**УТВЕРЖДЕНО**
решением Ученого совета факультета математики,
информационных и авиационных технологий
от «__18__» _____мая_____2021_г., протокол №__4/21_
Председатель _____/М.А.Волков
*(подпись, расшифровка подписи)*
«__18__» _____мая_____2021_г.

# РАБОЧАЯ ПРОГРАММА ДИСЦИПЛИНЫ

| | |
|---|---|
| Дисциплина | Модели данных и прикладные алгоритмы |
| Факультет | Математики, информационных и авиационных технологий |
| Кафедра | Информационные технологии |
| Курс | 1 |

Направление (специальность)_ 02.03.03 - «Математическое обеспечение и администрирование информационных систем».
*код направления (специальности), полное наименование*

Направленность (профиль/специализация)__Технология программирования_____
*полное наименование*

Форма обучения___очная_____
*очная, заочная, очно-заочная (указать только те, которые реализуются)*

Дата введения в учебный процесс УлГУ: «_01_»___сентября___2021__г.
Программа актуализирована на заседании кафедры: протокол №_____ от ____ 20_____г.
Программа актуализирована на заседании кафедры: протокол №_____ от ____ 20_____г.
Программа актуализирована на заседании кафедры: протокол №_____ от ____ 20_____г.

Сведения о разработчиках:

| ФИО | Кафедра | Должность,<br>ученая степень, звание |
|---|---|---|
| Жаркова Галина Алексеевна | Информационных технологий | Профессор, д.пед.н., доцент |

| | СОГЛАСОВАНО | СОГЛАСОВАНО |
|---|---|---|
| | Заведующий кафедрой информационных технологий, реализующей дисциплину | Заведующий выпускающей кафедрой информационных технологий |
| | /_____/__Волков М.А._/<br>*(подпись)* *(Ф.И.О.)*<br>«18 »__мая__20_21 г. | /_____/_Волков М.А._/<br>*(подпись)* *(Ф.И.О.)*<br>«18 »__мая__20_21 г. |

## 1. ЦЕЛИ И ЗАДАЧИ ОСВОЕНИЯ ДИСЦИПЛИНЫ

Целью преподавания дисциплины является:

- формирование у студентов знаний в области программирования и теории алгоритмов, являющихся основой математического обеспечения современных компьютерных и информационных технологий;
- получение представлений об основах объектно-ориентированного программирования и теории алгоритмов как базе для изучения специализированных курсов;
- приобретение представлений о новейших тенденциях развития технологий программирования.

**Задачи освоения дисциплины:**

- получить информацию об алгоритмах и структурах данных, используемых в программировании
- изучить объектно-ориентированный подход к программированию
- изучить работу с конечными автоматами Мура и Мили
- получить навыки практической работы по использованию структур данных: стеки, очереди, списки, деревья, графы, конечные автоматы.

## 2. МЕСТО ДИСЦИПЛИНЫ В СТРУКТУРЕ ОПОП

Курс входит в дисциплины по выбору Блока 1 Основной Профессиональной Образовательной Программы бакалавриата по направлению подготовки 02.03.03 – «Математическое обеспечение и администрирование информационных систем» профиль «Технология программирования» по очной форме обучения.

Для изучения этой дисциплины необходимы знания базовых возможностей языка программирования C++, основных моделей и алгоритмов обработки данных.

Дисциплина закладывает знания, необходимые для изучения всех основных курсов по программированию, а также других дисциплин вариативной части профессионального цикла этой ОПОП. При изучении данной дисциплины закладываются знания и формируются компетенции для изучения следующих дисциплин: высокоуровневые методы информатики и программирования (ПК-4: знать методы структурного и объектно-ориентированного программирования, уметь программировать стандартные алгоритмы обработки данных, иметь навыки работы с различными структурами данных), объектно-ориентированное программирования (ПК-4), методы программирования современных информационных систем (ПК-4), Программирование для Интернет (ПК-4).

## 3. ПЕРЕЧЕНЬ ПЛАНИРУЕМЫХ РЕЗУЛЬТАТОВ ОБУЧЕНИЯ ПО ДИСЦИПЛИНЕ (МОДУЛЮ), СООТНЕСЕННЫХ С ПЛАНИРУЕМЫМИ РЕЗУЛЬТАТАМИ ОСВОЕНИЯ ОСНОВНОЙ ПРОФЕССИОНАЛЬНОЙ ОБРАЗОВАТЕЛЬНОЙ ПРОГРАММЫ

| Код и наименование реализуемой компетенции | Перечень планируемых результатов обучения по дисциплине (модулю), соотнесенных с индикаторами достижения компетенций |
| --- | --- |
| ПК-4 – Способен использовать основные концептуальные | **Знать:** основные концептуальные положения функционального, логического, объектно-ориентированного и визуального направлений программирования; |

| | |
|---|---|
| положения функционального, логического, объектно-ориентированного и визуального направлений программирования, методы, способы и средства разработки программ в рамках этих направлений | **Уметь:** методы, способы и средства разработки программ в рамках этих направлений;<br>**Владеть:** приёмами и различными типами программирования для использования в профессиональной деятельности. |
| ПК-5 – Способен использовать современные методы разработки и реализации конкретных алгоритмов математических моделей на базе языков программирования и пакетов прикладных программ моделирования | **Знать:** приёмы программирования на С++, стандартные алгоритмы, основные методы и средства автоматизации проектирования, реализации и испытаний программных средств;<br>**Уметь:** строить алгоритмы заданной задачи и довести её до работоспособного состояния, проводить адаптацию готовых компонент ПО к решаемой задаче, проводить оценку качества программных продуктов;<br>**Владеть:** приёмами и алгоритмами решения различного класса задач, сопровождения программных продуктов, модернизации и администрирования информационных систем. |

## 4. ОБЩАЯ ТРУДОЕМКОСТЬ ДИСЦИПЛИНЫ

### 4.1. Объем дисциплины в зачетных единицах (всего): 6

### 4.2. Объем дисциплины по видам учебной работы (в часах): 216

| Вид учебной работы | Количество часов (форма обучения: очная) | | |
|---|---|---|---|
| | Всего по плану | в т.ч. по семестрам | |
| | | 1 | 2 |
| Контактная работа обучающихся с преподавателем в соответствии с УП | 80/80* | | 80/80* |
| Аудиторные занятия: | 80/80* | | 80/80* |
| лекции | 16/16* | | 16/16* |
| Семинары и практические занятия | 32/32* | | 32/32* |
| Лабораторные работы, практикумы | 32/32* | | 32/32* |
| Самостоятельная работа | 100 | | 100 |
| Форма текущего контроля знаний и контроля | Проверка лабораторных работ, | | Проверка лабораторных работ, |

| | | | |
|---|---|---|---|
| самостоятельной работы: тестирование, контр.работа, коллоквиум,рефераты др.(не менее 2 видов) | проверка заданий | | проверка заданий |
| Курсовая работа | - | | - |
| Виды промежуточной аттестации (экзамен, зачет) | 36<br>экзамен | | 36<br>экзамен |
| Всего часов по дисциплине | 216 | | 216 |

*Количество часов работы ППС с обучающимися в дистанционном формате с применением электронного обучения

В случае необходимости использования в учебном процессе частично/исключительно дистанционных образовательных технологий в таблице через слеш указывается количество часов работы ППС с обучающимися для проведения занятий в дистанционном формате с применением электронного обучения.

**4.3. Содержание дисциплины (модуля.) Распределение часов по темам и видам учебной работы:**

Форма обучения: очная

| Название разделов и тем | Всего | Виды учебных занятий | | | | | Форма текущего контроля знаний |
|---|---|---|---|---|---|---|---|
| | | Аудиторные занятия | | | Занятияв интерактивной форме | Самостоятельная работа | |
| | | Лекции | Практические занятия, семинары | Лабораторные работы,практикумы | | | |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | |
| Раздел 1. УКАЗАТЕЛИ. АДРЕСНАЯ АРИФМЕТИКА | | | | | | | |
| 1. Указатели. | 5 | 1 | 1 | | | 3 | Домашние задания. |
| 2. Динамическое выделение памяти. | 6 | | 2 | | | 4 | Домашние задания. |
| 3. Динамические одномерные массивы. | 7 | 1 | 2 | | | 4 | Домашние задания. |
| 4. Динамические двумерные массивы. | 6 | | 1 | 1 | 1 | 4 | Домашние задания. Лабораторные работы |
| Раздел 2. СТРУКТУРЫ ДАННЫХ | | | | | | | |
| 5. Списки. Основные | 11 | | 2 | 2 | 1 | 7 | Домашние задания. |

