МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Магнитогорский государственный технический университет им. Г.И. Носова»

УТВЕРЖДАЮ:
Директор института
С.Е. Лукьянов
21 июня 2017 г.

# РАБОЧАЯ ПРОГРАММА ДИСЦИПЛИНЫ

*ФУНКЦИОНАЛЬНОЕ ПРОГРАММИРОВАНИЕ*

Направление подготовки
09.03.01 Информатика и вычислительная техника

Профиль программы
Программное обеспечение средств вычислительной техники и автоматизированных систем

Уровень высшего образования – бакалавриат

Программа подготовки – академический бакалавриат

Форма обучения
Очная

| | |
|---|---|
| Институт | *энергетики и автоматизированных систем* |
| Кафедра | *вычислительной техники и программирования* |
| Курс | *2* |
| Семестр | *4* |

Магнитогорск
2017 г.

Рабочая программа составлена на основе ФГОС ВО по направлению подготовки (специальности) 09.03.01 Информатика и вычислительная техника, утвержденного приказом МО и Н РФ от 12.01.2016 № 5.

Рабочая программа рассмотрена и одобрена на заседании кафедры вычислительной техники и программирования «26» сентября 2017 г., протокол № 2.

Зав. кафедрой _____ / О.С. Логунова/

Рабочая программа одобрена методической комиссией института энергетики и автоматизированных систем «27» сентября 2017 г., протокол № 2.

Председатель _____ / С.И. Лукьянов/

Рабочая программа составлена: доцентом каф. ВтиП

_____ / М.В. Зарецким/
*(подпись)* *(И.О. Фамилия)*

Рецензент: начальник отдела инновационных разработок ЗАО «КонеОмСКС», канд. техн. наук

_____ / А.Н. Панов/

**Лист актуализации рабочей программы**

Рабочая программа пересмотрена, обсуждена и одобрена для реализации в 2017-2018 учебном году на заседании кафедры Вычислительной техники и программирования

Протокол от 26 09 2017 г. № 2
Зав. кафедрой _____ О.С. Логунова

Рабочая программа пересмотрена, обсуждена и одобрена для реализации в 2018 - 2019 учебном году на заседании кафедры Вычислительной техники и программирования

Протокол от 5 09 2018 г. № 1
Зав. кафедрой _____ О.С. Логунова

Рабочая программа пересмотрена, обсуждена и одобрена для реализации в 2019 - 2020 учебном году на заседании кафедры Вычислительной техники и программирования

Протокол от 19 02 2020 г. № 5
Зав. кафедрой _____ О.С. Логунова

Рабочая программа пересмотрена, обсуждена и одобрена для реализации в 2020 - 2021 учебном году на заседании кафедры Вычислительной техники и программирования

Протокол от 19 08 2020 г. № 5
Зав. кафедрой _____ О.С. Логунова

**1 Цели освоения дисциплины (модуля)**

Целями освоения дисциплины (модуля) «Функциональное программирование» являются:

- формирование у студентов понимания роли функциональной парадигмы программирования в теории и практике разработки, сопровождения и эксплуатации программного обеспечения;
- выработка умения применять технологии функционального программирования для решения практических задач.
- освоение взаимосвязей функциональной и объектно-ориентированной парадигм программирования;
- освоение современных методов проектирования программных продуктов на основе функциональной парадигмы.

Для достижения поставленных целей в курсе «Функциональное программирование» решаются задачи:

- изучение языка функционального программирования (LISP с учетом диалектов);
- изучение функциональных расширений современных языков программирования (Python, JavaScript, встроенный язык Matlab);
- изучение современных применений функциональной парадигмы программирования.

**2 Место дисциплины (модуля) в структуре образовательной программы подготовки бакалавра (магистра, специалиста)**

Дисциплина «Функциональное программирование» входит в вариативную часть блока 1 образовательной программы.

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

- математики (базовая часть блока 1 образовательной программы). Знания, умения и владения, полученные при изучении данной дисциплины, позволят обучающимся осмысленно применять понятие функции, грамотно строить суперпозиции функций;
- информатики (базовая часть блока 1 образовательной программы). Знания, умения и владения, полученные при изучении данной дисциплины, являются основой для освоения средств обработки информации в соответствии с функциональной парадигмой;
- прикладного программирования (базовая часть блока 1 образовательной программы). Знания, умения и владения, полученные при изучении данной дисциплины, являются основой для освоения методологии разработки программ в функциональной парадигме.

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

- систем автоматизированного проектирования (базовая часть блока 1 образовательной программы). Знания, умения и владения, полученные при изучении дисциплины «Функциональное программирование» позволят обучающимся осмысленно применять общепринятые в современных системах автоматизированного проектирования методы представления информации о проектируемом объекте в виде сложных вложенных списков;
- методы управления знаниями (базовая часть блока 1 образовательной программы). Знания, умения и владения, полученные при изучении дисциплины «Функциональное программирование» позволят обучающимся осмысленно применять современные программные средства инженерии знаний, основанные на функциональной парадигме;
- методы анализа информации (базовая часть блока 1 образовательной программы). Знания, умения и владения, полученные при изучении дисциплины «Функциональное программирование» позволят обучающимся осмысленно применять современные программные средства компьютерного анализа неструктурированной информации, основанные на объектной парадигме.

**3 Компетенции обучающегося, формируемые в результате освоения**

**дисциплины (модуля) и планируемые результаты обучения**

В результате освоения дисциплины (модуля) «Функциональное программирование обучающийся должен обладать следующими компетенциями:

| Структурный элемент компетенции | Планируемые результаты обучения |
|---|---|
| **ПК-2. Обладает способностью разрабатывать компоненты аппаратно-программных комплексов и баз данных, используя современные инструментальные средства и технологии программирования.** | |
| Знать | – основные элементы функциональной парадигмы: функция, суперпозиция функций, λ – исчисление, редукция, аппликативный порядок редукции, нормальный порядок редукции;<br>– связь понятий аппликативного и нормального порядков редукции и понятий энергичных и ленивых вычислений, разработанного в соответствии с указанными понятиями;<br>– связь между функциональной и объектно-ориентированной парадигмами программирования, методологию применения функциональной парадигмы программирования в разработке мультипарадименных программных систем. |
| Уметь | – определять целесообразность применения функциональной парадигмы, строить суперпозиции функций;<br>– разрабатывать функциональными средствами рационально организованный программный продукт;<br>– разрабатывать сложные программные системы, основанные на рационально основанной редукции суперпозиции функций |
| Владеть | – навыками применения современных инструментальных средств разработки функциональных программ;<br>– навыками применения не менее двух существенно отличающихся функциональных языков программирования;<br>– применения современных функциональных средств в процессе проектирования, программирования, отладки и модернизации сложных программных систем. |

**4 Структура и содержание дисциплины (модуля)**

Общая трудоемкость дисциплины составляет 4 зачетные единицы 144 акад. часа, в том числе:

- контактная работа – 54,15 акад. часов:
  - аудиторная – 51 акад. час;
  - внеаудиторная – 3,15 акад. часа
- самостоятельная работа – 54,15 акад. часа;
- подготовка к экзамену – 35,7 акад. часа

| Раздел/ тема дисциплины | Семестр | Аудиторная контактная работа (в акад. часах) | | | Самостоятельная работа (в акад. часах) | Вид самостоятельной работы | Форма текущего контроля успеваемости и промежуточной аттестации | Код и структурный элемент компетенции |
|---|---|---|---|---|---|---|---|---|
| | | лекции | лаборат. занятия | практич. занятия | | | | |
| 1. Раздел 1. Функциональная парадигма программирования | | | | | | | | |
| 1.1. Тема. Программа как суперпозиция функций. Рекурсивные функции и λ-исчисление А. Черча. Программирование в функциональных обозначениях. Редукция и ее виды. Редексы. Нормальный порядок редукции. Аппликативный порядок редукции. Теорема Черча - Россера. | 4 | 2 | 2 | | 6 | Самостоятельное изучение учебной и научной литературы. | Беседа – обсуждение. Устный опрос. | *ПК-2 – зув* |
| 1.2. Тема. Функциональные языки. Строго функциональный язык. Функциональные компоненты нефункциональных языков программирования | 4 | 2 | 2 | | 6 | Самостоятельное изучение учебной и научной литературы. Подготовка к лабораторному занятию. Выполнение лабораторной работы. | Беседа – обсуждение. Анализ программного кода. Устный опрос. | *ПК-2 – зув* |
| **Итого по разделу** | **4** | **4** | **4** | | **12** | | **Проверка индивидуальных заданий** | |
| 2. Раздел 2. Работа со списками | 4 | | | | | | | |

| Раздел/ тема дисциплины | Семестр | Аудиторная контактная работа (в акад. часах) | | | Самостоятельная работа (в акад. часах) | Вид самостоятельной работы | Форма текущего контроля успеваемости и промежуточной аттестации | Код и структурный элемент компетенции |
|---|---|---|---|---|---|---|---|---|
| | | лекции | лаборат. занятия | практич. занятия | | | | |
| 2.1. Тема. Основные понятия языка LISP. Атомы, точечные пары, списки. S-выражения. Функции в LISP.Данные в языке LISP. Типы данных. Арифметические функции, логические функции, функции работы со строковыми данными | 4 | 2 | 4/2И | | 6 | Самостоятельное изучение учебной и научной литературы. Подготовка к лабораторному занятию. Выполнение лабораторной работы. | Беседа – обсуждение. Анализ программного кода. Устный опрос. | *ПК-2 – зув* |
| 2.2. Тема Функции работы со списками и их суперпозиции в языке LISP. Функции работы со списками в языке Python. Глобальные и локальные переменные в языках LISP и Python. | 4 | 2 | 4/2И | | 6 | Самостоятельное изучение учебной и научной литературы. Подготовка к лабораторному занятию. Выполнение лабораторной работы. | Беседа – обсуждение. Анализ программного кода. Устный опрос. | *ПК-2 – зув* |
| **Итого по разделу** | **4** | **4** | **8/4И** | | **12** | | **Проверка индивидуальных заданий** | |
| 3. Раздел Рекурсия. Ассоциативные списки. | | | | | | | | |
| 3.1. Тема Виды рекурсии. Реализация рекурсии в языках LISP и Python. | 4 | 2 | 6 | | 6 | Самостоятельное изучение учебной и научной литературы. Подготовка к лабораторному занятию. Выполнение лабораторной работы. | Беседа – обсуждение. Анализ программного кода. Устный опрос. | *ПК-2 – зув* |
| 3.2. Тема. Ассоциативные списки в языке LISP. Словари в языке Python. Функции для ассоциативного поиска. Hash – таблицы в языках LISP и Python. Анонимные функции в язы- | 4 | 2 | 6/2И | | 6 | Самостоятельное изучение учебной и научной литературы. Подготовка к лабораторному занятию. Выполнение лабораторной ра- | Беседа – обсуждение. Анализ программного кода. Устный опрос. | *ПК-2 – зув* |

