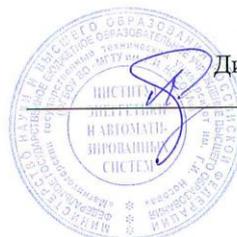




МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ  
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение  
высшего образования  
«Магнитогорский государственный технический университет им. Г.И. Носова»



УТВЕРЖДАЮ  
Директор ИЭиАС  
С.И. Лукьянов

26.02.2020 г.

**РАБОЧАЯ ПРОГРАММА ДИСЦИПЛИНЫ (МОДУЛЯ)**

***ПРОГРАММИРОВАНИЕ И ОСНОВЫ АЛГОРИТМИЗАЦИИ***

Направление подготовки (специальность)  
27.03.04 УПРАВЛЕНИЕ В ТЕХНИЧЕСКИХ СИСТЕМАХ

Направленность (профиль/специализация) программы  
Системы и средства автоматизации технологических процессов

Уровень высшего образования - бакалавриат  
Программа подготовки - академический бакалавриат

Форма обучения  
очная

Институт/ факультет	Институт энергетики и автоматизированных систем
Кафедра	Автоматизированных систем управления
Курс	2
Семестр	3

Магнитогорск  
2020 год

Рабочая программа составлена на основе ФГОС ВО по направлению подготовки 27.03.04 УПРАВЛЕНИЕ В ТЕХНИЧЕСКИХ СИСТЕМАХ (уровень бакалавриата) (приказ Минобрнауки России от 20.10.2015 г. № 1171)

Рабочая программа рассмотрена и одобрена на заседании кафедры Автоматизированных систем управления

12.02.2020, протокол № 6

Зав. кафедрой \_\_\_\_\_ С.М. Андреев

Рабочая программа одобрена методической комиссией ИЭиАС

26.02.2020 г. протокол № 5

Председатель \_\_\_\_\_ С.И. Лукьянов

Рабочая программа составлена:

ст. преподаватель кафедры АСУ, \_\_\_\_\_ И.Г. Самарина

Рецензент:

зам. директора ЗАО "КонсОМ СКС" , канд. техн. наук  
Ю.Н. Волщук



## Лист актуализации рабочей программы

---

Рабочая программа пересмотрена, обсуждена и одобрена для реализации в 2020 - 2021 учебном году на заседании кафедры Автоматизированных систем управления

Протокол от \_\_\_\_\_ 20\_\_ г. № \_\_\_\_  
Зав. кафедрой \_\_\_\_\_ С.М. Андреев

---

Рабочая программа пересмотрена, обсуждена и одобрена для реализации в 2021 - 2022 учебном году на заседании кафедры Автоматизированных систем управления

Протокол от \_\_\_\_\_ 20\_\_ г. № \_\_\_\_  
Зав. кафедрой \_\_\_\_\_ С.М. Андреев

---

Рабочая программа пересмотрена, обсуждена и одобрена для реализации в 2022 - 2023 учебном году на заседании кафедры Автоматизированных систем управления

Протокол от \_\_\_\_\_ 20\_\_ г. № \_\_\_\_  
Зав. кафедрой \_\_\_\_\_ С.М. Андреев

---

Рабочая программа пересмотрена, обсуждена и одобрена для реализации в 2023 - 2024 учебном году на заседании кафедры Автоматизированных систем управления

Протокол от \_\_\_\_\_ 20\_\_ г. № \_\_\_\_  
Зав. кафедрой \_\_\_\_\_ С.М. Андреев

### 1 Цели освоения дисциплины (модуля)

изучение приёмов создания алгоритмов программируемой системы и реализация их с помощью алгоритмического языка

### 2 Место дисциплины (модуля) в структуре образовательной программы

Дисциплина Программирование и основы алгоритмизации входит в вариативную часть учебного плана образовательной программы.

Для изучения дисциплины необходимы знания (умения, владения), сформированные в результате изучения дисциплин/ практик:

Информатика и информационные технологии

Учебная - практика по получению первичных профессиональных умений и навыков, в том числе первичных умений и навыков научно-исследовательской деятельности

Учебная - ознакомительная практика

Введение в направление

Основы объектно-ориентированного программирования

Знания (умения, владения), полученные при изучении данной дисциплины будут необходимы для изучения дисциплин/практик:

Операционные системы реального времени

Производственная - практика по получению профессиональных умений и опыта профессиональной деятельности

Теория автоматического управления

Системы автоматизации и управления

Автоматизированное управление в технических системах

Теория и техника инженерного эксперимента

Методы оптимизации

Моделирование систем

### 3 Компетенции обучающегося, формируемые в результате освоения дисциплины (модуля) и планируемые результаты обучения

В результате освоения дисциплины (модуля) «Программирование и основы алгоритмизации» обучающийся должен обладать следующими компетенциями:

Структурный элемент компетенции	Планируемые результаты обучения
ДПК-3 способностью разрабатывать новое программное обеспечение, необходимое для управления техническими системами и решения практических задач профессиональной деятельности	
Знать	- принципы программного управления компьютером; - методы формального представления алгоритмов, основные (типовые) алгоритмы обработки данных; - принципы структурного и модульного программирования с использованием операторов языка C/C++
Уметь	- разрабатывать алгоритмы решения прикладных задач на основе типовых структур алгоритмов; - разрабатывать прикладные программные продукты с помощью современных средств и языков программирования с применением современных информационных технологий обработки данных (включая СУБД)

Владеть	<ul style="list-style-type: none"> <li>- навыками работы в интегрированных средах разработки программного обеспечения (в т.ч. редактирования, компиляции, отладки программ);</li> <li>- навыками работы с современными инструментариями разработки прикладных программных продуктов на базе современных языков программирования;</li> <li>- методами построения современных проблемно-ориентированных прикладных программных средств</li> </ul>
ПК-2 способностью проводить вычислительные эксперименты с использованием стандартных программных средств с целью получения математических моделей процессов и объектов автоматизации и управления	
Знать	<ul style="list-style-type: none"> <li>- алгоритмы формирования выборки и обработки данных вычислительного эксперимента с целью создания на их основе модели технологического процесса;</li> <li>- особенности использования стандартных программных пакетов при создании моделей различных типов;</li> <li>- основные принципы и методологию разработки программного обеспечения, включая типовые способы организации данных и построения алгоритмов обработки данных с реальных объектов, синтаксис и семантику универсального алгоритмического языка программирования высокого уровня</li> </ul>
Уметь	<ul style="list-style-type: none"> <li>- использовать стандартные пакеты прикладных программ для решения практических задач на основе современных технологий программирования и алгоритмизации;</li> <li>- решать исследовательские и проектные задачи с использованием компьютеров и стандартных программных средств</li> </ul>
Владеть	<ul style="list-style-type: none"> <li>- навыками создания математических моделей процессов и объектов автоматизации и управления с использованием стандартных программных средств;</li> <li>- навыками работы и организации практического функционирования программных средств и систем автоматизации и управления</li> </ul>

#### 4. Структура, объём и содержание дисциплины (модуля)

Общая трудоемкость дисциплины составляет 4 зачетных единиц 144 акад. часов, в том числе:

