве базового регистра используется регистр BP, то подразумевается сегментный регистр SS, во всех остальных случаях – регистр DS.

Процессор посредством различных команд адресации к памяти обеспечивает доступ к отдельным байтам, к словам (2 смежным Байтам), к двойным словам (4 Байтам) или к четырем словам памяти (8 Байтам). Рассмотрим, как хранятся числа в памяти, например, целое двухбайтовое число без знака 5125 = 1405h. Понятно, что числа в памяти компьютера хранятся только в двоичном виде. Для удобства человека-пользователя данные числа в отладчике или языке Ассемблер представляются в более компактном шестнадцатеричном виде.



После выполнения следующих машинных команд:

MOV AX, 1405

MOV [1025], AX в оперативной памяти будет следующая картина:

05 14

1025h 1026h

(ячейка) (ячейка)

Числа как бы переворачиваются по адресам. Необходимо всегда помнить, что младшая часть числа всегда хранится в младшем адресе.

Порядок выполнения работы

- Составить математическую модель задачи.
- Выбрать и обосновать наиболее рациональный метод решения задачи;
- Разработать алгоритм для решения задачи.
- Написать и отладить программу.

Форма предоставления результата

- Блок – схема.
- Код программы.

Критерии оценки:

«Отлично» - теоретическое содержание курса освоено полностью, без пробелов, умения сформированы, все предусмотренные программой учебные задания выполнены, качество их выполнения оценено высоко.

–«Хорошо» - теоретическое содержание курса освоено полностью, без пробелов, некоторые умения сформированы недостаточно, все предусмотренные программой учебные задания выполнены, некоторые виды заданий выполнены с ошибками.

–«Удовлетворительно» - теоретическое содержание курса освоено частично, но пробелы не носят существенного характера, необходимые умения работы с освоенным материалом в основном сформированы, большинство предусмотренных программой обучения учебных заданий выполнено, некоторые из выполненных заданий содержат ошибки.

«Неудовлетворительно» - теоретическое содержание курса не освоено, необходимые умения не сформированы, выполненные учебные задания содержат грубые ошибки.

Лабораторное занятие №4. Использование потоков. Работа и использование отладчика AFD. Основные машинные команды

– команды передачи данных
– арифметические команды
– логические команды и команды сдвига
– команды передачи управления
– команды цикла.

Цель работы:

- получить навыки работы с основными командами отладчика.

Выполнив работу, Вы будете:

уметь:

- У2. Создавать программу по разработанному алгоритму как отдельный модуль.
- У1.Осуществлять разработку кода программного модуля на языках низкого уровня и высокого уровней;
- У3. Выполнять отладку и тестирование программы на уровне модуля.
- У6. Оформлять документацию на программные средства.
- У8. Применять инструментальные средства отладки программного обеспечения.

Материальное обеспечение:

- Мультимедийные средства хранения, передачи и представления информации.

Задание

ОСНОВНЫЕ МАШИННЫЕ КОМАНДЫ

В данном разделе приведен необходимый минимальный набор и назначение простейших команд. Для удобства практического применения и отражения специфики команды их принято делить на группы: передачи данных, арифметические, логические, цепочечные и т.д. Команда представляет собой некое число в двоичном виде. Человеку довольно трудно оперировать с последовательностями нулей и единиц и он использует их символьные модели – мнемоники.

С точки зрения процессора, нет принципиальной разницы между данными и командами. Данные и машинные команды находятся в одном пространстве памяти в виде последовательности нулей и единиц. Процессор, исполняя содержимое некоторых последовательных ячеек памяти, всегда пытается трактовать его как коды машинной команды, а если это не так, то происходит аварийное завершение программы, содержащей некорректный фрагмент. Надо четко себе представлять, где находятся данные, а где программа.

Смысл многих команд и алгоритм их работы далеко неочевиден, как может показаться на первый взгляд. Некоторые из них имеют свойства, которыми можно воспользоваться в ситуациях, когда команда применяется не по прямому назначению. Тех, кого заинтересует более глубокое понимание работы приведенных ниже машинных команд и алгоритм функционирования других команд можно рекомендовать учебники по ассемблеру [3 - 6].

1. Команды передачи данных

MOV <операнд назначение>, <операнд источник> Это основная команда пересылки данных. Она реализует самые разнообразные варианты пересылки. XCHG < операнд 1>, < операнд 2>

Переставляет или меняет содержимое операндов между собой. Операнды обоих этих команд должны быть согласованы по разрядности. Есть ряд особенностей применения этих команд.

Команда не может передавать данные между двумя адресами памяти;

Нельзя загрузить в сегментный регистр константу или данное из памяти;

Нельзя переслать данное из одного сегментного регистра в другой.

Команда PUSH <источник> - сохраняет значение слова в стеке для последующего использования. Регистр SP указывает на текущее слово в вершине стека (указатель стека). Команда PUSH автоматически уменьшает значение в регистре SP на 2 и передает слово из указанного операнда в новую вершину стека по адресу

SS:SP.

Например:

```
        PUSH    AX
        PUSH    CX
  PUSH        CS
  PUSH        1234 и т. п.
```

Команда POP <назначение> передает слово, помещенное ранее в стек, в указанный операнд. Регистр SP указывает на текущее слово в вершине стека. Команда POP извлекает слово из стека и увеличивает значение в регистре SP на 2. Например:

```
 PUSH DS  POP   ES
```

При этом SP увеличивается на 2. После выполнения этих команд сегментный регистр ES примет значение DS.

2. Арифметические команды

ADD < операнд1 >, < операнд2 > Команда сложения со следующим принципом действия:

операнд1= операнд1+операнд2 ADC < операнд1 >, < операнд2 >

Команда сложения с учетом флага переноса CF: операнд1= операнд1+операнд2+значение CF

Данные команды воздействуют на флаги AF, CF, OF, PF, SF и ZF. INC <операнд >

Операция инкремента, т.е. увеличения значения операнда на 1.

Команда воздействует на флаги AF, OF, PF, SF и ZF.

SUB < операнд1 >, < операнд2 >

Команда вычитания; ее принцип действия: операнд1= операнд1 – операнд2

Команда воздействует на флаги AF, CF, OF, PF, SF и ZF. SBB < операнд1 >, < операнд2 >

Вычитание с учетом заема (флага CF) операнд1= операнд1-операнд2 – значение CF

DEC <назначение>

Операция декремента, т.е. уменьшает содержимое операнда на 1.

Команда воздействует на флаги AF, OF, PF, SF и ZF.

MUL <сомножитель1 >

Беззнаковое умножение имеет один операнд (сомножитель 1). Второй операнд (сомножитель 2) задается неявно. Так как в общем случае результат умножения больше, чем любой из сомножителей, то его размер и местоположение должны быть тоже определены однозначно. Варианты размеров сомножителей и размещение второго операнда и результата приведены в табл.1.

При операции MUL левый единичный бит рассматривается как бит данных, а не как знаковый бит. Команда воздействует на флаги CF и OF. DIV <делитель >

Выполняет целочисленное деление чисел без знака. Местоположение делимого фиксировано, как и в команде умножения, зависит от размера делителя. Результатом команды деления являются значения частного и остатка. Варианты местоположения и размеров операндов операции деления показаны в табл.2.

Таблица 1

Расположение операндов и результата при умножении

Сомножитель 1	Сомножитель 2	Результат
Байт (8 бит)	AL	16 бит в AX: AL – младшая часть

		результата; AH – старшая часть результата
Слово (16 бит)	AX	32 бит в паре DX;AX: причем AX – младшая часть; DX – старшая часть
Двойное слово (32 бита)	EAX	64 бит в паре EDX;EAX: EAX – младшая часть, EDX – старшая часть

Таблица 2

Расположение операндов и результата при делении

Делимое	Делитель	Частное	Остаток
AX	Однобайтовый регистр или ячейка памяти размером 1 Б	AL	AH
DX – старшая часть; AX – младшая часть	Двухбайтовый регистр или ячейка памяти размером 2 Б	AX	DX
EDX – старшая часть; EAX – младшая часть	Двойное слово 4 Б, регистр или ячейка памяти	EAX	EDX

Левый единичный бит рассматривается как бит данных, а не как знаковый бит. После выполнения команды деления содержимое флагов неопределенно, но возможно возникновения немаскируемого аппаратного прерывания с номером 0, называемого «деление на ноль».

3.Логические операторы и команды сдвига

Три логических операции – AND (логическое И), OR (логическое ИЛИ), XOR (исключающее ИЛИ) дают возможность манипулировать отдельными битами в двоичных величинах. Можно устанавливать или сбрасывать единичные биты без влияния на другие, выделять из байта или слова один или группу битов и другие операции. Эти команды имеют два операнда, первый из которых является источником и приемником.

Команда AND часто применяется для маскирования (выделения) битов в байтах и словах. Например, выделить младший полубайт, находящийся в регистре AH:

AND AH, 0F или в двоичном виде: AND AH, 00001111b

Четыре старшие двоичные цифры будут установлены в «0» независимо оттого, что там было.

Логическая команда OR используется для изменений отдельных битов, не влияя при этом, на другие биты. Например, установить в регистре AX первый и 15 бит в «1»: OR AX, 8001 или в двоичном виде

OR AX, 1000000000000001b

Логическая команда XOR является удобным инструментом для переключения отдельных битов из состояния «off» в «on» и наоборот. Если оба бита имеют одинаковое значение, то результат операции XOR равен 0, в противном случае – 1. Это используется для наиболее быстрого обнуления данных регистра, например:

XOR SI, SI XOR AL, AL

Поскольку маска XOR изменяет на противоположный каждый бит начального значения, то повторное применение той же маски к результату восстанавливает первоначальное значение битов. Это широко используется в коммуникационных сетях, шифровании данных и программ.

Другой важной операцией над двоичными значениями является сдвиг битов влево и вправо и ротации через флаг переноса CF. Команды делятся на четыре группы:

♦ Простые сдвиги SHL, SHR;

♦ Арифметические сдвиги SAL, SAR; ♦ Простые ротации ROL, ROR; ♦ Ротации через флаг CF – RCR, RCL.

Общий вид этих команд следующий: КОМАНДА <операнд >,<колич. сдвигаемых бит>.

Выполняют сдвиг всех битов операнда влево или вправо. Сдвиги могут выполняться для однобайтового или двухбайтового операнда, находящегося в регистре или в памяти. Сдвиг на один бит кодируется в мнемонике константой – «1». Сдвиг более чем на один бит требует указания регистра CL, который содержит счетчик сдвига.

Команды SAL и SHL сдвигают биты влево определенное число раз, и правый, освобождающийся бит, заполняют нулевым значением. Команда SHR сдвигает биты вправо определенное число раз, и левый, освобождающийся бит, заполняет нулевым значением. Команда SAR сдвигает биты вправо определенное число раз, и левый, освобождающийся бит, всегда заполняется исходным значением знакового бита. Особенность работы циклических сдвигов ROL и ROR в том, что «уходящий» бит не теряется, а возвращается, но с другого конца. Во всех случаях значения битов, выдвигаемых за разрядную сетку, помещаются во флаг переноса CF.

4. Команды передачи управления

Команда безусловного перехода: JMP <адрес>

Она выполняет переход по указанному адресу при любых условиях.

Команды условного перехода передают управление на основе анализа некоторых условий или данных, имеют следующий вид:

Jxx <адрес>

Осуществляется переход по указанному адресу при выполнении условия заданного мнемоникой команды. Если заданное условие не выполняется, переход не осуществляется, а выполняется команда, следующая по порядку. Перед командой условного перехода обычно ставится команда: CMP <операнд1 >, <операнд2>

Она сравнивает два операнда. По своему действию эта команда аналогична команде SUB, но в отличие от нее не меняет содержимого операндов, но также воздействует на регистр флагов.

Все команды условного перехода действуют в зависимости от содержимого одного или нескольких флагов и перечислены вместе в табл.3. Например: Сравнить два байта по адресу 1000h и

1001h и если значения в них одинаковые, то перейти на метку

150h.

```
MOV     AL, [1000]
CMP     [1001], AL
JE      150  ; перейти на метку 150,
; если содержимое AL равно байту памяти
; по адресу ds:1001
```

Отличие команд «выше – ниже» и «больше – меньше» заключается в том, что первые команды выражают отношение между операндами без знака, а вторые со знаком.

Таблица 3

Команды условного перехода

Инструкция	Переход если	Анализируемые флаги
JA	Выше	(cf=0) и (zf=0)
JAE	Выше или равно	(cf=0)
JB	Ниже	(cf=1)
JBE	Ниже или равно	(cf=1) или (zf=1)
JE	Равно	(zf=1)
JNE	Не равно	(zf=0)
JG	Больше	(sf=of) и (zf=0)
JGE	Больше или равно	(sf=of)
JL	Меньше	(sf<>of)
JLE	Меньше или равно	(sf<>of) или (zf=1)

Порядок выполнения работы

- Составить математическую модель задачи.
- Выбрать и обосновать наиболее рациональный метод решения задачи;
- Разработать алгоритм для решения задачи.
- Написать и отладить программу.

Форма предоставления результата

- Блок – схема.
- Код программы.

Критерии оценки:

«Отлично» - теоретическое содержание курса освоено полностью, без пробелов, умения сформированы, все предусмотренные программой учебные задания выполнены, качество их выполнения оценено высоко.

–«Хорошо» - теоретическое содержание курса освоено полностью, без пробелов, некоторые умения сформированы недостаточно, все предусмотренные программой учебные задания выполнены, некоторые виды заданий выполнены с ошибками.

–«Удовлетворительно» - теоретическое содержание курса освоено частично, но пробелы не носят существенного характера, необходимые умения работы с освоенным материалом в основном сформированы, большинство предусмотренных программой обучения учебных заданий выполнено, некоторые из выполненных заданий содержат ошибки.

«Неудовлетворительно» - теоретическое содержание курса не освоено, необходимые умения не сформированы, выполненные учебные задания содержат грубые ошибки.

Лабораторное занятие №5. Использование потоков. Структура программы на языке Assembler

Цель работы:

- приобретение навыков выполнения операций в различных системах счисления.

Выполнив работу, Вы будете:

уметь:

- У2. Создавать программу по разработанному алгоритму как отдельный модуль.
- У1.Осуществлять разработку кода программного модуля на языках низкого уровня и высокого уровней;
- У3. Выполнять отладку и тестирование программы на уровне модуля.

- У6. Оформлять документацию на программные средства.
- У8. Применять инструментальные средства отладки программного обеспечения.

Материальное обеспечение:

- Мультимедийные средства хранения, передачи и представления информации.

Задание

Порядок выполнения работы

- Составить математическую модель задачи.
- Выбрать и обосновать наиболее рациональный метод решения задачи;
- Разработать алгоритм для решения задачи.
- Написать и отладить программу.

Форма предоставления результата

- Блок – схема.
- Код программы.

Критерии оценки:

«Отлично» - теоретическое содержание курса освоено полностью, без пробелов, умения сформированы, все предусмотренные программой учебные задания выполнены, качество их выполнения оценено высоко.

–«Хорошо» - теоретическое содержание курса освоено полностью, без пробелов, некоторые умения сформированы недостаточно, все предусмотренные программой учебные задания выполнены, некоторые виды заданий выполнены с ошибками.

–«Удовлетворительно» - теоретическое содержание курса освоено частично, но пробелы не носят существенного характера, необходимые умения работы с освоенным материалом в основном сформированы, большинство предусмотренных программой обучения учебных заданий выполнено, некоторые из выполненных заданий содержат ошибки.

«Неудовлетворительно» - теоретическое содержание курса не освоено, необходимые умения не сформированы, выполненные учебные задания содержат грубые ошибки.

Лабораторное занятие №6. Обмен данными. Ввод и вывод данных в Assembler

Цель работы:

- приобретение навыков выполнения операций в различных системах счисления.

Выполнив работу, Вы будете:

уметь:

- У2. Создавать программу по разработанному алгоритму как отдельный модуль.
- У1.Осуществлять разработку кода программного модуля на языках низкого уровня и высокого уровней;
- У3. Выполнять отладку и тестирование программы на уровне модуля.
- У6. Оформлять документацию на программные средства.
- У8. Применять инструментальные средства отладки программного обеспечения.

Материальное обеспечение:

- Мультимедийные средства хранения, передачи и представления информации.

Задание

Анализ таблицы кодов ASCII показывает следующее.

В системе кодировки ASCII для любой цифры от 0 до 9 справедливо соотношение

код (цифра) - код (‘0’) = цифра

Так как код символа 0 (‘0’) равен 30h, то ASCII-код любой цифры от 0 до 9 отличается от соответствующего двоичного представления числа на 30h.
Поэтому для преобразования ASCII-кода символа (‘0’..’9’) в число, следует из кода символа вычесть 30h.

При вводе с клавиатуры, например, цифры 5 недостаточно воспользоваться функцией 1h прерывания 21h, так как в регистре al в результате будет находиться код символа ’5’, а не число 5.
Для получения числа 5 необходимо еще вычесть 30h из полученного кода:

```
mov ah, 01h
int  21h     ; в al код символа ‘5’
sub  al,30h   ; теперь в al число 5
```

ЗАДАЧА 1

Пусть в сегменте данных под символическим именем N хранится беззнаковое десятичное число (от 0 до 255). Необходимо записать по адресу KOD цифры (как символы) из десятичной записи числа.