| Раздел/ тема дисциплины | Семестр | Аудиторная контактная работа (в акад. часах) | | | Самостоятельная работа (в акад. часах) | Вид самостоятельной работы | Форма текущего контроля успеваемости и промежуточной аттестации | Код и структурный элемент компетенции |
|---|---|---|---|---|---|---|---|---|
| | | лекции | лаборат. занятия | практич. занятия | | | | |
| ках LISP, Python встроенном языке Matlab. | | | | | | боты. | | |
| **Итого по разделу** | **4** | **4** | **12/2И** | | **12** | | **Проверка индивидуальных заданий** | |
| 4. Раздел. Функционалы. Классы и объекты. Практические реализации. | | | | | | | | |
| 4.1. Тема. Функционалы в языке LISP. Виды отображающих функционалов. Совместное применение функционалов и анонимных функций. | 4 | 2 | 4/2И | | 6 | Самостоятельное изучение учебной и научной литературы. Подготовка к лабораторному занятию. Выполнение лабораторной работы. | Беседа – обсуждение. Анализ программного кода. Устный опрос. | |
| 4.2. Тема. Записи, классы и объекты в языке LISP. Синтез функциональной и объектно-ориентированной парадигм программирования. Диалект Visual LISP для системы компьютерной графики AutoCAD. | 4 | 3 | 6/6И | | 12,15 | Самостоятельное изучение учебной и научной литературы. Подготовка к лабораторному занятию. Выполнение лабораторной работы. | Беседа – обсуждение. Анализ программного кода. Устный опрос. | |
| **Итого по разделу** | | **5** | **10/8И** | | **18,15** | | **Проверка индивидуальных заданий** | |
| **Итого за семестр** | | **17** | **34/14И** | | **54,15** | | **Экзамен** | |
| **Итого по дисциплине** | | **17** | **34/14И** | | **54,15** | | | |

## 5 Образовательные и информационные технологии

1. **Традиционные образовательные технологии** ориентируются на организацию образовательного процесса, предполагающую прямую трансляцию знаний от преподавателя к студенту (преимущественно на основе объяснительно-иллюстративных методов обучения). Учебная деятельность студента носит в таких условиях, как правило, репродуктивный характер.

**Формы учебных занятий с использованием традиционных технологий:**

Информационная лекция – последовательное изложение материала в дисциплинарной логике, осуществляемое преимущественно вербальными средствами (монолог преподавателя).

Семинар – беседа преподавателя и студентов, обсуждение заранее подготовленных сообщений по каждому вопросу плана занятия с единым для всех перечнем рекомендуемой обязательной и дополнительной литературы.

Практическое занятие, посвященное освоению конкретных умений и навыков по предложенному алгоритму.

Лабораторная работа – организация учебной работы с реальными материальными и информационными объектами, экспериментальная работа с аналоговыми моделями реальных объектов.

2. **Технологии проблемного обучения** – организация образовательного процесса, которая предполагает постановку проблемных вопросов, создание учебных проблемных ситуаций для стимулирования активной познавательной деятельности студентов.

3. **Интерактивные технологии** – организация образовательного процесса, которая предполагает активное и нелинейное взаимодействие всех участников, достижение на этой основе личностно значимого для них образовательного результата. Наряду со специализированными технологиями такого рода принцип интерактивности прослеживается в большинстве современных образовательных технологий. Интерактивность подразумевает субъект - субъектные отношения в ходе образовательного процесса и, как следствие, формирование саморазвивающейся информационно-ресурсной среды.

**Формы учебных занятий с использованием специализированных интерактивных технологий:**

Лекция «обратной связи» – лекция–провокация (изложение материала с заранее запланированными ошибками), лекция-беседа, лекция-дискуссия, лекция–пресс-конференция.

4. **Информационно-коммуникационные образовательные технологии** – организация образовательного процесса, основанная на применении специализированных программных сред и технических средств работы с информацией.

## 6 Учебно-методическое обеспечение самостоятельной работы обучающихся

**Задание к лабораторной работе по теме:**

**Программа как суперпозиция функций. Рекурсивные функции и λ-исчисление А. Черча. Программирование в функциональных обозначениях. Редукция и ее виды. Редексы. Нормальный порядок редукции. Аппликативный порядок редукции. Теорема Черча – Россера.**

Заданы функции:

1. $f(x) = e^{(x+3)(x-9)^2}$;
2. $f(x) = (\sin^2 3x \cdot e^{2x} + \cos\frac{x}{3}) \cdot \ln^3 5x$;
3. $f(x) = \frac{\left(x^2 - e^{3x}\right) \cdot \sin 5x}{2 \cdot x^2 + 5}$;
4. $f(x) = (\sin^3 4x + \ln\frac{x^2+2}{3x+7})(\cos 3x - 5)^2$;

5. $f(x)=\sin^3\frac{(x+3)(x-9)^2}{x^4+2}$;
6. $f(x)=\ln^4\frac{x^6+2}{2x^4+3}$;
7. $f(x)=\cos^3\frac{2x^5+7}{x^2+11}$;
8. $f(x)=tg^2(5x^2-3x+7)$;
9. $f(x)=e^{4^{\frac{x^2+3}{x^4+5}}}$;
10. $f(x)=\frac{(x^4+2)(2x-9)^3}{2x^4+3}$.

Аргумент каждой из представленных функций, в свою очередь, также является функцией:

1. $x=x(t)=t^2\cdot\cos^2(3t)$;
2. $x=x(t)=\sqrt{t}\cdot\ln^2(3t)$;
3. $x=x(t)=\sin(2t)\cdot\cos^2(3t)$;
4. $x=x(t)=\sqrt{t}\cdot\ln^2(3t)$;
5. $x=x(t)=e^{2t}\cdot\cos^2(2t)$;
6. $x=x(t)=e^{\sin(2t)}\cdot t^2$;
7. $x=x(t)=tg^2(3t)\cdot\frac{t^2+1}{t^4+3}$;
8. $x=x(t)=\frac{\cos^2(2t)+1}{t^2+4}$;
9. $x=x(t)=t^2\cdot\ln^3(1+t^4)$;
10. $x=x(t)=e^{(1+t^2)}\cdot\cos(2t)$.

В указанные преподавателем 3 функции из первого списка подставьте указанные преподавателем 3 функции из второго списка. Для каждой из получившихся функций выполните редукцию в каждом из возможных порядков. При заданном значении аргумента t доведите процесс редукции до получения числового результата.

Определите количество шагов, необходимых для получения результата при использовании каждого из порядков редукции.

**Задание к лабораторной работе по теме:**

**Функциональные языки. Строго функциональный язык. Функциональные компоненты нефункциональных языков программирования**

Опишите представленные фрагменты на языке Python в виде суперпозиции функций (если в тексте программы имеются циклы, их необходимо заменить рекурсией; импортируемые из внешних модулей функции следует считать имеющимися в наличии):

1.
```
def Start01(a,b):
    if a>b:
        c=a
    else:
        c=b
    print(a,b,c)
```
2.
```
def Start02():
    import keyword
```

```
    import builtins
    kword_list = keyword.kwlist
    for kw in kword_list:
        print(kw)
    q = dir(builtins)
    for bq in q:
        print(bq)
```

3.

```
def Start03():
    s = 'string'
    for i in s:
        print(i+'_',end = ' ')
    print('')
    d = {'w' : 0, 'x' : 1, 'y' : 2, 'z' : 3}
    for q in d:
        print(q, d[q])
```

4.

```
def Start04():
    table = {'Python': 'Guido van Rossum',
    'Perl':     'Larry Wall',
    'Tcl':      'John Ousterhout',
    'Pascal' : 'Niklaus Wirth' }
    for lang in table:
        print(lang, table[lang])
```

5.

```
def Start05(x,y):
    l = x is y
    print(l)
    print(x, y)
    y = 7
    print(x, y)
    l = x is y
    print(l)
```

6.

```
def Start06(x,y):
    print(x, y)
    y[0] = 17
    y[1] = 9
    print(x, y)
    l = x is y
    print(l)
```

7.

```
def Start07():
    x = [1, 7]
    y = x
    print(x, y)
    y[0] = 17
    y[1] = 9
```

```
    print(x, y)
    l = x is y
    print(l)

8.
def Start08(a,b):
    c = a & b
    print(c)
    d = a | b
    print(d)
    e = a ^ b
    print(e)

9.
def Start09(a,b):
    c = a >> b
    print(c)
    d = a << b
    print(d)

10.
def Start10():
    i, s = 0, 0
    while s < 9:
        i += 1
        s += 1/i
    print(i)

11.
def Start11():
    i, s = 0, 0
    while True:
        i += 1
        s += 1/i
        if s > 7:
            break
    print(i)

12.
def Start12():
    b= -5
    nb = b.bit_length()
#    bb = b.to_bytes(2, byteorder='big')
    bb = b.to_bytes(2, byteorder='big', signed = True)
    print(b, nb, bb)
    m = int.from_bytes(b'\x00\x10', byteorder='big')
    print(m)

13.
def Start13():
    import math
    q=3.17
    q1 = q.as_integer_ratio()
```

```
    print(q1)
    r = math.pi
    r1=r.as_integer_ratio()
    print(r1)
```

14.

```
def Start14(p):
    i, s = 0, 0
    while True:
        i += 1
        s += 1/i**0.5
        if s > p:
            break
    print(i)
```

15.

```
def Start15(str):
    for s in str:
        print(s)
    else:
        print ("\nCycle has been done")
```

16.

```
def Start16(x):
    if x % 2 == 0:
        return True
    else:
         return False
```

17.

```
def Start17(x):
    if x % 2 == 0:
        return False
    else:
         return True
```

18.

```
def Start18(arr):
    k=len(arr)
    for i in range(k):
        arr[i]*=2
    print (arr)
```

19.

```
def Start19(arr):
   for i, elem in enumerate(arr):
      if elem % 2 ==0:
          arr[i]*=2
   print(arr)
```

20.

```
def Start20(s):
    k=0
```

```
while s != "":
    print( s)
    k=k+1
s = s[1:-1]
print (k)
```

**Задание к лабораторной работе по теме:**

**Основные понятия языка LISP. Атомы, точечные пары, списки. S-выражения. Функции в LISP.Данные в языке LISP. Типы данных. Арифметические функции, логические функции, функции работы со строковыми данными**

Проанализируйте фрагменты программ на языке LISP (GNU Common LISP). Модифицируйте программы для работы в среде Steel Bank Common LISP и в среде Visual LISP.

