Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение
высшего образования
**«ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ» (ТУСУР)**

Кафедра автоматизации обработки информации (АОИ)

ИНФОРМАТИКА И ПРОГРАММИРОВАНИЕ

Методические указания к лабораторным работам, курсовой работе и организации самостоятельной работы
для студентов направления
«Бизнес-информатика»
(уровень бакалавриата)

2018

Пермякова Наталья Викторовна
Информатика и программирование: Методические указания к лабораторным работам, курсовой работе и организации самостоятельной работы для студентов направления «Бизнес-информатика» (уровень бакалавриата) / Н.В. Пермякова. — Томск, 2018. — 67 с.

Содержание

1 Введение

В методических указаниях к проведению лабораторных работ, выполнению курсовой работы и организации самостоятельной работы по дисциплине «Информатика и программирование» собраны методические рекомендации для поддержки аудиторных занятий и самостоятельной работы студентов.

Целью проведения лабораторных работ, выполнения курсовой работы и организации самостоятельной работы является формирование и развитие навыков структурного и объектно-ориентированного программирования.

По окончанию обучения дисциплины «Информатика и программирование» студент должен:

— **знать** основные факты, концепции, принципы и теории, связанные с информатикой; типы систем счисления; основные принципы и конструкции структурного программирования; основные принципы объектно-ориентированного подхода к программированию; графические способы представления алгоритмов;

— **уметь** разрабатывать алгоритмы решаемых задач; представлять алгоритмы в виде блок-схем, псевдокода, диаграмм Насси-Шнайдермана, программ на языке высокого уровня; использовать базовые алгоритмы для решения задач; выполнять объектно-ориентированный анализ предметной области;

— **владеть навыками** использования компьютера как средства управления информацией, работать с информацией из различных источников, в том числе в глобальных компьютерных сетях; реализации и отладки программ на алгоритмических языках программирования; использования различных структур данных при решении задач.

Цикл лабораторных работ по дисциплине можно условно разделить на две группы. Лабораторные работы первой группы направлены на закрепление навыков использования конструкций структурного программирования. Выполнение лабораторных работ второй группы формирует навыки объектно-ориентированного программирования.

Курсовая работа по дисциплине формирует навыки самостоятельной разработки программного продукта в соответствии с принципами структурного программирования.

Самостоятельная работа студентов по дисциплине содержит несколько видов деятельности — проработка лекционного материала, подготовка к контрольным работам, подготовка к лабораторным работам, выполнение домашних заданий, самостоятельное изучение тем, подготовка к экзамену.

2 Методические указания к проведению лабораторных работ

2.1 Общие положения

Целью проведения лабораторных работ является формирование и развитие навыков структурного и объектно-ориентированного программирования.

Основной формой проведения лабораторных работ является разработка алгоритма решения индивидуальной задачи и его программная реализация на языке Си/Си++. Процесс программной реализации включает в себя написание программы, отладку программы и тестирование программы.

К основным способам контроля формирования компетенций при выполнении лабораторных работ относятся индивидуальная защита выполненной работы, организация опроса студентов по теоретическому материалу дисциплины, практическое применение которого осуществляется в ходе выполнения лабораторной работы.

Для получения максимальной оценки за лабораторную работу необходимо выполнить и защитить работу во время, отведенное для ее выполнения, согласно расписанию занятий. Допускается досрочное выполнение лабораторной работы по предварительной договоренности с преподавателем.

Выполнение всех лабораторных работ, предусмотренных рабочей программой дисциплины, является условием допуска к итоговому контролю изучения дисциплины — экзамену.

2.2 Лабораторная работа «Создание консольного приложения в среде DEV-C++. Ввод-вывод информации»

Цель работы: ознакомиться с интегрированной средой Dev – C++, изучить основные типы данных языка Си, функции ввода и вывода информации, получить навыки написания программ на языке Си.

Форма проведения: выполнение индивидуального задания.

Подготовка к выполнению лабораторной работы: для выполнения лабораторной работы необходимо изучить теоретический материал, изложенный в [1]. Описание процесса создания проекта в IDE DEV-C++ рассматривается в главе 2 пособия (стр. 33 — 35). Описание структуры простой программы (стр. 81 — 82) и универсальных функций ввода-

вывода информации (стр. 77 — 81) в главе 5 пособия. Для реализации индивидуального задания необходимо ознакомиться с операторами языка Си (стр. 52 — 55) и основными типами данных (стр. 59 — 60).

Порядок выполнения работы

1. Получить индивидуальный вариант;
2. создать проект в Dev-C++;
3. написать программу на языке Си, выполняемые функции которой могут быть описаны следующей последовательностью шагов:
    3.1. описать входные и выходные данные;
    3.2. ввести данные с клавиатуры;
    3.3. вычислить значение функции;
    3.4. вывести полученное значение на экран;
    3.5. вывести личные данные.
4. выполнить компиляцию проекта;
5. выполнить тестирование проекта;
6. защитить работу.

Контрольные вопросы

1. Какое имя носит исполняемая функция Си?
2. Дайте определение понятия «переменная».
3. Дайте определение понятия «идентификатор».
4. Сколько переменных требуется описать в программе, если необходимо решить следующую задачу — «С клавиатуры вводятся три числа, необходимо вывести на экран значение минимального из этих трех чисел»?
5. Какая функция используется в Си для ввода информации?
6. Какая функция используется в Си для вывода информации?
7. Какой тип данных Си соответствует спецификатору «%d»?
8. Какой тип данных Си соответствует спецификатору «%f»?
9. Переменная *j* описана в программе следующим образом: *int j;* Запишите функцию *scanf ()* для считывания значения в переменную *j*.
10. Переменная *k* описана в программе следующим образом: *float k*; Запишите функцию *printf ()* для вывода значения переменной *k*.

Пример выполнения индивидуального варианта

Вариант 1: Ввести с клавиатуры целое число *x*. Вывести на экран значение функции $x^2 + 3.1x + 7.5$ и сообщение вида: «Программу выполнил ФИО».

В таблице 1 представлена программа, реализующая индивидуальное задание.