РЕШЕНИЕ

Пусть N=abc , где a, b, c - десятичные цифры числа N.
Для получения правой цифры c надо взять остаток от деления N на 10. Неполное частное от деления - это число ab, ели его разделить на 10, то неполное частное даст цифру a, а остаток - цифру b.
Чтобы получить эти цифры как символы (их затем можно будет вывести на экран), следует к цифре прибавить код ‘0’ (30h).

```
N  db  ?
KOD  db 3 dup(?)
. . .
mov bl,10
mov al,N
mov ah,0          ; расширение N в ax до слова (беззнаковое)
div bl            ; ah=c, al=ab
add ah,30h
mov KOD+2,ah  ; записали последнюю цифру
mov ah,0          ; al =ab, расширение ab до слова (беззнаковое)
div bl            ; ah=b, al=a
add ax,’00’       ; ah=b+’0’, al=a+’0’
mov KOD+1,ah   ; записали среднюю цифру
mov KOD,al        ; записали первую цифру
```

ЗАДАЧА 2

Ввести десятичное число и записать его в сегмент данных под символическим именем N (размером в один байт).

РЕШЕНИЕ

По очереди вводим цифры и формируем число по схеме Горнера.

Пусть уже введены первые цифры числа, например, 2 и 1, и по ним сформировано число 21. Пусть следующей введена цифра 3. Тогда умножаем предыдущее число на 10 и прибавляем к нему новую цифру: 21*10+3=213. И так для каждой новой цифры.

Замечание. Будем предполагать, что пользователь вводит число в диапазоне 0-255 корректно и ввод завершается нажатием клавиши Enter.

```
            N db ?
               . . .
            ah,01h
            int 21h ; в al - первый символ
            sub al,30h  ; теперь первая цифра
            mov ah,0  ; расширение до слова
            mov bx,10
            mov cx,ax ; в cx - первая цифра
     Loop:  ah,01h
            int 21h ;   в al следующий символ
            cmp al,0dh  ; сравнение с символом Enter
            je End      ; конец ввода
            sub al,30h  ; в al - следующая цифра
            cbw         ; расширение до слова
            xchg ax,cx  ; теперь в ax - предыдущее число, в cx - следующая
            mul bx      ; ax*10
            add cx,ax   ; cx=ax*10+cx
            jmp Loop    ; продолжение ввода
     End :  mov N,cx
            . . .
```

ЗАДАЧА 3

Ввести 16-ричное число и его же вывести.

РЕШЕНИЕ

```
.MODEL small
.STACK 64
.DATA
    org 100h
    tabl_ascii  db  '0123456789abcdef'

    org 130h
    db  0,1,2,3,4,5,6,7,8,9
    org 41h
    db  0ah,0bh, 0ch, 0dh, 0eh, 0fh


    org 150h
    x_ascii     db 20h dup(?)
    nl db 0dh, 0ah, "$"

    t1          db 0dh,0ah,"vvedite chislo i najmite enter"
```

```
                db 0dh, 0ah, "$"
        t2      db 0dh,0ah,"Vi vveli chislo",0dh,0ah
               db 0dh, 0ah, "$"

.CODE
   ;Главная процедура
g_k proc

        mov ax,@data
        mov ds, ax
        mov es, ax

d:      lea dx, t1
        mov ah,09h
        int 21h

        lea di, x_ascii
        call ink
        lea dx, t2
        mov ah,09h
        int 21h

       lea di, x_ascii
        call dis

g_k endp
        ink proc
        ;Процедура ввода числа
                xor             cx,cx
            l1:
                mov              ah,1
                int                21h
                stosb
                inc             cx
                cmp              al,0dh
                jnz                l1
                dec              cx
                ret
          ink endp

        dis proc
        ;Процедура вывода на экран числа
           lea dx, nl
           mov ah,09h
           int 21h
           r1: mov dl,[di]
           mov ah,2
           int 21h
           inc di
           loop r1
           ret
           dis endp
```

end g_k

ЗАДАЧА 4

Перевод числа из 8-ой в 2-ую систему счисления. При запуске программы вводится число в 8-ой системе счисления, затем программа переводит это число в бинарную систему счисления и выводит.

РЕШЕНИЕ

```
.model small
.386

.data
var db 5 dup(?)
msg1 db 0Ah,0Dh,'input (oct): $'
msg2 db 0Ah,0Dh,'output (bin): $'

.stack
db 255 dup(?)

.code
start:
mov ax,@data
mov ds,ax
mov es,ax

mov ah,09h
lea dx,msg1
int 21h

lea di,var
xor cx,cx
input:
mov ah,01h
int 21h

cmp al,0Dh
je ready

cmp al,'7'
jbe skip

mov ah,02h
mov dl,08h ;bin
int 21h

mov ah,02h
mov dl,20h ;oct
int 21h
```

```
41
42 mov ah,02h
43 mov dl,08h ;hex
44 int 21h
45
46 jmp input
47
48 skip:
49 and al,0Fh
50
51 stosb
52
53 inc cx
54
55 cmp cx,5
56 je ready
57 jmp input
58
59 ready:
60 or cx,cx
61 je exit
62
63 push cx
64
65 lea si,var
66 xor di,di
67 xor ax,ax
68
69 lodsb
70 mov di,ax
71
72 dec cx
73
74 mov bx,8
75 collect:
76 or cx,cx
77 je two
78
79 mov ax,di
80
81 mul bx
82
83 mov di,ax
84
85 xor ah,ah
86 lodsb
87
88 add di,ax
89
```

```
90  dec cx
91  jmp collect
92
93  two:
94  pop cx
95
96  mov ax,3
97
98  mul cx
99
100 mov dx,ax
101
102 mov cx,16
103 sub cx,ax
104
105 mov bx,di
106 shl bx,cl
107
108 mov cx,dx
109
110 mov ah,09h
111 lea dx,msg2
112 int 21h
113 bit:
114 shl bx,1
115 jc one
116
117 zero:
118 mov ah,02h
119 mov dl,'0'
120 int 21h
121
122 jmp good
123
124 one:
125 mov ah,02h
126 mov dl,'1'
127 int 21h
128
129 good:
130 loop bit
131
132 exit:
133 mov ah,02h
134 mov dl,0Ah
135 int 21h
136
137 mov ah,4Ch
138 mov al,00h
```

```
139 int 21h
140 end start
```

Порядок выполнения работы

- Составить математическую модель задачи.
- Выбрать и обосновать наиболее рациональный метод решения задачи;
- Разработать алгоритм для решения задачи.
- Написать и отладить программу.

Форма предоставления результата

- Блок – схема.
- Код программы.

Критерии оценки:

«Отлично» - теоретическое содержание курса освоено полностью, без пробелов, умения сформированы, все предусмотренные программой учебные задания выполнены, качество их выполнения оценено высоко.

–«Хорошо» - теоретическое содержание курса освоено полностью, без пробелов, некоторые умения сформированы недостаточно, все предусмотренные программой учебные задания выполнены, некоторые виды заданий выполнены с ошибками.

–«Удовлетворительно» - теоретическое содержание курса освоено частично, но пробелы не носят существенного характера, необходимые умения работы с освоенным материалом в основном сформированы, большинство предусмотренных программой обучения учебных заданий выполнено, некоторые из выполненных заданий содержат ошибки.

«Неудовлетворительно» - теоретическое содержание курса не освоено, необходимые умения не сформированы, выполненные учебные задания содержат грубые ошибки.

Лабораторное занятие №7. Обмен данными. Подпрограммы в Assembler

Цель работы:

- Получить навыки создания процедур с использованием подпрограмм.

Выполнив работу, Вы будете:

уметь:

- У2. Создавать программу по разработанному алгоритму как отдельный модуль.
- У1.Осуществлять разработку кода программного модуля на языках низкого уровня и высокого уровней;
- У3. Выполнять отладку и тестирование программы на уровне модуля.
- У6. Оформлять документацию на программные средства.
- У8. Применять инструментальные средства отладки программного обеспечения.

Материальное обеспечение:

- Мультимедийные средства хранения, передачи и представления информации.

Задание

Процедуры могут получать или не получать параметры из вызывающей процедуры и могут возвращать или не возвращать результаты. Процедуры, которые что-либо возвращают, называют функциями (Паскаль).

Различают два основных механизма передачи параметров процедуре:

- по значению;

- по ссылке.

Пусть procedure - имя процедуры, а value - имя некоторой области памяти (переменная). Тогда вызов процедуры с передачей параметра по значению выглядит следующим образом:

```
mov  AX, word ptr value ; процедура получает копию
call procedure; входного параметра
```

Вызов с передачей параметра по ссылке будет выглядеть так:

```
mov  AX, offset value ; процедура получает адрес
call procedure ; входного или выходного параметра
```

Для обмена информацией между вызывающим кодом и процедурой могут использоваться такие каналы:

- регистры;
- стек;
- глобальные переменные;
- поток кода;
- блок параметров.

Передача параметров в регистрах

Если число параметров и сами параметры невелики, лучшим метсом для их передачи являются регистры. Примерами могут служить вызовы функций DOS и BIOS. Языки высокого уровня обычно используют регистр AX (EAX) для размещения результата, который возвращается функцией.

Передача параметров через стек

Параметры помещаются в стек перед вызовом процедуры. Именно такой метод используют языки высокого уровня (C, Pascal). Для чтения параметров из стека применяют не команду POP, а регистр BP, в который помещают адрес вершины стека после входа в процедуру:

```
push  параметр1; поместить параметр в стек
push  параметр2
call  procedure
add   SP,  4 ; освободить стек от параметров
.....
procedure  PROC  near
push  BP
mov   BP,  SP
 (команды, которые могут использовать стек)
mov   AX,  [BP+4] ; считать параметр2. Его адрес [BP+4], потому что ; при выполнении команды call в стек поместили адрес возврата (2 байта для
; процедуры типа NEAR), а потом еще и BP - 2 байта.
mov   BX,  [BP+6]      ; считать параметр1.
 (остальные команды)
pop   BP
ret
procedure  ENDP
```

При использовании стека для передачи параметров процедуры возникает два вопроса: кто должен удалять параметры из стека (процедура или вызывающий код) и в каком порядке помещать параметры в стек. Все возможные варианты ответа на этот вопрос имеют свои "за" и "против". Так, например, если стек освобождает процедура (команда ret число_байтов), то код программы получается меньшим, а если за освобождение стека от параметров отвечает вызывающая функция, то становится возможным последовательными командами CALL вызвать несколько функций с одними и теми же параметрами. Первый способ - более строгий (Pascal), второй - дает больше возможностей для оптимизации (C). Кроме того, в языке C параметры помещают в стек в обратном порядке (справа налево), так что становятся возможными функции с изменяемым числом параметров.

Передача параметров в потоке кода

Передаваемые данные размещаются прямо в коде программы, сразу после команды call, например (так реализовна процедура print в одной из стандартных библиотек процедур для ассемблера UCRLIB):

```
call print
DB  "This ASCIZ-line will be printed", 0
(следующая команда)
```

Чтобы прочитать параметр, процедура должна использовать его адрес, который автоматически передается в стеке как адрес возврата из процедуры. При этом адрес возврата должен быть изменен на первый байт после конца блока данных перед выполнением команды RET.

Передача параметров в потоке кода (так же как и передача параметров в стеке в обратном порядке) позволяет передавать различное число параметров или один параметр различной длины.

Передача параметров в блоке параметров

Блок параметров - участок памяти, содержащий параметры и располагающийся обычно в сегменте данных. Процедура получает адрес начала такого блока. Таким методом реализованы многие функции DOS и BIOS (поиск файла - использует блок параметров DTA, загрузка и исполнение программы - используется блок параметров EPB).

Локальные переменные

Наиболее распространенный способ хранения локальных переменных - стек. Принято располагать локальные переменные сразу после сохраненного значения регистра BP, так что на них можно ссылаться как [BP-2], [BP-4], [BP-6] и т.д.

Рассмотрим процедуру examp, в которой используются параметры x, y, z и локальные переменные a, b, c (все эти величины предполагаются 2-х байтовыми):

```
examp    proc   near
x  equ [BP+8] ; параметры процедуры
y equ  [BP+6]
z equ [BP+4]
a equ  [BP- 2]  ; локальные переменные
b  equ [BP- 4]
cequ  [BP- 6]
pushBP ; сохранить BP
mov BP,  SP   ; установить BP для себя
sub  SP,  6 ; зарезервировать 6 байт для локальных переменных
 (тело процедуры)
mov  SP,  BP   ; выбросить из стека локальные переменные
pop BP  ; восстановить BP вызвавшей процедуры
ret 6 ; возврат с удалением трех параметров из стека
examp  endp
```

При вызове процедуры examp стек будет выглядеть следующим образом:

```
 [BP+8]  =   x
 [BP+6]  =   y
 [BP+4]  =   z
 [BP+2]  =   IP
BP®     [BP+0]  =   BP
 [BP- 2]  =   a
 [BP- 4]  =   b
SP®      [BP- 6]  =   c
```

В связи с тем, что последовательности команд

```
push      BP
```

```
mov     BP, SP
sub     SP, 6
и
mov     SP, BP
pop     BP
```

используются очень часто при построении процедур, были введены специальные команды ENTER и LEAVE, которые их заменяют. С использованием этих команд процедура examp будет выглядеть следующим образом:

```
examp   proc  near
x       equ     [BP+8]
y       equ     [BP+6]
z       equ     [BP+4]
a       equ     [BP- 2]
b       equ     [BP- 4]
c       equ     [BP- 6]
enter   6, 0                ; второй параметр - уровень вложенности процедуры
 (тело процедуры)
leave
ret     6
examp   endp
```

Активизационной записью процедуры называется следующая информация, размещаемая в стеке: параметры процедуры, IP (адрес возврата), BP (предыдущее значение).

Стековый кадр - область стека, отведенная для активизационной записи и локальных переменных процедуры.

Порядок выполнения работы

- Составить математическую модель задачи.
- Выбрать и обосновать наиболее рациональный метод решения задачи;
- Разработать алгоритм для решения задачи.
- Написать и отладить программу.

Форма предоставления результата

- Блок – схема.
- Код программы.

Критерии оценки:

«Отлично» - теоретическое содержание курса освоено полностью, без пробелов, умения сформированы, все предусмотренные программой учебные задания выполнены, качество их выполнения оценено высоко.

–«Хорошо» - теоретическое содержание курса освоено полностью, без пробелов, некоторые умения сформированы недостаточно, все предусмотренные программой учебные задания выполнены, некоторые виды заданий выполнены с ошибками.

–«Удовлетворительно» - теоретическое содержание курса освоено частично, но пробелы не носят существенного характера, необходимые умения работы с освоенным материалом в основном сформированы, большинство предусмотренных программой обучения учебных заданий выполнено, некоторые из выполненных заданий содержат ошибки.

«Неудовлетворительно» - теоретическое содержание курса не освоено, необходимые умения не сформированы, выполненные учебные задания содержат грубые ошибки.

Лабораторное занятие №8. Обмен данными. Макросы в Assembler

Цель работы:

- изучить возможности оптимизации программы, написанной на языке Assembler.

Выполнив работу, Вы будете:

уметь:

- У2. Создавать программу по разработанному алгоритму как отдельный модуль.
- У1.Осуществлять разработку кода программного модуля на языках низкого уровня и высокого уровней;
- У3. Выполнять отладку и тестирование программы на уровне модуля.
- У6. Оформлять документацию на программные средства.
- У8. Применять инструментальные средства отладки программного обеспечения.

Материальное обеспечение:

- Мультимедийные средства хранения, передачи и представления информации.

Задание

Альтернативы секции .const

Главной альтернативой секции .const является простое объявление символьной константы.

```
CONST_VALUE = 678h
.data
Dd CONST_VALUE
.code
...more code...
Mov edi, CONST_VALUE
Xor eax, CONST_VALUE
```

Вторая не менее важная альтернатива – это определение "макросимвола" (я это сам придумал, а, по-научному, абсолютный символ или "прозвище"). Он задаётся через директиву equ.

примеры

```
CONST1 equ 0123h
CONST2 equ 14d*15h
CONST3 equ "slovo"
CONST4 equ 56-45
CONST5 equ (offset metka1)
CONST6 equ (offset metka2+offset metka3)
CONST7 equ (CONST2+10b)
CONST8 equ CONST7/2
CONST9 equ (offset metka1-offse metka5)
CONST10 equ (offset metka4+CONST4)
CONST11 equ add edx,edi
```

... и так далее

Директива equ используется в там же, где определяются символьные константы. Это ещё не все возможности этой директивы. Можно дать прозвище некоторой команде, например:

Command1 equ mov eax, esi.

После этого при каждом упоминании command1 будет подразумеваться команда move eax, esi.

Пример

```
Mov eax, CONST1
Add edi, CONST2
Xor ebp, CONST7
CONST11
Sub edx, CONST8
```

Объявлять equ надо в начале файла там же где объявляются структуры и константы.

Макросы.