1.

```
(defun Sample01()
 (setq a (cons "a" 7))
 (print a)
 (setq b (atom a))
 (print b) )
```

2.

```
(defun Sample02()
 (setq a (cons "a" NIL))
 (print a)
 (setq b (atom a))
 (print b) )
```

3.

```
(defun Sample03()
 (setq a (cons "a" "b"))
 (print a)
 (setq b (atom a))
 (print b))
```

4.

```
(defun Sample04()
 (setq a (cons '("a"."b") nil))
 (print a)
 (setq b (atom a))
 (print b))
```

5.

```
defun Sample05()
  (setq  a #b110/11 b 18/10 c 21/7 d #o12/5 e #xa/2 f #3r22/2)
  (print a)
  (print b)
  (print c)
  (print d)
  (print e)
  (print f))
```

6.

```
(defun Sample6()
  (setq  a 123. b 123.0 c 1.23s2 d 1.23f-2 e 1.23d-3 f 1.23l4 g 1.23e-2)
  (print a)
```

```
  (print b)
  (print c)
  (print d)
  (print e)
  (print f)
  (print g))
```

7.

```
defun Sample7()
  (setq  a  #c(#b101 #xa1)  b #c(8/6 5) c #c(-1.75d-1 2.33l-2) d #c(1.75 0))
  (print a)
  (print b)
  (print c)
  (print d))
```

8.

```
(defun Sample8()
 (setq a #c(1 2) b 3 c (+ a b) d (* a b) e (/ a b))
 (print c)
 (print d)
 (print e) )
```

9.

```
(defun Sample9()
 (setq a #c(1 2) b #c(3 -4) c (+ a b) d (* a b) e (/ a b))
 (print c)
 (print d)
 (print e) )
```

10.

```
(defun Sample10()
 (setq a 21 b 4 c 2.7 d -5.2 )
(setq e (mod a b) f (rem a b)
g1 (floor c)  h1 (ceiling c) k1 (truncate c) l1 (round c)
g2 (floor d)  h2 (ceiling d) k2 (truncate d) l2 (round d))
 (print c)
 (print d)
 (print e)
 (print g1)
 (print h1)
 (print k1)
 (print l1)
 (print g2)
 (print h2)
 (print k2)
 (print l2))
```

11.

```
defun Sample11()
 (setq c1 (complex 1 1))
 (setq c2 (conjugate c1))
 (setq r (abs c1))
```

```
  (setq term (/ 0.18l3 pi))
  (setq phiRad (phase c1))
  (setq phiGrad (* term phiRad))
  (print c1)
  (print c2)
  (print r)
  (print phiRad)
  (print phiGrad))
```

12.

```
(defun Sample12()
  (setq s1 "This" s2 " is" s3 " a" s4 " text"
    s_All (concatenate 'string s1 s2 s3 s4))
  (print s_All)
  (setq U1 (subseq s_All 5))
  (setq U2 (subseq s_All 0 3))
  (setq U3 (subseq s_All 4 9))
  (print u1)
  (print u2)
  (print u3) )
```

13.

```
(defun Sample13()
   (setq r1 76.2f2)
   (setq s1 (write-to-string r1))
   (print s1)
   (setq s2 "15.7")
   (setq r2 (read-from-string s2) r2 (+ r2 1))
   (print r2))
```

14.

```
(defun Sample14()
      (setq a #c(1 2) b #c(3 -4) c (+ a b) d (* (* 7 a) (- b 5)) e (/ a b))
      (print c)
      (print d)
      (print e) )
```

15.

```
(defun Sample15()
   (setq r1 -777)
   (setq s1 (write-to-string r1))
   (print s1)
   (setq s2 "159")
   (setq r2 (read-from-string s2) r2 (+ r2 1))
   (print r2))
```

16.

```
(defun Sample16(a b)
      (setq c (and (> a 0) (< b 10)))
      (setq d (or (> a 0) (< b 10)))
      (print c)
      (print d))
```

17.

```
(defun Sample17(a b)
    (setq c (numberp a) d (number b))
    (print c)
    (print d))
```

18.

```
(defun Sample18(s)
    (setq s "This" s2 " is" s3 " a" s4 " text"
    (setq u1 (subseq s 5))
    (setq u2 (subseq s 0 3))
    (setq u3 (subseq s 4 9))
    (print u1)
    (print u2)
    (print u3) )
```

19.

```
(defun Sample19(a b c)
    (setq c (and (< a (+ b c)) (< b (+ a c)) (< c (+ a b))))
    (print c)
```

20.

```
(defun Sample20(a b c)
    (setq p1 (eq (* a a) (+ (* b b) (* c c))))
    (setq p2 (eq (* b b) (+ (* a a) (* c c))))
    (setq p3 (eq (* c c) (+ (* a a) (* b b))))
    (setq p (or p1 p2 p3))
    (print c) )
```

**Задание к лабораторной работе по теме:**

**Функции работы со списками и их суперпозиции в языке LISP. Функции работы со списками в языке Python. Глобальные и локальные переменные в языках LISP и Python.**

Проанализируйте фрагменты программ на языке LISP (GNU Common LISP). Модифицируйте программы для работы в среде Steel Bank Common LISP и в среде Visual LISP (если такая модификация требуется). Напишите программы – аналоги на языке Python. Следует учесть, что программа на языке Python, скорее всего, должна будет существенно отличаться от прототипа на любом из диалектов LISP.

1.

```
(defun Sample01()
    (setq a1 (cons 3 (cons 7 (cons 9 (cons 19 (cons 21 nil))))))
    (setq a2 (list 3 7 9 19 21))
    (setq a3 (quote (3 7 9 19 21)))
    (setq a4 '(3 7 9 19 21))
    (print a1)
    (print a2)
    (print a3)
    (print a4))
```

2.

```
(defun Sample02()
    (setq a1 (make-list 5))
    (setq a2 (make-list 5 :initial-element 3))
```

```
    (print a1)
    (print a2))
```

3.

```
(defun Sample03()
    (setq a1 (make-sequence 'list 5))
    (setq a2 (make-sequence 'list 5 :initial-element 3))
    (print a1)
    (print a2))
```

4.

```
(defun Sample04()
    (setq w nil)
    (push 3 w)
    (push '(1 9) w)
    (push "r" w)
    (print w)
    (setq b1 (pop w))
    (print w)
    (setq b2 (pop w))
    (print w)
    (setq b3 (pop w))
    (print w)
    (print b1)
    (print b2)
    (print b3)
    (terpri)
    )
```

5.

```
defun Sample05()
    (setq a '(1  2 3))
    (setq b1 (append a 4))
    (setq b2 (append a '(4)))
    (setq l1 (length b1))
    (setq l2 (length b2))
    (print a)
    (print b1)
    (print b2)
    (print l1)
    (print l2))
```

6.

```
(defun Sample06()
    (setq a '(1  2 3))
    (setq b (append '(4) a))
    (print a)
    (print b))
    )
```

7.

```
(defun Sample07()
    (setq a '(1  2 3 4 2))
```

```
    (setq b (remove 2 a))
    (print a)
    (print b)
    (terpri)
    )
```

8.

```
(defun Sample088()
    (setq a '(1  2 3 4 2))
    (setq b (delete-if  '#oddp a))
    (print a)
    (print b))
    )
```

9.

```
(defun List_9()
    (setq a '(1 7 2 3 4 2))
    (setq b (delete-if-not  #'oddp a))
    (print a)
    (print b))
    )
```

10.

```
(defun Sample10()
    (setq a '(1 7 2 3 4 2))
    (setq b (delete-if  #'evenp a))
    (print a)
    (print b))
    )
```

11.

```
(defun Sample11()
    "It is me!"
    (setq  x  1 y  (+  x  4))
    (print "Before LET")
    (print x)
    (print y)
    ( let ((x 5) (y (- x 12)) )
    (print "Let")
    (print x)
     (print y)
    )
    (print "After LET")
    (print x)
    (print y))
```

12.

```
(defun Sample12()
    "It is me!"
    (setq x 21)
    (print "Before")
    (print x)
    (dotimes  (x 10) (format t  "~d" x))
```

```
    (print "After")
    (print x))

13.
(defun Sample13()
    (let ( (l1 7) (l2 1) (l3 17))
    (format t "~% l1   ~a" l1)
    (format t "~% l2   ~a" l2)
    (format t "~% l3   ~a" l3)
    (setq a (+ l1 l2 l3))
    )

14.
    (defvar  *a1* 10)
    (defparameter *a2* 20)
    (defconstant +a3+ 17)
(defun Sample14()
    "It is me!"
    (setq a0 7 )
    (rotatef a0  *a1*  *a2* )
    (setq b (list a0  *a1*  *a2*))
    (print b)
    (shiftf a0 *a1* *a2*)
    (setq c (list a0  *a1*  *a2*) )
    (print c))

15.
    (defconstant +N+ 7)
    (defvar *S-Sum*)
(defun Sample15 ()
    "This is the simple sample"
    (let ((r (read)))
        (setq *S-Sum* (* pi r r +N+)))
    (format t "~%  ~a" *S-Sum*))

16.
    (defparameter *S*)
(defun Sample16()
    "This is the simple sample"
    (let ((r (read)))
        (setq *S* (* pi r r)))
    (format t "~%  ~a" *S*))

17.
(defun Sample17 ()
    "This is the sample of LET"
    (setq x 3)
    (format t "Before  Let: ~a~%" x)
    (let* ((x 20) (y (+ x 7)) (z (+ x y 10)))
    (format t " LET: ~a~%" y)
    (format t " LET: ~a~%" z)
    )
    (format t "After Let:  ~a~%"  y)
```

```
    (format t "After Let:  ~a~%"  z))
```