Таблица 1 — Этапы выполнения лабораторной работы

№	Название этапа	Описание результатов выполнения этапа
2	Создание проекта	Запустить Dev – C++, создать новый проект, дополнить код программы вызовом функции `system ("chcp 1251");` — смена кодировки страницы.
3	Реализация программы	
3.1	Описание переменных	*int x;* *float y;*
3.2	Ввод данных с клавиатуры	`printf(“Введите значение переменной x: ”);` `scanf(“%d”,&x);`
3.3	Вычисление значения функции	`y = x*x+3.1*x + 7.5;`
4	Вывод результата	`printf(“Значение функции: %7.2f\n”,y);`
5	Вывод личных данных	`printf(“Программу выполнил Иванов Андрей Сергеевич\n”);`

После набора представленной выше программы в шаблоне функции `main` () (рис. 1) выполните компиляцию проекта.

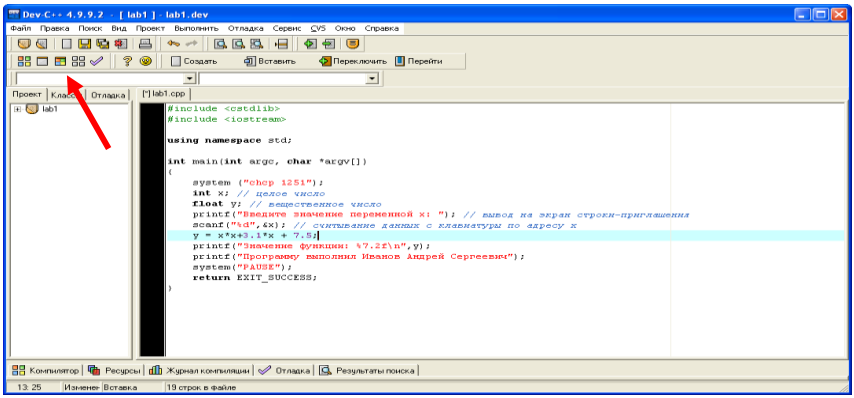


Рисунок 1 — Проект выполнения индивидуального задания

Определите имя файла, содержащего функцию `main()`. В рассматриваемом примере имя файла определено как `lab1` (рис. 2).

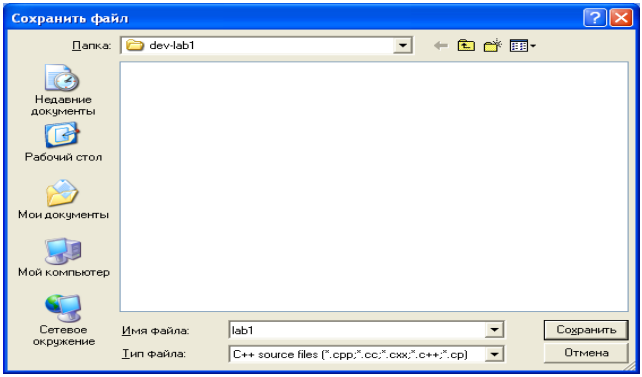


Рисунок 2 — Определение имени файла

Если этап компиляции прошел успешно, программа автоматически выполнится. Для смены кодировки страницы зайдите в свойства консольного окна, выберите вкладку «Шрифт» и выберите шрифт Lucida Console.

Результат работы программы представлен на рисунке 3.

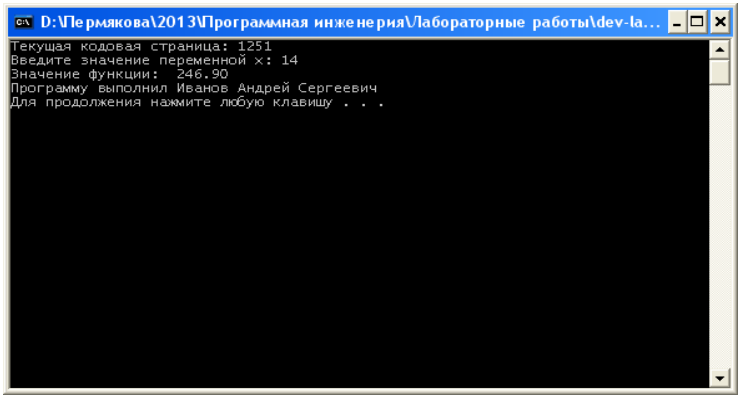


Рисунок 3 — Тестирование программы при x =14

При выполнении лабораторной работы в консольном окне выведено полученное значение переменной и личные данные студента, выполняющего работу.

2.3 Лабораторная работа «Проверка ошибок ввода в языке программирования Си»

Цель работы: ознакомиться с возможностями функции `scanf()`. Научиться составлять условные алгоритмы на примере алгоритма проверки ошибок ввода данных. Реализовать алгоритм на языке Си.

Форма проведения: выполнение индивидуального задания.

Рекомендации по подготовке к лабораторной работе: для выполнения лабораторной работы необходимо изучить теоретический материал, изложенный в [1]. Основные конструкции структурного программирования рассматриваются в главе 1 пособия (стр. 13 — 15). Описание синтаксиса конструкции проверки условия в языке Си (стр. 87 — 90) в главе 6 пособия. Возможности функции `scanf` для организации проверки корректности ввода данных описаны на стр. 80 — 81.

Порядок выполнения работы

1. Получить индивидуальный вариант;
2. по индивидуальному варианту определить типы и значения данных, являющиеся некорректными для задачи;
3. составить и записать алгоритм решения задачи;
4. составить программу, реализующую алгоритм:
    4.1. описать входные и выходные данные;
    4.2. ввести данные с клавиатуры;
    4.3. проверить входные данные;
    4.4. вычислить значение функции;
    4.5. вывести полученное значение на экран;
    4.6. вывести личные данные;
    4.7. выполнить компиляцию проекта;
5. защитить работу.

Пример выполнения индивидуального варианта

Вариант 1. Составить и записать алгоритм решения индивидуального задания с проверкой корректности данных. По составленному алгоритму написать программу на языке Си.