| | | | | | | | |
| --- | --- | --- | --- | --- | --- | --- | --- |
| понятия. Способы представления в компьютере. Создание структур. | | | | | | | Лабораторные работы |
| 6. Стеки. Основные понятия. Способы представления в компьютере. Создание структур. | 13 | 1 | 3 | 2 | 1 | 7 | Домашние задания. Лабораторные работы |
| 7. Графы. Основные понятия. Способы представления в компьютере. Создание структур. | 14 | 1 | 3 | 2 | 1 | 8 | Домашние задания. Лабораторные работы |
| 8. Деревья. Основные понятия. Способы представления в компьютере. Создание структур. | 15 | | 3 | 4 | 1 | 8 | Домашние задания. Лабораторные работы |
| Раздел 3. СТАНДАРТНАЯ БИБЛИОТЕКА ШАБЛОНОВ | | | | | | | |
| 9. Вектора. Алгоритмы обработки. | 10 | 1 | 1 | 2 | 1 | 6 | Домашние задания. Лабораторные работы |
| 10. Строки. Алгоритмы обработки. | 10 | | 1 | 2 | 1 | 7 | Домашние задания. Лабораторные работы |
| 11. Списки. Алгоритмы обработки. | 10 | | 1 | 2 | 1 | 7 | Домашние задания. Лабораторные работы |
| Раздел 4. АЛГОРИТМЫ ОБХОДА МАТРИЦ | | | | | | | |
| 12. Поиск в глубину и в | 7 | 1 | 1 | 2 | 1 | 3 | Лабораторная |

| | | | | | | | |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ширину. | | | | | | | работа. Домашние задания |
| 13. Алгоритм Дейкстра. | 4 | | 1 | | | 3 | Домашние задания |
| 14. Алгоритм Прима-Краскала. | 8 | 1 | 1 | 2 | 1 | 4 | Домашние задания Лабораторные работы. |
| Раздел 5. КОНЕЧНЫЕ АВТОМАТЫ | | | | | | | |
| 15. Формальные системы. Исчисление предикатов. Метатеория логических исчислений. Абстрактные формальные системы. Формальные грамматики. Семантика формальных языков. | 5 | 1 | 1 | | | 3 | Домашние задания. |
| 16. Основные понятия конечных автоматов, операции. Распознаваемость множеств автоматами. Сети из автоматов, их анализ и синтез. Автономные автоматы. Неавтономные автоматы. Автоматы с переменной структурой. | 5 | 1 | 1 | | | 3 | Домашние задания. |
| 17. Вероятностные автоматы. | 10 | 1 | 1 | 4 | 1 | 4 | Домашние задания. Лаборато |

| | | | | | | | |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Программная реализация конечных автоматов. | | | | | | | рные работы |
| Раздел 6. ОСНОВЫ ОБЪЕКТНО-ОРИЕНТИРОВАННОГО ПРОГРАММИРОВАНИЯ | | | | | | | |
| 18. Основные понятия объектно-ориентированного программирования. Конструктор и деструктор, конструктор копирования. | 4 | 1 | 1 | | | 2 | Домашние задания. |
| 19. Наследование классов. Множественное наследование. | 6 | 1 | 1 | 2 | 1 | 2 | Домашние задания. Лабораторные работы |
| 20. Доступ к объектам иерархии. Виртуальные методы. | 5 | 1 | 1 | 1 | 1 | 2 | Домашние задания. Лабораторные работы |
| 21. Абстрактные классы. | 6 | 1 | 1 | 1 | 1 | 3 | Домашние задания. Лабораторные работы |
| 22. Перегрузка операторов. | 6 | 1 | 1 | 1 | 1 | 3 | Домашние задания. Лабораторные работы |
| 23. Шаблоны классов | 7 | 1 | 1 | 2 | 1 | 3 | Домашние задания. Лабораторные работы |
| Экзамен | 36 | | | | | | |
| Итого: | 216 | 16 | 32 | 32 | 16 | 100 | |

## 5. СОДЕРЖАНИЕ ДИСЦИЛИНЫ (МОДУЛЯ)

**Раздел 1. УКАЗАТЕЛИ. АДРЕСНАЯ АРИФМЕТИКА**

**Тема 1.** Указатели.

**Тема 2.** Динамическое выделение памяти.

**Тема 3.** Динамические одномерные массивы.
**Тема 4.** Динамические двумерные массивы.
**Раздел 2. СТРУКТУРЫ ДАННЫХ**
**Тема 5.** Списки. Основные понятия. Способы представления в компьютере. Создание структур.
**Тема 6.** Стеки. Основные понятия. Способы представления в компьютере. Создание структур.
**Тема 7.** Графы. Основные понятия. Способы представления в компьютере. Создание структур.
**Тема 8.** Деревья. Основные понятия. Способы представления в компьютере. Создание структур.
**Раздел 3. СТАНДАРТНАЯ БИБЛИОТЕКА ШАБЛОНОВ**
**Тема 9.** Вектора. Алгоритмы обработки.
**Тема 10.** Строки. Алгоритмы обработки.
**Тема 11.** Списки. Алгоритмы обработки.
**Раздел 4. АЛГОРИТМЫ ОБХОДА МАТРИЦ**
**Тема 12.** Поиск в глубину и в ширину.
**Тема 13.** Алгоритм Дейкстры.
**Тема 14.** Алгоритм Прима-Краскала.
**Раздел 5. КОНЕЧНЫЕ АВТОМАТЫ**
**Тема 15.** Формальные системы. Исчисление предикатов. Метатеория логических исчислений. Абстрактные формальные системы. Формальные грамматики. Семантика формальных языков.
**Тема 16.** Основные понятия конечных автоматов, операции. Распознаваемость множеств автоматами. Сети из автоматов, их анализ и синтез. Автономные автоматы. Неавтономные автоматы. Автоматы с переменной структурой.
**Тема 17.** Вероятностные автоматы. Программная реализация конечных автоматов.
**Раздел 6. ОСНОВЫ ОБЪЕКТНО-ОРИЕНТИРОВАННОГО ПРОГРАММИРОВАНИЯ**
**Тема 18.** Основные понятия объектно-ориентированного программирования. Конструктор и деструктор, конструктор копирования.
**Тема 19.** Наследование классов. Множественное наследование.
**Тема 20.** Доступ к объектам иерархии. Виртуальные методы.
**Тема 21.** Абстрактные классы.
**Тема 22.** Перегрузка операторов.
**Тема 23.** Шаблоны классов

## 6. ТЕМЫ ПРАКТИЧЕСКИХ И СЕМИНАРСКИХ ЗАНЯТИЙ

**Раздел 1. УКАЗАТЕЛИ. АДРЕСНАЯ АРИФМЕТИКА**
**Тема 1.** Указатели.
**Тема 2.** Динамическое выделение памяти.
**Тема 3.** Динамические одномерные массивы.
**Тема 4.** Динамические двумерные массивы.
**Раздел 2. СТРУКТУРЫ ДАННЫХ**
**Тема 5.** Списки. Основные понятия. Способы представления в компьютере. Создание структур.
**Тема 6.** Стеки. Основные понятия. Способы представления в компьютере.

Создание структур.
**Тема 7.** Графы. Основные понятия. Способы представления в компьютере. Создание структур.
**Тема 8.** Деревья. Основные понятия. Способы представления в компьютере. Создание структур.
**Раздел 3. СТАНДАРТНАЯ БИБЛИОТЕКА ШАБЛОНОВ**
**Тема 9.** Вектора. Алгоритмы обработки.
**Тема 10.** Строки. Алгоритмы обработки.
**Тема 11.** Списки. Алгоритмы обработки.
**Раздел 4. АЛГОРИТМЫ ОБХОДА МАТРИЦ**
**Тема 12.** Поиск в глубину и в ширину.
**Тема 13.** Алгоритм Дейкстры.
**Тема 14.** Алгоритм Прима-Краскала.
**Раздел 5. КОНЕЧНЫЕ АВТОМАТЫ**
**Тема 15.** Формальные системы. Исчисление предикатов. Метатеория логических исчислений. Абстрактные формальные системы. Формальные грамматики. Семантика формальных языков.
**Тема 16.** Основные понятия конечных автоматов, операции. Распознаваемость множеств автоматами. Сети из автоматов, их анализ и синтез. Автономные автоматы. Неавтономные автоматы. Автоматы с переменной структурой.
**Тема 17.** Вероятностные автоматы. Программная реализация конечных автоматов.
**Раздел 6. ОСНОВЫ ОБЪЕКТНО-ОРИЕНТИРОВАННОГО ПРОГРАММИРОВАНИЯ**
**Тема 18.** Основные понятия объектно-ориентированного программирования. Конструктор и деструктор, конструктор копирования.
**Тема 19.** Наследование классов. Множественное наследование.
**Тема 20.** Доступ к объектам иерархии. Виртуальные методы.
**Тема 21.** Абстрактные классы.
**Тема 22.** Перегрузка операторов.
**Тема 23.** Шаблоны классов

## 7. ЛАБОРАТОРНЫЕ РАБОТЫ, ПРАКТИКУМЫ

**Лабораторная работа 1. Работа с одномерными и двумерными массивами**

**Цель работы:** освоить:

- приемы обработки двумерных и одномерных массивов
- навыки работы с одномерными и двумерными массивами.

**Указания к выполнению работы.**

Массивы содержат n или nxm целых чисел. Необходимо создать динамический одномерный или двумерный массив. Ввод чисел осуществляется с консоли. Вывести числа после ввода, а затем после обработки. Вывод одномерного массива должен осуществляться в трочку, а вывод двумерного массива в виде таблицы.