- контактная работа – 76,1 акад. часов;
- аудиторная – 72 акад. часов;
- внеаудиторная – 4,1 акад. часов
- самостоятельная работа – 32,2 акад. часов;
- подготовка к экзамену – 35,7 акад. часа

Форма аттестации - экзамен

Раздел/ тема дисциплины	Семестр	Аудиторная контактная работа (в акад. часах)			Самостоятельная работа студента	Вид самостоятельной работы	Форма текущего контроля успеваемости и промежуточной аттестации	Код компетенции
		Лек.	лаб. зан.	практ. зан.				
1. Основные понятия программирования								
1.1 Алгоритмы, определение, способы записи. Методы разработки	3	2		2	1,2	Самостоятельное изучение учебной литературы	Выполнение практической работы	ДПК-3, ПК-2
1.2 Основные виды, этапы проектирования и жизненный цикл программных продуктов. Стандарты на разработку		2		2	3	Самостоятельное изучение учебной литературы	Выполнение практической работы, КР	ДПК-3, ПК-2
Итого по разделу		4		4	4,2			
2. Структурно-модульное программирование								
2.1 Программирование на языках С/С++. Алфавит, типы данных. Основные положения структурного программирования: операторы управления	3	2		4	2	Самостоятельное изучение учебной литературы	Выполнение практической работы	ДПК-3, ПК-2
2.2 Массивы: одномерные и многомерные массивы. Структуры, синтаксис. Объявление типа: typedef. Перечисления. Объединения		4		6/2И	3	Самостоятельное изучение учебной литературы	Выполнение практической работы	ДПК-3, ПК-2
2.3 Функции. Объявления и определения функции. Структуры и массивы как параметры функции		4		6/2И	4	Самостоятельное изучение учебной литературы	Выполнение практической работы	ДПК-3, ПК-2
2.4 Ввод/вывод в С/С++. Открытие и закрытие потоков. Файловый ввод/вывод		4		4	4	Самостоятельное изучение учебной литературы	Выполнение практической работы, КР	ДПК-3, ПК-2
Итого по разделу		14		20/4И	13			
3. Объектно-ориентированное программирование (ООП)								

3.1 Основы ООП. Классы. Описание класса и определение объектов. Конструкторы и деструкторы	3	2		4/4И	3	Самостоятельное изучение учебной литературы	Выполнение практической работы	ДПК-3, ПК-2
3.2 Наследование. Виртуальные классы		4		2/2И	3	Самостоятельное изучение учебной литературы	Выполнение практической работы	ДПК-3, ПК-2
3.3 Перегрузка функций. Конструктор копий. Перегрузка оператор. Применение полиморфизма. Виртуальные функции		4		4/4И	3	Самостоятельное изучение учебной литературы	Выполнение практической работы, КР	ДПК-3, ПК-2
Итого по разделу		10		10/10И	9			
4. Прикладное программирование								
4.1 Динамические структуры. Сортировка	3	4		1	3	Самостоятельное изучение учебной литературы	Выполнение практической работы	ДПК-3, ПК-2
4.2 Рекурсия и итерация. Рекурсия как метод вычислений. Графы. Поиск, постановка задачи, виды		4		1	3	Самостоятельное изучение учебной литературы	Выполнение практической работы	ДПК-3, ПК-2
Итого по разделу		8		2	6			
Итого за семестр		36		36/14И	32,2		экзамен	
Итого по дисциплине		36		36/14И	32,2		экзамен	ДПК-3, ПК-2

## **5 Образовательные технологии**

Для реализации предусмотренных видов учебной работы в качестве образовательных технологий в преподавании дисциплины «Программирование и основы алгоритмизации» используются традиционная и модульно-компетентностная технологии.

Передача необходимых теоретических знаний и формирование основных представлений по курсу «Программирование и основы алгоритмизации» происходит с использованием мультимедийного оборудования.

Теоретический курс включает: вводную лекцию, первое представление о предмете и знакомит студентов с назначением и задачами курса; проблемные лекции являются результатом усвоения полученной информации посредством постановки проблемного вопроса и поиска путей его решения; лекции – консультации, изложение нового материала сопровождается постановкой вопросов и дискуссией в поисках ответов на эти вопросы.

В ходе проведения лекционных занятий предусматривается:

- использование электронного демонстрационного материала по темам объектно-ориентированного программирования;
- использование электронных учебников по отдельным темам занятий;
- встречи с представителями проектных и обслуживающих предприятий: встречи с представителями проектных и обслуживающих предприятий: ООО «ОСК», ООО «Информсервис ММК», ЗАО «КонсОМ»; предполагаемые темы встреч «Применение технологии объектно-ориентированного программирования для разработки обучающих программ-тренажеров», «Программное обеспечение современной системы управления»;
- активные и интерактивные формы обучения: вариативный опрос, дискуссии, устный опрос, контрольная работа, тестовый опрос, индивидуальная «защита» практических работ и т.д.

Лекционный материал закрепляется в ходе практических работ, на которых выполняются групповые или индивидуальные задания по пройденной теме. Часть практических занятий проводится в интерактивной форме с использованием следующих методов:

- работа в команде, предусматривает совместную деятельность студентов в группе под руководством лидера, направленную на решение общей задачи с делением ответственности и полномочий;
- проблемное обучение, которое заключается в стимулировании студентов к самостоятельной «добыче» знаний, необходимых для решения конкретной проблемы;
- контекстное обучение, которое позволяет усвоить материал путем выявления связей между конкретным знанием и его применением;
- обучение на основе опыта, активизация познавательной деятельности студентов за счет ассоциации их собственного опыта с предметом изучения.

Самостоятельная работа стимулирует студентов к самостоятельной проработке в процессе выполнения домашних и контрольных работ, а также в процессе подготовки к устному опросу, тестированию и итоговой аттестации

## **6 Учебно-методическое обеспечение самостоятельной работы обучающихся**

Представлено в приложении 1.

## **7 Оценочные средства для проведения промежуточной аттестации**

Представлены в приложении 2.

## **8 Учебно-методическое и информационное обеспечение дисциплины (модуля)**

#### **а) Основная литература:**

1. Самарина, И. Г. Программирование и основы алгоритмизации : учебное пособие. Ч. 1. Курс лекций / И. Г. Самарина. - Магнитогорск : МГТУ, 2013. - 1 электрон. опт. диск (CD-ROM). - Загл. с титул. экрана. - URL: <https://magtu.informsystema.ru/uploader/fileUpload?name=908.pdf&show=dcatalogues/1/1118881/908.pdf&view=true> (дата обращения: 14.05.2020). - Макрообъект. - Текст : электронный. - Сведения доступны также на CD-ROM.
2. Давыдова, Н. А. Программирование / Давыдова Н.А., Боровская Е.В., - 3-е изд., (эл.) - Москва : БИНОМ. ЛЗ, 2015. - 241 с.: ISBN 978-5-9963-2647-1. - Текст : электронный. - URL: <https://znanium.com/catalog/product/544438> (дата обращения: 18.09.2020). – Режим доступа: по подписке.