Макрос - это набор команд. С помощью equ можно создавать "прозвище" только для одной команды, а с помощью макросов можно создавать "прозвище" для нескольких команд. Для создания макроса надо использовать директивы macro и соответственно endm.

```
Firstmacro macro
                Sub ebp, esp
                Mov eax, ebp
              Endm
```

Теперь если компилятор встречает слово Firstmacro, то он автоматически заменяет его на тело макроса. Также макросу можно передавать параметры.

```
Secondmacro macro param1, param2
                        Add edi, param1
                        Sub esi, param2
                endm
.code    ;  использование
.........
Secondmacro 55h,edx
```

У макросов очень много возможностей.

Включаемые файлы.

Иногда надоедает (особенно при создании оконных приложений) объявлять в каждой программе одни и те же структуры и константы. Хотелось бы один раз их задать, а потом их использовать в каждой своей программе как в Си или Паскале. Для этого предназначены включаемые файлы *.inc. В них можно задать структуры, константы, макросы. Для включения такого файла надо использовать директиву include. После этой директивы надо указать путь к включаемому файлу абсолютный или относительный. При написании такого файла просто думайте, что вы находитесь после директивы .model и до .data.

Пример:

Файл sample.asm

```
;=======[CUT HERE]========
.386
.model  flat, stdcall

include sample.inc

extrn MessageBoxA:PROC

.data

Msg       db "First ASSEMBLER program",0h
Ttl       db 'Hello, World!!!!',0h
RCTNGL RECT ?
.code

start:
  call MessageBoxA,0,offset Msg,offset Ttl,MB_OKCANCEL
  call ExitProcess, 0
end    start
```

```
;=====[CUT HERE]==========
```

Файл sample.inc

```
;=======[CUT HERE]========
```

```
extrn ExitProcess:PROC

UINT    EQU <dd>  ; 32 bits for WIN32

RECT    struc
        rcLeft          UINT ?
        rcTop           UINT ?
        rcRight         UINT ?
        rcBottom        UINT ?
RECT    ends

MB_OK                   = 0000H
MB_OKCANCEL             = 0001H
MB_ABORTRETRYIGNORE     = 0002H
MB_YESNOCANCEL          = 0003H
MB_YESNO                = 0004H
MB_RETRYCANCEL          = 0005H
;=====[CUT HERE]===========
```

Директива include заменяется на всё содержимое включаемого файла.

Упрощённый вызов API функций в TASM

Если в директиве .model укажем модель вызова функций, то вызывать API будет намного проще

Call <функция>, <параметр1>,<параметр2>,<параметр3>

Теперь первая программа будет иметь такой вид:

```
  .386
.model  flat, stdcall

extrn MessageBoxA:PROC
extrn ExitProcess:PROC
.data

Msg     db "First ASSEMBLER program",0h
Ttl     db 'Hello, World!!!!',0h


.code

start:
  call MessageBoxA,0,offset Msg,offset Ttl,0
  call ExitProcess, 0
end   start
```

Команда call func, param1,pram2, paramn при компиляции автоматически преобразуется в

```
  Push paramn
  Push param2
  Push param1
  Call func
```

Порядок выполнения работы

- Составить математическую модель задачи.
- Выбрать и обосновать наиболее рациональный метод решения задачи;
- Разработать алгоритм для решения задачи.

– Написать и отладить программу.

Форма предоставления результата

– Блок – схема.
– Код программы.

Критерии оценки:

«Отлично» - теоретическое содержание курса освоено полностью, без пробелов, умения сформированы, все предусмотренные программой учебные задания выполнены, качество их выполнения оценено высоко.

–«Хорошо» - теоретическое содержание курса освоено полностью, без пробелов, некоторые умения сформированы недостаточно, все предусмотренные программой учебные задания выполнены, некоторые виды заданий выполнены с ошибками.

–«Удовлетворительно» - теоретическое содержание курса освоено частично, но пробелы не носят существенного характера, необходимые умения работы с освоенным материалом в основном сформированы, большинство предусмотренных программой обучения учебных заданий выполнено, некоторые из выполненных заданий содержат ошибки.

«Неудовлетворительно» - теоретическое содержание курса не освоено, необходимые умения не сформированы, выполненные учебные задания содержат грубые ошибки.

Лабораторное занятие №9. Обмен данными. Работа со стеком в Assembler

Цель работы:

– приобретение навыков выполнения операций в различных системах счисления.

Выполнив работу, Вы будете:

уметь:

– У2. Создавать программу по разработанному алгоритму как отдельный модуль.
– У1.Осуществлять разработку кода программного модуля на языках низкого уровня и высокого уровней;
– У3. Выполнять отладку и тестирование программы на уровне модуля.
– У6. Оформлять документацию на программные средства.
– У8. Применять инструментальные средства отладки программного обеспечения.

Материальное обеспечение:

– Мультимедийные средства хранения, передачи и представления информации.

Задание

Организация и модели памяти, адресация

Память – способность объекта обеспечивать хранение данных.

Все объекты, над которыми выполняются команды, как и сами команды, хранятся в памяти компьютера.

Память состоит из ячеек, в каждой из которых содержится 1 ***бит*** информации, принимающий одно из двух значений: 0 или 1. Биты обрабатывают группами фиксированного размера. Для этого группы бит могут записываться и считываться за одну базовую операцию. Группа из 8 бит называется **.**

Байты последовательно располагаются в памяти компьютера.

- 1 килобайт (Кбайт) = 2^{10} = 1 024 байт
- 1 мегабайт (Мбайт) = 2^{10} Кбайт = 2^{20} байт = 1 048 576 байт
- 1 гигабайт (Гбайт) = 2^{10} Мбайт = 2^{30} байт = 1 073 741 824 байт

Для доступа к памяти с целью записи или чтения отдельных элементов информации используются ***идентификаторы***, определяющие их расположение в памяти. Каждому идентификатору в соответствие ставится ***адрес***. В качестве адресов используются числа из диапазона от 0 до 2^k-1 со значением k, достаточным для адресации всей памяти компьютера .Все 2^k адресов составляют ***адресное пространство компьютера.***

Способы адресации байтов

Существует прямой и обратный способы адресации байтов.

При ***обратном*** способе адресации байты адресуются слева направо, так что самый старший (левый) байт слова имеет наименьший адрес.

Прямым способом называется противоположная система адресации. Компиляторы высокоуровневых языков поддерживают прямой способ адресации.

Объект занимает целое слово. Поэтому для того, чтобы обратиться к нему в памяти, нужно указать адрес, по которому этот объект хранится.

Организация памяти

Физическая память, к которой микропроцессор имеет доступ по шине адреса, называется ***оперативной памятью*** ОП (или оперативным запоминающим устройством — ОЗУ).

Механизм управления памятью полностью аппаратный, т.е. программа сама не может сформировать физический адрес памяти на адресной шине. Микропроцессор аппаратно поддерживает несколько моделей использования оперативной памяти:

- сегментированную модель
- страничную модель
- плоскую модель

В сегментированной модели память для программы делится на непрерывные области памяти, называемые ***сегментами***. Программа может обращаться только к данным, которые находятся в этих сегментах.

Сегмент представляет собой независимый, поддерживаемый на аппаратном уровне блок памяти.

Сегментация — механизм адресации, обеспечивающий существование нескольких независимых адресных пространств как в пределах одной задачи, так и в системе в целом для защиты задач от взаимного влияния.

Каждая программа в общем случае может состоять из любого количества сегментов, но непосредственный доступ она имеет только к 3 основным сегментам и к 3 дополнительным сегментам, обслуживаемых 6 сегментными регистрами. К основным сегментам относятся:

- Сегмент кодов (.CODE) – содержит машинные команды для выполнения. Обычно первая выполняемая команда находится в начале этого сегмента, и операционная система передает управление по адресу данного сегмента для выполнения программы. Регистр сегмента кодов (CS) адресует данный сегмент.
- Сегмент данных (.DATA) – содержит определенные данные, константы и рабочие области, необходимые программе. Регистр сегмента данных (DS) адресует данный сегмент.
- Сегмент стека (.STACK). Стек содержит адреса возврата как для программы (для возврата в операционную систему), так и для вызовов подпрограмм (для возврата в главную программу). Регистр сегмента стека (SS) адресует данный сегмент. Адрес текущей вершины стека задается регистрами SS:ESP.

Регистры дополнительных сегментов (ES, FS, GS), предназначены для специального использования.

Для доступа к данным внутри сегмента обращение производится относительно начала сегмента линейно, т.е. начиная с 0 и заканчивая адресом, равным размеру сегмента. Для обращения к любому адресу в программе, компьютер складывает адрес в регистре сегмента и ***смещение*** — расположение требуемого адреса относительно начала сегмента. Например, первый байт в сегменте кодов имеет смещение 0, второй байт – 1 и так далее.

Таким образом, для обращения к конкретному физическому адресу ОЗУ необходимо определить адрес начала сегмента и смещение внутри сегмента.

Физический адрес принято записывать парой этих значений, разделенных двоеточием

сегмент : смещение

Страничная модель памяти – это надстройка над сегментной моделью. ОЗУ делится на блоки фиксированного размера, кратные степени 2, например 4 Кб. Каждый такой блок называется ***страницей***. Основное достоинство страничного способа распределения памяти — минимально возможная фрагментация. Однако такая организация памяти не использует память достаточно эффективно за счет фиксированного размера страниц.

Плоская модель памяти предполагает, что задача состоит из одного сегмента, который, в свою очередь, разбит на страницы. Достоинства:

- при использовании плоской модели памяти упрощается создание и операционной системы, и систем программирования;
- уменьшаются расходы памяти на поддержку системных информационных структур.

В абсолютном большинстве современных 32(64)-разрядных операционных систем (для микропроцессоров Intel) используется плоская модель памяти.

Модели памяти

Директива .MODEL определяет модель памяти, используемую программой. После этой директивы в программе находятся директивы объявления сегментов (.DATA, .STACK, .CODE, SEGMENT). Синтаксис задания модели памяти

.MODEL модификатор **МодельПамяти** СоглашениеОВызовах

Параметр **МодельПамяти** является обязательным.

Основные модели памяти:

Модель памяти	**Адресация кода**	**Адресация данных**	**Операци-онная система**	**Чередование кода и данных**
TINY	NEAR	NEAR	MS-DOS	Допустимо
SMALL	NEAR	NEAR	MS-DOS, Windows	Нет
MEDIUM	FAR	NEAR	MS-DOS, Windows	Нет
COMPACT	NEAR	FAR	MS-DOS, Windows	Нет
LARGE	FAR	FAR	MS-DOS, Windows	Нет
HUGE	FAR	FAR	MS-DOS, Windows	Нет
FLAT	NEAR	NEAR	Windows NT, Windows 2000, Windows XP, Windows Vista	Допустимо

Модель tiny работает только в 16-разрядных приложениях MS-DOS. В этой модели все данные и код располагаются в одном физическом сегменте. Размер программного файла в этом случае не превышает 64 Кбайт.

Модель small поддерживает один сегмент кода и один сегмент данных. Данные и код при использовании этой модели адресуются как near (ближние).

Модель medium поддерживает несколько сегментов программного кода и один сегмент данных, при этом все ссылки в сегментах программного кода по умолчанию считаются дальними (far), а ссылки в сегменте данных — ближними (near).

Модель compact поддерживает несколько сегментов данных, в которых используется дальняя адресация данных (far), и один сегмент кода с ближней адресацией (near).

Модель large поддерживает несколько сегментов кода и несколько сегментов данных. По умолчанию все ссылки на код и данные считаются дальними (far).

Модель huge практически эквивалентна модели памяти large.

Особого внимания заслуживает модель памяти flat, которая используется только в 32-разрядных операционных системах. В ней данные и код размещены в одном 32-

разрядном сегменте. Для использования в программе модели flat перед директивой .model flat следует разместить одну из директив:

.386
.486
.586
.686

Желательно указывать тот тип процессора, который используется в машине, хотя это не является обязательным требованием. Операционная система автоматически инициализирует сегментные регистры при загрузке программы, поэтому модифицировать их нужно только в случае если требуется смешивать в одной программе 16-разрядный и 32-разрядный код. Адресация данных и кода является ближней (near), при этом все адреса и указатели являются 32-разрядными.

Параметр **модификатор** используется для определения типов сегментов и может принимать значения use16 (сегменты выбранной модели используются как 16-битные) или use32 (сегменты выбранной модели используются как 32-битные).

Параметр **СоглашениеОВызовах** используется для определения способа передачи параметров при вызове процедуры из других языков, в том числе и языков высокого уровня (C++, Pascal). Параметр может принимать следующие значения:

C,
BASIC,
FORTRAN,
PASCAL,
SYSCALL,
STDCALL.

При разработке модулей на ассемблере, которые будут применяться в программах, написанных на языках высокого уровня, обращайте внимание на то, какие соглашения о вызовах поддерживает тот или иной язык. Используются при анализе интерфейса программ на ассемблере с программами на языках высокого уровня.

Ассемблерная программа, например, получает число имеющихся дисковых накопителей следующим образом. При этом, нельзя забывать о восстановлении первоначального значения в порте B.

```
IN   AL,61H         ;получаем значение из порта B
OR   AL,10000000B   ;устанавливаем бит 7 в 1
OUT  61H,AL         ;заменяем байт
IN   AL,60H         ;получаем значение из порта A
MOV  CL,6           ;подготовка для сдвига AL
SHR  AL,CL          ;сдвигаем 2 старших бита на 6 позиций
INC  AL             ;начинаем счет с 1, а не с 0
MOV  NUM_DRIVES,AL  ;получаем число накопителей
IN   AL,61H         ;подготовка к восстановлению порта B
AND  AL,01111111B   ;сбрасываем бит 7
OUT  61H,AL         ;восстанавливаем байт
```

Определение числа и типа периферийных устройств.

При старте компьютера ROM-BIOS проверяет присоединенное оборудование, сообщая о результатах своей проверки в регистр статуса. Этот регистр занимает два байта, начиная с 0040:0010. Ниже приведены значения битов относятся ко всем машинам, пока не оговорено обратное:

бит 0 если 1, то присутствует НГМД
1 XT,AT:1 = есть мат. сопроцессор (PC,PCjr: не использ.)

2-3 11 = базовая память 64K (AT: не используется)
4-5 Активный видеоадаптер
6-7 число НГМД (если бит 0 = 1)
8 PCjr:0 = есть DMA (PC,XT,AT: не используется)
9-11 число адаптеров коммуникации
12 1 = есть игровой порт (AT: не используется)
13 PCjr :есть серийный принтер (PC,XT,AT: не использ.)
14-15 число присоединенных принтеров

Число портов коммуникации может быть получено из области данных BIOS. BIOS отводит четыре 2-байтных поля для хранения базовых адресов вплоть до четырех COM портов. Базовый адрес – это младший из адресов портов, относящихся к группе портов, имеющих доступ к данному каналу коммуникации. Эти четыре поля начинаются с адреса 0040:0008. Порту COM1 соответствует адрес :0008, а COM2 - 000A. Если это поле содержит 0, то соответствующий порт отсутствует. Таким образом, если слово по адресу: 0008 отлично от нуля, а по адресу 000A - нулевое, то имеется один порт коммуникации.

Ревизия количества памяти.

Информация о количестве банков памяти может быть прочитана через порт B микросхемы интерфейса с периферией 8255. BIOS хранит двухбайтную переменную по адресу 0040:0013, которая сообщает число килобайт используемой
памяти. Для PCjr бит 3 порта 62H (порт C микросхемы 8255) равен нулю, когда машина имеет добавочные 64K памяти. AT дает особо полную информацию о памяти. Регистры 15H (младший) и 16H (старший) микросхемы информации о конфигурации говорят сколько памяти установлено на системной плате. Память канала ввода/вывода для AT сообщается регистрами 17H и 18H (с инкрементом 512K). Память сверх 1 мегабайта доступна через регистры 30H и 31H (также с инкрементом 512K, вплоть до 15 мегабайт). Если AT имеет 128K на плате расширения, то установлен бит 7 регистра 33. Во всех случаях надо сначала послать номер регистра в порт 70H, а затем прочитать значение из порта 71H.

Порядок выполнения работы

- Составить математическую модель задачи.
- Выбрать и обосновать наиболее рациональный метод решения задачи;
- Разработать алгоритм для решения задачи.
- Написать и отладить программу.

Форма предоставления результата

- Блок – схема.
- Код программы.

Критерии оценки:

«Отлично» - теоретическое содержание курса освоено полностью, без пробелов, умения сформированы, все предусмотренные программой учебные задания выполнены, качество их выполнения оценено высоко.

–«Хорошо» - теоретическое содержание курса освоено полностью, без пробелов, некоторые умения сформированы недостаточно, все предусмотренные программой учебные задания выполнены, некоторые виды заданий выполнены с ошибками.

–«Удовлетворительно» - теоретическое содержание курса освоено частично, но пробелы не носят существенного характера, необходимые умения работы с освоенным материалом в основном сформированы, большинство предусмотренных программой обучения учебных заданий выполнено, некоторые из выполненных заданий содержат ошибки.

«Неудовлетворительно» - теоретическое содержание курса не освоено, необходимые умения не сформированы, выполненные учебные задания содержат грубые ошибки.