18.

```
(defun Sample18()
    "Initialization of 1 variable"
    (format t "~%  This is Sample18 ")
    (setq l 7)
    (format t "~% l =  ~a" l)
    (help_fun)
    (format t "~% l =  ~a" l))
(defun help_fun()
    (format t "~%  This is a help_fun ")
    (let ((l 5))
    (format t "~% l =  ~a" l))
```

19.

```
(defun Sample19()
    (let* ( (l1 7) (l2 1) (l3 (+ l1 (* 2 l2))) (l4 (* 7(+ l1 l2 l3))))
    (format t "~% l1   ~a" l1)
    (format t "~% l2   ~a" l2)
    (format t "~% l3   ~a" l3)
    (format t "~% l4   ~a" l4)
    )
```

20.

```
(defun Sample20()
    (let* ( (l1 7) (l2 1) (l3 (+ 17 l1 l2)))
    (format t "~% l1   ~a" l1)
    (format t "~% l2   ~a" l2)
    (format t "~% l3   ~a" l3)
    (setq a (+ l1 l2 l3)))
    (format t "~% a   ~a" a))
```

**Задание к лабораторной работе по теме:**

**Виды рекурсии. Реализация рекурсии в языках LISP и Python.**

Проанализируйте фрагменты программ на языке LISP (GNU Common LISP или Steel Bank Common LISP). Модифицируйте программы, написанные на диалекте GNU Common LISP, для работы в среде Steel Bank Common LISP и в среде Visual LISP (если такая модификация требуется). Напишите программы – аналоги на языке Python. Следует учесть, что программа на языке Python, скорее всего, должна будет существенно отличаться от прототипа на любом из диалектов LISP.

1.

```
(defun copy-list(l)
        (cond ((null l) nil)
            (t (cons (car l)
              (copy-list (cdr l))))))
```

2.

```
(defun is_a_member(a l)
        (cond ((null l) nil)
            ((equal (car l) a) l)
              (t (is_a_member a
```

```
                    (cdr l)))))
```

3.

```
(defun my_Reverse(l)
        (cond ((null l) nil)
            (t (append (my_Reverse (cdr l))
                    (cons (car l) nil)))))
```

4.

```
(defun Rev_Append(l1 l2)
    (cond ((null l1) l2)
            (t (Rev_Append (cdr l1)
                    (cons (car l1) l2)))))
```

5.

```
(defun First_Atom(l)
    (cond ((null l) nil)
            ((atom (car l)) (car l))
            (t (First_Atom (car l)))))
```

6.

```
(defun Sub_List(l1 l2)
    (cond ((null l2) nil)
            ((SubL l1 l2))
            (t (Sub_List l1 (cdr l2)))))

(defun SubL (l1 l2)
    (cond ((null l1) t)
            ((null l2) nil)
            ((eql (car l1) (car l2))
              (SubL (cdr l1) (cdr l2)))))
```

7.

```
(defun TranOp (X)
    (cond ((eq X 'TRUE) t)
            ((eq X 'FALSE) nil)
            ((atom X)  X )
            (t (TranExpr X))))

(defun TranExpr(X)
    (cond ((eq (cadr X) '&)
            (list 'AND (TranOp (car X))
                    (TranOp (caddr X))))
            (t (list 'OR (TranOp (car X))
                    (TranOp (caddr X))))))
```

8.

```
(defun Member_in_All_Levels (A  L)
    (cond ((atom l) (eql A L))
            (t  (or (Member_in_All_Levels  A (car L))
                    (Member_in_ALL_Levels A (cdr L))))))
```

9.

```
(defun my_Equal (X  Y)
        (cond ((atom X) (eql X Y))
              ((atom Y) nil)
              (t  (and (my_Equal (car X) (car Y))
                   (my_Equal (cdr X) (cdr Y))))))
```

10.

```
(defun my_SumPr_1 (X)
        (cond ((null  X) (cons 0 1))
              (t  (cons  (+ (car X) (car (my_SumPr_1 (cdr X))))
                         (* (car X) (cdr (my_SumPr_1 (cdr X))))))))
```

11.

```
(defun my_SumPr_2 (X)
        (cond ((null  X) (cons 0 1))
              (t  (let ((Y (car X)) (Z  (my_SumPr_2  (cdr  X)))
                    (cons (+ Y (car Z)) (*  Y  (cdr Z))))))))
```

12.

```
(defun my_SumPr_3 (X)
       (Accum X 0 1))
(defun Accum(X S P)
        (cond ((null  X) (cons S P))
              (t  (Accum  (cdr X) (+ S (car X))
                                  (*  P  (car X))))))
```

13.

```
(defun my_Reverse_All (X)
        (cond ((atom  X) X)
              (t  (append  (my_Reverse_All (cdr X))
                           (cons (my_Reverse_All (car X)) nil)))))
```

14.

```
(defun my_RevDepth (X)
        (cond ((atom  X) X)
              (t  (my_RevBroad X nil))))
(defun my_RevBroad(Y Res)
        (cond ((null Y) Res)
              (t (my_RevBroad (cdr Y)
                  (cons (my_RevDepth (car Y)) Res)))))
```

15.

```
(defun my_Flatten (X)
        (cond ((null  X) nil)
              ((atom X) (cons X ()))
              (t  (append  (my_Flatten (car X))
                           (my_Flatten (cdr X)) )) ))
```

16.

```
(defun my_Flatten_1 (X)
       (my_Flat X nil))
(defun my_Flat (Y Res)
        (cond ((null  Y) Res)
```

```
        ((atom Y) (cons Y Res))
         (t  (my_Flat (car Y) (my_Flat (cdr Y) Res))))
```

17.

```
(defun my_Collect (X)
    (my_Col X nil))
    (defun my_Col(Y Res)
     (cond ((null  Y) Res)
            ((atom Y) (cond  ((member Y Res ) Res)
                        (t (cons Y Res))))
           (t  (my_Col (car Y) (my_Col (cdr Y) Res)))))
```

18.

```
(defun Gorner1(P x)
     (cond ((null (cdr P)) (car P))
          ((null (cddr P)) (+ (car P) (* x (cadr P))))
          (t (+ (car P) (* x (Gorner1 (cdr P) x))))))
(defun GornerStart(P x)
     (cond((and (listp P) (numberp x))
          (setq PVal (Gorner1 P x))
               (format t "~% Polynomial value is ~a" PVal))
               (t (format t "~% Wrong input data!"))) )
```

19.

```
(defun myRec-1(a)
      (cond ((null a) nil)
          ((null (cdr a)) (cons a nil))
               (t (list (car a) (myRec-1 (cdr a))))))
```

20.

```
(defun myRec-2(a)
   (cond ((null a) nil)
       ((null (cdr a)) (cons (car a) nil))
            (t (cons (car a) (myRec-2 (cadr a))))))
```

**Задание к лабораторной работе по теме:**

**Ассоциативные списки в языке LISP. Словари в языке Python. Функции для ассоциативного поиска. Hash – таблицы в языках LISP и Python. Анонимные функции в языках LISP, Python встроенном языке Matlab.**

Проанализируйте фрагменты программ на языке LISP (GNU Common LISP или Steel Bank Common LISP). Модифицируйте программы, написанные на диалекте GNU Common LISP, для работы в среде Steel Bank Common LISP и в среде Visual LISP (если такая модификация требуется). Напишите программы – аналоги на языке Python. Следует учесть, что программа на языке Python, скорее всего, должна будет существенно отличаться от прототипа на любом из диалектов LISP.

1.

```
(defun anonymous_1(arg)
    (setq q (sin ((lambda(arg_grad) (/ (* pi arg_grad) 180) ) arg)))
    (return-from anonymous_1 q))
```

2.

```
(defun assoc_sample (w)
```

```
(setq myList '((name box) (width 5) (depth 4.9) (size 7.7) (11 9.99)))
(setq search w)
(setq z (assoc search mylist))
(prin1 z)
(terpri)
(setq zval (cadr z))
(prin1 zval))
```

3.

```
(defun  my-hash-01()
(setq h (make-hash-table))
(setf (gethash 'Kate h) '(programmer (202 Sunny Avenue)))
(setf (gethash 'Nick h) '(driver (101 Moon Street)))
(setf (gethash 'Diana h) '(pilot (303 Star Street)))
(setq h1 (gethash 'Diana h))
(format t "~% ~a" h1)
(setq h11 (car h1))
(format t "~% ~a" h11)
(setq h12 (cdr h1))
(format t "~% ~a" h12)
(setq h122 (cdadr h1))
(format t "~% ~a" h122))
```

4.

```
(defun Apply_001 ()
 (setq      b  (apply #'(lambda (u v w) (abs (+ u (* 2 v) (* 3 w))))
                #(1 2  7)))
 (format t "~% ~a" b))
```

5.

```
(defun lambdaSample ()
 (setq     a (mapcar #'(lambda (u v w) (abs (+ u (* 2 v) (* 3 w))))
              '(1 2 3)
              '(-2 4 -9 1 8)
              '(1 0 3 -7)))
 (print a))
```

6.

```
(defun lambdaSample2()
 (mapcar #'(lambda (x y)
      (+ x 3 y))
   '(10 20 30) '(7 1 9)))
```

7.

```
(defun lambdaSample3()
 (setq a (+ ((lambda (x) (+ (* x x) (* 2 x) 1)) 3) 5 6))
 (print a))
```