1. Элементы массива M(n) упорядочены по неубыванию. Для заданного x найти наименьшее k такое, что $m_k \le x \le m_{k+1}$, либо показать (выдать сообщение), что

такового нет. Для поиска полезно применить метод дихотомии (метод деления отрезка пополам).

2. В каждой строке матрицы A(n,n) найти наибольший элемент и поменять его местами с соответствующим диагональным элементом.
3. Последовательность $a_1, a_2, ..., a_k$, называется пилообразной, если $a_1 < a_2 > a_3 < a_4 > ... > a_k$ либо $a_1 > a_2 < a_3 > a_4 < ... < a_k$. В массиве A(m) найти самую длинную пилообразную последовательность.
4. Последовательность $a_1, a_2, ..., a_k$, называется монотонной, если $a_1 \le a_2 \le ... \le a_k$ либо $a_1 \ge a_2 \ge ... \ge a_k$. В массиве A(m) найти самую длинную монотонную последовательность.
5. Утверждается, что массив A(m) целиком (как последовательность) встречается в массиве B(n), n>m. Найти место массива A в массиве B или показать, что его в массиве B нет.
6. Найти все числа из массива B(n), встречающиеся более чем в одной строке матрицы A(m,n).
7. В массиве Z(n) найти наиболее длинную цепочку стоящих подряд попарно различных элементов.
8. В массиве P(n) найти самую длинную последовательность, которая является арифметической или геометрической прогрессией.
9. В массиве A(2n+1), не содержащем одинаковых элементов, найти средний по величине элемент, т.е. такой, что в массиве A ровно n элементов меньше его и столько же элементов больше его. Массив A сохранить (не сортировать), дополнительных массивов не использовать.
10. В массиве H(n) хранятся значения высот некоторого профиля местности (ее вертикального сечения) с постоянным шагом по горизонтали. Найти области (номера точек измерения высоты), невидимые для наблюдателя, находящегося в точке h.
11. Черный квадрат. В матрице A(m,n), состоящей из нулей и единиц, найти квадрат заданного размера (квадратную подматрицу), состоящий целиком из нулей.
12. Матрицу M(m,n) заполнить натуральными числами от 1 до $m \cdot n$ по спирали, начинающейся в левом верхнем углу и закрученной по часовой стрелке.
13. Матрицу K(m,n) заполнить следующим образом. Элементам, находящимся на периферии (по периметру матрицы), присвоить значение 1; периметру оставшейся подматрицы – значение 2 и так далее до заполнения всей матрицы.
14. Поворот матрицы. Сдвинуть элементы заданной матрицы в пределах периметра каждых вложенных подматриц на одну позицию по часовой стрелке.
15. В каждом столбце и каждой строке матрицы P(n,n) содержится строго по одному нулевому элементу. Перестановкой строк добиться расположения всех нулей по главной диагонали матрицы.
16. Касса. В массиве K(n) в порядке убывания представлены достоинства денежных знаков (купюр и монет) валютной системы некоторой страны. Реализовать выдачу в этой системе заданной суммы m минимальным числом денежных знаков.
17. Колокол. В массиве A(n) наименьший элемент поместить на первое место, наименьший из оставшихся – на последнее место, следующий по величине – на второе место, следующий – на предпоследнее и так далее – до середины массива.
18. С внешнего устройства (с клавиатуры, из файла) вводятся последовательно числа, количество которых велико и заранее неизвестно. Требуется сохранять и в

процессе ввода каждого числа распечатывать не более m последних введенных чисел (в порядке их поступления).

19. Магический квадрат. Магическим квадратам порядка n называется квадратная таблица размером n x n, состоящая из чисел $1,2,...n^2$ так, что суммы по каждому столбцу, каждой строке и каждой из двух диагоналей равны между собой. Проверить, является ли заданная целочисленная квадратная матрица магическим квадратом.
20. В трехмерном массиве K(l,m,n), состоящем из нулей и единиц, храниться сеточное изображение некоторого трехмерного тела. Получить в двумерных массивах три проекции (тени) этого тела.
21. Автостоп. Из пункта A в пункт B, между которыми s км, выехал велосипедист с постоянной скоростью $\upsilon_0$ км/ч. Одновременно с ним в том де направлении другой путник решил добраться “автостопом” – на разных видах попутного транспорта. Перед каждым участком пути он $\tau_i$ минут “голосует”, затем движется $t_i$ часов со скоростью $\upsilon_i$ км/ч (величины $\tau_i, t_i, \upsilon_i, i = 1,..., n_i$ заданы в соответствующих массивах). В каких точках пути (в какие моменты времени) путники смогут помахать друг другу рукой?

**Лабораторная работа 2. Обработка файлов.**

**Цель работы.** Освоить работу с бинарными файлами, получить навыки обработки двоичных файлов и работы со структурами.

**Указания к работе:** Двоичный файл – файл, состоящий из записей фиксированного размера. Каждая запись в программе является структурой.

Разработать программу, позволяющую осуществлять в диалоговом режиме следующие основные действия:

- добавление в конец файла новых записей (файл либо создается вновь, либо открывается существующий; название файла и его местоположение задается в диалоговом режиме);
- просмотр существующих записей в исходном или результирующем файле (предусмотреть возможность прерывания просмотра файла);
- предусмотреть защиту от «дурака»;
- обработка данных входного типизированного файла (открывается существующий; название файла и его местоположение задается в диалоговом режиме) с выводом результатов в новый или существующий файл (результирующий файл либо создается вновь, либо открывается существующий в режиме добавления записей; название файла и его местоположение задается в диалоговом режиме);
- выход из программы.

**Обязательное требование к программе**:

- Программа должна обнаруживать и сообщать об ошибках, связанных с выполнением файловых операций. Например: ввод имени файла данных, отсутствующего в текущем каталоге, или просмотр входного файла, в который еще не введены данные и т. д.

**Примечание**. Перед сдачей лабораторной работы **обязательно подготовить** несколько примеров задания во входном файле.

**Варианты задания:**

**1.Построение индекса**

```
struct People
```

```
{     // структура входного файла
      int _id;          // уникальный идентификатор записи о человеке
      char _FIO[100];   // Фамилия, имя, отчество человека
};
```

Известно, что входной файл содержит уникальные _id для каждой записи. Необходимо создать индексный файл (по _id):

```
struct IndexPeople
{     // структура индексного (выходного) файла
      int _id;                // идентификатор записи о человеке входного
файла
      int record_position;    // номер позиции записи во входном файле
};
```

_id в индексном файле должны быть отсортированы по возрастанию.

Кроме того, необходимо реализовать функцию быстрого поиска соответствующей записи входного файла по созданному индексу (т.е. сначала осуществляется поиск нужного _id в индексном файле методом деления пополам, затем осуществляется позиционирование и считывание нужной записи входного файла).

### 2.Построение зависимой справочной таблицы

```
struct People
{     // структура входного файла
      int _id;                // уникальный идентификатор записи о человеке
      int _StateId;           // ключ - статус человека
      char _FIO[100];         // Фамилия, имя, отчество человека
};
```

Известно, что входной файл содержит уникальные _id и повторяющиеся _StateId. Необходимо создать выходной файл:

```
struct PeopleState
{     // структура выходного файла
      int _StateId;           // ключ - статус человека
      char StateName[20];     // наименование статуса
};
```

Сформировать выходной файл _StateId, в котором будут уникальные для каждой записи _StateId (упорядочивать не обязательно), поле StateName – предлагать вводить с клавиатуры пользователю.

### 3.Проверка целостности данных таблиц

```
struct People
{     // структура входного файла 1
      int _id;                // уникальный идентификатор записи о человеке
      int _StateId;           // ключ - статус человека
      char _FIO[100];         // Фамилия, имя, отчество человека
};
struct PeopleState
{     // структура входного файла 2
      int _StateId;           // ключ - статус человека
      char StateName[20];     // наименование статуса
};
```

Известно, что первый входной файл содержит уникальные _id и повторяющиеся _StateId. Второй входной файл содержит уникальные _StateId. Кроме того, может возникать ситуация, такая, что не для всех _StateId в записях первого файла имеются соответствующие записи во втором файле.

Необходимо создать файл:

```
struct People
{     // структура выходного файла
      int _id;                // уникальный идентификатор записи о человеке
```

```
      char StateName[20];      // наименование статуса
      char _FIO[100];          // Фамилия, имя, отчество человека
};
```

для отсутствующих кодов _StateId выдавать предупреждение на экран и оставлять StateName пустым.

### 4.Проверка целостности первичного ключа

```
struct People
{     // структура входного файла
      int _id;                 // уникальный идентификатор записи о человеке
      char _FIO[100];          // Фамилия, имя, отчество человека
};
```

Необходимо проверить файл на уникальность по полю _id. Если существуют две и более записей с одинаковыми _id – необходимо вывести соответствующее предупреждение на экран и предложить пользователю оставить одну из них. На основе данной обработки создать выходной файл той же структуры, но удовлетворяющий условию уникальности записей по _id.