Лабораторное занятие №10. Обмен данными. Аппаратные прерывания. Приоритет прерываний. Запрет и маскирование аппаратных прерываний

Цель работы:

- приобретение навыков выполнения операций в различных системах счисления.

Выполнив работу, Вы будете:

уметь:

- У2. Создавать программу по разработанному алгоритму как отдельный модуль.
- У1.Осуществлять разработку кода программного модуля на языках низкого уровня и высокого уровней;
- У3. Выполнять отладку и тестирование программы на уровне модуля.
- У6. Оформлять документацию на программные средства.
- У8. Применять инструментальные средства отладки программного обеспечения.

Материальное обеспечение:

- Мультимедийные средства хранения, передачи и представления информации.

Задание

Организация и модели памяти, адресация

Память – способность объекта обеспечивать хранение данных.

Все объекты, над которыми выполняются команды, как и сами команды, хранятся в памяти компьютера.

Память состоит из ячеек, в каждой из которых содержится 1 ***бит*** информации, принимающий одно из двух значений: 0 или 1. Биты обрабатывают группами фиксированного размера. Для этого группы бит могут записываться и считываться за одну базовую операцию. Группа из 8 бит называется **.**

Байты последовательно располагаются в памяти компьютера.

- 1 килобайт (Кбайт) = 2^{10} = 1 024 байт
- 1 мегабайт (Мбайт) = 2^{10} Кбайт = 2^{20} байт = 1 048 576 байт
- 1 гигабайт (Гбайт) = 2^{10} Мбайт = 2^{30} байт = 1 073 741 824 байт

Для доступа к памяти с целью записи или чтения отдельных элементов информации используются ***идентификаторы***, определяющие их расположение в памяти. Каждому идентификатору в соответствие ставится ***адрес***. В качестве адресов используются числа из диапазона от 0 до 2^k-1 со значением k, достаточным для адресации всей памяти компьютера .Все 2^k адресов составляют ***адресное пространство компьютера***.

Способы адресации байтов

Существует прямой и обратный способы адресации байтов.

При ***обратном*** способе адресации байты адресуются слева направо, так что самый старший (левый) байт слова имеет наименьший адрес.

Прямым способом называется противоположная система адресации. Компиляторы высокоуровневых языков поддерживают прямой способ адресации.

Объект занимает целое слово. Поэтому для того, чтобы обратиться к нему в памяти, нужно указать адрес, по которому этот объект хранится.

Организация памяти

Физическая память, к которой микропроцессор имеет доступ по шине адреса, называется ***оперативной памятью*** ОП (или оперативным запоминающим устройством — ОЗУ).

Механизм управления памятью полностью аппаратный, т.е. программа сама не может сформировать физический адрес памяти на адресной шине.

Микропроцессор аппаратно поддерживает несколько моделей использования оперативной памяти:

- сегментированную модель
- страничную модель
- плоскую модель

В сегментированной модели память для программы делится на непрерывные области памяти, называемые ***сегментами***. Программа может обращаться только к данным, которые находятся в этих сегментах.

Сегмент представляет собой независимый, поддерживаемый на аппаратном уровне блок памяти.

Сегментация — механизм адресации, обеспечивающий существование нескольких независимых адресных пространств как в пределах одной задачи, так и в системе в целом для защиты задач от взаимного влияния.

Каждая программа в общем случае может состоять из любого количества сегментов, но непосредственный доступ она имеет только к 3 основным сегментам и к 3 дополнительным сегментам, обслуживаемых 6 сегментными регистрами. К основным сегментам относятся:

- Сегмент кодов (.CODE) – содержит машинные команды для выполнения. Обычно первая выполняемая команда находится в начале этого сегмента, и операционная система передает управление по адресу данного сегмента для выполнения программы. Регистр сегмента кодов (CS) адресует данный сегмент.
- Сегмент данных (.DATA) – содержит определенные данные, константы и рабочие области, необходимые программе. Регистр сегмента данных (DS) адресует данный сегмент.
- Сегмент стека (.STACK). Стек содержит адреса возврата как для программы (для возврата в операционную систему), так и для вызовов подпрограмм (для возврата в главную программу). Регистр сегмента стека (SS) адресует данный сегмент. Адрес текущей вершины стека задается регистрами SS:ESP.

Регистры дополнительных сегментов (ES, FS, GS), предназначены для специального использования.

Для доступа к данным внутри сегмента обращение производится относительно начала сегмента линейно, т.е. начиная с 0 и заканчивая адресом, равным размеру сегмента. Для обращения к любому адресу в программе, компьютер складывает адрес в регистре сегмента и ***смещение*** — расположение требуемого адреса относительно начала сегмента. Например, первый байт в сегменте кодов имеет смещение 0, второй байт – 1 и так далее.

Таким образом, для обращения к конкретному физическому адресу ОЗУ необходимо определить адрес начала сегмента и смещение внутри сегмента.

Физический адрес принято записывать парой этих значений, разделенных двоеточием

сегмент : смещение

Страничная модель памяти – это надстройка над сегментной моделью. ОЗУ делится на блоки фиксированного размера, кратные степени 2, например 4 Кб. Каждый такой блок называется ***страницей***. Основное достоинство страничного способа распределения памяти — минимально возможная фрагментация. Однако такая организация памяти не использует память достаточно эффективно за счет фиксированного размера страниц.

Плоская модель памяти предполагает, что задача состоит из одного сегмента, который, в свою очередь, разбит на страницы. Достоинства:

- при использовании плоской модели памяти упрощается создание и операционной системы, и систем программирования;

• уменьшаются расходы памяти на поддержку системных информационных структур.

В абсолютном большинстве современных 32(64)-разрядных операционных систем (для микропроцессоров Intel) используется плоская модель памяти.

Модели памяти

Директива .MODEL определяет модель памяти, используемую программой. После этой директивы в программе находятся директивы объявления сегментов (.DATA, .STACK, .CODE, SEGMENT). Синтаксис задания модели памяти

.MODEL модификатор **МодельПамяти** СоглашениеОВызовах

Параметр **МодельПамяти** является обязательным.

Основные модели памяти:

Модель памяти	**Адресация кода**	**Адресация данных**	**Операци-онная система**	**Чередование кода и данных**
TINY	NEAR	NEAR	MS-DOS	Допустимо
SMALL	NEAR	NEAR	MS-DOS, Windows	Нет
MEDIUM	FAR	NEAR	MS-DOS, Windows	Нет
COMPACT	NEAR	FAR	MS-DOS, Windows	Нет
LARGE	FAR	FAR	MS-DOS, Windows	Нет
HUGE	FAR	FAR	MS-DOS, Windows	Нет
FLAT	NEAR	NEAR	Windows NT, Windows 2000, Windows XP, Windows Vista	Допустимо

Модель tiny работает только в 16-разрядных приложениях MS-DOS. В этой модели все данные и код располагаются в одном физическом сегменте. Размер программного файла в этом случае не превышает 64 Кбайт.

Модель small поддерживает один сегмент кода и один сегмент данных. Данные и код при использовании этой модели адресуются как near (ближние).

Модель medium поддерживает несколько сегментов программного кода и один сегмент данных, при этом все ссылки в сегментах программного кода по умолчанию считаются дальними (far), а ссылки в сегменте данных — ближними (near).

Модель compact поддерживает несколько сегментов данных, в которых используется дальняя адресация данных (far), и один сегмент кода с ближней адресацией (near).

Модель large поддерживает несколько сегментов кода и несколько сегментов данных. По умолчанию все ссылки на код и данные считаются дальними (far).

Модель huge практически эквивалентна модели памяти large.

Особого внимания заслуживает модель памяти flat, которая используется только в 32-разрядных операционных системах. В ней данные и код размещены в одном 32-разрядном сегменте. Для использования в программе модели flat перед директивой .model flat следует разместить одну из директив:

.386
.486
.586
.686

Желательно указывать тот тип процессора, который используется в машине, хотя это не является обязательным требованием. Операционная система автоматически инициализирует сегментные регистры при загрузке программы, поэтому модифицировать их нужно только в случае если требуется смешивать в одной программе 16-разрядный и 32-разрядный код. Адресация данных и кода является ближней (near), при этом все адреса и указатели являются 32-разрядными.

Параметр **модификатор** используется для определения типов сегментов и может принимать значения use16 (сегменты выбранной модели используются как 16-битные) или use32 (сегменты выбранной модели используются как 32-битные).

Параметр **СоглашениеОВызовах** используется для определения способа передачи параметров при вызове процедуры из других языков, в том числе и языков высокого уровня (C++, Pascal). Параметр может принимать следующие значения:

C,
BASIC,
FORTRAN,
PASCAL,
SYSCALL,
STDCALL.

При разработке модулей на ассемблере, которые будут применяться в программах, написанных на языках высокого уровня, обращайте внимание на то, какие соглашения о вызовах поддерживает тот или иной язык. Используются при анализе интерфейса программ на ассемблере с программами на языках высокого уровня.

Ассемблерная программа, например, получает число имеющихся дисковых накопителей следующим образом. При этом, нельзя забывать о восстановлении первоначального значения в порте B.

```
IN   AL,61H         ;получаем значение из порта B
OR   AL,10000000B   ;устанавливаем бит 7 в 1
OUT  61H,AL         ;заменяем байт
IN   AL,60H         ;получаем значение из порта A
MOV  CL,6           ;подготовка для сдвига AL
SHR  AL,CL          ;сдвигаем 2 старших бита на 6 позиций
INC  AL             ;начинаем счет с 1, а не с 0
MOV  NUM_DRIVES,AL  ;получаем число накопителей
IN   AL,61H         ;подготовка к восстановлению порта B
AND  AL,01111111B   ;сбрасываем бит 7
OUT  61H,AL         ;восстанавливаем байт
```

Определение числа и типа периферийных устройств.

При старте компьютера ROM-BIOS проверяет присоединенное оборудование, сообщая о результатах своей проверки в регистр статуса. Этот регистр занимает два байта, начиная с 0040:0010. Ниже приведены значения битов относятся ко всем машинам, пока не оговорено обратное:

бит 0 если 1, то присутствует НГМД
1 XT,AT:1 = есть мат. сопроцессор (PC,PCjr: не использ.)
2-3 11 = базовая память 64K (AT: не используется)
4-5 Активный видеоадаптер
6-7 число НГМД (если бит 0 = 1)
8 PCjr:0 = есть DMA (PC,XT,AT: не используется)
9-11 число адаптеров коммуникации
12 1 = есть игровой порт (AT: не используется)
13 PCjr :есть серийный принтер (PC,XT,AT: не использ.)
14-15 число присоединенных принтеров

Число портов коммуникации может быть получено из области данных BIOS. BIOS отводит четыре 2-байтных поля для хранения базовых адресов вплоть до четырех COM портов. Базовый адрес – это младший из адресов портов, относящихся к группе портов, имеющих доступ к данному каналу коммуникации. Эти четыре поля начинаются с адреса 0040:0008. Порту COM1 соответствует адрес :0008, а COM2 - 000A. Если это поле содержит 0, то соответствующий порт отсутствует. Таким образом,

если слово по адресу: 0008 отлично от нуля, а по адресу 000A - нулевое, то имеется один порт коммуникации.

Ревизия количества памяти.

Информация о количестве банков памяти может быть прочитана через порт B микросхемы интерфейса с периферией 8255. BIOS хранит двухбайтную переменную по адресу 0040:0013, которая сообщает число килобайт используемой
памяти. Для PCjr бит 3 порта 62H (порт C микросхемы 8255) равен нулю, когда машина имеет добавочные 64K памяти. AT дает особо полную информацию о памяти. Регистры 15H (младший) и 16H (старший) микросхемы информации о конфигурации говорят сколько памяти установлено на системной плате. Память канала ввода/вывода для AT сообщается регистрами 17H и 18H (с инкрементом 512K). Память сверх 1 мегабайта доступна через регистры 30H и 31H (также с инкрементом 512K, вплоть до 15 мегабайт). Если AT имеет 128K на плате расширения, то установлен бит 7 регистра 33. Во всех случаях надо сначала послать номер регистра в порт 70H, а затем прочитать значение из порта 71H.

Порядок выполнения работы

- Составить математическую модель задачи.
- Выбрать и обосновать наиболее рациональный метод решения задачи;
- Разработать алгоритм для решения задачи.
- Написать и отладить программу.

Форма предоставления результата

- Блок – схема.
- Код программы.

Критерии оценки:

«Отлично» - теоретическое содержание курса освоено полностью, без пробелов, умения сформированы, все предусмотренные программой учебные задания выполнены, качество их выполнения оценено высоко.

–«Хорошо» - теоретическое содержание курса освоено полностью, без пробелов, некоторые умения сформированы недостаточно, все предусмотренные программой учебные задания выполнены, некоторые виды заданий выполнены с ошибками.

–«Удовлетворительно» - теоретическое содержание курса освоено частично, но пробелы не носят существенного характера, необходимые умения работы с освоенным материалом в основном сформированы, большинство предусмотренных программой обучения учебных заданий выполнено, некоторые из выполненных заданий содержат ошибки.

«Неудовлетворительно» - теоретическое содержание курса не освоено, необходимые умения не сформированы, выполненные учебные задания содержат грубые ошибки.

Лабораторное занятие №11. Обмен данными. Программный доступ к CMOS-памяти.

Цель работы:

- приобретение навыков выполнения операций в различных системах счисления.

Выполнив работу, Вы будете:

уметь:

- У2. Создавать программу по разработанному алгоритму как отдельный модуль.
- У1.Осуществлять разработку кода программного модуля на языках низкого уровня и высокого уровней;
- У3. Выполнять отладку и тестирование программы на уровне модуля.
- У6. Оформлять документацию на программные средства.
- У8. Применять инструментальные средства отладки программного обеспечения.

Материальное обеспечение:

– Мультимедийные средства хранения, передачи и представления информации.

Лабораторное занятие №12. Обмен данными. Программирование клавиатуры.

Цель работы:

– приобретение навыков выполнения операций в различных системах счисления.

Выполнив работу, Вы будете:

уметь:

– У2. Создавать программу по разработанному алгоритму как отдельный модуль.
– У1.Осуществлять разработку кода программного модуля на языках низкого уровня и высокого уровней;
– У3. Выполнять отладку и тестирование программы на уровне модуля.
– У6. Оформлять документацию на программные средства.
– У8. Применять инструментальные средства отладки программного обеспечения.

Материальное обеспечение:

– Мультимедийные средства хранения, передачи и представления информации.

Задание

Организация и модели памяти, адресация

Память – способность объекта обеспечивать хранение данных.

Все объекты, над которыми выполняются команды, как и сами команды, хранятся в памяти компьютера.

Память состоит из ячеек, в каждой из которых содержится 1 ***бит*** информации, принимающий одно из двух значений: 0 или 1. Биты обрабатывают группами фиксированного размера. Для этого группы бит могут записываться и считываться за одну базовую операцию. Группа из 8 бит называется **.**

Байты последовательно располагаются в памяти компьютера.

- 1 килобайт (Кбайт) = 2^{10} = 1 024 байт
- 1 мегабайт (Мбайт) = 2^{10} Кбайт = 2^{20} байт = 1 048 576 байт
- 1 гигабайт (Гбайт) = 2^{10} Мбайт = 2^{30} байт = 1 073 741 824 байт

Для доступа к памяти с целью записи или чтения отдельных элементов информации используются ***идентификаторы***, определяющие их расположение в памяти. Каждому идентификатору в соответствие ставится ***адрес***. В качестве адресов используются числа из диапазона от 0 до 2^k-1 со значением k, достаточным для адресации всей памяти компьютера .Все 2^k адресов составляют ***адресное пространство компьютера***.

Способы адресации байтов

Существует прямой и обратный способы адресации байтов.

При ***обратном*** способе адресации байты адресуются слева направо, так что самый старший (левый) байт слова имеет наименьший адрес.

Прямым способом называется противоположная система адресации. Компиляторы высокоуровневых языков поддерживают прямой способ адресации.

Объект занимает целое слово. Поэтому для того, чтобы обратиться к нему в памяти, нужно указать адрес, по которому этот объект хранится.

Организация памяти

Физическая память, к которой микропроцессор имеет доступ по шине адреса, называется ***оперативной памятью*** ОП (или оперативным запоминающим устройством — ОЗУ).

Механизм управления памятью полностью аппаратный, т.е. программа сама не может сформировать физический адрес памяти на адресной шине.

Микропроцессор аппаратно поддерживает несколько моделей использования оперативной памяти:

- сегментированную модель
- страничную модель
- плоскую модель

В сегментированной модели память для программы делится на непрерывные области памяти, называемые ***сегментами***. Программа может обращаться только к данным, которые находятся в этих сегментах.

Сегмент представляет собой независимый, поддерживаемый на аппаратном уровне блок памяти.

Сегментация — механизм адресации, обеспечивающий существование нескольких независимых адресных пространств как в пределах одной задачи, так и в системе в целом для защиты задач от взаимного влияния.