8.

```
(defun lambdaSample41()
(setq a '(1 7 2 3 4 2))
(setq b (delete-if-not  #'(lambda(x) (< (* 2 x) 7)) a))
(print a)
```

```
        (print b))

        9.
(defun lambdaSample3()
        (setq a (+ ((lambda (x) (+ (* (+ x 1) (+ x 3)) (* 2 x) 1)) 9) 7 2))
        (print a))

        10.
(defun mul-N (N)
          #'(lambda (x) (* x N)))

        11.
(defun add-N (N)
         #'(lambda (x) (+ x N)))

        12.
(defun lambdaMul()
        (mapcar #'(lambda (x y)
        (* x y))
        '(10 20 30) '(7 1 9)))

        13.
(defun lambdaDiv()
        (mapcar #'(lambda (x y)
              (/ x  y))
           '(10 20 30) '(7 1 9)))

        14.
(defun lambdaSumSq()
        (mapcar #'(lambda (x y)
            (+ (* x x) (* y y)))
          '(-1 7 2) '(2 4 0)))

        15.
(defun lambdaInv()
        (mapcar #'(lambda (x y)
        (+ (/ 1 x) (/ 1 y)))
        '(-1 7 2) '(2 4 0)))

        16.
(defun lambdaInvSq()
        (mapcar #'(lambda (x y)
        (+ (/ 1 (* x x)) (/ 1 (* y y))))
        '(-1 7 2) '(2 4 0)))

        17.
(defun lambdaSinCos()
        (mapcar #'(lambda (x y)
        (+ (sin x) (cos y)))
        ‘(-0.1 0.7 0.2) '(0.2 0.4 0.3)))

        18.
(defun lambdaExp()
```

```
(mapcar #'(lambda (x y)
(exp (+ x y))
(-0.1 0.7 0.2) '(0.2 0.4 0.3)))
```

19.

```
(defun lambdaSinPlus()
(mapcar #'(lambda (x y)
(sin (+ x y))
(-0.1 0.7 0.2) '(0.2 0.4 0.3)))
```

20.

```
(defun lambdaCosPlus()
(mapcar #'(lambda (x y)
(cos (+ x y))
(-0.1 0.7 0.2) '(0.2 0.4 0.3)))
```

**Задание к лабораторной работе по теме:**

**Функционалы в языке LISP. Виды отображающих функционалов. Совместное применение функционалов и анонимных функций.**

Проанализируйте фрагменты программ на языке LISP (GNU Common LISP или Steel Bank Common LISP). Модифицируйте программы, написанные на диалекте GNU Common LISP, для работы в среде Steel Bank Common LISP и в среде Visual LISP (если такая модификация требуется). Напишите программы – аналоги на языке Python. Следует учесть, что программа на языке Python, скорее всего, должна будет существенно отличаться от прототипа на любом из диалектов LISP.

1.

```
(defun next_el_plus(x1)
 (cond
   (x1 (cons (1+ (car x1))
   (next_el_plus (cdr x1)))))
```

2.

```
(defun map-el(fn xl)
 (cond
   (xl (cons (funcall fn (car xl))
   (map-el fn (cdr xl)))))
```

3.

```
(defun next-el(xl)  ; поэлементное преобразование xl с помощью fn
(map-el '1+ xl) )
(defun square_1 (xl) (map-el #'(lambda (x) (* x x)) xl)
)
```

4.

```
(defun decart_1 (x y)
 (map-el #'(lambda (i)
          (map-el #'(lambda (j) (list i j)) y)
          ) x)
```

5.

```
(defun list_up (ll)
 (cond
```

```
   (ll (append (car ll)
   (list_up (cdr ll))
  )))
 )
```

6.

```
(defun map_up (fn ll)
 (cond
   (ll (append (funcall fn (car ll))
   (map_up fn (cdr ll))))))
```

7.

```
(defun myMap()
        (setq a '("lower" "UPPER" "" "123"))
        (setq b (map 'vector #'string-upcase a))
        (format t "~% ~a" b))
```

8.

```
(defun myMapcan1()
        (setq l1 '(nil nil nil d e) l2  '(1 2 3 4 5 6))
        (setq a (mapcan #'(lambda (x y) (if (null x) nil (list x y)))
              l1 l2))
        (format t "~%  ~a"  a)
        (format t "~%  ~a"  l1)
        (format t "~%  ~a"  l2))
```

9.

```
(defun myMapcar1()
        (setq a (mapcar   #'* '(1 2 3 4 5) '(10 9 8 7 6)))
        (format t "~%  ~a" a)
        (setq b (mapcar   #'+  '(1 2 3 4 5) '(10 9 8 7 6) '(4 7 11)))
        (format t "~%   ~a" b))
```

10.

```
(defun mMapcar2()
         (setq     a (mapcar #'(lambda (u v w) (abs (+ u (* 2 v) (* 3 w))))
                     '(1 2 3)
                     '(-2 4 -9 1 8)
                     '(1 0 3 -7)
            ) )
        (format t "~%  ~a" a))
```

11.

```
(defun myMapcar3 ()
        (setq a (mapcar #'car '((1 a) (2 b) (3 c))))
        (format t "~%  ~a" a)
        )
```

12.

```
(defun myMapcar4()
        (setq a (mapcar #'car '((1 a) (2 b) (3 c)) '((7 d) (8 e) (9 f))))
        (format t "~%  ~a" a))
```

13.

```
(defun myMapcon1()
	(setq l1 '(nil nil nil d e) l2  '(1 2 3 4 5 6))
	(setq a (mapcon #'(lambda (x y) (if (null x) nil (list x y)))
		l1 l2))
	(format t "~%  ~a"  a)
	(format t "~%  ~a"  l1)
	(format t "~%  ~a"  l2))

	14.
(defun mapcon_02()
	(setq l1 '(nil nil nil d e) l2  #(1 2 3 4 5 6))
	(setq a (mapcon #'(lambda (x y) (if (null x) nil (list x y)))
		l1 l2))
	(format t "~%  ~a"  a)
	(format t "~%  ~a"  l1)
	(format t "~%  ~a"  l2))

	15.
(defun myMapl1()
	(setq Something nil)
	(setq a (mapl #'(lambda (x) (push x Something)) '(1 2 3 4)))
	(format t "~%  ~a" a)
	(format t "~%  ~a" Something))

	16.
(defun myMapl2()
	(setq Something nil)
	(setq a (mapl #'(lambda (x) (push x Something)) #(1 2 3 4)))
	(format t "~%  ~a" a)
	(format t "~%  ~a" Something)
	)

	17.
(defun myMaplist01()
	(setq a (maplist  #'(lambda(y)  (cons 'w y)) '(1 2 3 4 5)))
	(format t "~% ~a" a))

	18.
(defun myMaplist2()
	(setq a (maplist  #'(lambda(y)  (cdr y)) '(1 2 3 4 5)))
	(format t "~% ~a" a))

	19.
(defun myMaplist3()
	(setq a (maplist  #'(lambda(y)  (car y)) '(1 2 3 4 5)))
	(format t "~% ~a" a))

	20.
(defun myMaplist4(argList1 argList2 argList3)
	(setq a (maplist  #'list
		argList1 argList2 argList3))
	(format t "~% ~a" a))
```

**Задание к лабораторной работе по теме:**

**Записи, классы и объекты в языке LISP. Синтез функциональной и объектно-ориентированной парадигм программирования. Диалект Visual LISP для системы компьютерной графики AutoCAD.**

Проанализируйте фрагменты программ на языке LISP (GNU Common LISP или Steel Bank Common LISP). Модифицируйте программы, написанные на диалекте GNU Common LISP, для работы в среде Steel Bank Common LISP (если такая модификация требуется). Напишите программы – аналоги на языке Python. Следует учесть, что программа на языке Python, скорее всего, должна будет существенно отличаться от прототипа на любом из диалектов LISP. В Visual LISP отсутствуют средства для работы с записями, классами и объектами. В Python отсутствуют средства для взаимодействия со средой AutoCAD.

1.

```
(defun start_class()
        (defclass a-point()
            (x y z)))
```

2.

```
(defun init_values(some-point x y z)
        (setf  (slot-value some-point 'x) x
              (slot-value some-point 'y) y
              (slot-value some-point 'z) z))
```

3.

```
(defun distance_class(some-point)
        (with-slots (x y z)
          some-point
         (sqrt (+ (* x x) (* y y) (* z z)))))
```

4.

```
(defun distance_class_1(some-point)
        (setq xx (slot-value some-point 'x)
             yy (slot-value some-point 'y)
             zz (slot-value some-point 'z)
             dd (sqrt (+ (* xx xx) (* yy yy) (* zz zz)))))
```

5.

```
(defun work_class(coord-x coord-y coord-z)
        (setq the_first_point (make-instance 'a-point))
        (init_values the_first_point  coord-x coord-y coord-z)
        (setq dd (distance_class_1 the_first_point))
        (print dd))
```

6.

```
(defun start-abstract-animal()
        (defclass abstract-animal()
                ((legs :reader leg-count :initarg :legs)
                  (comes-from :reader comes-from :initarg comes-from))))
```

7.

```
(defun start-mammal()
        (defclass mammal(abstract-animal)
                ((diet :initform 'antelopes  :initarg :diet))))
```

8.

```
(defun start-aardvark()
        (defclass aardvark(mammal)
                ((cute-p :accessor cute-p  :initform nil))))
```

9.

```
(defun start-aardvark()
        (defclass aardvark(mammal)
                ((cute-p :accessor cute-p  :initform nil))))
```

10.

```
(defun start-figurine-aardvark()
        (defclass figurine-aardvark(aardvark figurine)
                ((name :reader aardvark-name  :initarg :aardvark-name)
                  (diet :initform nil))))
```

11.

```
(defun Init-instance()
         (setf Eric (make-instance 'figurine-aardvark
                                :legs 4
                                :made-by "Jen"
                                :made-in "China"
                                :aardvark-name "Eric")))
```

12.

```
(defun make-struct(x-val y-val z-val)
        (setf new-point (make-a-point :x x-val :y y-val :z z-val))
       )
```