### 5.Управление хранением записей в таблице

```
struct People
{     // структура входного файла
      int _id;                 // уникальный идентификатор записи о человеке
      int _StateId;            // ключ - статус человека
      char _FIO[100];          // Фамилия, имя, отчество человека
      char _IsDeleted;         // 0 - запись актуальна, 1 - запись удалена
};
```

Известно, что входной файл содержит уникальные _id. Необходимо реализовать операции добавления/изменения/удаления записей. При удалении запись физически остается в файле, изменяется лишь поле _IsDeleted. При просмотре показываются только «не удаленные» записи. Нужно реализовать возможность просмотра только «удаленных» записей. В выходной файл той же структуры вывести все «не удаленные» записи.

### 6.Построение «кластерного» индекса

```
struct People
{     // структура входного файла
      int _id;                 // уникальный идентификатор записи о человеке
      int _StateId;            // ключ - статус человека
      char _FIO[100];          // Фамилия, имя, отчество человека
};
```

Известно, что первый входной файл содержит уникальные _id и повторяющиеся _StateId. Необходимо создать индекс по полю _StateId, состоящий из следующих двух выходных файлов. В первом выходном файле записи должны быть отсортированы по _StateId. При обновлении входного файла – выходной переформируется.

```
struct SimpleClusteredIndexMaster
{     // структура первого выходного файла
      int _StateId;                    // ключ - статус человека
      int _StartPositionIndex;         // начальная позиция хранения
индексов _id
      int _Length;                     // количество записей
};
struct SimpleClusteredIndexSlave
{
      int _id;                         // уникальный идентификатор записи о
человеке
};
```

Кроме того, необходимо реализовать функцию быстрого построения выборки записей по заданному _StateId (т.е. сначала осуществляется поиск нужного _StateId в индексном (первом) файле методом деления пополам, затем осуществляется позиционирование и считывание нужных записей входного файла).

### 7.Страничная организация хранения записей

```
const int c_StdCount = 10;    // Стандартное количество записей на странице
struct DataPage
{     // структура входного файла
      int _id;          // номер страницы
      int _Count;       // количество записей на странице (всего)
      int _CountFree;   // количество свободных записей на странице
      struct People
      {     // информационная запись о человеке
            int _id;          // уникальный идентификатор записи о человеке
            int _StateId;     // ключ - статус человека
            char _FIO[100];   // Фамилия, имя, отчество человека
            char _IsDeleted;  // 0 - запись актуальна, 1 - удалена, 2 -
свободна
      } peoples[c_StdCount];
};
```

Входной файл содержит записи о людях, организованные в страницы. Все операции ввода/вывода осуществляются постранично. В выходном файле того же типа необходимо обеспечить постраничную сортировку записей по _id и очистку удаленных записей (т.е. записи должны быть отсортированы по ключу в пределах страницы, свободные записи хранятся в конце страницы). Организовать буферизованный поиск записи по полям _id, _FIO (т.е. поиск сначала осуществляется в памяти, методом деления пополам, а затем, по необходимости, поднимаются остальные страницы).

### 8.Индекс с постраничным хранением

```
struct People
{     // структура входного файла
      int _id;                // уникальный идентификатор записи о человеке
      int _StateId;           // ключ - статус человека
      char _FIO[100];         // Фамилия, имя, отчество человека
      char _IsDeleted;        // 0 - запись актуальна, 1 - удалена
};
```

Известно, что входной файл содержит уникальные _id для каждой записи. Необходимо создать индексный файл, обеспечив постраничную сортировку записей по _id (т.е. записи должны быть отсортированы по ключу в пределах страницы, свободные записи хранятся в конце страницы). Помимо этого, при добавлении/удалении/обновлении записей во входном файле, необходимо обновлять индексный файл, сохраняя свойство постраничной сортировки. Все операции ввода/вывода осуществляются постранично.

```
const int c_StdCountIndex = 20;
struct IndexPage
{     // структура индексного (выходного) файла
      int _id;          // номер страницы
      int _Count;       // количество записей на странице (всего)
      int _CountFree;   // количество свободных записей на странице
      struct IndexPeople
      {
            int _id;                // идентификатор записи о человеке
входного файла
            int record_position;    // номер позиции записи во входном файле
            char _IsDeleted;        // 0 - запись актуальна, 1 – удалена, 2 -
свободна
```

```
      } indexes[c_StdCountIndex];
};
```

Кроме того, необходимо реализовать функцию быстрого поиска соответствующей записи входного файла по созданному индексу (т.е. сначала осуществляется поиск нужного _id в индексном файле методом деления пополам, затем осуществляется позиционирование и считывание нужной записи входного файла).

### 9.Кластерный индекс с постраничным хранением

```
struct People
{     // структура входного файла
      int _id;                // уникальный идентификатор записи о человеке
      int _StateId;           // ключ - статус человека
      char _FIO[100];         // Фамилия, имя, отчество человека
      char _IsDeleted;        // 0 - запись актуальна, 1 - удалена
};
```

Известно, что входной файл содержит уникальные _id для каждой записи. Необходимо создать индексный файл (по полю _StateId), обеспечив постраничную сортировку записей по _StateId (т.е. записи должны быть отсортированы по ключу в пределах страницы, свободные записи хранятся в конце страницы). Помимо этого, при добавлении/удалении/обновлении записей во входном файле, необходимо обновлять индексный файл, сохраняя свойство постраничной сортировки. Все операции ввода/вывода осуществляются постранично.

```
const int c_StdCountIndex = 20;
struct IndexPage
{     // структура индексного (выходного) файла
      int _id;          // номер страницы
      int _Count;       // количество записей на странице (всего)
      int _CountFree;   // количество свободных записей на странице
      struct IndexPeople
      {
            int _StateId;           // идентификатор статуса человека во
входном файле
            int record_position;    // номер позиции записи во входном файле
            char _IsDeleted;        // 0 - запись актуальна, 1 - удалена, 2 -
свободна
      } indexes[c_StdCountIndex];
};
```

Кроме того, необходимо реализовать функцию быстрого построения выборки записей по заданному _StateId (т.е. сначала осуществляется поиск нужного _StateId в индексном (первом) файле методом деления пополам, затем осуществляется позиционирование и считывание нужных записей входного файла).

### 10. Вычисление факториалов длинных чисел

```
struct FactInput
{
      int _id;          // Номер записи в файле
      char Number[3];   // Число в десятичном представлении (т.е. коды
символов '0'..'9')
};
```

Во входном файле содержатся коды символов ‘0’…’9’ в десятичном представлении (в виде С-строки). Необходимо вычислить для каждого числа его факториал и в том же виде записать в выходной файл.

```
struct FactOutput
{
      int _id;          // Номер записи в файле
      char Number[200]; // Факториал числа в десятичном представлении
```

```
};
```

### 11. Сумма длинных целых чисел

```
struct LongNumbers
{
      int _id;                // Номер записи в файле
      char Number1[100];      // Число 1 в десятичном представлении
      char Number2[100];      // Число 2 в десятичном представлении
      char Number3[100];      // Число 3 в десятичном представлении
};
```

Во входном файле содержатся коды символов ‘0’…’9’ в десятичном представлении (в виде С-строки). Необходимо вычислить выражение Number1+Number2+Number3 и в том же виде записать в выходной файл.

```
struct LongNumberResult
{
      int _id;          // Номер записи в файле
      char Number[200]; // Число в десятичном представлении
};
```

### 12. Вычитание длинных целых чисел

```
struct LongNumbers
{
      int _id;                // Номер записи в файле
      char Number1[100];      // Число 1 в десятичном представлении
      char Number2[100];      // Число 2 в десятичном представлении
      char Number3[100];      // Число 3 в десятичном представлении
};
```

Во входном файле содержатся коды символов ‘0’…’9’ в десятичном представлении (в виде С-строки). Необходимо вычислить выражение Number1-Number2-Number3 и в том же виде записать в выходной файл.

```
struct LongNumberResult
{
      int _id;          // Номер записи в файле
      char Number[200]; // Число в десятичном представлении
};
```

### 13. Деление длинных целых чисел

```
struct LongNumbers
{
      int _id;                // Номер записи в файле
      char Number1[100];      // Число 1 в десятичном представлении
      char Number2[100];      // Число 2 в десятичном представлении
};
```

Во входном файле содержатся коды символов ‘0’…’9’ в десятичном представлении (в виде С-строки). Необходимо вычислить выражение Number1/Number2 и в том же виде записать в выходной файл, десятичная точка – «.».

```
struct LongNumberResult
{
      int _id;          // Номер записи в файле
      char Number[200]; // Число в десятичном представлении
};
```

### 14. Умножение длинных целых чисел

```
struct LongNumbers
{
      int _id;                // Номер записи в файле
      char Number1[100];      // Число 1 в десятичном представлении
```

```
	char Number2[100];	// Число 2 в десятичном представлении
};
```