Каждая программа в общем случае может состоять из любого количества сегментов, но непосредственный доступ она имеет только к 3 основным сегментам и к 3 дополнительным сегментам, обслуживаемых 6 сегментными регистрами. К основным сегментам относятся:

- Сегмент кодов (.CODE) – содержит машинные команды для выполнения. Обычно первая выполняемая команда находится в начале этого сегмента, и операционная система передает управление по адресу данного сегмента для выполнения программы. Регистр сегмента кодов (CS) адресует данный сегмент.
- Сегмент данных (.DATA) – содержит определенные данные, константы и рабочие области, необходимые программе. Регистр сегмента данных (DS) адресует данный сегмент.
- Сегмент стека (.STACK). Стек содержит адреса возврата как для программы (для возврата в операционную систему), так и для вызовов подпрограмм (для возврата в главную программу). Регистр сегмента стека (SS) адресует данный сегмент. Адрес текущей вершины стека задается регистрами SS:ESP.

Регистры дополнительных сегментов (ES, FS, GS), предназначены для специального использования.

Для доступа к данным внутри сегмента обращение производится относительно начала сегмента линейно, т.е. начиная с 0 и заканчивая адресом, равным размеру сегмента. Для обращения к любому адресу в программе, компьютер складывает адрес в регистре сегмента и ***смещение*** — расположение требуемого адреса относительно начала сегмента. Например, первый байт в сегменте кодов имеет смещение 0, второй байт – 1 и так далее.

Таким образом, для обращения к конкретному физическому адресу ОЗУ необходимо определить адрес начала сегмента и смещение внутри сегмента.

Физический адрес принято записывать парой этих значений, разделенных двоеточием

сегмент : смещение

Страничная модель памяти – это надстройка над сегментной моделью. ОЗУ делится на блоки фиксированного размера, кратные степени 2, например 4 Кб. Каждый такой блок называется ***страницей***. Основное достоинство страничного способа распределения памяти — минимально возможная фрагментация. Однако такая организация памяти не использует память достаточно эффективно за счет фиксированного размера страниц.

Плоская модель памяти предполагает, что задача состоит из одного сегмента, который, в свою очередь, разбит на страницы. Достоинства:

- при использовании плоской модели памяти упрощается создание и операционной системы, и систем программирования;

• уменьшаются расходы памяти на поддержку системных информационных структур.

В абсолютном большинстве современных 32(64)-разрядных операционных систем (для микропроцессоров Intel) используется плоская модель памяти.

Модели памяти

Директива .MODEL определяет модель памяти, используемую программой. После этой директивы в программе находятся директивы объявления сегментов (.DATA, .STACK, .CODE, SEGMENT). Синтаксис задания модели памяти

.MODEL модификатор **МодельПамяти** СоглашениеОВызовах

Параметр **МодельПамяти** является обязательным.

Основные модели памяти:

Модель памяти	**Адресация кода**	**Адресация данных**	**Операци-онная система**	**Чередование кода и данных**
TINY	NEAR	NEAR	MS-DOS	Допустимо
SMALL	NEAR	NEAR	MS-DOS, Windows	Нет
MEDIUM	FAR	NEAR	MS-DOS, Windows	Нет
COMPACT	NEAR	FAR	MS-DOS, Windows	Нет
LARGE	FAR	FAR	MS-DOS, Windows	Нет
HUGE	FAR	FAR	MS-DOS, Windows	Нет
FLAT	NEAR	NEAR	Windows NT, Windows 2000, Windows XP, Windows Vista	Допустимо

Модель tiny работает только в 16-разрядных приложениях MS-DOS. В этой модели все данные и код располагаются в одном физическом сегменте. Размер программного файла в этом случае не превышает 64 Кбайт.

Модель small поддерживает один сегмент кода и один сегмент данных. Данные и код при использовании этой модели адресуются как near (ближние).

Модель medium поддерживает несколько сегментов программного кода и один сегмент данных, при этом все ссылки в сегментах программного кода по умолчанию считаются дальними (far), а ссылки в сегменте данных — ближними (near).

Модель compact поддерживает несколько сегментов данных, в которых используется дальняя адресация данных (far), и один сегмент кода с ближней адресацией (near).

Модель large поддерживает несколько сегментов кода и несколько сегментов данных. По умолчанию все ссылки на код и данные считаются дальними (far).

Модель huge практически эквивалентна модели памяти large.

Особого внимания заслуживает модель памяти flat, которая используется только в 32-разрядных операционных системах. В ней данные и код размещены в одном 32-разрядном сегменте. Для использования в программе модели flat перед директивой .model flat следует разместить одну из директив:

.386
.486
.586
.686

Желательно указывать тот тип процессора, который используется в машине, хотя это не является обязательным требованием. Операционная система автоматически инициализирует сегментные регистры при загрузке программы, поэтому модифицировать их нужно только в случае если требуется смешивать в одной программе 16-разрядный и 32-разрядный код. Адресация данных и кода является ближней (near), при этом все адреса и указатели являются 32-разрядными.

Параметр **модификатор** используется для определения типов сегментов и может принимать значения use16 (сегменты выбранной модели используются как 16-битные) или use32 (сегменты выбранной модели используются как 32-битные).

Параметр **СоглашениеОВызовах** используется для определения способа передачи параметров при вызове процедуры из других языков, в том числе и языков высокого уровня (C++, Pascal). Параметр может принимать следующие значения:

C,
BASIC,
FORTRAN,
PASCAL,
SYSCALL,
STDCALL.

При разработке модулей на ассемблере, которые будут применяться в программах, написанных на языках высокого уровня, обращайте внимание на то, какие соглашения о вызовах поддерживает тот или иной язык. Используются при анализе интерфейса программ на ассемблере с программами на языках высокого уровня.

Ассемблерная программа, например, получает число имеющихся дисковых накопителей следующим образом. При этом, нельзя забывать о восстановлении первоначального значения в порте B.

```
IN   AL,61H         ;получаем значение из порта B
OR   AL,10000000B   ;устанавливаем бит 7 в 1
OUT  61H,AL         ;заменяем байт
IN   AL,60H         ;получаем значение из порта A
MOV  CL,6           ;подготовка для сдвига AL
SHR  AL,CL          ;сдвигаем 2 старших бита на 6 позиций
INC  AL             ;начинаем счет с 1, а не с 0
MOV  NUM_DRIVES,AL  ;получаем число накопителей
IN   AL,61H         ;подготовка к восстановлению порта B
AND  AL,01111111B   ;сбрасываем бит 7
OUT  61H,AL         ;восстанавливаем байт
```

Определение числа и типа периферийных устройств.

При старте компьютера ROM-BIOS проверяет присоединенное оборудование, сообщая о результатах своей проверки в регистр статуса. Этот регистр занимает два байта, начиная с 0040:0010. Ниже приведены значения битов относятся ко всем машинам, пока не оговорено обратное:

бит 0 если 1, то присутствует НГМД
1 XT,AT:1 = есть мат. сопроцессор (PC,PCjr: не использ.)
2-3 11 = базовая память 64K (AT: не используется)
4-5 Активный видеоадаптер
6-7 число НГМД (если бит 0 = 1)
8 PCjr:0 = есть DMA (PC,XT,AT: не используется)
9-11 число адаптеров коммуникации
12 1 = есть игровой порт (AT: не используется)
13 PCjr :есть серийный принтер (PC,XT,AT: не использ.)
14-15 число присоединенных принтеров

Число портов коммуникации может быть получено из области данных BIOS. BIOS отводит четыре 2-байтных поля для хранения базовых адресов вплоть до четырех COM портов. Базовый адрес – это младший из адресов портов, относящихся к группе портов, имеющих доступ к данному каналу коммуникации. Эти четыре поля начинаются с адреса 0040:0008. Порту COM1 соответствует адрес :0008, а COM2 - 000A. Если это поле содержит 0, то соответствующий порт отсутствует. Таким образом,

если слово по адресу: 0008 отлично от нуля, а по адресу 000A - нулевое, то имеется один порт коммуникации.

Ревизия количества памяти.

Информация о количестве банков памяти может быть прочитана через порт B микросхемы интерфейса с периферией 8255. BIOS хранит двухбайтную переменную по адресу 0040:0013, которая сообщает число килобайт используемой
памяти. Для PCjr бит 3 порта 62H (порт C микросхемы 8255) равен нулю, когда машина имеет добавочные 64K памяти. AT дает особо полную информацию о памяти. Регистры 15H (младший) и 16H (старший) микросхемы информации о конфигурации говорят сколько памяти установлено на системной плате. Память канала ввода/вывода для AT сообщается регистрами 17H и 18H (с инкрементом 512K). Память сверх 1 мегабайта доступна через регистры 30H и 31H (также с инкрементом 512K, вплоть до 15 мегабайт). Если AT имеет 128K на плате расширения, то установлен бит 7 регистра 33. Во всех случаях надо сначала послать номер регистра в порт 70H, а затем прочитать значение из порта 71H.

Порядок выполнения работы

- Составить математическую модель задачи.
- Выбрать и обосновать наиболее рациональный метод решения задачи;
- Разработать алгоритм для решения задачи.
- Написать и отладить программу.

Форма предоставления результата

- Блок – схема.
- Код программы.

Критерии оценки:

«Отлично» - теоретическое содержание курса освоено полностью, без пробелов, умения сформированы, все предусмотренные программой учебные задания выполнены, качество их выполнения оценено высоко.

–«Хорошо» - теоретическое содержание курса освоено полностью, без пробелов, некоторые умения сформированы недостаточно, все предусмотренные программой учебные задания выполнены, некоторые виды заданий выполнены с ошибками.

–«Удовлетворительно» - теоретическое содержание курса освоено частично, но пробелы не носят существенного характера, необходимые умения работы с освоенным материалом в основном сформированы, большинство предусмотренных программой обучения учебных заданий выполнено, некоторые из выполненных заданий содержат ошибки.

«Неудовлетворительно» - теоретическое содержание курса не освоено, необходимые умения не сформированы, выполненные учебные задания содержат грубые ошибки.

Лабораторное занятие №13. Обмен данными. Микросхема таймера Intel 8253 и ее программирование.

Цель работы:

- Изучить возможности работы с таймером и звуком.

Выполнив работу, Вы будете:

уметь:

- У2. Создавать программу по разработанному алгоритму как отдельный модуль.
- У1.Осуществлять разработку кода программного модуля на языках низкого уровня и высокого уровней;
- У3. Выполнять отладку и тестирование программы на уровне модуля.
- У6. Оформлять документацию на программные средства.
- У8. Применять инструментальные средства отладки программного обеспечения.

Материальное обеспечение:

– Мультимедийные средства хранения, передачи и представления информации.

Задание

Встроенный динамик – это системное устройство, поэтому в отличие от музыкальных сопроцессоров не требует никаких драйверов и доступно всегда. В архитектуре IBM PC существуют две возможности управлять «спикером»:

с помощью первого бита порта 61h;

при программировании второго канала таймера Intel 8253

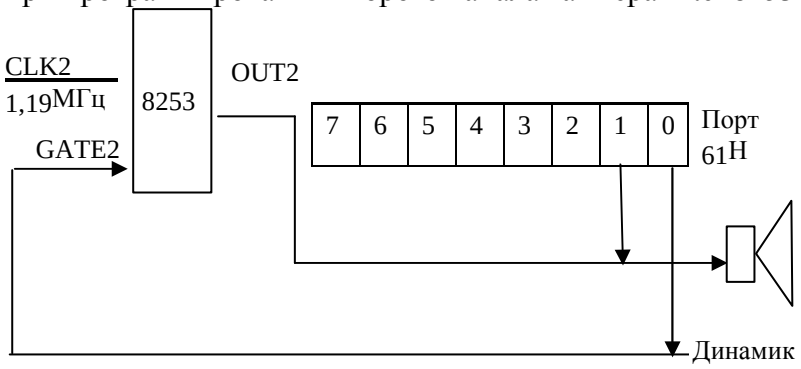


Рис. 1. Получение звука через PC Speaker

Комбинируя эти возможности, можно получить специальные звуковые эффекты. Аппаратные возможности не позволяют изменить громкость звука, издаваемого динамиками.

Системный порт 61h работает как на чтение, так и на запись.

Ворота GATE2 закрываются записью «0» в нулевой бит 61h порта.

Это позволяет производить ряд манипуляций со звуком. Первый бит порта 61h связан с PC speaker и также может быть использован для генерации звука. Спикер в отличие от звуковой платы является системным устройством, а не драйверным, и присутствует в любой компьютерной системе, поэтому он широко используется для сигнализации различных событий, программных ошибок и аппаратных сбоев.

Генерация звука с помощью порта 61h

Метод состоит в периодическом включении и выключении с желаемой частотой первого бита порта 61h. Если переключать значение бита с максимально возможной частотой, то мы вряд ли что услышим, так как получаемая частота будет довольно высокой.

Слышимые человеком звуки ограничены диапазоном от 20 до 20000 Гц. Звуки, воспроизводимые SPEAKER, имеют еще более узкий диапазон 50–12000 Гц. Между переключениями необходимо вставлять задержки по времени τвкл, τвыкл. (рис. 4.5). Нулевой бит порта 61h управляет воротами канала 2 таймера, который связан с динамиком. Этот бит сбрасываем в 0, чтобы таймер не влиял на получаемый звук.

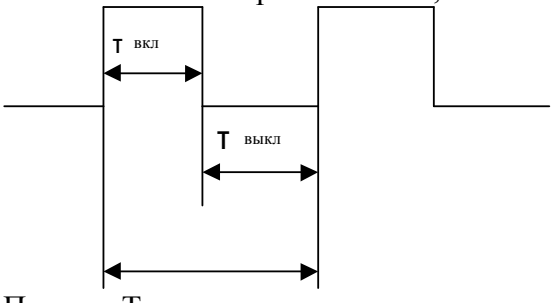


Рис. 2. Формирование прямоугольных импульсов, подаваемых на системный динамик: период получаемых колебаний T=τвкл +τвыкл; полученная частота звука f=1/T Гц

Задача 6

Составить программу для генерации звука посредством порта 61h.

Задержки по времени осуществляются программно, поэтому, чем производительнее процессор, тем выше будет тон получаемого звука и меньше времени он будет продолжаться. Фрагмент этой программы можно использовать как простейший тест быстродействия компьютера.

Решение:

```
 ; Основной фрагмент программы zvuk_prt.asm
cli                     ;       запрет       аппаратных
        прерываний
mov         DX, 1000    ; длительность тонов в
        периодах
in          AL, 61h     ;   получаем   значение
        порта 61H
and         AL,    1111111Ob    ;    отключение
metka1: динамика от 8253
or          AL, 00000010b  ; включим динамик
out         61h, AL
; задержка на половину периода
MOV   CX, 300h metka2:
LOOP metka2
  AND  AL, 11111101b        ; выключение динамика
  OUT  61h, AL
; задержка на половину периода
MOV   CX, 300h metka3:
LOOP metka3
  DEC  DX
  JNZ  metka1
  STI               ; разрешить аппаратные прерывания
```

Генерация звука посредством таймера

Более предпочтительным является способ получения звука посредством таймера, т.к. процессор «не участвует» в получении самого звука и может заниматься другой работой.

Задача 7

Получить звук с частотой f=440 Гц. Звук прекратить нажатием любой клавиши на клавиатуре.

Решение:

```
  ;Основной фрагмент программы zvuk_t.asm
; Биты 0 и 1 порта 61h предварительно необходимо уста-
новить в «1»

in          AL, 61h         ; читаем порт 61H
or          AL, 00000011b   ; установление двух    ;
      младших битов в «1»
out         61h, AL
mov         AL,  10110110b          ;  управляющий
      регистр для 8253
out         43h, al
mov         AX, A91h        ;
      1190000/440=2705=A91h
out         42h, AL         ;   посылаем   младший
      байт
```

```
mov         AL, AH
out         42h, AL          ;   посылаем   старший
        байт

 ; слышим звук с частотой 440 Гц

  mov    AX, 0C08h      ; очистка буфера,
  ; ввод символа без ЭХА на экран
int       21h              ; ждем нажатие клавиши in        AL, 61h
  and     AL, 11111100b   ; сбрасываем два младших бита
  out     61h, AL
; звук выключается
```

Порядок выполнения работы

- Составить математическую модель задачи.
- Выбрать и обосновать наиболее рациональный метод решения задачи;
- Разработать алгоритм для решения задачи.
- Написать и отладить программу.

Форма предоставления результата

- Блок – схема.
- Код программы.

Критерии оценки:

«Отлично» - теоретическое содержание курса освоено полностью, без пробелов, умения сформированы, все предусмотренные программой учебные задания выполнены, качество их выполнения оценено высоко.

–«Хорошо» - теоретическое содержание курса освоено полностью, без пробелов, некоторые умения сформированы недостаточно, все предусмотренные программой учебные задания выполнены, некоторые виды заданий выполнены с ошибками.

–«Удовлетворительно» - теоретическое содержание курса освоено частично, но пробелы не носят существенного характера, необходимые умения работы с освоенным материалом в основном сформированы, большинство предусмотренных программой обучения учебных заданий выполнено, некоторые из выполненных заданий содержат ошибки.

«Неудовлетворительно» - теоретическое содержание курса не освоено, необходимые умения не сформированы, выполненные учебные задания содержат грубые ошибки.

Лабораторное занятие №14. Обмен данными. Работа системных часов.

Цель работы:

- приобретение навыков обращения к системным характеристикам.