#### **б) Дополнительная литература:**

1. Полубенцева М. С/С++. Процедурное программирование / М. Полубенцева. - Санкт-Петербург : БХВ-Петербург, 2010. - 448 с. - ISBN 978-5-9775-0145-3. - URL: <https://ibooks.ru/reading.php?productid=18410> (дата обращения: 18.09.2020). - Текст: электронный
2. Павловская Т. С/С++. Структурное и объектно-ориентированное программирование. Практикум / Т. Павловская, Ю. Щупак. - Санкт-Петербург : Питер, 2011. - 352 с. - ISBN 978-5-459-00613-1. - URL: <https://ibooks.ru/reading.php?productid=21762> (дата обращения: 18.09.2020). - Текст: электронный.
3. Ашарина И. В. Объектно-ориентированное программирование в С++: лекции и упражнения. Учебное пособие для вузов / И.В. Ашарина. - Москва: Горячая Линия–Телеком, 2012. - 320 с. - ISBN 978-5-9912-7001-4. - URL: <https://ibooks.ru/reading.php?productid=333353> (дата обращения: 18.09.2020). - Текст: электронный
4. Хабибуллин И. Программирование на языке высокого уровня. С/С++ / И. Хабибуллин. - Санкт-Петербург : БХВ-Петербург, 2010. - 512 с. - ISBN 5-94157-559-9. - URL: <https://ibooks.ru/reading.php?productid=18532> (дата обращения: 18.09.2020). - Текст: электронный
5. Кубенский А. Структуры и алгоритмы обработки данных: объектно ориентированный подход и реализация на С++ / А. Кубенский. - Санкт-Петербург : БХВ-Петербург, 2010. - 464 с. - ISBN 5-94157-506-8. - URL: <https://ibooks.ru/reading.php?productid=18563> (дата обращения: 18.09.2020). - Текст: электронный

#### **в) Методические указания:**

1. Самарина, И. Г. Программирование и основы алгоритмизации : учебное пособие. Ч. 3 / И. Г. Самарина, А. Р. Бондарева ; МГТУ. - Магнитогорск : МГТУ, 2015. - 1 электрон. опт. диск (CD-ROM). - Загл. с титул. экрана. - URL: <https://magtu.informsystema.ru/uploader/fileUpload?name=1485.pdf&show=dcatalogues/1/1124014/1485.pdf&view=true> (дата обращения: 14.05.2020). - Макрообъект. - Текст: электронный. - Сведения доступны также на CD-ROM
2. Самарина, И. Г. Программирование и основы алгоритмизации: учебное пособие / И. Г. Самарина, А. Р. Бондарева; МГТУ. - Магнитогорск: МГТУ, 2018. - 71 с.: табл., схемы, диагр. - Текст : непосредственный

**г) Программное обеспечение и Интернет-ресурсы:**

**Программное обеспечение**

Наименование ПО	№ договора	Срок действия лицензии
MS Windows 7 Professional(для классов)	Д-1227-18 от 08.10.2018	11.10.2021
MS Office 2007 Professional	№ 135 от 17.09.2007	бессрочно
Kaspersky Endpoint Security для бизнеса-Стандартный	Д-300-18 от 21.03.2018	28.01.2020
7Zip	свободно распространяемое ПО	бессрочно
MS Visual Studio 2010 Professional(для класса)	Д-1227-18 от 08.10.2018	11.10.2021

**Профессиональные базы данных и информационные справочные системы**

Название курса	Ссылка
Электронная база периодических изданий East View Information Services, ООО «ИВИС»	<a href="https://dlib.eastview.com/">https://dlib.eastview.com/</a>
Национальная информационно-аналитическая система – Российский индекс научного цитирования	URL: <a href="https://elibrary.ru/project_risc.asp">https://elibrary.ru/project_risc.asp</a>
Поисковая система Академия Google (Google Scholar)	URL: <a href="https://scholar.google.ru/">https://scholar.google.ru/</a>
Информационная система - Единое окно доступа к информационным ресурсам	URL: <a href="http://window.edu.ru/">http://window.edu.ru/</a>
Федеральное государственное бюджетное учреждение «Федеральный институт промышленной собственности»	URL: <a href="http://www1.fips.ru/">http://www1.fips.ru/</a>
Российская Государственная библиотека. Каталоги	<a href="https://www.rsl.ru/ru/4readers/catalogues/">https://www.rsl.ru/ru/4readers/catalogues/</a>
Электронные ресурсы библиотеки МГТУ им. Г.И. Носова	<a href="http://magtu.ru:8085/marcweb2/Default.asp">http://magtu.ru:8085/marcweb2/Default.asp</a>
Федеральный образовательный портал – Экономика. Социология. Менеджмент	<a href="http://ecsocman.hse.ru/">http://ecsocman.hse.ru/</a>
Университетская информационная система РОССИЯ	<a href="https://uisrussia.msu.ru">https://uisrussia.msu.ru</a>
Международная наукометрическая реферативная и полнотекстовая база данных научных изданий «Web of science»	<a href="http://webofscience.com">http://webofscience.com</a>
Международная реферативная и полнотекстовая справочная база данных научных изданий «Scopus»	<a href="http://scopus.com">http://scopus.com</a>
Международная база полнотекстовых журналов Springer Journals	<a href="http://link.springer.com/">http://link.springer.com/</a>

Международная коллекция научных протоколов по различным отраслям знаний Springer Protocols	<a href="http://www.springerprotocols.com/">http://www.springerprotocols.com/</a>
Международная база научных материалов в области физических наук и инжиниринга SpringerMaterials	<a href="http://materials.springer.com/">http://materials.springer.com/</a>
Международная база справочных изданий по всем отраслям знаний SpringerReference	<a href="http://www.springer.com/references">http://www.springer.com/references</a>
Международная реферативная база данных по чистой и прикладной математике zbMATH	<a href="http://zbmath.org/">http://zbmath.org/</a>
Международная реферативная и полнотекстовая справочная база данных научных изданий «Springer Nature»	<a href="https://www.nature.com/siteindex">https://www.nature.com/siteindex</a>
Архив научных журналов «Национальный электронно-информационный консорциум» (НП НЭИКОН)	<a href="https://archive.neicon.ru/xmlui/">https://archive.neicon.ru/xmlui/</a>

**6 Учебно-методическое обеспечение самостоятельной работы студентов**

По дисциплине «Программирование и основы алгоритмизации» предусмотрена аудиторная и внеаудиторная самостоятельная работа обучающихся.

Аудиторная самостоятельная работа студентов на лабораторных занятиях происходит под контролем преподавателя в ходе выполнения практических работ, при решении задач и выполнении упражнений, которые для студентов определяет преподаватель.

Внеаудиторная самостоятельная работа осуществляется в виде проработки материала практических занятий, выполнения домашних заданий и при консультациях с преподавателем.