Во входном файле содержатся коды символов ‘0’…’9’ в десятичном представлении (в виде С-строки). Необходимо вычислить выражение Number1*Number2 и в том же виде записать в выходной файл.

```
struct LongNumberResult
{
	int _id;		// Номер записи в файле
	char Number[200]; // Число в десятичном представлении
};
```

### 15. Поиск корней многочленов

```
struct Equation
{
	int _id;		// Номер записи в файле
	int a[10];		// Коэффициенты многочлена
	double Left;		// Левая граница отрезка на оси абсцисс
	double Right;		// Правая граница отрезка на оси абсцисс
	double Epsilon;	// Точность вычислений
};
```

Во входном файле содержится информация о многочлене. Известно, что на заданном отрезке функция, заданная многочленом имеет один корень. Определить этот корень с точностью до Epsilon методом деления пополам и найти количество итераций («с точностью до Epsilon» означает необходимость вычисления корня многочлена до тех пор, пока результат многочлена не окажется меньшим или равным Epsilon). Результаты записать в выходной файл:

```
struct EquationRoot
{
	int _id;			// Номер записи в файле
	char root[30];		// Значение корня в виде строки
	int CalcIterations;	// Количество итераций, необходимых для
вычисления корня
};
```

### 16. Линейные операции над матрицами

```
struct InputMatrixes
{
	int _id;					// Номер записи в файле
	int m,n,k;					// Размерность матриц (< 10)
	double A[10][10], B[10][10], C[10][10];	// Матрицы A,B,C
	double V[10];				// Вектор V
	double S;					// Коэффициент
};
```

Во входном файле содержится информация о матрицах: A[m,n], B[m,k], C[n,k], векторе V[k,1] и некотором коэффициенте S. Размерность – [строк, столбцов]. Вычислить и записать в файл, посчитав количество операций сложения и умножения: $(A * C) * S * B^T * V$.

```
struct OutputMatrix
{
	int _id;			// Номер записи в файле
	int m,n;			// Размерность матриц (< 10)
	double A[10][10];		// Матрица
	int CalcOperations;	// Количество операций умножения и сложения
};
```

### 17. Перевод числительных в словесную форму

```
struct Product
{
	int _id;		// Номер записи в файле
	char Name[200];		// Наименование товара
	double RetailPrice;	// Розничная цена товара
};
```

Во входном файле содержится информация о товарах. Необходимо сформировать выходной файл, в котором сформировать цену «прописью» (целая часть числа) и цифрами вывести две первых цифры (с округлением) после запятой. Пример: 91234.356 (девяносто одна тысяча двести тридцать четыре руб. 36 коп.)

```
struct ProductPriceInfo
{
	int _id;			// Номер записи в файле
	char Name[200];			// Наименование товара
	double RetailPrice;		// Розничная цена товара
	char sRetailPrice[300];		// Розничная цена товара в письменном виде
};
```

### 18. Перевод чисел в другую систему исчисления

```
struct NumberInfo
{
	int _id;			// Номер записи в файле
	long Number;			// Число в десятичной системе исчисления
	int Foundation;			// Основание новой системы исчисления
};
```

Во входном файле содержится информация о числах. Необходимо сформировать выходной файл со строковым представлением тех же чисел, но в указанной системе исчисления. Дополнительные цифры отображать буквами ‘A’, ‘B’, ‘C’, ‘D’ и т. д.

```
struct NumberInfoResult
{
	int _id;		// Номер записи в файле
	char Number[100];	// Строковое представление числа в новой системе
исчисления
};
```

### 19. Решение системы линейных уравнений методом Гаусса

```
struct EquationSystem
{	// структура входного файла
	int _id;		// Номер записи в файле
	double Matrix[6][6];	// Коэффициенты
	double Vector[6];	// Свободные члены
	int EquationCount;	// Количество уравнений
};
```

Во входном файле в каждой записи определена система из `EquationCount` уравнений с `EquationCount` неизвестными. Известно, что определитель матрицы не равен нулю и элемент $Matrix[0][0]$ не равен нулю. Найти решение каждой системы методом Гаусса и поместить их в выходной файл.

```
struct EquationSystemRoot
{	// структура выходного файла
	int _id;		// Номер записи в файле
	double Root[6];		// Корни уравнения
	int EquationCount;	// Количество корней уравнения
};
```

### 20. Вычисление формулы

```
struct Expression
```

```
{
	int _id;		// Номер записи в файле
	char Number1[100];	// Формула (С-строка)
	double x,y,z;		// переменные x,y,z
};
```

Дана безошибочная запись формулы, записанной в виде С-строки. Формула заканчивается знаком «=» и задана грамматикой:

<формула> :=: <терм> | <терм> + <терм> | <терм> - <терм>
<терм> :=: <идентификатор> | (<формула>) | [<формула>] | <формула>
<идентификатор> :=: x | y | z

Результаты вычислений записать в выходной файл.

```
struct Result
{
	int _id;		// Номер записи в файле
	double res;		// Результат вычисления формулы
};
```

**21. Группировка информации**

```
struct People
{	// структура входного файла
	int _id;		// уникальный идентификатор записи о человеке
	int _StateId;		// ключ - статус человека
	int _GroupId;		// Группа
	char _FIO[100];		// Фамилия, имя, отчество человека
	double _Salary;		// Оклад
	char _IsDeleted;  // 0 - запись актуальна, 1 - запись удалена
};
```

Известно, что входной файл содержит уникальные _id и повторяющиеся _StateId, _GroupId. Необходимо создать выходной файл:

```
struct Group
{	// структура выходного файла
	int _GroupId;		// Группа
	double _Salary;		// Средний Оклад
};
```

Сформировать выходной файл _GroupId, в котором будут уникальные для каждой записи _GroupId (упорядочивать не обязательно), поле _Salary – среднее значений _Salary в соответствующей группе исходного файла.

**Лабораторная работа 3. Работа с конечными автоматами.**

**Цель работы.** Освоить обработку символьных последовательностей с использованием механизма конечных автоматов.

**Указания к работе.** Для каждого задания необходимо создать нагруженный граф Мура, получить таблицы входов, переходов состояний и реализовать в программе конечный автомат. В программе предусмотреть исправление ошибок в символьной строке.

**Варианты заданий.**

| | |
| --- | --- |
| 1. | Защищенный ввод десятичных чисел.<br>Дана строка, содержащая несколько целых или дробных десятичных чисел, возможно со знаком «-» (знак «+» не допускается), и, если надо, десятичной запятой. Пример правильной строки: 32 -45; 0,14 -99,01<br>Перед запятой и после нее должна быть хотя бы одна цифра. Числа |

| | |
|---|---|
| | разделяются одним или несколькими пробелами или одной «;».<br>По концу строки напечатать количество введенных чисел и их среднее значение. Ошибочно набранные числа дать возможность исправлять на месте. |
| 2. | Перевод конечных или бесконечных периодических десятичных дробей в рациональные дроби вида *m/n.*<br>Дана строка, содержащая выражения вида: 127= 3,14= 0,(3)= 1,23(100)=.<br>Перед запятой должна быть хотя бы одна цифра. Десятичная точка не допускается. Период (конечный набор десятичных цифр) указывается в круглых скобках. Числа считаются положительными (без знака). Число заканчивается обязательным знаком «=», после которого может быть произвольное число пробелов.<br>По концу строки каждое число напечатать как рациональную дробь (то есть отдельно числитель и знаменатель). Ошибочно набранные числа дать возможность исправлять на месте. |
| 3. | Перевод углов из градусной меры в радианную.<br>Дана строка, содержащая одно или несколько выражений (чисел в градусной форме) вида: 12^35’19’’= (12 градусов, 35 минут, 19 секунд).<br>Секунды могут отсутствовать, а если их нет, то могут отсутствовать и минуты. Секунды и минуты могут быть только в пределах от 0 до 59. Числа могут разделяться произвольным числом пробелов. Пробелы внутри чисел не допускаются.<br>По концу строки каждое число напечатать в радианной мере. При вычислениях использовать приближенное значение числа  хотя бы с шестью верными значащими цифрами. Ошибочно набранные выражения дать возможность исправлять на месте. |
| 4. | Арифметика 16-х чисел.<br>Дана строка, содержащая числа без знака в 16-тиричном виде, например: 9h+15Bh - D3F0h.<br>Между числами должны стоять знаки арифметических операций «+» или «-», окруженные произвольным числом пробелов. Пробелы внутри чисел не допускаются.<br>По концу строки напечатать результат указанных действий также в 16-тиричном виде. Ошибочно набранные выражения дать возможность исправлять на месте. |
| 5. | Арифметика в различных системах счисления.<br>Дана строка, содержащая выражения вида: 62(8) - 921(10) + 1001(2).<br>Числа записаны в системах счисления с разными основаниями (основания указываются в скобках и все основания не превосходят 10). Цифры только десятичные. Внутри чисел пробелы не допускаются. Все цифры числа должны быть меньше основания. Между числами должны стоять знаки арифметических операций «+» или «-», окруженные произвольным числом пробелов.<br>По концу строки напечатать результат этих действий в десятичном виде. Ошибочно набранные числа дать возможность исправлять на месте. |
| 6. | Заполнение массива целых чисел.<br>Пусть в программе описан массив X из десяти элементов (от 0 до 9-го). Дана строка, содержащая команды вида: X[3]=54 X[0] = -8.<br>Команды идут в произвольном порядке, могут вообще отсутствовать, заканчиваются по концу строки. Внутри команды пробелы возможны только вокруг знака «=», команды разделяются произвольным числом пробелов. Числа |