Выполнив работу, Вы будете:

уметь:

- У2. Создавать программу по разработанному алгоритму как отдельный модуль.
- У1.Осуществлять разработку кода программного модуля на языках низкого уровня и высокого уровней;
- У3. Выполнять отладку и тестирование программы на уровне модуля.
- У6. Оформлять документацию на программные средства.
- У8. Применять инструментальные средства отладки программного обеспечения.

Материальное обеспечение:

Мультимедийные средства хранения, передачи и представления информации.

Задание

Задача 1

Прочитать текущее значение секунд, часов реального времени из CMOS-памяти и вывести в виде десятичного числа на экран монитора.

Решение:

```
    MASM
    ;  Программа    r_cmos.asm
CODE_SEG    segment
            assume      cs:CODE_SEG
            org     100h
    start:
            mov     BL, 00
            mov     CX, 10  ; количество выводимых секунд
    @@1:    mov     AL, 0
            mov     DX, 70h
            out     DX, al

            mov     DX, 71h
            in      AL, DX  ; читаем текущее значение секунд
            cmp     AL, BL
            jz      @@1     ; переходим, пока
                            ; не изменится содержимое секунд
            mov     BL, AL
            mov     DL, AL
            call    write_BCD
            loop    @@1
            ret

    ; процедура вывода двоично-кодированного числа (BCD),
    ; находящегося в DL, на экран в десятичной форме

write_BCD   proc    near
            push    ax
            push    cx      ; сохранение используемых
                            ; в процедуре
            push    dx      ; регистров в стеке
            and     dl, 0f0h
            mov     cl, 4
            shr     dl, cl
            call    wr_cifra
            pop     dx
            and     dl, 0fh
            call    wr_cifra
            pop     cx      ; восстанавливаем регистры
                            ; из стека
            pop     ax
            ret
write_BCD   endp
    ; Подпрограмма вывода 1 цифры на экран с
    ; использованием 2 функции прерывания 21h
    ; цифра передается в регистре DL
    wr_cifra    proc    near
            push    ax
            add     dl, 30h
            mov     ah, 02
            int     21h
            pop     ax

            ret
    wr_cifra    endp
CODE_SEG    ends
            end     start
```

Содержание некоторых ячеек RTC и CMOS RAM:

00 – текущее значение секунд (показание часов реального времени) в формате BCD;
02 – минуты;
04 – часы;
06 – текущий день недели;
07 – день месяца;
08 – номер месяца;
09 – год (две цифры);
10h – тип гибких дисков;
12h – тип жестких дисков;
15h – размер базовой памяти; **17h** – размер расширенной памяти и т.д.

Порядок выполнения работы

- Составить математическую модель задачи.
- Выбрать и обосновать наиболее рациональный метод решения задачи;
- Разработать алгоритм для решения задачи.
- Написать и отладить программу.

Форма предоставления результата

- Блок – схема.
- Код программы.

Критерии оценки:

«Отлично» - теоретическое содержание курса освоено полностью, без пробелов, умения сформированы, все предусмотренные программой учебные задания выполнены, качество их выполнения оценено высоко.

–«Хорошо» - теоретическое содержание курса освоено полностью, без пробелов, некоторые умения сформированы недостаточно, все предусмотренные программой учебные задания выполнены, некоторые виды заданий выполнены с ошибками.

–«Удовлетворительно» - теоретическое содержание курса освоено частично, но пробелы не носят существенного характера, необходимые умения работы с освоенным материалом в основном сформированы, большинство предусмотренных программой обучения учебных заданий выполнено, некоторые из выполненных заданий содержат ошибки.

«Неудовлетворительно» - теоретическое содержание курса не освоено, необходимые умения не сформированы, выполненные учебные задания содержат грубые ошибки.

Лабораторное занятие №15. Обмен данными. Использование счетчика тактов процессора в качестве таймера.

Цель работы:

- приобретение навыков выполнения операций в различных системах счисления.

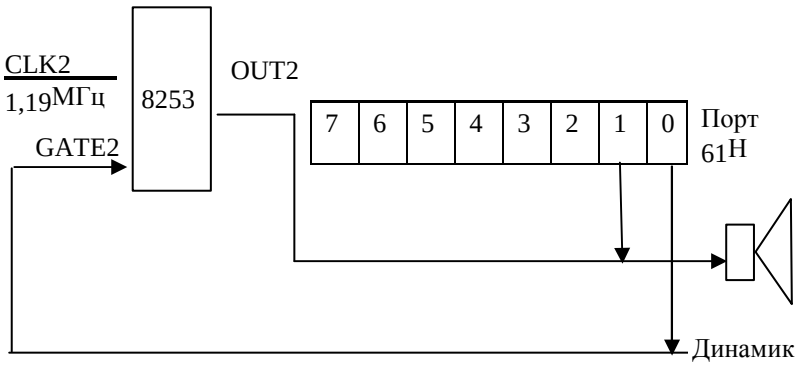
Выполнив работу, Вы будете:

уметь:

- У2. Создавать программу по разработанному алгоритму как отдельный модуль.
- У1.Осуществлять разработку кода программного модуля на языках низкого уровня и высокого уровней;
- У3. Выполнять отладку и тестирование программы на уровне модуля.
- У6. Оформлять документацию на программные средства.
- У8. Применять инструментальные средства отладки программного обеспечения.

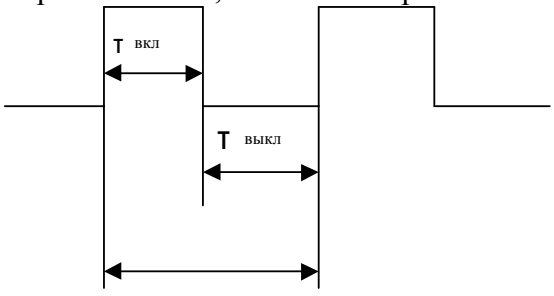
Материальное обеспечение:

- Мультимедийные средства хранения, передачи и представления информации.
- **Задание**
- Встроенный динамик – это системное устройство, поэтому в отличие от музыкальных сопроцессоров не требует никаких драйверов и доступно всегда. В архитектуре IBM PC существуют две возможности управлять «спикером»:
- с помощью первого бита порта 61h;
- при программировании второго канала таймера Intel 8253



– Рис. 1. Получение звука через PC Speaker

– Комбинируя эти возможности, можно получить специальные звуковые эффекты. Аппаратные возможности не позволяют изменить громкость звука, издаваемого динамиками.
– Системный порт 61h работает как на чтение, так и на запись.
– Ворота GATE2 закрываются записью «0» в нулевой бит 61h порта.
– Это позволяет производить ряд манипуляций со звуком. Первый бит порта 61h связан с PC speaker и также может быть использован для генерации звука. Спикер в отличие от звуковой платы является системным устройством, а не драйверным, и присутствует в любой компьютерной системе, поэтому он широко используется для сигнализации различных событий, программных ошибок и аппаратных сбоев.
– Генерация звука с помощью порта 61h
– Метод состоит в периодическом включении и выключении с желаемой частотой первого бита порта 61h. Если переключать значение бита с максимально возможной частотой, то мы вряд ли что услышим, так как получаемая частота будет довольно высокой.
– Слышимые человеком звуки ограничены диапазоном от 20 до 20000 Гц. Звуки, воспроизводимые SPEAKER, имеют еще более узкий диапазон 50–12000 Гц. Между переключениями необходимо вставлять задержки по времени τвкл, τвыкл. (рис. 4.5). Нулевой бит порта 61h управляет воротами канала 2 таймера, который связан с динамиком. Этот бит сбрасываем в 0, чтобы таймер не влиял на получаемый звук.



– Период T

– Рис. 2. Формирование прямоугольных импульсов, подаваемых на системный динамик: период получаемых колебаний T=τвкл +τвыкл; полученная частота звука f=1/T Гц

– Задача 6
– Составить программу для генерации звука посредством порта 61h.
– Задержки по времени осуществляются программно, поэтому, чем производительнее процессор, тем выше будет тон получаемого звука и

меньше времени он будет продолжаться. Фрагмент этой программы можно использовать как простейший тест быстродействия компьютера.

- Решение:
- ; Основной фрагмент программы zvuk_prt.asm

```
cli             ; запрет аппаратных прерываний
mov     DX, 1000    ; длительность тонов в периодах
in      AL, 61h     ; получаем значение порта 61H
and     AL, 1111111Ob   ; отключение
metka1:     динамика от 8253
or      AL, 00000010b  ; включим динамик
out     61h, AL
```

- ; задержка на половину периода
- MOV CX, 300h metka2:
- LOOP metka2
- AND AL, 11111101b ; выключение динамика
- OUT 61h, AL
- ; задержка на половину периода
- MOV CX, 300h metka3:
- LOOP metka3
- DEC DX
- JNZ metka1
- STI ; разрешить аппаратные прерывания
- Генерация звука посредством таймера
- Более предпочтительным является способ получения звука посредством таймера, т.к. процессор «не участвует» в получении самого звука и может заниматься другой работой.
- Задача 7
- Получить звук с частотой f=440 Гц. Звук прекратить нажатием любой клавиши на клавиатуре.
- Решение:
- ;Основной фрагмент программы zvuk_t.asm

```
; Биты 0 и 1 порта 61h предварительно необходимо установить в «1»

in      AL, 61h     ; читаем порт 61H
or      AL, 00000011b  ; установление двух ;
    младших битов в «1»
out     61h, AL
mov     AL, 10110110b   ; управляющий
    регистр для 8253
out     43h, al
mov     AX, A91h    ;
    1190000/440=2705=A91h
out     42h, AL     ; посылаем младший
    байт
mov     AL, AH
out     42h, AL     ; посылаем старший
    байт
```

–
– ; слышим звук с частотой 440 Гц
–
– mov AX, 0C08h ; очистка буфера,
– ; ввод символа без ЭХА на экран
– int 21h ; ждем нажатие клавиши in AL, 61h
– and AL, 11111100b ; сбрасываем два младших бита
– out 61h, AL
– ; звук выключается

Порядок выполнения работы

- Составить математическую модель задачи.
- Выбрать и обосновать наиболее рациональный метод решения задачи;
- Разработать алгоритм для решения задачи.
- Написать и отладить программу.

Форма предоставления результата

- Блок – схема.
- Код программы.

Критерии оценки:

«Отлично» - теоретическое содержание курса освоено полностью, без пробелов, умения сформированы, все предусмотренные программой учебные задания выполнены, качество их выполнения оценено высоко.

–«Хорошо» - теоретическое содержание курса освоено полностью, без пробелов, некоторые умения сформированы недостаточно, все предусмотренные программой учебные задания выполнены, некоторые виды заданий выполнены с ошибками.

–«Удовлетворительно» - теоретическое содержание курса освоено частично, но пробелы не носят существенного характера, необходимые умения работы с освоенным материалом в основном сформированы, большинство предусмотренных программой обучения учебных заданий выполнено, некоторые из выполненных заданий содержат ошибки.

«Неудовлетворительно» - теоретическое содержание курса не освоено, необходимые умения не сформированы, выполненные учебные задания содержат грубые ошибки.

Лабораторное занятие №16. Обмен данными. Управление звуком.

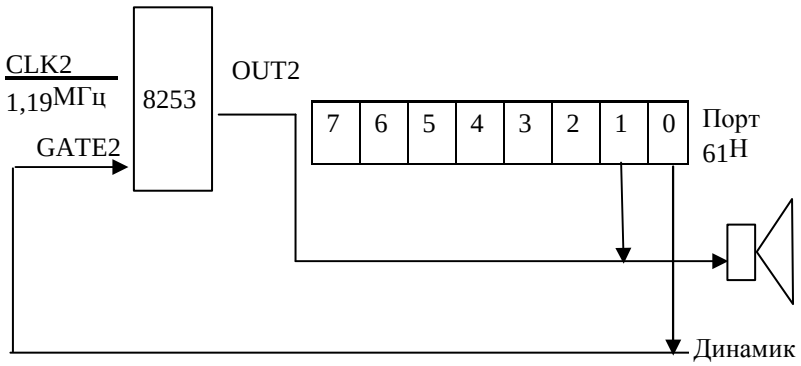
Цель работы:

- приобретение навыков выполнения операций в различных системах счисления.

Выполнив работу, Вы будете:

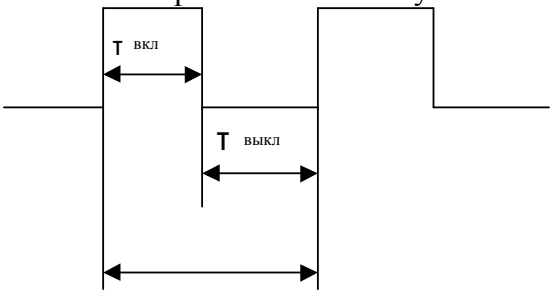
уметь:

- У2. Создавать программу по разработанному алгоритму как отдельный модуль.
- У1.Осуществлять разработку кода программного модуля на языках низкого уровня и высокого уровней;
- У3. Выполнять отладку и тестирование программы на уровне модуля.
- У6. Оформлять документацию на программные средства.
- У8. Применять инструментальные средства отладки программного обеспечения.
- **Задание**
- Встроенный динамик – это системное устройство, поэтому в отличие от музыкальных сопроцессоров не требует никаких драйверов и доступно всегда. В архитектуре IBM PC существуют две возможности управлять «спикером»:
- с помощью первого бита порта 61h;
- при программировании второго канала таймера Intel 8253



– Рис. 1. Получение звука через PC Speaker

– Комбинируя эти возможности, можно получить специальные звуковые эффекты. Аппаратные возможности не позволяют изменить громкость звука, издаваемого динамиками.
– Системный порт 61h работает как на чтение, так и на запись.
– Ворота GATE2 закрываются записью «0» в нулевой бит 61h порта.
– Это позволяет производить ряд манипуляций со звуком. Первый бит порта 61h связан с PC speaker и также может быть использован для генерации звука. Спикер в отличие от звуковой платы является системным устройством, а не драйверным, и присутствует в любой компьютерной системе, поэтому он широко используется для сигнализации различных событий, программных ошибок и аппаратных сбоев.
– Генерация звука с помощью порта 61h
– Метод состоит в периодическом включении и выключении с желаемой частотой первого бита порта 61h. Если переключать значение бита с максимально возможной частотой, то мы вряд ли что услышим, так как получаемая частота будет довольно высокой.
– Слышимые человеком звуки ограничены диапазоном от 20 до 20000 Гц. Звуки, воспроизводимые SPEAKER, имеют еще более узкий диапазон 50–12000 Гц. Между переключениями необходимо вставлять задержки по времени τвкл, τвыкл. (рис. 4.5). Нулевой бит порта 61h управляет воротами канала 2 таймера, который связан с динамиком. Этот бит сбрасываем в 0, чтобы таймер не влиял на получаемый звук.



– Период T

– Рис. 2. Формирование прямоугольных импульсов, подаваемых на системный динамик: период получаемых колебаний T=τвкл +τвыкл; полученная частота звука f=1/T Гц

– Задача 6
– Составить программу для генерации звука посредством порта 61h.
– Задержки по времени осуществляются программно, поэтому, чем производительнее процессор, тем выше будет тон получаемого звука и меньше времени он будет продолжаться. Фрагмент этой программы можно использовать как простейший тест быстродействия компьютера.

– Решение:
– ; Основной фрагмент программы zvuk_prt.asm

```
cli                 ; запрет аппаратных
                    прерываний
mov     DX, 1000    ; длительность тонов в
                    периодах
in      AL, 61h     ; получаем значение
                    порта 61H
and     AL, 1111111Ob   ; отключение
metka1:             динамика от 8253
or      AL, 00000010b  ; включим динамик
out     61h, AL
```

– ; задержка на половину периода
– MOV CX, 300h metka2:
– LOOP metka2
– AND AL, 11111101b ; выключение динамика
– OUT 61h, AL
– ; задержка на половину периода
– MOV CX, 300h metka3:
– LOOP metka3
– DEC DX
– JNZ metka1
– STI ; разрешить аппаратные прерывания
– Генерация звука посредством таймера
– Более предпочтительным является способ получения звука посредством таймера, т.к. процессор «не участвует» в получении самого звука и может заниматься другой работой.
– Задача 7
– Получить звук с частотой f=440 Гц. Звук прекратить нажатием любой клавиши на клавиатуре.
– Решение:
– ;Основной фрагмент программы zvuk_t.asm

```
; Биты 0 и 1 порта 61h предварительно необходимо
установить в «1»

in      AL, 61h     ; читаем порт 61H
or      AL, 00000011b  ; установление двух  ;
        младших битов в «1»
out     61h, AL
mov     AL, 10110110b   ; управляющий
        регистр для 8253
out     43h, al
mov     AX, A91h    ;
        1190000/440=2705=A91h
out     42h, AL     ; посылаем младший
        байт
mov     AL, AH
out     42h, AL     ; посылаем старший
        байт
```

–
– ; слышим звук с частотой 440 Гц

–
– mov AX, 0C08h ; очистка буфера,
– ; ввод символа без ЭХА на экран
– int 21h ; ждем нажатие клавиши in AL, 61h
– and AL, 11111100b ; сбрасываем два младших бита
– out 61h, AL
– ; звук выключается

Материальное обеспечение:

– Мультимедийные средства хранения, передачи и представления информации.