Наименование раздела дисциплины	Перечень практических работ
Основные понятия программирования	1. Разработать алгоритм по заданию
Структурно-модульное программирование	1. Операции и выражения 2. Условные операторы 3. Операторы циклов 4. Операторы циклов 5. Массивы 6. Структуры 7. Указатели 8. Функции
Объектно-ориентированное программирование	1. Создание объекта типа class 2. Конструкторы и деструкторы 3. Наследование 4. Множественное наследование 5. Виртуальные классы 6. Перегрузка функций 7. Перегрузка операторов
Прикладное программирование	1. Библиотека стандартных шаблонов (STL – Standard template library)

**Пример экзаменационного билета**

1. Алфавит языка C/C++
2. Понятие области видимости класс и прав доступа (public, private, protected)
3. Дан массив: а) вывести на экран сначала его неотрицательные элементы, затем отрицательные; б) верно ли, что сумма элементов, которые больше 20, превышает а (функция)

**Пример варианта контрольной работы №1**

1. Функция, определение
2. В каком случае не требуется прототип функции? Пример
3. Глобальные переменные, пример
4. Найти ошибку:

```
int f(int a, int b);
```

```

void main()

    {.....}
int f(int a, int b)

    {.....
return }

```

5. Решить с помощью функции:

$$z = \begin{cases} x - a, & \text{если } x \geq 0; \\ x / 2, & \text{если } x \leq 0; \end{cases} \text{ , где } a \text{ вводится с клавиатуры}$$

6. Написать программу, в которой функция находит сумму элементов массива, имеющих нечетные значения

### Пример варианта контрольной работы №2

1. Инкапсуляция –
2. Теряет ли оператор при перегрузке что-либо из своих функциональных исходных возможностей?
3. Правильен ли фрагмент (создание виртуальной функции)?

```

class B {

public: virtual int f (int a) = 0;

    .....};
class D : public B {
public: int f (int a, int b) { return a*b; }

    .....};

```

4. Перегрузить оператор +

### Пример варианта домашней работы №1

Программирование циклов и условных операторов. Найти сумму ряда при различных и заданных значениях переменной ряда и заданном числе его членов:

$$S = \frac{(2 \cdot x)^2}{2} + \frac{(2 \cdot x)^4}{24} + \dots + (-1)^n \cdot \frac{(2 \cdot x)^{2n}}{(2n)!}; \quad 0,1 \leq x \leq 1; n_{\max} = 15$$

### Пример варианта домашней работы №2

Программа из двух функций; в первой функции (main) вводятся конкретные массивы, вызывается вторая функция. Вторая функция производит заданные операции над элементами массивов, переданными в функцию, и возвращает результат:

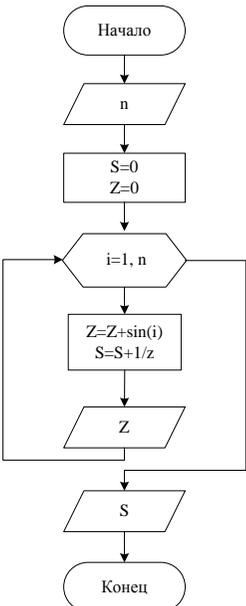
Для заданной матрицы размером 8 на 8 найти такое k, чтобы k-тая строка матрицы совпадала с k-тым столбцом; найти сумму элементов в тех строках, которые содержат хотя бы один отрицательный элемент.

7 Оценочные средства для проведения промежуточной аттестации

а) Планируемые результаты обучения и оценочные средства для проведения промежуточной аттестации:

Структурный элемент компетенции	Планируемые результаты обучения	Оценочные средства
<b>ДПК-3: способностью разрабатывать новое программное обеспечение, необходимое для управления техническими системами и решения практических задач профессиональной деятельности</b>		
Знать	<ul style="list-style-type: none"> <li>– принципы программного управления компьютером;</li> <li>– методы формального представления алгоритмов, основные (типовые) алгоритмы обработки данных;</li> <li>– принципы структурного и модульного программирования с использованием операторов языка C/C++</li> </ul>	<p><i>Перечень теоретических вопросов к экзамену:</i></p> <ol style="list-style-type: none"> <li>1. Понятие алгоритма</li> <li>2. Классификация алгоритмов</li> <li>3. Язык программирования</li> <li>4. Классификация языков</li> <li>5. Способы записи алгоритмов</li> <li>6. Алгоритм линейной структуры, пример</li> <li>7. Алгоритм разветвляющейся структуры, пример</li> <li>8. Алгоритм циклической структуры, пример</li> <li>9. Принципы проектирования алгоритмов</li> <li>10. Алфавит языка C/C++</li> <li>11. Идентификаторы и ключевые (служебные) слов</li> <li>12. Константы языка C/C++, задание определение и использование</li> <li>13. Типы данных</li> <li>14. Спецификаторы класса памяти (auto, static, register, extern) и их влияние на время жизни переменной</li> <li>15. Понятие указателя в C/C++: определение, инициализация, разыменованние</li> <li>16. Указатель на тип void, его использование с объектами разных типов</li> <li>17. Перечислимый тип в C/C++: определение типа, переменных этого типа и их использование</li> <li>18. Понятие массива, определение одномерного массива, обращение к отдельным элементам, инициализация</li> <li>19. Многомерный массив (двух и трёхмерный), расположение элементов в памяти, инициализация при определении</li> <li>20. Имя массива как указатель; доступ к элементам массива по указателю</li> <li>21. Определение типа структуры и переменных типа структуры; инициализация</li> </ol>

Структурный элемент компетенции	Планируемые результаты обучения	Оценочные средства
		<p>структуры при определении</p> <p>22. Понятие объединения (union): определение объединения, инициализация объединения, обращение к элементам объединения</p> <p>23. Введение новых типов с помощью typedef</p> <p>24. Понятие выражения; первичные элементы выражения</p> <p>25. Операции инкремента и декремента (++ , -- ); префиксный и постфиксный инкремент</p> <p>26. Встроенная функция sizeof; её использование для определения размера переменной определённого типа</p> <p>27. Унарные операции(операторы) в C/C++. Порядок их выполнения в C/C++</p> <p>28. Бинарные операции в C/C++: арифметические операции</p> <p>29. Операции (операторы) побитого правого и левого сдвига операнда целого типа</p> <p>30. Операции (операторы) отношения в C/C++; порядок их выполнения. Понятие true и false в C/C++.</p> <p>31. Побитовые логические операции</p> <p>32. Логические операции в C/C++</p> <p>33. Тернарная операция ?: и её использование взамен оператора if</p> <p>34. Операция запятая и её использование в операторах (инструкциях) цикла</p> <p>35. Понятие функции как многократно используемого участка программы (подпрограммы). Выделение в стеке памяти для передачи фактических параметров</p> <p>36. Описание функции (прототип). Список формальных параметров, допустимые типы формальных параметров</p> <p>37. Определение функции. Тело функции использование оператора return</p> <p>38. Вызов функции. Механизм передачи фактических параметров по значению. Использование указателей для передачи параметров по ссылке</p> <p>39. Операторы выбора: условный оператор if</p> <p>40. Оператор выбора: переключатель switch</p> <p>41. Операторы цикла: for, while, do ... while</p> <p>42. Операторы передачи управления: return, continue</p> <p>43. Обращение к элементам массива по указателю</p> <p>44. Передача массива в функцию с помощью указателя. Обращение к элементу двумерного массива по указателю. Операторы new и delete</p>