| | |
|---|---|
| | допускаются только целые, возможно со знаком «-».<br>По концу строки напечатать значения всех элементы массива. Ошибочно набранные команды дать возможность исправлять на месте. |
| 7. | Арифметика в массиве целых чисел.<br>Пусть в программе описан массив X из десяти элементов (от 0 до 9-го). Дана строка, содержащая команды вида: X[3] + X[0] - X[9].<br>Внутри команды пробелы допускаются только около знаков арифметических операций «+» или «-».<br>По концу строки напечатать результат этих действий для каждой команде. Ошибочно набранные команды дать возможность исправлять на месте. |
| 8. | Логические выражения.<br>Пусть в программе описаны логические переменные A, B, C, D, которые все равны True. Дана строка, содержащая несколько записей логических выражений вида: A&B= A!C&D= ^B!^D&^A=.<br>Каждое выражение включает в себя обозначения логических переменных A B C или D, соединенных двуместными операциями & или ! Возможна одноместная операция ^. Внутри выражений пробелы не допускаются, но выражения разделяются произвольным числом пробелов.<br>По концу строки напечатать значения всех введенных логических выражений. Ошибочно набранные выражения дать возможность исправлять на месте. |
| 9. | Сравнения.<br>Пусть в программе описана целая переменная X и ей присвоено значение 10. Дана строка, содержащая несколько записей сравнений вида: X<34 X>=9.<br>Допускаются сравнения: > < >= <= = <> ><. Внутри сравнений пробелы не допускаются, но разделяются произвольным числом пробелов. Числа только целые положительные.<br>По концу строки напечатать количество верных введенных сравнений. Ошибочно набранные сравнения дать возможность исправлять на месте. |
| 10. | Арифметика рациональных чисел вида *m/n*.<br>Дана строка, содержащая выражения с положительными рациональными числами, записанные в виде: 28 + 7/13- 15/8.<br>Внутри чисел пробелы не допускаются. Между числами стоят знаки арифметических операций «+» или «-», окруженные произвольным числом пробелов.<br>По концу строки напечатать значения этих выражений в рациональном виде. Ошибочно набранные числа дать возможность исправлять на месте. |
| 11. | Значения многочленов.<br>Дана строка, содержащая конструкции вида: [12,-8,0](4).<br>Это означает значение многочлена $12x^2-8x$ в точке $x=4$. Многочлен может быть произвольной степени. Внутри конструкций пробелы не допускаются. Все числа должны быть только целые, возможно со знаком «-». Между конструкциями стоит произвольное число пробелов.<br>По концу строки напечатать значения всех введенных многочленов. Ошибочно набранные конструкции дать возможность исправлять на месте. |
| 12. | Решение квадратного уравнения.<br>Дана строка, содержащая выражения вида: 4*x*^2-4*x*+1=0 3+*x*^2-*x*=0 12*x*^2-3=0.<br>Каждое выражение подразумевает квадратное уравнение с одним, двумя или |

| | |
| --- | --- |
| | тремя неприводимыми одночленами, которые могут идти в произвольном порядке. Внутри конструкций пробелы допускаются только около знаков «+», «-» или «=». Выражения разделяются произвольным числом пробелов. Все коэффициенты могут быть только целыми числами, возможно со знаком «-». Отсутствие коэффициента перед *x* означает, что он равен 1.<br>По концу строки напечатать дискриминант и значения действительных корней (если они есть) для всех выражений. Ошибочно набранные конструкции дать возможность исправлять на месте. |
| 13 | Перевод конечных цепных дробей в рациональные дроби вида *m/n*.<br>Дана строка, содержащая выражения цепных (непрерывных) дробей вида: 12[7]= 3[1;4]= 0[3;12;5]=.<br>Вычисление значений цепной дроби покажем на примерах: 12[7]=12+1/7, 3[1;4]=3+1/(1+1/4))=19/5, 0[3;12;5]= 1/(3+1/(12+1/5))=61/188. Внутри конструкций пробелы могут быть только вокруг знака «+».<br>По концу строки каждое число напечатать как рациональную дробь *m/n*. Ошибочно набранные конструкции дать возможность исправлять на месте. |
| 14 | Суммирование членов арифметической прогрессии.<br>Дана строка, содержащая одно или несколько выражений вида: {2*n*-1}[10]= {*n*}[100]= {33-3*n*}[3]= .<br>Это означает, что в фигурных скобках задан общий член арифметической прогрессии, и у этой прогрессии надо подсчитать сумму стольких членов, сколько указано в квадратных скобках, начиная с первого. Внутри конструкций пробелы запрещены только перед *n*.<br>По концу строки напечатать значения всех введенных сумм. Ошибочно набранные конструкции дать возможность исправлять на месте. |
| 15 | Вычисление периметра многоугольника.<br>Многоугольник задан перечнем вершин своими координатами. Дана строка, содержащая выражение вида: P{ (21;0), (-3;4), (-1; 30) , (0; 1) } =.<br>Это означает, что в фигурных скобках заданы координаты всех вершин многоугольника, и у него надо подсчитать периметр. Внутри конструкций пробелы запрещены только внутри чисел. Числа должны быть целые, возможно, со знаком «-». Количество вершин неограниченно.<br>По концу строки напечатать значения периметров всех введенных многоугольников. Ошибочно набранные конструкции дать возможность исправлять на месте. |
| 16 | Арифметика биномиальных коэффициентов.<br>Дана строка, содержащая выражения вида: [10,6] + [10,5] =.<br>В квадратных скобках заданы параметры биномиального коэффициента (например, [7,2]=21 есть число сочетаний из 7 предметов по 2). Эти коэффициенты можно складывать и вычитать. Пробелы не допускаются только внутри чисел.<br>По концу строки напечатать значения всех введенных выражений. Ошибочно набранные конструкции дать возможность исправлять на месте. |
| 17 | Арифметика комплексных чисел.<br>Дана строка, содержащая выражения вида: (2+3*i*)*(4-*i*)= (*i*)/(1+0*i*)= (*i*-1)+(-2*i*)=.<br>Комплексное число обязательно записывается в круглых скобках. Действительная и мнимая части в числе идут в любом порядке. Действительная |

| | |
|---|---|
| | часть может отсутствовать, мнимая единица *i* обязательна. Действительная и мнимая части могут быть только целыми числами, возможно со знаком «-». Допускаются четыре арифметические операции: сложение «+», вычитание «-», умножение «*» и деление «/». Пробелы не допускаются только между мнимой частью и *i*.<br>По концу строки напечатать значения всех введенных выражений. Ошибочно набранные конструкции дать возможность исправлять на месте. |
| 18 | Защищенный ввод чисел в «научной нотации».<br>Дана строка, содержащая выражения вида: 3.14E01 1.25E-01.<br>Эти выражения представляют собой действительные десятичные положительные числа. Выражение включает в себя целую часть (обязательно от 1 до 9), десятичную точку, дробную часть, обозначение экспоненты (E) и порядок (обязательно две цифры, возможно со знаком «-»). Пробелы внутри чисел не допускаются.<br>По концу строки напечатать значения всех введенных выражений в виде конечных десятичных дробей: 3.14 0.125. Ошибочно набранные выражения дать возможность исправлять на месте. |
| 19 | Двоичные числа в «сжатом» формате.<br>Дана строка, содержащая выражения вида: 1(3)01 (5)100(2)1.<br>Эти выражения представляют собой двоичные положительные числа: 10001 111110011. Число в скобках (натуральное) означает повторение указанное количество раз одной цифры, следующей за закрывающей скобкой. Пробелы внутри чисел не допускаются. Числа разделяются произвольным числом пробелов.<br>По концу строки напечатать значения всех введенных выражений в десятичном виде. Ошибочно набранные выражения дать возможность исправлять на месте. |
| 20 | Печать регулярных событий.<br>Дана строка, содержащая выражения вида: abc ab(c)*(2)acb (ab)*(3)cb ((a)*(4)b)*(2)cb .<br>Эти выражения представляют собой слова с маленькими буквами латинского алфавита, возможно с итерацией (повторением) части слова: abc abccacb abababcb aaaabaaaabcb. Число в скобках (натуральное) означает повторение указанное количество раз подслова, стоящего в скобках. Глубина вложения инверсий не более двух. Пробелы внутри выражений не допускаются. Выражения разделяются произвольным числом пробелов.<br>По концу строки напечатать в развернутом виде все введенные выражения. Ошибочно набранные выражения дать возможность исправлять на месте. |