Даны *x, y, z*. Вычислить *a, b*, если $a = (\frac{2y}{z-x} - |x|)(\sqrt[4]{x - \frac{y}{\sqrt{x}}})$, $b = a - \frac{x}{2a} + \frac{x^2}{a-z}$. Значения *x,y,z* вводить с клавиатуры.

В таблице 2 описано поэтапное выполнение лабораторной работы.

Таблица 2 — Этапы выполнения лабораторной работы

№	Название этапа	Описание результата выполнения этапа
2.	Определение некорректных данных	Символьные данные; $z - x = 0$— ошибка деления на 0; $x \leq 0$— ошибка извлечения квадратного корня, ошибка деления на 0; $x - \frac{y}{\sqrt{x}} < 0$— ошибка извлечения корня четвертой степени $a = 0$ — ошибка деления на 0; $a = z$— ошибка деления на 0.
3.	Алгоритм	алг Лабораторная работа 2 нач ввод x,y,z если x или y или z — не число, то рез “Введены не числа” если z-x=0 или x = 0, то рез “Ошибка деления на 0” если$x - \frac{y}{\sqrt{x}} < 0$или $x < 0$ то рез “Ошибка извлечения корня” $a = (\frac{2y}{z-x} - \lvert x \rvert)(\sqrt[4]{x - \frac{y}{\sqrt{x}}})$ рез a если a = 0 или a = z то рез “Ошибка деления на 0” $b = a - \frac{x}{2a} + \frac{x^2}{a-z}$ рез b конец
4.	Составление программы	Для использования математических функций и констант подключите библиотеку математических функций: `#include <math.h>`
4.6.	Вывод личных данных	`printf(“Программу выполнил Иванов Андрей Сергеевич\n”);`
4.1.	Описание переменных	`float x,y,z,a,b;`

4.2.	Ввод данных с клавиатуры	printf(“Введите значения переменных: ”); int m = scanf(“%f%f%f”,&x,&y,&z);

4.3.	Проверка входных данных	if (m!=3) {
		printf (“Введены не числа\n”);
		system(“pause”);
		return 0;}
		if (x==0 ||z-x==0) {
		printf (“Ошибка деления на 0\n”);
		system(“pause”);
		return 0;}
		float temp = x – y/sqrt(x);
		if (temp<0||x<0) {
		printf (“Ошибка извлечения корня\n”);
		system(“pause”);
		return 0;}

4.4	Вычисление значения функции	a = 2*y/(z-x)-fabs(x); a = a*pow(temp,1/4.);
4.5	Вывод результата	printf(“Значение a = %9.3f\n”,a);
4.3.	Проверка входных данных	if (a==0||a==z) { printf (“Ошибка деления на 0\n”); system(“pause”); return 0;}
4.4	Вычисление значения функции	b = a – x/(2*a)+ x*x/(a-z);
4.5	Вывод результата	printf(“Значение b = %9.3f\n”,b);

Контрольные вопросы

1. Что возвращает функция *scanf()*?
2. Запишите функцию *scanf ()* для ввода трех переменных.
3. Что Вы понимаете под некорректными данными?
4. Какие данные будут некорректными для решения следующей задачи — “Даны длины трех сторон треугольника. Найдите площадь треугольника.”
5. Как в Си реализована условная конструкция структурного программирования?
6. Опишите синтаксис конструкции if else языка Си.
7. Какое значение примет переменная m после выполнения следующего фрагмента программы:

…

```
float i;
int j;
int m = scanf(“%f%d”,&i, &j);
…
```

если с клавиатуры были введены значения 3 2?

8. Какое значение примет переменная *m* после выполнения следующего фрагмента программы:

```
…
float i;
int j;
int m = scanf(“%f%d”,&i, &j);
…
```

если с клавиатуры были введены значения 3 d?

9. Какое значение примет переменная x после выполнения следующего фрагмента программы:

```
…
int x = 10;
int k = 12,z = 74;
if (k<z) x = 1; else x = 0;
…
```

2.4 Лабораторная работа «Проверка условий. Геометрия на плоскости»

Цель работы: закрепление навыков построения разветвляющихся алгоритмов.

Форма проведения: выполнение индивидуального задания.

Проверка расположения точки с координатами (x, y) относительно прямой: пусть уравнение прямой задано в каноническом виде *y = ax + b*. Тогда все точки, лежащие на линии прямой (рис. 4), подчиняются условию $y = \mathrm{a}x + b$. На рисунке это условие выполняется для точки с координатами (x_3, y_3). Все точки, лежащие левее линии прямой, подчиняются условию $y < \mathrm{a}x + b$, это условие выполняется для точки с координатами (x_1, y_1). Все точки, лежащие правее линии прямой, подчиняются условию $y > \mathrm{a}x + b$. Это условие является истинным для точки с координатами (x_2, y_2). Тогда для выбранных трех точек являются истинными условия: $y_3 = \mathrm{a}x_3 + b; y_2 > \mathrm{a}x_2 + b; y_1 < \mathrm{a}x_1 + b.$

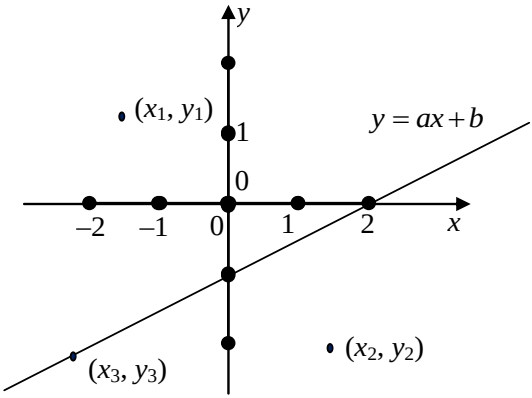


Рисунок 4 — Расположение точки относительно прямой

Для прямой, изображенной на рис. 4, составим уравнение прямой по двум заданным точкам: прямая проходит через точки с координатами (0, –1) и (2, 0). Найдем коэффициенты уравнения a и b. Для этого решим систему уравнений:

$$\begin{cases} -1 = a \cdot 0 + b \\ 0 = a \cdot 2 + b \end{cases} \Rightarrow \begin{cases} b = -1 \\ a = \dfrac{-b}{2} \end{cases} \Rightarrow \begin{cases} b = -1 \\ a = 0.5 \end{cases} \Rightarrow \quad y = 0.5x - 1.$$