Структурный элемент компетенции	Планируемые результаты обучения	Оценочные средства
		45. Объявление переменных на внешнем уровне, их область видимости
Уметь	<ul style="list-style-type: none"> <li>– разрабатывать алгоритмы решения прикладных задач на основе типовых структур алгоритмов;</li> <li>– разрабатывать прикладные программные продукты с помощью современных средств и языков программирования с применением современных информационных технологий обработки данных (включая СУБД)</li> </ul>	<p><b>Примеры практических заданий для экзамена:</b></p> <ol style="list-style-type: none"> <li>1. Составить алгоритм вычисления по формуле <math>S = X \cdot Y^2</math></li> <li>2. Составить алгоритм решения для функции <math>Z(X) = X</math> при <math>X &gt; 0</math> и <math>Z(X) = X^2</math> при <math>X \leq 0</math></li> <li>3. Структура спецификация, поля структуры: позиция, наименование технического средства, количество. Программа выводит необходимое техническое средство, по выбранной позиции</li> <li>4. Реализовать блок-схему на языке C++</li> </ol> <div style="text-align: center;">  <pre> graph TD     Start([Начало]) --&gt; Input[/n/]     Input --&gt; Init[S=0 Z=0]     Init --&gt; Loop{i=1, n}     Loop --&gt; Calc[Z=Z+sin(i) S=S+1/z]     Calc --&gt; OutZ[/Z/]     OutZ --&gt; Loop     OutZ --&gt; OutS[/S/]     OutS --&gt; End([Конец]) </pre> </div>
Владеть	– навыками работы в интегрированных средах разработки	<p><b>Перечень практических работ:</b></p> <ol style="list-style-type: none"> <li>1. Разработать алгоритм по заданию</li> </ol>

Структурный элемент компетенции	Планируемые результаты обучения	Оценочные средства
	<p>программного обеспечения (в т.ч. редактирования, компиляции, отладки программ);</p> <ul style="list-style-type: none"> <li>– навыками работы с современными инструментариями разработки прикладных программных продуктов на базе современных языков программирования;</li> <li>– методами построения современных проблемно-ориентированных прикладных программных средств</li> </ul>	<ol style="list-style-type: none"> <li>2. Операции и выражения</li> <li>3. Условные операторы</li> <li>4. Операторы циклов</li> <li>5. Операторы циклов</li> <li>6. Массивы</li> <li>7. Структуры</li> <li>8. Указатели</li> <li>9. Функции</li> <li>10. Создание объекта типа class</li> <li>11. Конструкторы и деструкторы</li> <li>12. Наследование</li> <li>13. Множественное наследование</li> <li>14. Виртуальные классы</li> <li>15. Перегрузка функций</li> <li>16. Перегрузка операторов</li> </ol>
<b>ПК-2 способностью проводить вычислительные эксперименты с использованием стандартных программных средств с целью получения математических моделей процессов и объектов автоматизации и управления</b>		
Знать	<ul style="list-style-type: none"> <li>– алгоритмы формирования выборки и обработки данных вычислительного эксперимента с целью создания на их основе модели технологического процесса;</li> <li>– особенности использования стандартных программных пакетов при создании моделей различных типов;</li> <li>– основные принципы и методологию разработки программного обеспечения, включая типовые способы организации данных и построения алгоритмов обработки данных с реальных объектов, синтаксис и семантику универсального</li> </ul>	<p><b>Перечень теоретических вопросов к экзамену:</b></p> <ol style="list-style-type: none"> <li>1. Основные этапы алгоритмизации. Постановка задачи. Построение математической модели. Разработка алгоритма решения зада. Программирование</li> <li>2. Этапы работ по созданию программных продуктов</li> <li>3. Составление технического задания на программирование</li> <li>4. Технический проект по созданию программных продуктов</li> <li>5. Рабочая документация (рабочий проект). Основные виды</li> <li>6. Жизненный цикл программных продуктов</li> <li>7. Маркетинг и спецификация программного продукта</li> <li>8. Проектирование структуры программного продукта</li> <li>9. Программирование, тестирование и отладка программ</li> <li>10. Документирование программного продукта</li> <li>11. Выход программного продукта на рынок программных средств</li> <li>12. Эксплуатация и сопровождение программного продукта</li> </ol>

Структурный элемент компетенции	Планируемые результаты обучения	Оценочные средства
	алгоритмического языка программирования высокого уровня	13. Снятие программного продукта с продажи и отказ от сопровождения 14. Основные виды, этапы проектирования и жизненный цикл программных продуктов 15. Стандарты на разработку. Стандарты на разработку прикладных программных средств. Документирование, сопровождение и эксплуатация программных средств 16. От C к C++. Понятие объектно-ориентированного программирования 17. Перегрузка функций (статическая) 18. Понятие конструктора. Использование конструкторов для инициализации вновь созданной переменной типа класс 19. Понятие деструктора. Использование деструктора 20. Понятие о перегрузках операторов. Пример перегрузки оператора + 21. Перегрузка функций 22. Понятие области видимости класс и прав доступа (public, private, protected) 23. Механизм наследования. 24. Виртуальные базовые классы 25. Понятие полиморфизма, механизм. Примеры 26. Виртуальные функции
Уметь	– использовать стандартные пакеты прикладных программ для решения практических задач на основе современных технологий программирования и алгоритмизации; – решать исследовательские и проектные задачи с использованием компьютеров и стандартных программных средств	<b>Перечень практических работ:</b> 1. Библиотека стандартных шаблонов (STL – Standard template library): распределители памяти, предикаты, функции сравнения и объекты-функции 2. Библиотека стандартных шаблонов (STL – Standard template library): строковый класс 3. Библиотека стандартных шаблонов (STL – Standard template library): класс vector 4. Библиотека стандартных шаблонов (STL – Standard template library): класс list 5. Динамические структуры. Сортировка 6. Рекурсия и итерация. Рекурсия как метод вычислений 7. Графы. Поиск, постановка задачи, виды
Владеть	– навыками создания математических моделей процессов и объектов автоматизации и управления с	<b>Перечень практических заданий на экзамен:</b> 1. Определить выходной сигнал терморезистора в заданном температурном диапазоне. Вывести в два столбца, начальное сопротивление и температурный

Структурный элемент компетенции	Планируемые результаты обучения	Оценочные средства
	<p>использованием стандартных программных средств;</p> <p>– навыками работы и организации практического функционирования программных средств и систем автоматизации и управления</p>	<p>коэффициент задать как именованные константы</p> <p>2. Структура спецификация, поля структуры: позиция, наименование технического средства, количество. Программа выводит необходимое техническое средство, по выбранной позиции</p> <p>3. Рассчитать и вывести относительную погрешность <math>n</math> измерений тока и определить укладывается ли данная погрешность в класс точности прибора</p> <p>4. Оценить <math>n</math> количество измерений температуры, на наличие грубой погрешности</p> <p>5. Рассчитать выходной сигнал заданного регулятора, расчет выполняет функция</p> <p>6. Определить выходной сигнал нормирующего преобразователя (на основе неинвертирующего операционного усилителя), работающего совместно с термоэлектрическим преобразователем (считать, что термопара инерционное звено 1-го порядка, с заданной постоянной времени)</p>

**б) Порядок проведения промежуточной аттестации, показатели и критерии оценивания:**

Промежуточная аттестация по дисциплине «Электроника в управляющих устройствах» включает теоретические вопросы, позволяющие оценить уровень усвоения обучающимися знаний, и практические задания, выявляющие степень сформированности умений и владений, проводится в форме экзамена.