Порядок выполнения работы

– Составить математическую модель задачи.
– Выбрать и обосновать наиболее рациональный метод решения задачи;
– Разработать алгоритм для решения задачи.
– Написать и отладить программу.

Форма предоставления результата

– Блок – схема.
– Код программы.

Критерии оценки:

«Отлично» - теоретическое содержание курса освоено полностью, без пробелов, умения сформированы, все предусмотренные программой учебные задания выполнены, качество их выполнения оценено высоко.

–«Хорошо» - теоретическое содержание курса освоено полностью, без пробелов, некоторые умения сформированы недостаточно, все предусмотренные программой учебные задания выполнены, некоторые виды заданий выполнены с ошибками.

–«Удовлетворительно» - теоретическое содержание курса освоено частично, но пробелы не носят существенного характера, необходимые умения работы с освоенным материалом в основном сформированы, большинство предусмотренных программой обучения учебных заданий выполнено, некоторые из выполненных заданий содержат ошибки.

«Неудовлетворительно» - теоретическое содержание курса не освоено, необходимые умения не сформированы, выполненные учебные задания содержат грубые ошибки.

Лабораторное занятие №17. Обмен данными. Программирование мыши.

Цель работы:

– приобретение навыков программирования мыши.

Выполнив работу, Вы будете:

уметь:

– У2. Создавать программу по разработанному алгоритму как отдельный модуль.
– У1.Осуществлять разработку кода программного модуля на языках низкого уровня и высокого уровней;
– У3. Выполнять отладку и тестирование программы на уровне модуля.
– У6. Оформлять документацию на программные средства.
– У8. Применять инструментальные средства отладки программного обеспечения.

Материальное обеспечение:

– Мультимедийные средства хранения, передачи и представления информации.

Задание:

1 Должны быть реализованы три потока, каждый из которых осуществляет передвижение собственной надписи по главному окну. Все надписи должны быть различны и двигаться с разной скоростью.

2. Главное окно должно быть разделено на четыре части. Также должны быть созданы четыре потока, каждый из которых раз в секунду изменяет цвет фона в своей части окна.

1.Запустить несколько заданий (например, команд просмотра файлов less), возвращаясь в командную строку комбинацией клавиш Ctrl-Z и изучить действие команд ps, jobs, fg, bg, kill, killall.

2.Обеспечить синхронизацию процессов и передачу данных между ними на примере двух приложений «клиент» и «сервер», создав два процесса (два исполняемых файла) – процесс «клиент» (первый исполняемый файл) и процесс «сервер» (второй исполняемый файл). С помощью механизмов межпроцессного взаимодействия обеспечить передачу информации от «клиента» к «серверу» и наоборот. В качестве типа передаваемой информации можно использовать: данные, вводимые с клавиатуры; данные, считываемые из файла; данные, генерируемые случайным образом и
т.п.

3.Обмен данными между процессами «клиент»-«сервер» осуществить следующим образом:

- с использованием программных каналов (именованных либо неименованных, по указанию преподавателя);
- с использованием (по указанию преподавателя) одного из перечисленных вариантов:
- разделяемая память (обязательна синхронизация процессов, например с помощью семафоров);
- очередь сообщений.

1. Создать приложение, состоящее из двух процессов:

−первый процесс записывает текстовый файл и растровый рисунок в буфер обмена;
−второй процесс считывает информацию из буфера обмена и отображает в окне процесса.

Текстовый файл и файл с растровым рисунком взять из предыдущих лабораторных работ

Порядок выполнения работы

- Составить математическую модель задачи.
- Выбрать и обосновать наиболее рациональный метод решения задачи;
- Разработать алгоритм для решения задачи.
- Написать и отладить программу.

Форма предоставления результата

- Блок – схема.
- Код программы.

Критерии оценки:

«Отлично» - теоретическое содержание курса освоено полностью, без пробелов, умения сформированы, все предусмотренные программой учебные задания выполнены, качество их выполнения оценено высоко.

−«Хорошо» - теоретическое содержание курса освоено полностью, без пробелов, некоторые умения сформированы недостаточно, все предусмотренные программой учебные задания выполнены, некоторые виды заданий выполнены с ошибками.

−«Удовлетворительно» - теоретическое содержание курса освоено частично, но пробелы не носят существенного характера, необходимые умения работы с освоенным материалом в основном сформированы, большинство предусмотренных программой

обучения учебных заданий выполнено, некоторые из выполненных заданий содержат ошибки.

–«Неудовлетворительно» - теоретическое содержание курса не освоено, необходимые умения не сформированы, выполненные учебные задания содержат грубые ошибки.

Лабораторное занятие№ 18. Сетевое программирование сокетов. Реализация архитектуры клиент-сервер на основе интерфейса сокетов Windows Sockets API.

Цель работы:

- Научиться формировать алгоритмы разработки программных модулей в соответствии с техническим заданием;
- Разрабатывать программные модули в соответствии с техническим заданием;
- Выполнять отладку программных модулей с использованием специализированных программных средств;
- Выполнять тестирование программных модулей;
- Осуществлять рефакторинг и оптимизацию программного кода;
- Разрабатывать модули программного обеспечения для мобильных платформ.

Выполнив работу, Вы будете:

уметь:

- У2. Создавать программу по разработанному алгоритму как отдельный модуль.
- У1.Осуществлять разработку кода программного модуля на языках низкого уровня и высокого уровней;
- У3. Выполнять отладку и тестирование программы на уровне модуля.
- У6. Оформлять документацию на программные средства.
- У8. Применять инструментальные средства отладки программного обеспечения.

Материальное обеспечение:

- Мультимедийные средства хранения, передачи и представления информации.

Краткие теоретические сведения:

Изучить:

- возможности реализации архитектуры клиент-сервер на основе интерфейса сокетов Windows Sockets API;
- типы сокетов TCP/IP;
- основные методики и API-функции для разработки сетевых приложений с использованием Winsock.

1. Постановка задачи

2.Изучить методические указания к лабораторной работе, материалы лекций и рекомендуемую литературу.

3.Разработать консольное клиент-серверное приложение, демонстрирующее взаимодействие на основе потоковых сокетов.

4.Разработать консольное клиент-серверное приложение, демонстрирующее взаимодействие на основе дейтаграммных сокетов.

Задание 1.

Разработать TCP-сервер, создающий сокет, привязывающий его к локальному IP-адресу и порту и прослушивающий соединения клиентов. Номер порта и IP-адрес вводить с клавиатуры. IP-адрес задавать в десятично-точечной нотации.

Учесть, что функция accept возвращает новый дескриптор сокета, соответствующий принятому клиентскому соединению. Для всех последующих операций с данным клиентским соединением должен применяться новый сокет. Исходный прослушивающий сокет используется для приема других клиентских соединений и

продолжает находиться в режиме прослушивания. При получении от клиента запроса на установление соединения вывести на экран IP-адрес и номер порта клиентского сокета.

Разработать приложение–клиент, соединяющееся с заданным TCP-сервером. Все отправленные и полученные по сокету данные вывести на экран.

При вызове API-функций выполнять проверку и обработку ошибок.

Нарисовать блок-схему серверной и клиентской части программы.

Задание 2.

Разработать серверное приложение, выполняющее получение данных через сокет без установления соединения по протоколу UDP.

Разработать клиентское приложение, выполняющее отправку UDP-дейтаграмм. IP-адрес и порт удаленного получателя должен задаваться пользователем.

При вызове API-функций выполнять проверку и обработку ошибок.

Адреса сокетов вывести на экран.

Нарисовать блок-схему серверной и клиентской части программы.

2. Написать программные коды
3. Запустить и отладить программу

Варианты заданий:

1) Удаленный калькулятор (+,-,*,/);
2) Работа с массивом чисел (количество элементов в массиве, получение значения элемента массива по номеру, найти номер элемента в массиве по значению, увеличить значения элементов на заданное число);

Порядок выполнения работы

- Составить математическую модель задачи.
- Выбрать и обосновать наиболее рациональный метод решения задачи;
- Разработать алгоритм для решения задачи.
- Написать и отладить программу.

Форма предоставления результата

- Блок – схема.
- Код программы.

Критерии оценки:

«Отлично» - теоретическое содержание курса освоено полностью, без пробелов, умения сформированы, все предусмотренные программой учебные задания выполнены, качество их выполнения оценено высоко.

–«Хорошо» - теоретическое содержание курса освоено полностью, без пробелов, некоторые умения сформированы недостаточно, все предусмотренные программой учебные задания выполнены, некоторые виды заданий выполнены с ошибками.

–«Удовлетворительно» - теоретическое содержание курса освоено частично, но пробелы не носят существенного характера, необходимые умения работы с освоенным материалом в основном сформированы, большинство предусмотренных программой обучения учебных заданий выполнено, некоторые из выполненных заданий содержат ошибки.

–«Неудовлетворительно» - теоретическое содержание курса не освоено, необходимые умения не сформированы, выполненные учебные задания содержат грубые ошибки.

Лабораторное занятие. Основные методики для разработки сетевых приложений с использованием Winsock

Цель работы:

- приобретение навыков разработки сетевых приложений с использованием Winsock.

Выполнив работу, Вы будете:

уметь:

- У2. Создавать программу по разработанному алгоритму как отдельный модуль.
- У1.Осуществлять разработку кода программного модуля на языках низкого уровня и высокого уровней;
- У3. Выполнять отладку и тестирование программы на уровне модуля.
- У6. Оформлять документацию на программные средства.
- У8. Применять инструментальные средства отладки программного обеспечения.

Материальное обеспечение:

- Мультимедийные средства хранения, передачи и представления информации.

Задание1:

Создайте сценарий WMI, выполняющий запись в файл сведений о количестве процессоров и скорости процессора, о размерах КЭШей различных уровней.

Задание2:

Использовать классы - Win32_Processor, Win32_CacheMemory

Создайте сценарий WMI, выполняющий запись в файл сведений о скринсейвере и разрешении экрана, наименование клавиатуры и количество функциональных клавиш.

Использовать классы - Клавиатура (Win32_Keyboard), Монитор (Win32_DesktopMonitor)

Порядок выполнения работы

- Составить математическую модель задачи.
- Выбрать и обосновать наиболее рациональный метод решения задачи;
- Разработать алгоритм для решения задачи.
- Написать и отладить программу.

Форма предоставления результата

- Блок – схема.
- Код программы.

Критерии оценки:

«Отлично» - теоретическое содержание курса освоено полностью, без пробелов, умения сформированы, все предусмотренные программой учебные задания выполнены, качество их выполнения оценено высоко.

–«Хорошо» - теоретическое содержание курса освоено полностью, без пробелов, некоторые умения сформированы недостаточно, все предусмотренные программой учебные задания выполнены, некоторые виды заданий выполнены с ошибками.

–«Удовлетворительно» - теоретическое содержание курса освоено частично, но пробелы не носят существенного характера, необходимые умения работы с освоенным материалом в основном сформированы, большинство предусмотренных программой обучения учебных заданий выполнено, некоторые из выполненных заданий содержат ошибки.

–«Неудовлетворительно» - теоретическое содержание курса не освоено, необходимые умения не сформированы, выполненные учебные задания содержат грубые ошибки.

Лабораторное занятие №19. API-функции для разработки сетевых приложений с использованием Winsock

Цель работы:

- приобретение навыков разработки сетевых приложений с использованием Winsock.

Выполнив работу, Вы будете:

уметь:

- У2. Создавать программу по разработанному алгоритму как отдельный модуль.
- У1.Осуществлять разработку кода программного модуля на языках низкого уровня и высокого уровней;
- У3. Выполнять отладку и тестирование программы на уровне модуля.
- У6. Оформлять документацию на программные средства.
- У8. Применять инструментальные средства отладки программного обеспечения.

Материальное обеспечение:

- Мультимедийные средства хранения, передачи и представления информации.

Порядок выполнения работы

- Составить математическую модель задачи.
- Выбрать и обосновать наиболее рациональный метод решения задачи;
- Разработать алгоритм для решения задачи.
- Написать и отладить программу.

Форма предоставления результата

- Блок – схема.
- Код программы.

Критерии оценки:

«Отлично» - теоретическое содержание курса освоено полностью, без пробелов, умения сформированы, все предусмотренные программой учебные задания выполнены, качество их выполнения оценено высоко.

–«Хорошо» - теоретическое содержание курса освоено полностью, без пробелов, некоторые умения сформированы недостаточно, все предусмотренные программой учебные задания выполнены, некоторые виды заданий выполнены с ошибками.

–«Удовлетворительно» - теоретическое содержание курса освоено частично, но пробелы не носят существенного характера, необходимые умения работы с освоенным материалом в основном сформированы, большинство предусмотренных программой обучения учебных заданий выполнено, некоторые из выполненных заданий содержат ошибки.

«Неудовлетворительно» - теоретическое содержание курса не освоено, необходимые умения не сформированы, выполненные учебные задания содержат грубые ошибки.

Лабораторное занятие №20. Разработка серверного приложения, выполняющего получение данных через сокет без установления соединения по протоколу UDP

Цель работы:

- приобретение навыков разработки серверного приложения, выполняющего получение данных через сокет без установления соединения по протоколу UDP.

Выполнив работу, Вы будете:

уметь:

- У2. Создавать программу по разработанному алгоритму как отдельный модуль.
- У1.Осуществлять разработку кода программного модуля на языках низкого уровня и высокого уровней;

- У3. Выполнять отладку и тестирование программы на уровне модуля.
- У6. Оформлять документацию на программные средства.
- У8. Применять инструментальные средства отладки программного обеспечения.

Материальное обеспечение:

- Мультимедийные средства хранения, передачи и представления информации.

Порядок выполнения работы

- Составить математическую модель задачи.
- Выбрать и обосновать наиболее рациональный метод решения задачи;
- Разработать алгоритм для решения задачи.
- Написать и отладить программу.

Форма предоставления результата

- Блок – схема.
- Код программы.

Критерии оценки:

«Отлично» - теоретическое содержание курса освоено полностью, без пробелов, умения сформированы, все предусмотренные программой учебные задания выполнены, качество их выполнения оценено высоко.

–«Хорошо» - теоретическое содержание курса освоено полностью, без пробелов, некоторые умения сформированы недостаточно, все предусмотренные программой учебные задания выполнены, некоторые виды заданий выполнены с ошибками.

–«Удовлетворительно» - теоретическое содержание курса освоено частично, но пробелы не носят существенного характера, необходимые умения работы с освоенным материалом в основном сформированы, большинство предусмотренных программой обучения учебных заданий выполнено, некоторые из выполненных заданий содержат ошибки.

«Неудовлетворительно» - теоретическое содержание курса не освоено, необходимые умения не сформированы, выполненные учебные задания содержат грубые ошибки.

Лабораторное занятие №21. Разработка приложения с графическим интерфейсом

Цель работы:

- приобретение навыков разработки приложения с графическим интерфейсом.

Выполнив работу, Вы будете:

уметь:

- У2. Создавать программу по разработанному алгоритму как отдельный модуль.
- У1.Осуществлять разработку кода программного модуля на языках низкого уровня и высокого уровней;
- У3. Выполнять отладку и тестирование программы на уровне модуля.
- У6. Оформлять документацию на программные средства.
- У8. Применять инструментальные средства отладки программного обеспечения.

Материальное обеспечение:

- Мультимедийные средства хранения, передачи и представления информации.

Задание

Все возможности видеосистемы компьютера можно реализовать с помощью видеофункций BIOS прерывания int 10h. Прерывание int 10h обеспечивает: смену

видеорежима (текстовый или графический); вывод символьной и текстовой информации; смену шрифтов, настройку цветовой палитры, работу с графическим изображением. Программирование видеосистемы с помощью средств BIOS более громоздко, однако большие возможности и высокая скорость вывода обуславливают широкое использование этого метода в прикладных программах.

В данной работе рассматриваются функции BIOS для обслуживания видеосистемы компьютера, а также функции для работы с клавиатурой. Перечислим функции, являющиеся предметом рассмотрения в лабораторной работе.

Int 10h:

- функция 00h - установка видеорежима;
- функция 02h - установка позиции курсора;
- функция 03h - считывание позиции и размера курсора;
- функция 05h - установка видеостраницы;
- функция 06h (07h) - инициализация или прокрутка окна вверх (вниз);
- функция 08h - чтение символа и атрибута в позиции курсора;
- функция 09h - запись символа и атрибута в позицию курсора;
- функция 0Ah - запись символа в позицию курсора с текущим атрибутом;
- функция 0Eh - запись символа в режиме телетайпа с текущим атрибутом;
- функция 0Fh - получить режим дисплея;
- функция 1003h - переключение назначения старшего бита байта атрибута: мерцание/яркость,
- функция 13h - запись строки с заданным атрибутом в режиме телетайпа.

Int 16h:

- функция 00h (10h) - чтение символа с клавиатуры с ожиданием;
- функция 01h(11h)- проверка буфера клавиатуры на наличие в нём символа;
- функция 02h (12h) - получение флагов (расширенной) клавиатуры.