Таким образом, проверить местоположение произвольной точки с координатами (x, y) можно, написав следующий код:

```
…
if (y<0.5*x-1)
printf(“Точка расположена левее прямой”);
   else   if(y>0.5*x-1) )
        printf(“Точка расположена правее прямой”);
        else printf(“Точка расположена на прямой”);
   …
```

Проверка расположения точки относительно окружности с заданным центром известного радиуса: каноническое уравнение окружности выглядит следующим образом:

$$R^2 = (x - x_1)^2 + (y - y_1)^2, \quad (1)$$

где R — радиус окружности,

(x_1, y_1) — координаты центра окружности.

Тогда (рис. 5) для точки с координатами (x_4, y_4), выполняется равенство: $R^2 = (x_4 - x_1)^2 + (y_4 - y_1)^2$.

Выражение (1) истинно для всех точек, лежащих на линии окружности. Для точки с координатами (x_2, y_2) и для всех точек, лежащих за окружностью, выполняется неравенство:

$R^2 < (x_2 - x_1)^2 + (y_2 - y_1)^2$.

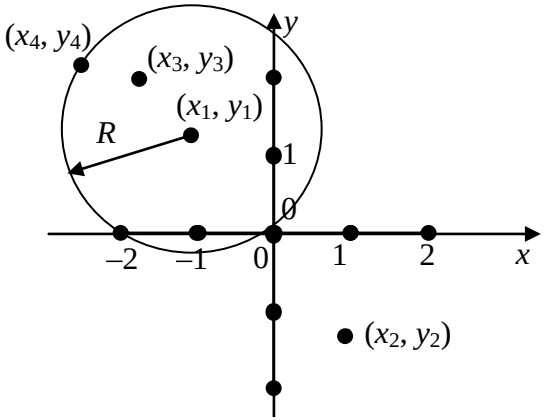


Рисунок 5 — Расположение точки относительно окружности

Из этого неравенства следует, что радиус R окружности меньше радиуса окружности с центром в точке (x_1, y_1), на которой лежит точка с координатами (x_2, y_2). Соответственно, для точки с координатами (x_3, y_3) выполняется неравенство: $R^2 > (x_3 - x_1)^2 + (y_3 - y_1)^2$.

То есть радиус R окружности, на которой лежит точка с координатами (x_3, y_3) больше радиуса окружности с центром в точке (x_1, y_1).

Порядок выполнения работы

1. Получить индивидуальный вариант.
2. Определить условия вхождения точки в заданную область.
3. Составить и записать алгоритм решения задачи.
4. Составить программу, реализующую алгоритм:
   4.1.описать входные и выходные данные;
   4.2.ввести данные с клавиатуры;
   4.3.проверить входные данные;

4.4.проверить условие вхождения точки в заданную область;
4.5.вывести результат проверки на экран;
4.6.вывести личные данные.

5. Выполнить компиляцию проекта.
6. Защитить работу.

Примеры проверки вхождения точки с заданными координатами в фигуру

Пример 1. Напишите алгоритм, проверяющий вхождение точки в область, изображенную на рисунке 6.

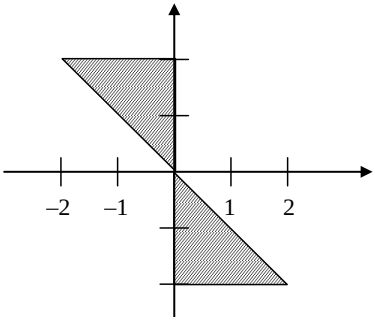


Рисунок 6 — Пример 1

Построим условия вхождения точки в заданную область:

Уравнение прямой, на которой лежат гипотенузы прямоугольных треугольников, образующих фигуру: y = –x.

Разобьем фигуру на две части. Точка будет считаться принадлежащей фигуре, если она попадет в первую или вторую часть.

Первую (верхнюю часть) можно ограничить следующими условиями: *(y >= –x) и (x <= 0) и (y <= 2).*

Первое условие описывает гипотенузу, второе и третье условие описывают катеты. Условия связаны между собой связками **И** (логическое умножение).

Вторую (нижнюю часть) можно ограничить условиями:

(y <= –x) и (x >= 0) и (y >= –2).

Общее условие для двух частей будет выглядеть следующим образом:

если ($y >= -x$) **и** ($x <= 0$) **и** ($y <= 2$) **или**

($y <= -x$) **и** ($x >= 0$) **и** ($y >= -2$),

то «Точка принадлежит заданной области»,

иначе «Точка не принадлежит заданной области».

Тогда алгоритм проверки вхождения точки с заданными координатами будет выглядеть следующим образом:

алг Пример 1 нач

вещ *x*,*y*

ввод *x*,*y*

если ($y >= -x$) и ($x <= 0$) и ($y <= 2$) или

($y <= -x$) и ($x >= 0$) и ($y >= -2$),

то «Точка принадлежит заданной области»,

иначе «Точка не принадлежит заданной области»

кон

Пример 2. Рассмотрим еще один пример, заданная область изображена на рисунке 7. В этом случае: уравнение прямой $y = x$; уравнение окружности $1 = (x - 1)^2 + (y - 1)^2$.

Ограниченная область находится правее прямой $(y < x)$ и внутри окружности $(1 > (x - 1)^2 + (y - 1)^2)$.