Экзамен проводится в устной форме по теоретическим вопросам и задачам.

**Показатели и критерии оценивания экзамена:**

– на оценку *«отлично»* (5 баллов) – обучающийся должен полно раскрыть содержание материала в объеме программы дисциплины, чётко и правильно дать определения, привести доказательства на основе математических и логических выкладок, показать навыки исследовательской деятельности. Ответ должен быть самостоятельным, при ответе использованы знания, приобретённые ранее;

– на оценку *«хорошо»* (4 балла) – обучающийся должен раскрыть содержание материала в объеме программы дисциплины, в основном правильно дать основные определения и понятия предмета. При ответе допущены неточности, нарушена последовательность изложения, допущены небольшие неточности при выводах и использовании терминов, практические навыки нетвёрдые;

– на оценку *«удовлетворительно»* (3 балла) – обучающийся должен усвоить основное содержание материала. При ответе определения и понятия даны не чётко, допущены ошибки при промежуточных математических выкладках в выводах, практические навыки слабые;

– на оценку *«неудовлетворительно»* (2 баллов) – обучающийся демонстрирует знания не более 20% теоретического материала, допускает существенные ошибки, не может показать интеллектуальные навыки решения простых задач. При ответе допущены грубые ошибки в определениях, доказательства теорем не проведено, не даны ответы на дополнительные вопросы преподавателя, отсутствуют навыки исследовательской деятельности;

– на оценку *«неудовлетворительно»* (1 балл) – не может показать знания на уровне воспроизведения и объяснения информации, не может показать интеллектуальные навыки решения простых задач, основное содержание учебного материала не раскрыто.

## Приложение 3

### Методические указания для выполнения практических работ

#### Лабораторная работа №1

#### Операции и выражения

**Цель работы:** Получить навыки программирования линейных алгоритмов

#### Основные сведения

Выражение в языке C/C++ – это последовательность операндов, операций и символов-разделителей. Разделителями в C++ являются символы [ ] ( ) { } , ; : ... \* = #, каждый из которых выполняет свою функцию. Выражение может состоять из одного или более операций.

По числу операндов, участвующих в операции, различают:

- унарные операции (один операнд);
- бинарные операции (два операнда);
- тернарные операции (три операнда).

#### *По типу выполняемых операций различают:*

– арифметические операции – сложения (+), вычитания (-), умножения (\*), деления (/), определение остатка (%); инкремента (++) и декремента (--);

– логические операции – логическое И (&&), логическое ИЛИ (||), логическое НЕ (!);

– операции отношения – больше (>), меньше (<), равно (==), не равно (!=) и т.д.);

– операцию условия (?:);

– операцию присваивания (=);

– операцию sizeof;

– операцию присваивания типов.

Пример 1 – Операция инкремента (две формы – префиксная и постфиксная)

```
#include<iostream>
using namespace std;
```

```
void main()
{ float a, b, c;
  cout<<"Введите a, b, c";
  cin>>a>>b>>c;

      a=b+c++/5;          // a=b+++c/5;
  cout<<"a=><a<< b=><b<<\"n\";
}
```

#### *Пример 2 – Логические операции*

```
#include<iostream>
using namespace std;
```

```
void main()
{ float p1, p2;
  cout<<"Введите p1, p2";
```

```

cin>>p1>>p2;
cout<<"p1 > p2 результат"<<( p1>p2)<<"\n";
cout<<"p1 < p2 результат"<<( p1<p2)<<"\n";
cout<<"p1 == p2 результат"<<( p1==p2)<<"\n";
cout<<"p1 != p2 результат"<<( p1!=p2)<<"\n";
cout<<"p1 || p2 результат"<<( p1||p2)<<"\n";
cout<<"p1 && p2 результат"<<( p1&&p2)<<"\n";

}

```

### Порядок выполнения работы

1. Набрать примеры 1, 2 и продемонстрировать работу.
2. Получить у преподавателя номера самостоятельных заданий.
3. Написать программы на языке C/C++, результаты работы показать преподавателю.
4. Записать в тетрадь проверенные задачи.

### Задания для самостоятельного выполнения

1. Составить программу вывода на экран числа, вводимого с клавиатуры. Выводимому числу должно предшествовать сообщение "Вы ввели число"
2. Вывести на одной строке числа 1 13 и 49 с одним пробелом между ними
3. Составить программу вывода на экран в одну строку трех любых чисел с двумя пробелами между ними
4. Вывести на экран числа 5 11 и 10 одно под другим
5. Составить программу вывода на экран «столбиком» 4 любых чисел

## Лабораторная работа №2

### Условные операторы

**Цель работы:** Получить навыки программирования ветвящихся алгоритмов (разветвлений)

### Основные сведения

Для программирования разветвлений в языке C/C++ предназначены условный оператор **if**, операция условия **?:** и оператор переключатель (выбора) **switch**.

Оператор **if** имеет следующую общую форму:

**if** (логическое выражение) оператор 1; [else оператор 2;]

Пример 3 – Проверка правильности ввода переменной, в диапазоне от 1 до 31

```

.....
cin>>den;
if(den<1||den>31) cout<<"Ошибка!";

```

.....

### Пример 4 – Найти максимум из трех чисел

.....

```

if(a>b&& a>c) max = a;

```

```

else if(b>c) max = b;

```

```

else max = c;

cout<<"max="<<max;

.....

```

**Структура switch имеет следующий вид:**

```

switch(выражение выбора)

{
case значение 1: оператор 1; break;

.....
case значение n: оператор n; break;

[default: оператор;]

}

```

*Пример 5 – Проанализировать значение переменной rez.*

```

.....
switch(rez)

{
case 5: cout<<"Оценка – отлично"; break;
case 4: cout<<"Оценка – хорошо"; break;
case 3: cout<<"Оценка – удовлетворительно"; break;
case 2: cout<<"Оценка – неудовлетворительно"; break;

default: cout<<"Неверное значение rez";

}

.....

```

### **Порядок выполнения работы**

1. Доработать и набрать примеры 3, 4, 5 и продемонстрировать работу.
2. Пример 3 выполнить, так же используя операцию условия.
3. Получить у преподавателя номера самостоятельных заданий.
4. Написать программы на языке C/C++, результаты работы показать преподавателю.
5. Записать в тетрадь проверенные задачи.

### **Задания для самостоятельного выполнения**

1. Даны три вещественных числа a, b, c. Проверить: а) выполняется ли неравенство  $a < b < c$ ; б) выполняется ли неравенство  $b > a > c$ .
2. Определить, является ли число a делителем числа b
3. Определить, верно ли, что при делении неотрицательного целого числа a на положительное число b получается остаток, равный одному из двух заданных чисел c или d
4. Даны три вещественных числа a, b, c. Определить, имеется ли среди них хотя бы одна пара равных между собой чисел

5. Определить, является ли треугольник со сторонами a, b, c равносторонним

### Лабораторная работа №3 Операторы циклов

**Цель работы:** Получить навыки программирования циклических алгоритмов

#### Основные сведения

При выполнении программы возникает необходимость неоднократного повторения однотипных вычислений над различными данными. Для этого используются циклы.