13.

```
(defun declare-struct()
        (defstruct a-point
            x y z))
```

14.

```
(defun change-struct(some-struct new-x)
        (setf (a-point-x some-struct) new-x))
```

15.

```
(defun HelloCAD       (dx dy)
        (setq p1 (getpoint "\nEnter the first point"))
        (setq p2x (+ (car p1) dx))
        (setq p2y (+ (cadr p1) dy))
        (setq p2 (list p2x p2y))
        (command "_line" p1 p2 "")
       )
```

16.

```
(defun MyLine (a b)
        (setq col (acad_colordlg 3))
        (command "color" col)
        (setq p1 (getpoint '(100 100) "\n¬ведите точку"))
        (setq     x (+ a (car p1))
         y (+ b (cadr p1))
```

```
  )
  (setq p2 (list x y))
  (command "_line" p1 p2 "")
 )

17.
(defun id ()
  (setq p1 (getpoint "\nInitial point:   "))
  (setq l (getdist "\nInitial length:   "))
 )
                                        ;
(defun Quadr (l a  / p2 p3 p4)
  (setq p2 (polar p1 a l))
  (setq p3 (polar p2 (+ a (/ pi 2)) l))
  (setq p4 (polar p3 (+ a pi) l))
  (command "_line" p1 p2 p3 p4 "c")
 )
                                        ;
(defun qn ()
  (setq b 0.0)
  (id)
  (quadr l b )
  (while (<= b (* 2 pi))
    (quadr l b )
    (setq b (+ b (* pi 0.1)))
    (setq l (* l 0.9))
  )
 )

18.
(defun multistepD ()
  (setvar "CMDECHO" 0)
  (setvar "BLIPMODE" 0)
  (command "_limits" '(0 0) '(297 210) "zoom" "a")
  (textscr)
  (setq     n  (getint "\nEnter number of steps N=  ")
   Xt (getreal "\nEnter the X-coordinate of the base point Xt= ")
   Yt (getreal "\nEnter the Y-coordinate of the base point Yt= ")
   Bt (list Xt Yt)
  )
  (repeat n
    (setq d (getreal "\nEnter the diameter of step D= ")
     l (getreal "\nEnter the length of step L= ")
     r (* 0.5 d)
    )
    (setq t1 (polar bt (/ pi 2) r)
     t2 (polar t1 0 l)
     t4 (polar bt (* 1.5 pi) r)
     t3 (polar t4 0 l)
    )
    (command "_pline" bt "_w" 0.1 0.1 t1 t2 t3 t4 "_c")
    (setq bt (polar bt 0 l))
  )
```

```
)
```

19.

```
(defun vlcmdfSample()
  (setq p1 (list 10 10) p2 (list 100 70) p3 (list (+ 100 100) 70))
  (vl-cmdf "_line" p1 p2 p3       ""  )
 )
```

20.

```
(defun init-food()
  (defclass food () ()
  )
(defun init-pie ()
    (defclass pie (food)
    ((filling :accessor pie-filling :initarg :filling
                        :initform 'apple))
    )
    (defmethod cook ((p pie))
     (print "Cooking a pie!")
     (setf (pie-filling p) (list 'cooked (pie-filling p)))
    )
    (defmethod cook :before ((p pie))
      (print "A pie is about to be cooked!")
    )
    (defmethod cook :after ((p pie))
      (print "A pie has been cooked!")
    )
  )
(defun start-pie()
  (init-food)
  (init-pie)
  (setq American-pie (make-instance 'pie :filling 'apple))
  (cook American-pie)
  )
```

**Индивидуальные задание к разделу 1.**

Запрограммируйте представленные $\lambda$ - выражения на языках LISP и Python:

1. $\lambda x.\lambda y$ **(* (+ *x* 2) (– *y* 3))**;
2. $\lambda x.\lambda y$ **(* *x y*) (– *y* 3))**;
3. $\lambda x.\lambda y$ **(/ (+ *x* 2) (– *y x*))**;
4. $\lambda x.\lambda y$ **(* (+(* 3 *x* 2) (– *y* 3))**;
5. $\lambda x.\lambda y$ **(* (+ *x* 2) (– *y* 3 7))**;
6. $\lambda x.\lambda y$ **(* (+ *x* 2 *y*) (– *y* 3))**;
7. $\lambda x.\lambda y$ **(* (/ *x* 2 *y*) (– *y* 3))**;
8. $\lambda x.\lambda y$ **(* (* *x* 7) (+ *y* 9))**;
9. $\lambda x.\lambda y$ **(* (+ *x y*) (– *y* 3 *x*))**;
10. $\lambda x.\lambda y$ **(* (* *x y* 2) (+ *y x* 3))**;
11. $\lambda x.\lambda y$ **(* (/ (* *y x*) 2) (– *y* 9))**;
12. $\lambda x.\lambda y$ **(* (+ *x* 2) (– *y* 3))**;

13. $\lambda x.\lambda y$ **(/ (* *x y* 2) (– *y x* 7))**;
14. $\lambda x.\lambda y$ **(* (/ *x y* 2) (– *y x* 5))**;
15. $\lambda x.\lambda y$ **(/ (+ *x y* 9) (– *y* (* *x* 3)))**;
16. $\lambda x.\lambda y$ **(+ (* *x y* 7) (– *y x* 3))**;
17. $\lambda x.\lambda y$ **(/ (/ *x y* 7) (– *y x* 3))**;
18. $\lambda x.\lambda y$ **(* (* *x* 17) (+ *y* (* *x* 5) 3))**;
19. $\lambda x.\lambda y$ **(+ (* *x y* 7) (– *y x* 3))**;
20. $\lambda x.\lambda y$ **(– (* *x* (+ *y* 7) 9) (– *y x* 3))**.

**Индивидуальные задание к разделу 2.**

Задания выполняются на языках LISP и Python.

1. Написать функцию, которая проверяет, является ли список палиндромом. Пример списка-палиндрома: (2 3 “q” (1 7) 5 (1 7) “q” 3 2);
2. Написать функцию, которая список ‘(a1 a2 a3 …aN) преобразует в сложный список (a1 (a2 (a3 ( (…(aN)…)));
3. Написать функцию, которая «упаковывает» повторяющиеся элементы списка в подсписки следующего формата: (“Packed” <number_of_elements> <packed_element>). Например, список (7 7 7 “a” “b” “b” “b” “b” “b”) должен быть преобразован в список ((“Packed” 3 7) “a” (“Packed” 5 “b”));
4. Дан список из n чисел и натуральное число m<n. Для каждой группы из m элементов, которые находятся рядом, вычислить ее сумму. Написать функцию, которая выдает список из всех возможных сумм. Пример: (7 1 4 2 3), m=3. S= (12 7 9).
5. Написать функцию, которая список ‘(a1 a2 a3 …aN) преобразует в сложный список ((((aN)…a3) a2) a1).
6. Написать функцию, которая находит наибольшее по модулю число, содержащееся в списке с подсписками.
7. Написать функцию (gorner n lst x) — вычисления по схеме Горнера значения многочлена степени n в точке x. Коэффициенты многочлена должны быть заданы в списке lst.
8. Написать функцию — разложение числа n на простые множители. Вернуть список простых чисел, произведение которых есть n.
9. Написать функцию, которая список ‘(a1 a2…aN) преобразует в сложный список ((a1 a2) (a2 a3) …). При нечетном N послндний подсписок должен содержать 1 элемент.
10. Написать функцию, которая по заданному списку с подсписками возвращает список из сумм числовых элементов подсписков.
11. Написать функцию, которая определяет атом, который чаще всего встречается в списке с подсписками.
12. Написать функцию, которая определяет максимальную длину подсписка в списке.
13. Написать функцию, которая список ‘(x1 + x2 + x3 +… xN) преобразует в список ‘(+ x1 x2 x3 …xN)
14. Написать функцию, которая по списку с подсписками строит списки из положительных числовых элементов, отрицательных числовых элементов и выводит количество нулевых элементов.
15. Написать функцию, которая находит подсписок, реже всего встречающийся в данном списке.
16. Написать функцию, которая находит в данном списке подсписок минимальной длины.
17. Написать функцию, которая находит в данном списке с подсписками количество чисел, кратных заданному целому числу.
18. Написать функцию, которая находит в данном списке с подсписками количество подсписков, содержащих заданное число.
19. Написать функцию, которая находит в данном списке количество подсписков, содер-

жащих не менее заданного количества элементов.

20. Написать функцию, которая находит в данном массиве количество подсписков, сумма числовых элементов которых превосходит заданное число.

**Индивидуальные задание к разделу 3.**

Задания выполняются на языках LISP и Python.

1. Написать функцию, которая принимает целое число и возвращает из заданного списка с подсписками и массивами наименьшее из четных чисел, больших аргумента.
2. Написать функцию, которая принимает число и возвращает из заданного списка с подсписками и массивами наименьшее по модулю из чисел списка, но с модулем, большим модуля аргумента.
3. Написать функцию, которая находит произведение двух матриц, заданных в виде массивов. Предусмотреть проверку корректности ввода исходных данных.
4. Написать функцию, которая проверяет, состоят ли два заданных списка из одних и тех же элементов (независимо от порядка их расположения).
5. Написать функцию, которая меняет местами элементы с заданными номерами заданного списка или одномерного массива.
6. Написать функцию, которая проверяет расположены ли числовые элементы списка или одномерного массива в порядке убывания.
7. Написать функцию для нахождения максимального из числовых элементов двумерного массива.
8. Написать функцию для вычисления значения синуса для заданного аргумента с помощью ряда Тейлора. Предусмотреть проверку корректности данных.
9. Написать функцию, отбирающую числовые элементы двумерного массива, принадлежащие заданному интервалу. Предусмотреть проверку корректности данных.
10. Написать функцию, которая переставляет местами столбцы матрицы, заданной в виде двумерного массива. Столбцы матрицы задаются своими номерами. Предусмотреть проверку корректности данных.
11. Написать функцию для нахождения коэффициентов многочлена являющегося линейной комбинацией двух заданных многочленов (коэффициеты всех многочленов задаются в виде упорядоченных списков или одномерных массивов, степени многочленов могут не совпадать). Предусмотреть проверку корректности данных.
12. Написать функцию для нахождения коэффициентов многочлена являющегося производной заданного многочлена (коэффициеты всех многочленов задаются в виде упорядоченных списков или одномерных массивов, степени многочленов могут не совпадать). Предусмотреть проверку корректности данных.
13. Написать функцию, которая находит чаще всего встречающийся в двумерном массиве атом типа «строка».
14. Написать функцию, которая умножает на заданное число заданные своими номерами элементы из двумерного массива. Предусмотреть проверку корректности данных.
15. Задан список, состоящий из подсписков длины 3. Каждый подсписок содержит координаты 3 точек в пространстве. Проверить, являются ли заданные содержащимися в подсписках тройками точек плоскости параллельными.
16. Дан список, содержащий подсписки. Написать функцию, которая возвращает подсписок, сумма числовых элементов которого максимальна.
17. Написать функцию, которая по заданному списку с подсписками и одномерными массивами возвращает список из сумм числовых элементов подсписков.
18. Написать функцию, которая находит наибольший из отрицательных и наименьший из положительных элементов данного списка.
19. Написать функцию, которая проверяет, образуют ли числовые элементы списка или одномерного массива арифметическую прогрессию.
20. Координаты узлов ломаной линии на плоскости образуют список или одномерный массив. Найти длину ломаной. Проверить корректность данных.