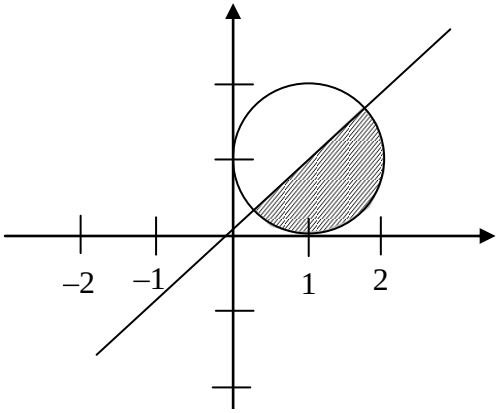


Рисунок 7 — Пример 2

Тогда общее условие будет выглядеть следующим образом:

если $y < x$ **и** $1 > (x-1)^2 + (y-1)^2$,

то «Точка принадлежит заданной области»,

иначе «Точка не принадлежит заданной области».

Тогда алгоритм проверки вхождения точки с заданными координатами будет выглядеть следующим образом:

```
алг Пример 2 нач
вещ x,y
ввод x,y
если    y < x и 1 > (x – 1)^2 + (y – 1)^2,
то «Точка принадлежит заданной области»,
иначе «Точка не принадлежит заданной области»
кон
```

2.5 Лабораторная работа «Цикл for»

Цель работы: закрепление навыков программирования циклических процессов с использованием цикла `for` языка Си.

Форма проведения: выполнение индивидуального задания.

Рекомендации по подготовке к лабораторной работе: для выполнения лабораторной работы необходимо изучить теоретический материал, изложенный в [1]. Основные конструкции структурного программирования рассматриваются в главе 1 пособия (стр. 13 — 15). Описание синтаксиса конструкции цикла `for` в языке Си (стр. 91 — 92) в главе 6 пособия.

Порядок выполнения работы

1. Получить индивидуальный вариант.
2. Разработать алгоритм решения поставленной задачи.
3. Написать и протестировать программу, составленную по разработанному алгоритму.
4. Защитить работу.

Пример выполнения задания

Задание: известна масса каждого предмета, загружаемого в автомобиль. Определить общую массу груза.

Алгоритм решения поставленной задачи представлен на рис. 8.

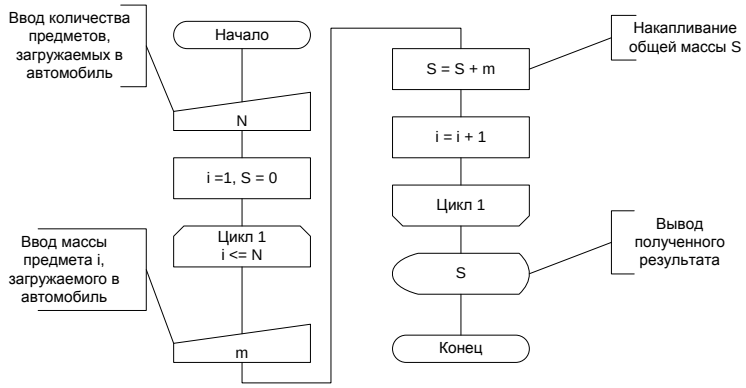


Рисунок 8 — Алгоритм накапливания суммы

Программная реализация

```
int main(int argc, char *argv[]) {
    system("chcp 1251");
    int N,i = 1,S = 0,m;
    printf("Введите количество предметов, загружаемых в автомобиль: ");
    scanf("%d", &N);
    for(i=1;i<=N;i++){
        printf("Вес предмета %d: ",i);
        scanf("%d", &m);
        S = S + m;
    }
    printf("Общий вес: %d\n",S);
    return 0;
}
```

Результат работы программы представлен на рис. 9.

```
Текущая кодовая страница: 1251
Введите количество предметов, загружаемых в автомобиль: 5
Вес предмета 1: 2
Вес предмета 2: 4
Вес предмета 3: 1
Вес предмета 4: 6
Вес предмета 5: 7
Общий вес: 20
```

Рисунок 9 — Результат выполнения программы

На вход программы подается общее количество предметов, равное 5. Далее в цикле вводятся массы пяти предметов: 2, 4, 1, 6, 7. Общая масса всех предметов равна 20.

2.6 Лабораторная работа «Циклы while и do while»

Цель работы: закрепление навыков работы с циклами по условию на примере задач целочисленной арифметики.

Форма проведения: выполнение индивидуального задания.

Рекомендации по подготовке к лабораторной работе: для выполнения лабораторной работы необходимо изучить теоретический материал, изложенный в [1]. Правила языка Си для работы с массивами — стр. 93 — 95.

Порядок выполнения работы

1. Получить индивидуальное задание.
2. Составить алгоритм решения задания и реализовать его графическое представление (блок-диаграмма, диаграмма Насси-Шнайдермана или псевдокод).
3. Составить программу на языке Си.
4. Выполнить отладку и тестирование программы.
5. Защитить работу.

Пример выполнения задания

Задание: напишите программу, которая печатает цифры произвольного числа.

Алгоритм решения поставленной задачи представлен на рис. 10.

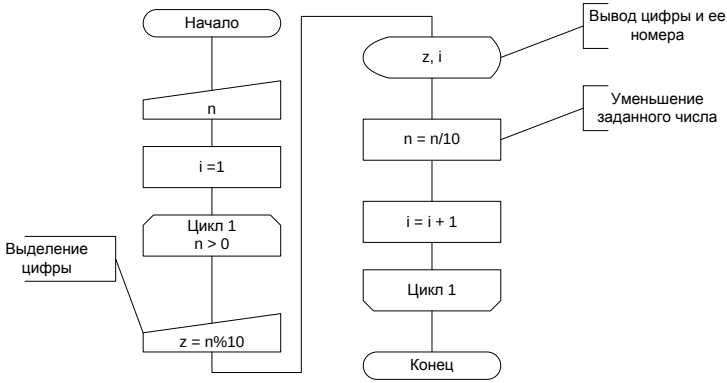


Рисунок 10 — Алгоритм выделения цифр числа

Программная реализация

```
int main(int argc, char *argv[])
{
  system("chcp 1251");
  int n = 152348;
  int z,i=1;
  printf("n = %d\n", n);
  while(n>0){
            z = n%10;
            printf("Цифра %d - %d\n",i,z);
            n=n/10;
            i++;
            }
  system("PAUSE");
  return 0;
}
```

Результат работы программы представлен на рис. 11.

```
Текущая кодовая страница: 1251
n = 152348
Цифра 1 - 8
Цифра 2 - 4
Цифра 3 - 3
Цифра 4 - 2
Цифра 5 - 5
Цифра 6 - 1
Для продолжения нажмите любую клавишу . . .
```

Рисунок 11 — Результат выполнения программы

Программа выводит цифры заданного числа слева направо, от младшего разряда к старшему.