Цикл представляет собой участок программы, в котором одни и те же вычисления реализуются неоднократно над различными значениями одних и тех же переменных (объектов).

Для организации циклов в C/C++ используются следующие операторы: **while**, **for** и **do – while**.

**Цикл типа while имеет следующую форму записи:**

while (логическое условие) { операторы }

*Пример 6 – Дается 10 попыток для угадывания заданного числа. Цикл выполняется до тех пор, пока не угадано число или не исчерпано количество попыток*

```
.....  
i = 1; rez = 1;  
while (i++<=10&&rez!=25)  
{ cout<<"Введите число";  
cin>>rez;  
  
}  
if (i ==12) cout<<"Вы не угадали";  
else cout<<"Вы угадали";
```

.....

**Цикл for имеет следующую структуру:**

for (выражение1; выражение2; выражение3) операторы  
Пример 7 – Вычислить  $y^{10}$  (возможный вариант решения)

```
.....  
for (i = 1, rez = 1; i<=10; i++) rez = rez*y;  
cout<<"rez="<<rez;
```

.....

**Форма записи цикла do – while:**

do оператор while (логическое условие);  
Пример 8 – Ввод дней месяца с проверкой правильности ввода

```
.....  
do cin>>day;
```

```
while (day<1||day>31);
cout<<day
```

.....

### Порядок выполнения работы

1. Набрать и доработать примеры 6, 7, 8, продемонстрировать работу.
2. Получить у преподавателя номера самостоятельных заданий.
3. Написать программы на языке C++, результаты работы показать преподавателю.
4. Записать в тетрадь проверенные задачи.

### Задания для самостоятельного выполнения

1. Напечатать таблицу соответствия между весом в фунтах и весом в килограммах для значений 1, 2, ... 10 фунтов (1 фунт = 453 г)
2. Напечатать таблицу перевода 1, 2, ... 20 долларов США в рубли по текущему курсу (значение курса вводится с клавиатуры)
3. Считая, что Земля – идеальная сфера с радиусом  $R = 6350$  км. Определить расстояние до линии горизонта от точки с высотой над Землей, равной 1, 2, ... 10 км
4. Найти: а) сумму всех целых чисел от 100 до 500; б) сумму всех целых чисел от  $a$  до 500 (значение  $a$  вводится с клавиатуры;  $a < 500$ ); в) сумму всех целых чисел от -10 до  $b$  (значение  $b$  вводится с клавиатуры;  $b > -10$ ); г) сумму всех целых чисел от  $a$  до  $b$  (значения  $a$  и  $b$  вводятся с клавиатуры;  $b > a$ )
5. Гражданин 1 марта открыл счет в банке, вложив 1000 руб. Через каждый месяц размер вклада увеличивается на 2% от имеющейся суммы. Определить: а) прирост суммы вклада за первый, второй, ... десятый месяц; б) сумму вклада через три, четыре, двенадцать месяцев

## Лабораторная работа №4

### Операторы циклов

**Цель работы:** Получить навыки программирования циклических алгоритмов

### Основные сведения

Пример 9 – Вычислить сумму ряда  $x + \frac{x^3}{2} + \dots + \frac{x^{2n+1}}{n^2 + 1}$  при  $0,1 \leq x \leq 1,0$ ;  $n_{max} = 10$

используя простейшие математические функции. Сумма ряда определяется  $S = S + C_n$ , где  $C_n$  –  $n$ -ый член ряда, который можно определить:

$$C_n = \frac{a_n}{a_{n-1}} = \frac{x^{2n+1}}{n^2 + 1} \cdot \frac{(n-1)^2 + 1}{x^{2(n-1)+1}} = \frac{x^{2n+1}}{n^2 + 1} \cdot \frac{(n-1)^2 + 1}{x^{2n-1}} = x^2 \frac{(n-1)^2 + 1}{n^2 + 1}$$

Для расчета используется два цикла: while по  $x$  и for по  $i$ .

```
#include<iostream>
#include<cmath>
#include<iomanip>
using namespace std;
#define MAX 1.0
#define MIN 0.1
```

```

#define NUMBER 10
using namespace std;
int main()
{ double i,x,s,step,c;
x=MIN;
step=(MAX-MIN)/NUMBER;
while(x<=MAX)

    {s=0.0;
    for(i=0;i<=10;i++)

        {c=pow(x,2.0)*(pow((i-1),2)+1)/(pow(i,2)+1);
        s=s+c;

        }
cout<< fixed <<setprecision(2)<<"x = "<<x<<setw(8)<<"s = "<<s<<"\n";
x+=step;

    }
}

```

#### **Порядок выполнения работы**

1. Доработать и набрать пример 9 и продемонстрировать работу.
2. Выполнить проверку правильности нахождения программой суммы ряда.
3. Получить у преподавателя вариант самостоятельного задания.
4. Написать программу на языке C++, результаты работы показать преподавателю, выполнить проверку.
5. Записать в тетрадь проверенные задачи.

### **Лабораторная работа №5**

#### **Массивы**

**Цель работы:** Приобретение навыков работы с массивами

#### **Основные сведения**

Массивы позволяют удобным образом организовать размещение и обработку больших объемов информации. Это набор однотипных объектов, имеющих общее имя и различающихся местоположением в этом наборе.

*Пример 10 – Описание одномерного массива и присваивание начальных значений его элементам*

```

.....
int mas[2];    //объявление массива
int a=10, b=5; //объявление переменных
.....
mas[0]= a;
mas[1]= b;
.....

```

В качестве массива может выступать символьная строка, последний элемент этой строки ‘\0’.

**Пример 11 – Определить количество символов в строке**

```
.....
{ int i;
char p[] = "Кафедра";
for(i=0;i++)
if (p[i]=='\0') break;
cout<<"Длина строки"<<i<<"символов";

}
```

Массивы могут быть многомерными. Многомерным называется массив, элементами которого являются одномерные массивы. Например, инициализируем двухмерный массив A2 размерностью 3x5: `int A2[3][5] = {{1,2,4,5,0},{2,4,5,6,-1},{4,-9,0,0,-2}}`;

**Пример 12 – Вычисление количества положительных, отрицательных и нулевых элементов двухмерного массива**

```
.....
{ float a[10];
int i = 0, n = 0, p = 0, zero = 0;

cout<<"\n Определить количество положительных и отрицательных элементов массива
a[10]\n";

for(i=0; i<10; i++)

{
cout<<"\n Введите a["<<i+1<<"]:";
cin>>a[i];

}

for(i=0; i<10; i++)

{
if(a[i]>0 p+=1; // Определение количества положительных
if(a[i]<0 n+=1; // Определение количества отрицательных
if(a[i]==0 zero+=1; // Определение количества нулевых

}

cout<<"\n Число положительных элементов ="<<p;
cout<<"\n Число отрицательных элементов ="<<n;
cout<<"\n Число нулевых элементов ="<<zero;

}
```

**Порядок выполнения работы**

1. Набрать и доработать примеры 10, 11, 12, продемонстрировать работу.
2. Получить у преподавателя номера самостоятельных заданий.