## 8. ТЕМАТИКА КУРСОВЫХ, КОНТРОЛЬНЫХ РАБОТ, РЕФЕРАТОВ

Не предусмотрены данной ОПОП

## 9. ПЕРЕЧЕНЬ ВОПРОСОВ К ЭКЗАМЕНУ

1. Адресная арифметика. Указатели
2. Динамическое выделение памяти
3. Динамические одномерные массивы
4. Динамические двумерные массивы
5. Списки, стеки
6. Графы, представление графов
7. Деревья

8. Вектора и алгоритмы
9. Строки библиотеки STL
10. Списки библиотеки STL
11. Поиск в глубину и в ширину
12. Алгоритм Прима-Краскала
13. Формальные теории, исчисление предикатов
14. Абстрактные формальные системы
15. Формальные грамматики
16. Операции над языками
17. Семантика формальных языков
18. Основные понятия и операции конечных автоматов
19. Распознаваемость множеств автоматами
20. Автономные автоматы
21. Неавтономные автоматы
22. Автоматы с переменной структурой
23. Вероятностные автоматы
24. Конструктор и деструктор, конструктор копирования
25. Наследование классов
26. Доступ к объектам иерархии
27. Виртуальные методы
28. Абстрактные классы
29. Перегрузка операторов
30. Шаблоны классов

## 10. САМОСТОЯТЕЛЬНАЯ РАБОТА ОБУЧАЮЩИХСЯ

Содержание, требования, условия и порядок организации самостоятельной работы обучающихся с учетом формы обучения определяются в соответствии с «Положением об организации самостоятельной работы обучающихся», утвержденным Ученым советом УлГУ (протокол №8/268 от 26.03.2019 г.).

Форма обучения: очная

| **Название разделов и тем** | **Вид самостоятельной работы** ***(проработка учебного материала, решение задач, реферат, доклад, контрольная работа, подготовка к сдаче зачета, экзамена и др.)*** | **Объем в часах** | **Форма контроля (*проверка решения задач, реферата и др.)*** |
| --- | --- | --- | --- |
| Раздел 1. | УКАЗАТЕЛИ. АДРЕСНАЯ АРИФМЕТИКА<br>*Гниденко, И. Г.* Технологии и методы программирования<br>С.169-189 | 13 | Проверка конспектов проработанного материала/ Лабораторная работа |
| | Контрольная работа на динамические структуры данных | 2 | Проверка решения |
| Раздел 2. | Проработка учебного материала по структурам данных<br>*Огнева, М. В.* Программирование на языке с++: практический курс<br>С.204-211 | 26 | Проверка конспектов проработанного материала, лабораторных |

| | | | |
| --- | --- | --- | --- |
| | | | работ |
| | Контрольная работа на работу со списками | 4 | Проверка выполненного задания |
| Раздел 3. | Проработка учебного материала по стандартной библиотеке шаблонов<br>*Огнева, М. В.* Программирование на языке с++: практический курс<br>С.316-324 | 20 | Проверка конспектов проработанного материала, лабораторных работ |
| Раздел 4. | Проработка материалов, исследование алгоритмов на обход матриц<br>*Гниденко, И. Г.* Технологии и методы программирования<br>С.230-240 | 10 | Проверка конспектов проработанного материала, лабораторных работ |
| Раздел 5. | Подготовка и выполнение контрольной работы на конечные автоматы<br>Жаркова, Г. А. Программная реализация конечных автоматов<br>С.3-18 | 10 | Проверка конспектов проработанного материала, лабораторных работ |
| Раздел 6. | Основы объектно-ориентированного программирования<br>*Огнева, М. В.* Программирование на языке с++: практический курс<br>С.248-258, 275-316 | 15 | Проверка конспектов проработанного материала, лабораторных работ |

## 11. УЧЕБНО-МЕТОДИЧЕСКОЕ И ИНФОРМАЦИОННОЕ ОБЕСПЕЧЕНИЕ ДИСЦИПЛИНЫ

### а) Список рекомендуемой литературы

**основная**

1. Огнева, М. В. Программирование на языке с++: практический курс : учебное пособие для бакалавриата и специалитета / М. В. Огнева, Е. В. Кудрина. — Москва : Издательство Юрайт, 2019. — 335 с. — (Бакалавр и специалист). — ISBN 978-5-534-05123-0. — Текст : электронный // ЭБС Юрайт [сайт]. — URL: https://urait.ru/bcode/438987
2. Гниденко, И. Г. Технологии и методы программирования : учебное пособие для прикладного бакалавриата / И. Г. Гниденко, Ф. Ф. Павлов, Д. Ю. Федоров. — Москва : Издательство Юрайт, 2019. — 235 с. — (Бакалавр. Прикладной курс). — ISBN 978-5-534-02816-4. — Текст : электронный // ЭБС Юрайт [сайт]. — URL: https://urait.ru/bcode/433611
3. Зыков, С. В. Программирование. Объектно-ориентированный подход : учебник и практикум для академического бакалавриата / С. В. Зыков. — Москва : Издательство Юрайт, 2019. — 155 с. — (Бакалавр. Академический курс). — ISBN 978-5-534-00850-0. — Текст : электронный // ЭБС Юрайт [сайт]. —

URL: https://urait.ru/bcode/434106

**дополнительная**

1. Мойзес, О. Е. Информатика. Углубленный курс : учебное пособие для прикладного бакалавриата / О. Е. Мойзес, Е. А. Кузьменко. — Москва : Издательство Юрайт, 2019. — 157 с. — (Университеты России). — ISBN 978-5-9916-7051-7. — Текст : электронный // ЭБС Юрайт [сайт]. — URL: https://urait.ru/bcode/434019
2. Ковалевская Е.В. Методы программирования [Электронный ресурс] : учебное пособие / Е.В. Ковалевская, Н.В. Комлева. — Электрон. текстовые данные. — М. : Евразийский открытый институт, 2011. — 320 с. — 978-5-374-00356-7. — Режим доступа: https://www.iprbookshop.ru/10784.html
3. Рацеев С. М. Программирование на языке СИ : учеб. пособие / С. М. Рацеев; УлГУ, ФМиИТ. - Ульяновск : УлГУ, 2015. - Загл. с экрана; Имеется печ. аналог. - Электрон. текстовые дан. (1 файл : 1,74 КБ). - Текст : электронный. Режим доступа: http://lib.ulsu.ru/MegaPro/Download/MObject/325
4. Жаркова Галина Алексеевна**.** Программная реализация конечных автоматов : учеб.-метод. пособие / Жаркова Галина Алексеевна**,** А. В. Жарков; УлГУ, Фак. матем. и информ. технологий, Каф. информ. технологий. - Ульяновск : УлГУ, 2011. - Имеется печ. аналог. - Электрон. текстовые дан. (1 файл : 350 Кб). - Текст : электронный. Режим доступа: http://lib.ulsu.ru/MegaPro/Download/MObject/653

1. Жаркова, Галина Алексеевна. Методы программирования и прикладные алгоритмы : учеб.-метод. пособие / Жаркова Галина Алексеевна, А. В. Жарков ; УлГУ, ФМИиАТ. - Ульяновск : УлГУ, 2018.

**учебно-методическая**

1. Жаркова Г. А. Методические указания для выполнения лабораторных работ студентов по дисциплине «Методы программирования и прикладные алгоритмы» для студентов бакалавриата по направлению подготовки 09.03.03 «Прикладная информатика», направленность (профиль/специализация) Информационная сфера очной формы обучения / Г. А. Жаркова; УлГУ, ФМИиАТ. - Ульяновск : УлГУ, 2019. - Загл. с экрана; Неопубликованный ресурс. - Электрон. текстовые дан. (1 файл : 459 КБ). - Текст : электронный. Режим доступа: http://lib.ulsu.ru/MegaPro/Download/MObject/7234
2. Жаркова Г. А. Методические указания для самостоятельной работы студентов по дисциплине «Методы программирования и прикладные алгоритмы» для студентов бакалавриата по направлению подготовки 09.03.03 «Прикладная информатика», направленность (профиль/специализация) Информационная сфера очной формы обучения / Г. А. Жаркова; УлГУ, ФМИиАТ. - Ульяновск : УлГУ, 2019. - Загл. с экрана; Неопубликованный ресурс. - Электрон. текстовые дан. (1 файл : 326 КБ). - Текст : электронный. Режим доступа: http://lib.ulsu.ru/MegaPro/Download/MObject/7243

Согласовано:

| гл. библ-рь ИБ УлГУ | Попина И.Ю. | | |
| --- | --- | --- | --- |
| должность сотрудника научной библиотеки | ФИО | подпись | дата |

**б)Программное обеспечение** Для образовательного процесса по данной дисциплине необходим стационарный класс ПК с установленным следующим программным обеспечением:

Microsoft Office
Microsoft Windows
ПО СОТСБИ
ЛПО «ТеМП»
NX Academic Perpetual License CAE+CAM
NX Academic Perpetual License Core+CAD
«Антиплагиат.ВУЗ»
КОМПАС-3D
Альт Рабочая станция
МойОфис Стандартный
SQL Server
Visual Studio
MATLAB
Embarcadero RAD Studio
Maple
Statistica
Средства защиты информации Secret Net Studio 8
Академическая лицензия на УМК ViPNet "Защита сетей"

**Список свободно распространяемого ПО:**

Qt Creator
JDK
PostgreSQL
Python IDLE
Scilab
Visual studio code
Code::Blocks IDE
Visual Studio Community
Ubuntu linux
Oracle VM VirtualBox
Xunbuntu
LibreOffice
Calculate Linux

**в) Профессиональные базы данных, информационно-справочные системы**

**1. Электронно-библиотечные системы:**

1.1. IPRbooks : электронно-библиотечная система : сайт / группа компаний Ай Пи Ар Медиа. - Саратов, [2021]. – URL: http://www.iprbookshop.ru. – Режим доступа: для зарегистрир. пользователей. - Текст : электронный.