**Индивидуальные задание к разделу 4.**

Задания выполняются на языках LISP и Python.

1. В кадровой службе предприятия содержатся сведения о работниках предприятия: табельный номер, ИНН, фамилия, дата рождения (в виде дд.мм.гггг), образование. Данные содержатся в виде списка. Данные о каждом работнике представлены в виде подсписка в общем списке. Организовать выбор работников по возрасту и образованию. Отобранные данные должны быть представлены в виде списков.
2. В торговой фирме исследуется спрос на непродовольственные товары. Для этого собираются сведения о покупках: код товара, код товарной группы, цена за единицу, количество проданных единиц. Данные содержатся в виде списка. Данные о каждой покупке представлены в виде подсписка в данном списке. Организовать выбор покупок по единичной цене, по потраченной денежной сумме. Отобранные данные должны быть представлены в виде списков.
3. В деканате содержатся сведения о студентах: номер студенческого билета, группа, фамилия, дата рождения, успеваемость в последнюю сессию. Успеваемость должна быть представлена в виде подсписка списка, состоящего из элементов (<код дисциплины> <код вида контроля> <оценка>). Организовать выбор студентов, не имеющих задолженностей по результатам сессии. Отобранные данные должны быть представлены в виде списков. Код вида контроля: 0 — зачет; 1 — экзамен. Оценки за экзамен имеют коды: 5 — «отлично»; 4 — «хорошо»; 3 — «удовлетворительно»; 2 — «неудовлетворительно». Результаты зачетов кодируются следующим образом: 1 — «зачтено»; 0 — «не зачтено».
4. Провайдер мобильной связи собирает сведения о клиентах: № SIM – карты, тарифный план, услуги, оплата услуг. Сведения об услугах и их оплате должны быть представлены подсписка (<порядковый номер услуги> <код услуги> <стоимость>). Организовать отбор клиентов по видам услуг. Коды услуг: 0 — входящие звонки; 1 — исходящие звонки; 2 — входящие sms; 3 — исходящие SMS; 4 — мобильный Интернет.
5. В банке собраны сведения о кредитной истории каждого из клиентов (физических лиц): расчетный счет, фамилия клиента, кредитная история. Кредитная история представлена в виде подсписка (<порядковый номер кредита> <код вида кредита> <сумма кредита> <код своевременности погашения>). Организовать выбор клиентов по своевременности возврата кредита. Коды вида кредита: 0 — потребительский; 1 — ипотечный. Коды своевременности погашения: 0 — своевременно; 1 — с опозданием; 2 — не погашен.
6. Процессинговый центр обслуживает расчеты по дебетовым карточкам. В каждом обращении указывается порядковый номер операции, код счета, код вида операции, запрашиваемая сумма. Организовать отбор запросов по запрашиваемой сумме, по виду операции. Коды вида операции: 0— выдача наличных; 1 — оплата кредита; 2 — оплата покупок; 3 — оплата услуг; 4 — перевод денег на другой счет.
7. Администрация Интернет – ресурса анализирует поток сообщений. Каждое сообщение характеризуется данными (порядковый номер сообщения, идентификационный код пользователя, объем сообщения в байтах, код претензий к сообщению службы модерации). Организовать отбор пользователей по объему сообщения, наличию претензий службы модерации. Коды претензий службы модерации: 0 — нет замечаний; 1 — устранимые замечания; 2 — вещание прекращено службой модерации.
8. В системе продажи билетов в кинотеатр хранятся сведения: номер сеанса, номер зрительного зала, код категории фильма, количество проданных билетов. Организовать отбор сеансов по количеству проданных билетов, по коду категории фильма. Коды категории фильма: 0 — 0+; 1 — 6+; 2 — 16+.
9. Авторемонтная фирма собирает сведения о ремонтах: индивидуальный номер ремонта, клиент, стоимость ремонта, код страховки, код страховщика. Организовать отбор сведений по стоимости ремонта, по страховщикам. Коды страховки: 0 — отсутствие страховки; 1 — ОСАГО; 2 — КАСКО.

10. Кадровое агентство занимается поиском квалифицированных специалистов. Требования к специалисту: квалификация по диплому (ИНН, фамилия, направление подготовки по диплому, свидетельство о прохождении повышения квалификации, стаж работы по специальности, степень владения иностранным языком) Организовать отбор данных по направлению подготовки и степени владения иностранным языком. Степени владения иностранным языком: A1 — может объясниться в простейшей бытовой ситуации; A2 — может объясниться в магазине, на вокзале, при занятии неквалифицированным трудом; B1 — уверенно общается в быту, читает и адекватно воспринимает несложные тексты, может объясниться при занятии квалифицированным трудом; B2 — читает и адекватно воспринимает сложные тексты, пишет без ошибок; C1 — владеет языком в достаточной мере для работы в должности инженера; C2 — владеет языком в достаточной мере для работы в должности юриста, журналиста.
11. Телевизионный канал ведет учет рейтинга программ с использованием фокус - групп. Учет ведется по схеме: личный код зрителя, возраст зрителя, код социального статуса зрителя, заявленный доход зрителя, длительность просмотра программ. Длительность просмотра программ представлена в виде списка: (<код программы> <длительность просмотра в минутах>). Предусмотреть отбор данных по коду социального статуса зрителя. Коды социального статуса зрителя: 0 — учащийся; 1 — работающий по найму; 2 — предприниматель; 3 — пенсионер.
12. В поликлинике ведется учет посетителей: номер полиса обязательного медицинского страхования, страховщик, фамилия, код категории посетителя, причина посещения, сумма, которую должна выплатить страховая компания за прием. Предусмотреть отбор по коду категории посетителя. Коды категории посетителя: 0 — первичный; 1 — состоит на амбулаторном учете; 2 — состоит на специальном учете.
13. Страховая компания производит выплаты по системе ОСАГО, исходя из следующих сведений: номер аварии в списке, личные данные виновника аварии (номер и серия паспорта), реквизиты договора об ОСАГО, оценка ущерба, выполненная независимым экспертом. Организовать отбор данных по оценке ущерба. Критерии отбора: до 30000 рублей; от 30000 рублей до 50000 рублей; от 50000 рублей до 100000 рублей; от 100000 рублей до 300000 рублей; свыше 300000 рублей.
14. Диспетчерская служба скорой помощи ведет учет вызовов по следующей схеме: номер вызова, данные пациента, диагноз, установленный бригадой, код действий бригады. Организовать отбор данных по кодам действий бригады. Коды действий бригады: 0 — оказана помощь на месте; 1 — пациент доставлен в стационар.
15. Медицинская страховая компания оплачивает работу амбулаторных медицинских учреждений исходя из следующих данных: номер выставленного счета, реквизиты медицинского учреждения, реквизиты полиса обязательного медицинского страхования пациента, код оказанной медицинской помощи. Организовать отбор данных по коду оказанной медицинской помощи. Коды медицинской помощи: 0 — амбулаторный прием; 1 — физиотерапевтические процедуры; 2 — хирургические процедуры; 3 — сложные обследования и процедуры (УЗИ, МРТ).
16. В магазине сложной бытовой техники ведется учет продаж по схеме: номер продажи, код товарной группы, код товара, цена товара, табельный номер продавца, сопровождавшего продажу. Организовать отбор данных по  отбор данных табельному номеру продавца. Коды товарной группы: 0 — стиральные машины; 1 — холодильники; 2 — кухонное электрооборудование; 3 — телевизоры; 4 — мобильные устройства; 5 — компьютерная техника.
17. Сеть продовольственных магазинов ведет учет продаж товаров по следующей схеме: наименование товара, товарная группа, объем продаж товара за месяц в денежном выражении, среднее время хранения товара в магазине. Организовать отбор по среднему времени хранения товара в магазине. Коды товарных групп: 0 — хлебобулочные изделия; 1 — макаронные изделия; 2 — крупы; 3 — молочные продукты; 4 — консервы; 5 — фрукты и овощи.

18. Аварийная служба энергосети ведет учет аварий по следующей схеме: номер вызова, адрес, код вида аварии, длительность ремонта, стоимость ремонта. Организовать отбор данных по длительности ремонта. Коды вида аварии: 0 — обрыв контактной сети; 1 — замыкание в сети; 2 — несанкционированный отбор из сети.
19. Вуз анализирует карьеру выпускников. Учет ведется по следующим параметрам: код вида трудоустройства, характеристика выпускником работы (для трудоустроенных по специальности), характеристика выпускника (для трудоустроенных по специальности). Организовать отбор данных по виду трудоустройства. Коды вида трудоустройства: 0 — по специальности; 1 — не по специальности; 2 — продолжает образование; 3 — проходит службу по призыву; 5 — находится в предродовом (послеродовом) отпуске, 6 — не трудоустроен.
20. В фирме по организации праздников ведется учет заявок по схеме: номер заявки, код вида праздника, список ролей организаторов, список реквизита, дата, цена. Организовать отбор данных по табельному номеру продавца. Коды вида праздника: 0 — день рождения ребенка; 2 — день рождения взрослого; 2 — свадьба; 3 — юбилей.