3. Написать программы на языке C++, результаты работы показать преподавателю.
4. Записать в тетрадь проверенные задачи.

### **Задания для самостоятельного выполнения**

1. Дан массив: а) вывести на экран сначала его неотрицательные элементы, затем отрицательные; б) верно ли, что сумма элементов, которые больше 20, превышает а
2. Дан массив целых чисел. Определить: а) количество элементов, отличных от последнего элемента; б) количество элементов, кратных а
3. Дан массив. Напечатать: а) второй, четвертый и т. д. элементы; б) третий, шестой и т. д. элементы
4. Дан массив целых чисел. а) вывести на экран сначала его четные элементы, затем нечетные; б) определить количество неотрицательных элементов
5. Дан массив. Определить частное от деления суммы положительных элементов массива на модуль суммы отрицательных элементов

## **Лабораторная работа №6**

### **Указатели**

**Цель работы:** Приобретение навыков работы с указателями

### **Основные сведения**

Указатель – это тип переменной, содержащей в памяти адрес того элемента, на который он указывает. При этом имя элемента отсылается к его значению прямо, а указатель косвенно – косвенная адресация.

Указатель может указывать на любые объекты: переменные, массивы, классы, структуры и функции.

Описание переменных типа указатель выполняется с помощью операторов следующей формы:

`<тип>*<имя указателя на переменную заданного типа>;`

Например, `int *r` – указатель на целое число.

Для инициализации указателей используется операция присваивания.

### **Пример 13 – Инициализация указателей**

```
.....  
int a=10, b;  
int *ptr=&a; //инициализация указателя адресом переменной a
```

```
.....  
cout<<"Указатель ="<<ptr<<"Значение a ="<<*ptr;  
ptr = &b; //теперь указатель указывает на переменную b
```

```
.....
```

С указателем связаны два специальных оператора: **&** и **\***. Эти операции унарные, то есть имеют один операнд, перед которым они ставятся. Операция **&** соответствует действию “взятия адреса”, операция **\*** – “значение, расположенное по адресу”.

#### **Пример 14 – Операции с указателями**

```
.....
{ int i = 100;
  int *ptrA, *ptrB;
  ptrA = &i;
  cout<<"Адрес i = "<<&i;
  cout<<"Значение ptrA = "<<ptrA;
  cout<<"Значение i = "<<i;
  cout<<"Значение по адресу ptrA = "<<*ptrA;

}
```

Указатели можно увеличивать (++), уменьшать (--), складывать с указателем целые числа (+ и +=), вычитать из него целые числа (- -=) или вычитать один указатель из другого.

#### **Пример 15 – Операции с указателями**

```
.....
{ int x;
  int *p, *p1;
  p = &x;
  p1 = p + 3;
  cout<<"Начальное значение p = "<<p;
  cout<<"Значение ++p"<<++p;
  cout<<"Значение—p = "<<--p;
  cout<<"Значение p1 = "<<p1;

}
```

В языке C++ указатели тесно связаны с массивами и могут использоваться почти эквивалентно, так как имя массива является константным указателем на первый элемент массива.

#### **Пример 16 – Найти сумму элементов массива, обращаясь к элементам массива по индексу и по указателю**

```
.....
{ int M[10] = {3, 2, 5, 4, 6, 0, 0, 1, 9, -9};
  char t[] = "Сумма элементов массива =\n";
  char *p_t = "Сумма элементов массива =\n";
  int i, S = 0;
  for(i=0; i<10; i++) S+=M[i];
  cout<<t<<S;

S = 0;
for(i=0; i<10; i++) S+=*(M + i);
```

```
cout<<p_t<<S;
}
```

Массивы, размер которых становится известен в процессе выполнения программы, называются динамическими. Для работы с этими массивами используются указатели и специальные операторы **new** – выделение памяти под динамический объект и **delete** – удаление из памяти.

*Пример 17 – Создание двумерного динамического массива целых чисел размерностью  $n \times m$ , который заполняется элементами  $a[i][j] = 10 \cdot (i+1)$*

```
.....
{int n, m;
cout<<"Введите количество строк n и столбцов m";
cin>>n>>m;
int **a = new int*[n];
for(int i=0; i<n; i++)
a[i] = new int[m];
for(int i=0; i<n; i++)
{
cout<<"\n";
for(int j=0; j<m; j++)
{
a[i][j] = 10*(i+1);
cout<<"  "<<*(a+i+j);
}
}
for(int i=0; i<n; i++)
delete[a[i];
delete[a;
}
```

*Пример 18 – Работа с указателями и символьными массивами (вывод осуществляется с помощью функции **printf()**)*

```
#include<iostream>
using namespace std;
#include<locale>
void main()
{locale::global(locale("rus"));
int i;
char *a[3]={"МИР!", "ТРУД!", NULL};
char *ptr="МАЙ!";
a[2]=ptr;
for(i=0; i<=2; i++)
```

```
printf(" %p %s\n",a[i],a[i]);
}
```

### **Пример 19 – Отсортировать элементы массива по возрастанию**

```
....
void main()
{int n,j,i, *mas, buf;
cout<<>>Введите размер массива\n»;
cin>>n;
mas=new int[n];
cout<<>>Введите элементы массива\n»;
for(i=0;i<n;i++)
    {cout<<>>mas["<<i<<"]="»;
    cin>>mas[i];
    }
for(j=0; j<n; j++)
    {for(i=1; i<n; i++)
if(mas[i-1] > mas[i])
    {buf = mas [i-1]; mas[i-1] = mas[i]; mas[i] = buf;
    }
    }
cout<<>>Результирующий массив:\n»;
for(i=0; i<n; i++) cout<<mas[i]<<>> "<<>>\n»;
delete mas;
}
```

### **Порядок выполнения работы**

1. Набрать и доработать примеры 13 – 19 продемонстрировать работу.
2. Получить у преподавателя номера самостоятельных заданий.
3. Написать программы на языке C++, результаты работы показать преподавателю.
4. Записать в тетрадь проверенные задачи.

### **Задания для самостоятельного выполнения**

1. Составить программу, которая подсчитывает число элементов в строке, строка вводится с клавиатуры
2. Составить программу, которая подсчитывает сумму элементов массива размера  $axb$ , элементы вводятся с клавиатуры
3. Упорядочить по возрастанию (убыванию) элементы одномерного динамического массива

4. Составить программу, которая подсчитывает произведение  $P = \prod_{j=1}^n (1 + x_j)$ , где  $x_1, \dots, x_n$  – динамический массив из n элементов. Значение n определяется при вводе.
5. Создать матрицу, элементы главной диагонали которой равны 0, а остальные элементы 1

### **Библиографический список**

1. Юркин А.Г. Задачник по программированию. – СПб.: «Питер», 2002. -182с.
2. Павловская Т.А. С/C++. Программирование на языке высокого уровня. Учебник. – М.: «Питер», 2002. -460с.
3. Глушаков С.В., Коваль А.В, Смирнов С.В. Язык программирования С++. Учебный курс. – Харьков: «Фолио», 2001. – 500с.
4. Павловская Т.А., Щупак Ю.А. С\C++. Объектно-ориентированное программирование . – СПб.: «Питер», 2005. -264с.