2.3.2. Прямое программирование видеобуфера в текстовом режиме

Современные видеоконтроллеры поддерживают разнообразные текстовые и графические режимы. Текстовые режимы различаются по разрешению (число отображаемых символов по горизонтали и вертикали) и цветовой палитре (монохромный или 16-цветный режим). Для графических режимов основным признаком классификации является количество одновременно отображаемых цветов и, соответственно, количество бит видеопамяти, отводимое на каждую точку (пиксел) изображения. Различают следующие типы графических режимов:

- монохромный (1-битное кодирование);
- 16-цветный *EGA/VGA* (4-битное кодирование);
- 256-цветный *SVGA* (8-битное кодирование);
- *HiColor* (16-битное кодирование);

- *True Color* (24-битное / 32-битное кодирование).

Всё, что изображено на мониторе - графика, текст - одновременно присутствует в памяти, встроенной в видеоадаптер. Для того чтобы изображение появилось на мониторе, оно должно быть записано в память видеоадаптера.

На экране отображается видеобуфер, соответствующий активной странице. В текстовых режимах для изображения каждого символа отводится 2 байта: байт с ASCII-кодом символа и байт с его атрибутом. Вообще при формировании изображения непосредственно в видеобуфере, в обход программ DOS и BIOS, все управляющие коды ASCII теряют свои управляющие функции и отображаются в виде соответствующих символов. Структура байта атрибутов приведена на рис. 2.1.

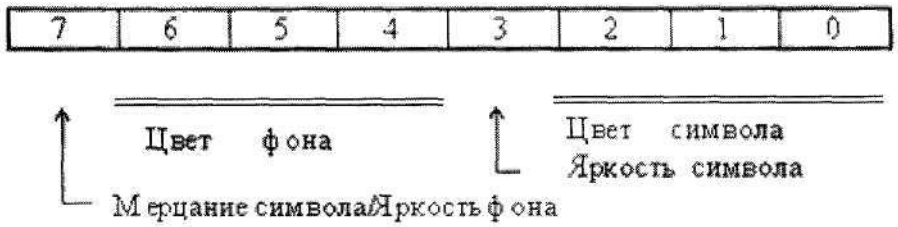


Рис. 2.1.- Структура байта атрибутов

Из рис. 2.1 следует, что каждый символ может принимать любой из 16 возможных цветов, определяемых сочетанием младших 4-х битов. Биты 4-6 байта атрибутов задают цвет фона под данным символом. Последний бит 7, в зависимости от режима видеоадаптера, определяет либо яркость фона под данным символом (тогда фон также может принимать 16 разных цветов), либо мерцание символа (устанавливается DOS no умолчанию).

При загрузке машины устанавливается стандартная палитра, коды цветов которой приведены в табл. 2.1. Рассмотрим некоторые примеры. Так, в режиме мерцания значение старшего полубайта атрибута 8h обозначает не серый фон, а чёрный при мерцающем символе, цвет которого по-прежнему определяется младшим полубайтом; значение старшего полубайта 0Ch - красный фон при мерцающем символе. Переключение назначения бита 7 осуществляется подфункцией 03h функции 10h прерывания int 10h.

Таблица 2.1 – Коды цветов стандартной палитры

Код	Цвет	Код	Цвет
0h	Чёрный	8h	Серый
lh	Синий	9h	Голубой
2h	Зелёный	0Ah	Салатовый
3h	Бирюзовый	0Bh	Светло-бирюзовый
4h	Красный	0Ch	Розовый
5h	Фиолетовый	0Dh	Светло-фиолетовый
6h	Коричневый	0Eh	Жёлтый
7h	Белый	0Fh	Ярко- белый

Двухбайтовые коды символов записываются в видеобуфер в том порядке, в каком они должны появиться на экране: первые 80*2 байт соответствуют первой строке экрана, вторые 80*2 байт - второй и т.д. При этом переход на следующую строку экрана определяется не управляющими кодами возврата каретки и перевода строки, а размещением кода в другом месте видеобуфера. Вычислить смещение ячейки в координатах "строка-столбец" (row, clm) можно так:

VidAddr= (row* 160) + (clm* 2)

При большом объёме выводимых данных, информационный кадр формируется заранее в буфере пользователя, располагающегося в сегменте данных программы.

2.3.3. Прерывание int 10h. Видеофункции BIOS

Функция 00h. Установка видеорежима (табл. 2.2) текущей видеостраницы с очисткой экрана *(быстрая очистка экрана реализуется функцией 06h и 07h, полная очистка экрана 00h).*

Вызов: *AH = 00h,*

Al = видеорежим (код режима задаётся в младших 7 битах, установка в 1 старшего бита запрещает очистку экрана).

Регистры *AX, BP, SI, и DI* – не используются в данной функции.

Функция 02h. Установка позиции курсора.

Задаёт положение курсора на экране в текстовых координатах, с которых в дальнейшем будет выводиться текст. Отсчёт номера строки и столбца ведётся от верхнего левого угла. Курсор можно установить как в текстовом, так и в графическом режиме, однако, в графическом режиме курсор не виден. BIOS поддерживает до восьми независимых курсоров - по одному на каждую страницу (см. табл. 2.2) независимо от того, какая страница является активной. Функцию *02h* BIOS можно использовать в комбинации с функциями DOS для организации вывода на экран.

Вызов: *AH = 02h; BH* = номер страницы (0,1,...7), **обычно 0;** *DH* = строка; *DL* = столбец.

Регистры *AX, BP, SI* и *DI* – не используются в данной функции.

Функция 03h. Считывание позиции и размера курсора.

Возвращает текущие координаты состояния курсора на выбранной странице. Это даёт возможность временно перейти для работы на другое место экрана, а затем вернуться на старое место. Функцию *03h* BIOS можно использовать в комбинации с функциями DOS для организации вывода на экран.

Вызов: *AH = 03h, BH* = номер страницы (0,1,...7), **обычно 0.**

Возврат: *DH, DL* = строка и столбец текущей позиции курсора, *CH, CL* = первая и последняя строки развёртки курсора – размер курсора.

Регистры AX, BP, SI и DI – не используются в данной функции.

Функция 08h. Чтение символа и атрибута в текущей позиции курсора на выбранной странице.

Вызов: *AH = 08h, (цвет) BH* = номер страницы (0,...,7), **обычно 0.**

Возврат: *AH* = атрибут символа, *AL = ASCII-код* символа.

Регистры *BP, SI* и *D* – не используются в данной функции.

Функция 09h. Запись символа с заданным атрибутом на экран в позицию курсора. Действует как в графическом, так и в текстовом режимах. В графическом режиме символы не должны переходить на следующую строку. *Все коды в AL рассматриваются как символьные и не управляют положением курсора. После вывода символа курсор смещается к следующей позиции функцией 02h.* Коэффициент повторения позволяет выводить строки одинаковых символов. В текстовом режиме символ выводится с указанным в *BL* атрибутом. В графическом - содержимое *BL* влияет только на цвет символа, но не на фон под ним. Графическое изображение под знакоместом затирается.

Вызов: *AH =09h, AL = ASCII-код* символа,

BL — атрибут символа (текстовый режим) или только цвет символа (графический режим),

BH = номер страницы (0,1,...7), *CX*= коэффициент повторения.

Регистры *AX, BP, SI* и *DI* – не используются в данной функции.

Функция 0Ah. Запись символа с текущим атрибутом на экран в позицию курсора. Функция действует как в графическом, так и в текстовом режимах. Символ принимает атрибут, установленный ранее для этой позиции. *Все ASCII-коды в AL рассматриваются как символьные и не управляют положением курсора (также как и в функции 09h). После вывода символа курсор смещается к следующей позиции функцией 02h.*

Вызов: *AH = 0Ah, AL = ASCII-код* символа,

BH = номер страницы (0,1,...7), *CX* = коэффициент повторения.

Регистры *AX, BP, SI* и *DI* – не используются в данной функции.

Функция 0Fh. Получить режим дисплея и номер текущей страницы.

Вызов: *AH = 0Fh.*

Возврат: *AL* = режим дисплея, *AH* = ширина экрана в текстовом формате

BH =номер активной страницы.

Регистры *BP, SI* и *DI* – не используются в данной функции.

Пример. Процедура установки позиции курсора на текущей странице.

Вход: *dh* = строка (0 - 25), *dl* = столбец (0 - 79)

```
Proc SetCursor
;Сохранить регистры (по необходимости)
Mov ah,0Fh
Int 10h
Mov ah,02h
Int l0h
;восстановить регистры
Endp SetCursor
```

Функция 10h. Подфункция *03h.* Переключение бита "мерцание/яркость".

Определяет назначение старшего бита 7 атрибута символа: мерцание символа или повышенная яркость фона.

Вызов: *AX*= 1003h, *BL* = назначение 7-го бита атрибута:

0 - повышенная яркость, 1 - мерцание *(устанавливается по умолчанию).*

Функция воздействует сразу на все символы экрана, у которых установлен старший бит атрибута фона.

Функция 13h. Запись строки символов с заданными атрибутами.

Записывает строку в текущую страницу видеобуфера, начиная с указанной позиции. Коды *ASCII: 07h* - звонок, *08h* - шаг назад, *0Ah* - перевод строки, *0Dh* - возврат каретки, рассматриваются как управляющие, остальные - как символьные.

Вызов: *AH = I3h, AL* = режим записи:

1. — атрибут символа в *BL,* строка содержит только коды символов, после записи курсор принимает исходное положение (т.е. вывод следующей строки, если не изменить позицию курсора, начинается с изначально установленной позиции);
2. - отличается от режима 0 тем, что после записи курсор остаётся в конце строки;
3. - строка содержит попеременно коды символов и атрибутов (т.е. каждый символ описывается 2 байтами — *ASCII-кодом* и атрибутом), после записи курсор принимает исходное положение;
4. - отличается от режима 2 тем, что по окончании вывода курсор остаётся в конце строки.

BH = номер страницы (0,1,.. .7), BL = атрибут для режимов 0 и 1,

CX - длина символьной строки (в длину входят только коды символов, но не байты атрибутов),

DX = DH.DL — координаты курсора (строка, столбец) в исходной точке вывода строки на экране,

ES:BP = адрес начала строки в памяти.

Пример программы индивидуального задания.

Нарисовать прямоугольник зелёного цвета в любом месте экрана

```
.model small
.stack
100h
VGA_mode equ 13h ; 320x200 256 цветный графический режим
color equ 2 ; цвет линий
```

```
x_sise equ 300 ; ширина прямоугольника в пикселах
y_sise equ 100 ; высота прямоугольника в пикселах
x_pos equ 10 ; положение нижнего левого угла прямоуг-ка
y_pos equ 50
.code
start:
set_mode:
mov ah,00h ; вызов нулевой функции BIOS
mov al,VGA_mode ; и инициализация графического режима
int 10h
set_proc:
mov ah,0Ch ; настройка параметров для вызова функции 0Ch
mov al,color
mov cx,x_pos
mov dx,y_pos
line_1: ;
int 10h
inc cx
cmp cx,(x_pos + x_sise)
jne line_1
line_2: ;
int
int10h
inc dx
cmp dx,(y_pos + y_sise)
jne line_2;
line_3: ;
int 10h
dec cx
cmp cx,x_pos
jne line_3;
line_4: ;
int 10h
dec dx
cmp dx,y_pos
jne line_4
anykey: ; блок отвечающий за завершение приложения
mov ah,1 ; при нажатии любой клавиши
int 16h ; вызов 16h прерывания BIOS, определения
jz anykey ; наличия введенного символа
int 21h
end start ; завершене программы
```

2.4. Варианты индивидуального задания:

1. Разработать программу рисования зеленого треугольника в любом месте экрана.
2. Разработать программу рисования красного прямоугольного треугольника в нижнем правом углу экрана.
3. Разработать программу рисования синего равнобедренного треугольника в верхнем правом углу экрана.
4. Разработать программу рисования белого ромба в любом месте экрана.
5. Разработать программу рисования желтого квадрата в центре экрана.
6. Разработать программу рисования белой трапеции в любом месте экрана.

7. Разработать программу рисования синего параллелограмма в любом месте экрана.
8. Разработать программу рисования двух любых букв русского алфавита в любом месте экрана.

Разработать программу рисования красного закрашенного квадрата в центре экрана.

Порядок выполнения работы

- Составить математическую модель задачи.
- Выбрать и обосновать наиболее рациональный метод решения задачи;
- Разработать алгоритм для решения задачи.
- Написать и отладить программу.

Форма предоставления результата

- Блок – схема.
- Код программы.

Критерии оценки:

«Отлично» - теоретическое содержание курса освоено полностью, без пробелов, умения сформированы, все предусмотренные программой учебные задания выполнены, качество их выполнения оценено высоко.

–«Хорошо» - теоретическое содержание курса освоено полностью, без пробелов, некоторые умения сформированы недостаточно, все предусмотренные программой учебные задания выполнены, некоторые виды заданий выполнены с ошибками.

–«Удовлетворительно» - теоретическое содержание курса освоено частично, но пробелы не носят существенного характера, необходимые умения работы с освоенным материалом в основном сформированы, большинство предусмотренных программой обучения учебных заданий выполнено, некоторые из выполненных заданий содержат ошибки.

«Неудовлетворительно» - теоретическое содержание курса не освоено, необходимые умения не сформированы, выполненные учебные задания содержат грубые ошибки.

Лабораторное занятие №22. Разработка мультимедийного приложения

Цель работы:

- приобретение навыков выполнения операций в различных системах счисления.

Выполнив работу, Вы будете:

уметь:

- У2. Создавать программу по разработанному алгоритму как отдельный модуль.
- У1.Осуществлять разработку кода программного модуля на языках низкого уровня и высокого уровней;
- У3. Выполнять отладку и тестирование программы на уровне модуля.
- У6. Оформлять документацию на программные средства.
- У8. Применять инструментальные средства отладки программного обеспечения.

Материальное обеспечение:

- Мультимедийные средства хранения, передачи и представления информации.

Задание

Программы, которые позволяют объединить отдельные кусочки в единое законченное целое мультимедиа-приложение, можно условно разделить на три группы:

- специализированные программы, предназначенные для быстрой подготовки определенных типов мультимедиа-приложений;
- авторские инструментальные средства мультимедиа;

- языки программирования.

Граница между этими тремя типами программ постепенно размывается, но все же достаточно заметна. Для создания презентаций и публикаций в Интернет используется первая группа программ. Для разработки других видов мультимедиа продуктов возможны второй и третий варианты.

Авторские инструментальные средства мультимедиа занимают место между программами мультимедиа-презентаций и языками программирования. Деление между программами мультимедиа-презентаций и авторскими инструментальными средствами достаточно условное. В общем, можно сказать, что первые ориентированы в основном на передачу информации в одном направлении (от компьютера к пользователю), а вторые служат для создания программных продуктов с высокой степенью взаимодействия с пользователем.

Использование авторских средств дает экономию средств и времени, но эффективность работы программы будет ниже. Программирование — более дорогостоящий и трудоемкий путь, но он дает больше возможностей реализации идеи автора. Авторские системы предлагают среду программирования на языке сценариев для разработки пользовательского интерфейса. От настоящих языков программирования их отличают ограниченные возможности. Вместе с тем, в последнее время появилось достаточно много систем, в которых программирование, пусть даже на специализированном, но все же языке программирования, не является обязательным, а служит дополнением к возможностям программ создать приложение на экране компьютера. Такими возможностями обладают и современные языки программирования, в них добавляются различные мастера для быстрого создания приложений, в задачу которых входит построение исходного текста программы на языке программирования после ввода пользователем исходной информации о внешнем виде приложения.

Таким образом, задача выбора необходимого средства разработки мультимедиа-приложения не так проста, как кажется на первый взгляд, и универсального решения, годного на все случаи жизни, не имеет.

Для разработки мультимедиа-продукта необходим набор технических средств, соответствующий небольшой мультимедиа-студии, в том числе:

- мультимедийный компьютер;
- цветной сканер, лучше планшетный, и необходимое для сканирования изображений программное обеспечение;
- записывающий накопитель на компакт-дисках;
- видеоплата для оцифровки видео (если ее нет, запись и оцифровку можно заказать).

Порядок выполнения работы

- Составить математическую модель задачи.
- Выбрать и обосновать наиболее рациональный метод решения задачи;
- Разработать алгоритм для решения задачи.
- Написать и отладить программу.

Форма предоставления результата

- Блок – схема.
- Код программы.

Критерии оценки:

«Отлично» - теоретическое содержание курса освоено полностью, без пробелов, умения сформированы, все предусмотренные программой учебные задания выполнены, качество их выполнения оценено высоко.

–«Хорошо» - теоретическое содержание курса освоено полностью, без пробелов, некоторые умения сформированы недостаточно, все предусмотренные программой учебные задания выполнены, некоторые виды заданий выполнены с ошибками.

–«Удовлетворительно» - теоретическое содержание курса освоено частично, но пробелы не носят существенного характера, необходимые умения работы с освоенным материалом в основном сформированы, большинство предусмотренных программой обучения учебных заданий выполнено, некоторые из выполненных заданий содержат ошибки.

«Неудовлетворительно» - теоретическое содержание курса не освоено, необходимые умения не сформированы, выполненные учебные задания содержат грубые ошибки.