2.7 Лабораторная работа «Организация хранения данных в массиве»

Цель работы: закрепление навыков работы со статическими массивами, знакомство с генератором случайных чисел, обращение к элементу массива.

Форма проведения: выполнение индивидуального задания.

Рекомендации по подготовке к лабораторной работе: для выполнения лабораторной работы необходимо изучить теоретический материал, изложенный в [1]. Правила языка Си для работы с массивами — стр. 64 — 66, 114 — 120.

Порядок выполнения работы

1. Получить индивидуальное задание.
2. Составить алгоритм решения задания и реализовать его графическое представление (блок-диаграмма, диаграмма Насси-Шнайдермана или псевдокод).
3. Составить программу на языке Си.
4. Выполнить отладку и тестирование программы.
5. Защитить работу.

Пример выполнения индивидуального задания

Задание: дан массив размерности n. Элементы массива заполнить случайными значениями от -20 до 25. Найти квадратный корень из любого заданного элемента массива. Определить сумму всех элементов массива.

Алгоритм решения поставленной задачи представлен на рис. 12.

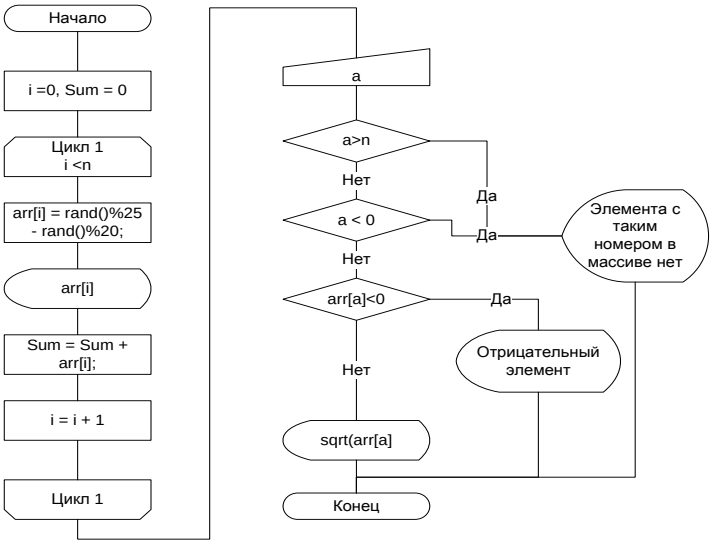


Рисунок 12 — Алгоритм решения задачи

Программная реализация

```
int main(int argc, char *argv[])
 {system("chcp 1251");
  int n = 10,i;
  int Sum=0, a, arr[n];
  printf("Элементы массива: \n");
  for(i=0; i<n; i++){
    arr[i] = rand()%25 - rand()%20;
    printf("%d ", arr[i]);
    Sum = Sum + arr[i]; }
  printf("Введите порядковый номер элемента: ");
  scanf("%d",&a);
  if (a>n || a<0)  printf("Элемента с таким номером в массиве нет\n");
  else
      if (arr[a-1]<0) printf("Отрицательный элемент\n");
  else
     printf("Корень квадратный V%d = %6.2f\n",arr[a-1],sqrt(arr[a-1]));
  printf("Сумма элементов массива %6d\n", Sum);
  }
```

Результат работы программы представлен на рис. 13.

```
Текущая кодовая страница: 1251
Элементы массива:
9 9 15 -15 8 0 -1 0 18 -14 Введите порядковый номер элемента: 1
Корень квадратный V9 =   3.00
Сумма элементов массива     29

--------------------------------
Process exited after 11.58 seconds with return value 31
Для продолжения нажмите любую клавишу . . .
```

```
Текущая кодовая страница: 1251
Элементы массива:
-4 -16 22 5 11 5 -1 9 1 5 Введите порядковый номер элемента: 2
Отрицательный элемент
Сумма элементов массива     37

--------------------------------
Process exited after 5.778 seconds with return value 31
Для продолжения нажмите любую клавишу . . .
```

```
Текущая кодовая страница: 1251
Элементы массива:
11 -9 3 11 9 -12 2 4 9 -9 Введите порядковый номер элемента: 12
Элемента с таким номером в массиве нет
Сумма элементов массива     19

--------------------------------
Process exited after 6.274 seconds with return value 31
Для продолжения нажмите любую клавишу . . .
```

Рисунок 13 — Результаты тестирования программы

Написанная программа корректно обрабатывает информацию о введенном значении номера элемента. При обработке первых тестовых данных программа вычисляет значение квадратного корня первого элемента массива. При обработке вторых тестовых данных программа определяет, что произошла попытка вычисления квадратного корня из отрицательного числа, при обработке третьего набора тестовых данных программа определяет, что элемента с заданным номером не существует.

2.8 Лабораторная работа «Организация поиска в одномерных массивах»

Цель работы: изучение способов работы с динамическими одномерными массивами; реализация простых алгоритмов поиска; организация отладки программы с помощью встроенного отладчика.

Форма проведения: выполнение индивидуального задания.

Рекомендации по подготовке к лабораторной работе: для выполнения лабораторной работы необходимо изучить теоретический материал, изложенный в [1]. В главе 8 пособия изложена справочная информация об организации работы с одномерными массивами в языке Си (стр. 114 — 120), классические алгоритмы поиска в наборах данных (стр. 20 — 27) описываются в первой главе пособия. Руководство по использованию встроенного отладчика в IDE описано в главе 2 (стр. 37 — 42). Дополнительную информацию по алгоритмам организации поиска в одномерных массивах можно найти [2], стр. 15 — 40.