1.2. ЮРАЙТ : электронно-библиотечная система : сайт / ООО Электронное издательство ЮРАЙТ. – Москва, [2021]. - URL: https://www.biblio-online.ru. – Режим доступа: для зарегистрир. пользователей. - Текст : электронный.

1.3. Консультант студента : электронно-библиотечная система : сайт / ООО Политехресурс. – Москва, [2021]. – URL: http://www.studentlibrary.ru/catalogue/switch_kit/x2019-128.html. – Режим доступа: для зарегистрир. пользователей. – Текст : электронный.

1.4. Лань : электронно-библиотечная система : сайт / ООО ЭБС Лань. – Санкт-Петербург, [2021]. – URL: https://e.lanbook.com. – Режим доступа: для зарегистрир. пользователей. – Текст : электронный.

1.5. **Znanium.com :** электронно-библиотечная система : сайт / ООО Знаниум. - Москва, [2021]. - URL: http://znanium.com. – Режим доступа : для зарегистрир. пользователей. - Текст : электронный.

1.6. Clinical Collection : коллекция для медицинских университетов, клиник, медицинских библиотек // EBSCOhost : [портал]. – URL: http://web.a.ebscohost.com/ehost/search/advanced?vid=1&sid=e3ddfb99-a1a7-46dd-a6eb-2185f3e0876a%40sessionmgr4008. – Режим доступа : для авториз. пользователей. – Текст : электронный.

**2. КонсультантПлюс** [Электронный ресурс]: справочная правовая система. /ООО «Консультант Плюс» - Электрон. дан. - Москва : КонсультантПлюс, [2021].

**3. Базы данных периодических изданий:**

3.1. База данных периодических изданий : электронные журналы / ООО ИВИС. - Москва, [2021]. – URL: https://dlib.eastview.com/browse/udb/12. – Режим доступа : для авториз. пользователей. – Текст : электронный.

3.2. eLIBRARY.RU: научная электронная библиотека : сайт / ООО Научная Электронная Библиотека. – Москва, [2021]. – URL: http://elibrary.ru. – Режим доступа : для авториз. пользователей. – Текст : электронный

3.3. «Grebennikon» : электронная библиотека / ИД Гребенников. – Москва, [2021]. – URL: https://id2.action-media.ru/Personal/Products. – Режим доступа : для авториз. пользователей. – Текст : электронный.

**4. Национальная электронная библиотека** : электронная библиотека : федеральная государственная информационная система : сайт / Министерство культуры РФ ; РГБ. – Москва, [2021]. – URL: https://нэб.рф. – Режим доступа : для пользователей научной библиотеки. – Текст : электронный.

**5. SMART Imagebase** // EBSCOhost : [портал]. – URL: https://ebsco.smartimagebase.com/?TOKEN=EBSCO-1a2ff8c55aa76d8229047223a7d6dc9c&custid=s6895741. – Режим доступа : для авториз. пользователей. – Изображение : электронные.

**6. Федеральные информационно-образовательные порталы:**

6.1. Единое окно доступа к образовательным ресурсам : федеральный портал / учредитель ФГАОУ ДПО ЦРГОП и ИТ. – URL: http://window.edu.ru/. – Текст : электронный.

6.2. Российское образование : федеральный портал / учредитель ФГАОУ ДПО ЦРГОП и ИТ. – URL: http://www.edu.ru. – Текст : электронный.

**7. Образовательные ресурсы УлГУ:**

7.1. Электронная библиотека УлГУ : модуль АБИС Мега-ПРО / ООО «Дата Экспресс». – URL: http://lib.ulsu.ru/MegaPro/Web. – Режим доступа : для пользователей научной библиотеки. – Текст : электронный.

7.2. Образовательный портал УлГУ. – URL: http://edu.ulsu.ru. – Режим доступа : для зарегистр. пользователей. – Текст : электронный.

Согласовано:

Заместитель начальника УИТиТ /Клочкова А.В. _____/_____

## 12. МАТЕРИАЛЬНО-ТЕХНИЧЕСКОЕ ИЛИ ИНФОРМАЦИОННОЕ ОБЕСПЕЧЕНИЕ ДИСЦИПЛИНЫ

Аудитории для проведения лекций, семинарских занятий, для проведения лабораторных работ, для проведения текущего контроля и промежуточной аттестации.

Аудитории укомплектованы специализированной мебелью, учебной доской. Аудитории для проведения лекций оборудованы мультимедийным оборудованием для представления информации большой аудитории. Помещения для самостоятельной работы оснащены компьютерной техникой с возможностью подключения к сети «Интернет» и обеспечением доступа к электронной информационно-образовательной среде, электронно-библиотечной системе.

Помещение 3/414. Аудитория для проведения практических и лекционных занятий, текущего контроля и промежуточной аттестации, групповых и индивидуальных консультаций с набором демонстрационного оборудования для обеспечения тематических иллюстраций. Помещение укомплектовано ученической доской и комплектом мебели (посадочных мест – 38). 432017, Ульяновская область, г. Ульяновск, ул. Набережная реки Свияги, д. 106 (3 корпус).

Помещение 303. Аудитория для проведения лабораторных занятий. Помещение укомплектовано ученической доской и комплектом мебели (посадочных мест – 10). Компьютеры, Wi-Fi с доступом к сети «Интернет», ЭИОС, ЭБС. Проектор, экран. 432017, Ульяновская область, г. Ульяновск, ул. Набережная реки Свияги, д. 106 (1 корпус).

Реализация программы дисциплины требует наличия учебной лаборатории. Оборудование учебной лаборатории: посадочные места по количеству студентов. Технические средства обучения: компьютеры с лицензионным программным обеспечением:
Microsoft Office
Microsoft Windows
ПО СОТСБИ
ЛПО «ТеМП»
NX Academic Perpetual License CAE+CAM
NX Academic Perpetual License Core+CAD
«Антиплагиат.ВУЗ»
КОМПАС-3D
Альт Рабочая станция
МойОфис Стандартный
SQL Server
Visual Studio
MATLAB
Embarcadero RAD Studio
Maple
Statistica
Средства защиты информации Secret Net Studio 8
Академическая лицензия на УМК ViPNet "Защита сетей"

**Список свободно распространяемого ПО:**

Qt Creator
JDK
PostgreSQL
Python IDLE

Scilab

Visual studio code
Code::Blocks IDE
Visual Studio Community
Ubuntu linux
Oracle VM VirtualBox
Xunbuntu
LibreOffice
Calculate Linux

## 13. СПЕЦИАЛЬНЫЕ УСЛОВИЯ ДЛЯ ОБУЧАЮЩИХСЯ С ОГРАНИЧЕННЫМИ ВОЗМОЖНОСТЯМИ ЗДОРОВЬЯ

Обучение по ОПОП ВО обучающихся с ограниченными возможностями здоровья осуществляется с учетом особенностей психофизического развития, индивидуальных возможностей и состояния здоровья таких обучающихся. Образование обучающихся с ограниченными возможностями здоровья может быть организовано как совместно с другими обучающимися, так и отдельно. В случае необходимости, обучающимся из числа лиц с ограниченными возможностями здоровья (по заявлению обучающегося) могут предлагаться одни из следующих вариантов восприятия информации с учетом их индивидуальных психофизических особенностей:

– для лиц с нарушениями зрения: в печатной форме увеличенным шрифтом; в форме электронного документа; в форме аудиофайла (перевод учебных материалов в аудиоформат); в печатной форме на языке Брайля; индивидуальные консультации с привлечением тифлосурдопереводчика; индивидуальные задания и консультации.

– для лиц с нарушениями слуха: в печатной форме; в форме электронного документа; видеоматериалы с субтитрами; индивидуальные консультации с привлечением сурдопереводчика; индивидуальные задания и консультации.

– для лиц с нарушениями опорно-двигательного аппарата: в печатной форме; в форме электронного документа; в форме аудиофайла; индивидуальные задания и консультации».

В случае необходимости использования в учебном процессе частично/исключительно дистанционных образовательных технологий, организация работы ППС с обучающимися с ОВЗ и инвалидами предусматривается в электронной информационно-образовательной среде с учетом их индивидуальных психофизических особенностей.

Разработчик _____ (подпись) профессор (должность) _____ (ФИО)