**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 |
|---|---|---|
| **ПК-2. Обладает способностью разрабатывать компоненты аппаратно-программных комплексов и баз данных, используя современные инструментальные средства и технологии программирования.** | | |
| Знать | – основные элементы функциональной парадигмы: функция, суперпозиция функций, λ – исчисление, редукция, аппликативный порядок редукции, нормальный порядок редукции;<br>– связь понятий аппликативного и нормального порядков редукции и понятий энергичных и ленивых вычислений, разработанного в соответствии с указанными понятиями;<br>– связь между функциональной и объектно-ориентированной парадигмами программирования, методологию применения функциональной парадигмы программирования в разработке мультипарадиминенных программных систем | Список теоретических вопросов:<br>- функциональная парадигма программирования. Программа, как суперпозиция функций;<br>- λ-функции;<br>- редукция, редексы, виды редукции;<br>- теорема Черча-Россера;<br>- аппликативный и нормальный порядок редукции, энергичные и ленивые вычисления;<br>- S-выражения в языке LISP: атомы, точечные пары, списки;<br>- построение точечных пар и списков в языке LISP;<br>- функции CAR, CDR и их суперпозиции в языке LISP;<br>- числовые типы данных в языке LISP, арифметические функции;<br>- логические операции;<br>- строковые данные в языке LISP, действия над данными строкового типа;<br>- функции проверки типа данных в языке LISP, преобразование типа данных в языке LISP;<br>- функции quote и eval в языке LISP, их применение;<br>- функции setq и setf в языке LISP, различия между ними;<br>- создание функции в языке LISP, строка документации и ее вызов, дизассемблирование функции, возврат результата выполнения функции;<br>- формальные параметры функций в языке LISP, виды формальных параметров;<br>- создание и использование глобальных переменных в языке LISP, функции defvar и defparameter, создание и использование констант в языке LISP;<br>- создание и использование локальных переменных в языке LISP, функции let и let*;<br>- создание и использование анонимных функций в языке LISP;<br>- функции проверки условий в языке LISP;<br>- рекурсия в языке LISP; |

| Структурный элемент компетенции | Планируемые результаты обучения | Оценочные средства |
|---|---|---|
| | | - ассоциативные списки в языке LISP;<br>- Hash-таблицы в языке LISP;<br>- поиск по ключу в ассоциативном списке;<br>- поиск по ключу в Hash-таблице;<br>- функционалы в языке LISP, функционалы и анонимные функции;<br>- функции высших порядков в языке LISP;<br>- массивы в языке LISP;<br>- циклические структуры в языке LISP;<br>- файловый ввод/вывод в языке LISP;<br>- форматирование вывода в языке LISP;<br>- диалект Visual LISP. Специфика. |
| Уметь | – определять целесообразность применения функциональной парадигмы, строить суперпозиции функций;<br>– разрабатывать функциональными средствами рационально организованный программный продукт;<br>разрабатывать сложные программные системы, основанные на рационально основанной редукции суперпозиции функций | Список практических умений:<br>- сформулировать в заданной задаче систему функций и записать ее средствами $\lambda$-исчисления;<br>- определить наиболее приемлемый в конкретных условиях порядок редукции — аплик5ативный или нормальный;<br>- разработать программный код на одном из функциональных языков программирования или функциональном расширении языка, не являющегося функциональным;<br>- разработать при необходимости эффективное взаимодействие функциональных и нефункциональных компонентов разрабатываемого программного продукта. |
| Владеть | – навыками применения современных инструментальных средств разработки функциональных программ;<br>– навыками применения не менее двух существенно отличающихся функциональных языков программирования;<br>– применения современных функциональных средств в процессе проектирования, программирования, отладки и модернизации сложных программных систем. | Список навыков:<br>- навык осмысленного использования программных сред GNU Common LISP, Steel Bank Common LISP, Visual LISP;<br>- навык осмысленного использования программных сред, представляющих функциональные возможности для языков, не являющихся функциональными: Python (Anaconda, PyCharm) с пакетом NumPy, Matlab/Octave, Java Script, C# (не менее двух);<br>- навык эффективной отладки как чисто функционального программного кода, так и мультипарадигменного программного кода, содержащего функциональную компоненту. |

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

Промежуточная аттестация по дисциплине «Функциональное программирование» включает теоретические вопросы, позволяющие оценить уровень усвоения обучающимися знаний, и практические задания, выявляющие степень сформированности умений и владений, проводится в форме экзамена и в форме выполнения и защиты курсовой работы.

Экзамен по данной дисциплине проводится в устной форме по экзаменационным билетам, каждый из которых включает два теоретических вопроса и одно практическое задание.

**Показатели и критерии оценивания экзамена:**

– на оценку **«отлично»** (5 баллов) – обучающийся демонстрирует высокий уровень сформированности компетенций, всестороннее, систематическое и глубокое знание учебного материала, свободно выполняет практические задания, свободно оперирует знаниями, умениями, применяет их в ситуациях повышенной сложности.

– на оценку **«хорошо»** (4 балла) – обучающийся демонстрирует средний уровень сформированности компетенций: основные знания, умения освоены, но допускаются незначительные ошибки, неточности, затруднения при аналитических операциях, переносе знаний и умений на новые, нестандартные ситуации.

– на оценку **«удовлетворительно»** (3 балла) – обучающийся демонстрирует пороговый уровень сформированности компетенций: в ходе контрольных мероприятий допускаются ошибки, проявляется отсутствие отдельных знаний, умений, навыков, обучающийся испытывает значительные затруднения при оперировании знаниями и умениями при их переносе на новые ситуации.

– на оценку **«неудовлетворительно»** (2 балла) – обучающийся демонстрирует знания не более 20% теоретического материала, допускает существенные ошибки, не может показать интеллектуальные навыки решения простых задач.

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

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

**а) Основная литература:**

1. Салмина, Н.Ю. Функциональное программирование и интеллектуальные системы : учебное пособие / Н.Ю. Салмина ; Томский Государственный университет систем управления и радиоэлектроники (ТУСУР), Факультет дистанционного обучения. – Томск : ТУСУР, 2016. – 100 с. : ил. – Режим доступа: по подписке. – URL: https://biblioclub.ru/index.php?page=book&id=480936 (дата обращения: 30.10.2020). – Библиогр.: с. 97. – Текст : электронный.

**Дополнительная литература:**

1. *Зыков, С. В.* Программирование. Функциональный подход : учебник и практикум для вузов / С. В. Зыков. — Москва : Издательство Юрайт, 2020. — 164 с. — (Высшее образование). — ISBN 978-5-534-00844-9. — Текст : электронный // ЭБС Юрайт [сайт]. — URL: https://urait.ru/bcode/451972 (дата обращения: 30.10.2020).

**Методические указания:**

1. Зарецкий М.В. Методические указания для выполнения самостоятельных работ по курсу «Объектно-ориентированное программирование» для студентов специальности 220400 / М.В. Зарецкий, Ю.Б. Кухта - Магнитогорск. МГТУ, 2005- 17 с.

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

*Программное обеспечение*: лицензионное программное обеспечение: операционная система; офисные программы; математические пакет, статистические пакеты, установленные на каждом персональном компьютере вычислительного центра ФГБОУ ВПО «МГТУ».

Перечень лицензионного программного обеспечения по ссылке: http://sps.vuz.magtu.ru/Shared%20Documents/Forms/AllItems.aspx?RootFolder=%2FShared%20Documents%2F%D0%9F%D0%BE%D0%B4%D0%B3%D0%BE%D1%82%D0%BE%D0%B2%D0%BA%D0%B0%20%D0%BA%20%D0%B0%D0%BA%D0%BA%D1%80%D0%B5%D0%B4%D0%B8%D1%82%D0%B0%D1%86%D0%B8%D0%B8%202020%2F%D0%A1%D0%B0%D0%BC%D0%BE%D0%BE%D0%B1%D1%81%D0%BB%D0%B5%D0%B4%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5%202019%D0%B3%2F%D0%9B%D0%B8%D1%86%D0%B5%D0%BD%D0%B7%D0%B8%D0%BE%D0%BD%D0%BD%D0%BE%D0%B5%20%D0%9F%D0%9E&InitialTabId=Ribbon.Document&VisibilityContext=WSSTabPersistence
Официальные сайты промышленных предприятий и организаций: http://www.mmk.ru, http://www.creditural.ru, http://www.magtu.ru, http://www.gks.ru и т.п.; разработчиков программных продуктов: http://www.statsoft.ru, http://www.microsoft.com, http://www.ptc.com и т.п; сайты лабораторий компьютерной графики http://graphics.cs.msu.ru , http://cgm.graphicon.ru.

## 9 Материально-техническое обеспечение дисциплины (модуля)

Материально-техническое обеспечение дисциплины включает:

| Тип и название аудитории | Оснащение аудитории |
|---|---|
| Лекционная аудитория | Мультимедийные средства хранения, передачи и представления информации |
| Компьютерный класс | Персональные компьютеры с пакетом Office, выходом в Интернет и с доступом в электронную информационно-образовательную среду университета |
| Аудитории для самостоятельной работы: компьютерные классы; читальные залы библиотеки | Все классы УИТ и АСУ с персональными компьютерами, выходом в Интернет и с доступом в электронную информационно-образовательную среду университета |
| Аудиторий для групповых и индивидуальных консультаций, текущего контроля и промежуточной аттестации | Ауд. 282 и классы УИТ и АСУ |
| Помещения для самостоятельной работы обучающихся, оснащенных компьютерной техникой с возможностью подключения к сети «Интернет» и наличием доступа в электронную информационно-образовательную среду организации | Классы УИТ и АСУ |
| Помещения для хранения и профилактического обслуживания учебного оборудования | Центр информационных технологий – ауд. 379 |