Порядок выполнения работы

1. Получить индивидуальный вариант.
2. Изучить теоретические аспекты лабораторной работы.
3. Изучить работу отладчика в среде DEV-CPP.
4. Разработать и реализовать на языке Си алгоритм решения предложенных задач.
5. Защитить работу, используя средства отладчика на тестовом примере с массивом из десяти элементов.

Пример выполнения индивидуального задания

Задание: В одномерном массиве, состоящем из целых элементов, вычислить:

А) сумму индексов четных (по значению) элементов массива.

Б) найти минимальный элемент массива.

Программная реализация

```
int main(int argc, char *argv[])
 {system("chcp 1251");
srand(time(NULL));
 int n = 10,i;
 int Sum=0, a, arr[n];
 printf("Элементы массива: \n");
 for(i=0; i<n; i++){
   arr[i] = rand()%25 - rand()%20;
   printf("%d ", arr[i]);
   Sum = Sum + arr[i]; }
 // сумма индексов четных (по значению) элементов
 for(i=0; i<n; i++){
   if(arr[i]%2==0)
   Sum = Sum + i; }
// поиск минимального элемента
 a = arr[0];
 for(i=1; i<n; i++){
   if(arr[i]<a)
   a = arr[i]; }
 printf("\nСумма индексов четных элементов %d \n",Sum);
 printf("Минимальное значение в массиве %d \n",a);
 }
```

2.9 Лабораторная работа «Простые сортировки на месте»

Цель работы: реализация простых обменных сортировок.

Форма проведения: выполнение индивидуального задания.

Рекомендации по подготовке к лабораторной работе: для выполнения лабораторной работы необходимо изучить теоретический материал, изложенный в [1]. В главе 8 пособия описаны алгоритмы простых сортировок (стр. 121 — 124). Дополнительную информацию о сортировках можно получить в [3], стр. 72 — 80.

Порядок выполнения работы

1. Получить индивидуальный вариант.
2. Написать программу, реализующую заданный алгоритм сортировки.

3. Отладить и протестировать программу.
5. Защитить работу.

Алгоритмы простых сортировок

Сортировка обменом

```
for (i=0; i<n-1; i++){
        flag = 0;
          for (j=0; j<n-i-1; j++)
               if (x[j]>x[j+1]) {
                                  flag = 1;
                                 temp = x[j];
                                 x[j]=x[j+1];
                                 x[j+1]=temp; }
 if (flag==0) break;
…
```

Сортировка выбором

```
…
for(int i=0;i<n-1;i++)
 { int k = i;
   for (int j=i+1;j<n;j++)
     if (x[k]>x[j])  k=j;
   if(k!=i) { int temp=x[i];
              x[i]=x[k] ;
              x[k]=temp;}
} …
```

Сортировка вставками

```
…
for(i=1;i<n;i++){
   buf = x[i];
   j=i-1;
   while(buf<x[j]&&j>=0)
       {x[j+1]=x[j];
        j--;}
   x[j+1]=buf;
        }
…
```

2.10 Лабораторная работа «Матрицы — 1»

Цель работы: закрепление навыков работы с двумерными массивами — создание матрицы с заданным количеством строк и столбцов, печать элементов матрицы, выделение областей матриц, поиск элементов с заданными характеристиками.

Форма проведения: выполнение индивидуального задания.

Рекомендации по подготовке к лабораторной работе: для выполнения лабораторной работы необходимо изучить теоретический материал, изложенный в [1]. В главе 8 пособия описаны правила языка Си для работы с матрицами (стр. 124 — 132).

Порядок выполнения работы

1. Получить индивидуальный вариант.
2. Составить алгоритм решения задачи.
3. Написать программу по составленному алгоритму.
4. Отладить и протестировать программу.
5. Защитить работу.

Пример построения алгоритма по индивидуальному заданию

Задание: Дан двумерный массив. Найти сумму элементов верхней половины матрицы.

Алгоритм решения задачи представлен на рис. 14.

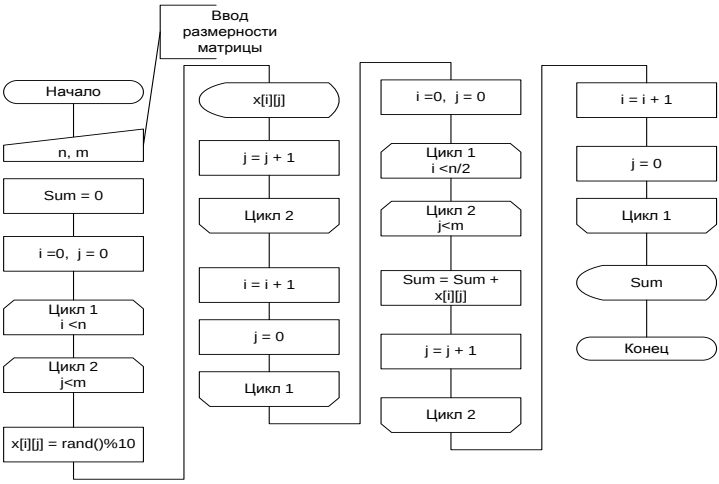


Рисунок 14 — Алгоритм обработки матрицы

2.11 Лабораторная работа «Матрицы — 2»

Цель работы: закрепление навыков работы с двумерными массивами, разработка и реализация алгоритмов поиска экстремальных элементов матрицы

Форма проведения: выполнение индивидуального задания.

Рекомендации по подготовке к лабораторной работе: для выполнения лабораторной работы необходимо изучить теоретический материал, изложенный в [1]. В главе 8 пособия описаны правила языка Си для работы с матрицами (стр. 124 — 127). Примеры работы с матрицами представлены на стр. 127 — 132.

Порядок выполнения работы

1. Получить индивидуальный вариант.
2. Составить алгоритм решения задачи.
3. Написать программу по составленному алгоритму.
4. Отладить и протестировать программу.
5. Защитить работу.

Пример программной реализации индивидуального задания

Задание: дан двумерный массив. Найти столбец с максимальной суммой элементов.

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
int main(int argc, char *argv[]) {
    system("chcp 1251");
    int n,m,**x,i,j,S,jmax, smax;
    printf("Введите количество строк и столбцов матрицы: ");
    scanf("%d%d", &n,&m);
    srand(time(NULL));
    // Выделение памяти под двумерный массив
    x = (int**) malloc(sizeof(int*)*n);
    for(i=0;i<n;i++) x[i] = (int*) malloc(sizeof(int)*m);
    // Формирование матрицы случайным образом и печать элементов
    for(i=0;i<n;i++) {
        for(j=0;j<m;j++)
        {
            x[i][j] = rand()%20;
            printf("%4d",x[i][j]);
        }
        printf("\n");
    }
```

```
    // Организация поиска столбца с максимальной суммой
        jmax = 0; smax = -1000;
        for(j=0;j<m;j++){
        S = 0;
        for(i=0;i<n;i++){
            S+=x[i][j];
        }
        if (smax<S) {
            jmax = j;
            smax = S;
        }
        }
    printf("Столбец с максимальной суммой: %d (%d)\n",jmax,smax);
    return 0;
}
```

2.12 Лабораторная работа «Матрицы — 3»

Цель работы: реализация алгоритмов обработки квадратных матриц.

Форма проведения: выполнение индивидуального задания.

Рекомендации по подготовке к лабораторной работе: для выполнения лабораторной работы необходимо изучить теоретический материал, изложенный в [1]. В главе 8 пособия описаны правила языка Си для работы с матрицами (стр. 124 — 127). Примеры работы с матрицами представлены на стр. 127 — 132.

Порядок выполнения работы

1. Получить индивидуальный вариант.
2. Составить алгоритм решения задачи.
3. Написать программу по составленному алгоритму.
4. Отладить и протестировать программу.
5. Защитить работу.

Примеры определения параметров циклов для выделения областей квадратных матриц

Главная диагональ

```
for(i=0;i<n;i++) Обращение к элементу x[i][i]
```

Побочная диагональ

```
for(i=0;i<n;i++) Обращение к элементу x[i][n-i-1]
```

Треугольная область матрицы под главной диагональю

```
for(i=0;i<n;i++)
   for(j=0;j<i;j++) Обращение к элементу x[i][j]
```

Треугольная область матрицы над главной диагональю

```
for(i=0;i<n;i++)
   for(j=i+1;j<n;j++) Обращение к элементу x[i][j]
```

Треугольная область матрицы под побочной диагональю

```
for(i=0;i<n;i++)
   for(j=n-i;j<n;j++) Обращение к элементу x[i][j]
```

Треугольная область матрицы под побочной диагональю

```
for(i=0;i<n;i++)
for(j=0;j<n-i-1;j++) Обращение к элементу x[i][j]
```

2.13 Лабораторная работа «Функции»

Цель работы: закрепление навыков разработки функций.

Форма проведения: выполнение индивидуального задания.

Рекомендации по подготовке к лабораторной работе: для выполнения лабораторной работы необходимо изучить теоретический материал, изложенный в [1]. В главе 7 пособия описаны основные моменты организации функций на языке Си (стр. 103 — 109).

Порядок выполнения работы

1. Получить индивидуальный вариант.
2. Написать программу, решающую поставленную задачу с использованием функций.
3. Отладить и протестировать программу.
4. Подготовить электронный вариант документа, содержащий описание функций.
5. Защитить работу.

Пример описания функции

int fprintf(FILE * stream, const char * format, ...);

Описание

Функция *fprintf* выполняет форматированный вывод в поток. Записывает в указанный поток последовательность символов в формате, указанном аргументом *format*. После параметра *format*, функция ожидает, по крайней мере, многие дополнительные аргументы, как указано в прототипе.

Параметры:

stream

Указатель на объект типа *FILE*, который связан с потоком.

format

Си-строка, содержащая текст, который будет выведен на поток. Опционально, строка может содержать встроенные метки форматирования, которые заменяются значениями, указанными в последующих дополнительных аргументах и отформатированы требуемым образом.

Дополнительные аргументы

В зависимости от формата строки, функция может принимать дополнительные аргументы, каждый из которых содержит одно значение. Вместо каждого %-тега, указанного в параметре *format*, в поток вывода будет вставлено значение соответствующего аргумента. Количество дополнительных аргументов должно соответствовать количеству %-тегов.

Возвращаемое значение

В случае успеха, возвращается общее число записанных символов.

В случае неудачи, возвращается отрицательное число.

2.14 Лабораторная работа «Обработка строк»

Цель работы: реализация алгоритмов работы со строками, изучение стандартных функций для работы со строками в языке Си.

Форма проведения: выполнение индивидуального задания.

Рекомендации по подготовке к лабораторной работе: для выполнения лабораторной работы необходимо изучить теоретический материал, в [1]. В главе 8 пособия содержится справочная информация по организации работы со строками (стр. 133 — 143). При подготовке к лабораторной работе обратите особое внимание на представление строки в памяти компьютера (стр. 134) и на примеры работы со строками (стр. 139 — 143).

Порядок выполнения работы

1. Получить индивидуальный вариант.

2. Написать программу, решающую поставленную задачу с использованием стандартных функций Си для работы со строками.

3. Отладить и протестировать программу.

4. Защитить работу.

2.15 Лабораторная работа «Многофайловая компиляция»

Цель работы: обобщение навыков обработки двумерных массивов на языке Си, формирование навыков работы с многофайловыми проектами, создание заголовочных файлов.

Форма проведения: выполнение индивидуального задания.

Рекомендации по подготовке к лабораторной работе: При выполнении задания может быть полезен материал пособия [1], стр. 42 — 43, теоретические и справочные материалы по теме «Функции» (стр. 103 — 110), по теме «Матрицы» (стр. 124 — 133).

Порядок выполнения работы

1. Получить индивидуальный вариант.

2. Создать проект на языке Си.

3. Написать заголовочный файл, содержащий прототипы реализуемых функций и добавить заголовочный файл в созданный проект.

4. Реализовать функции для работы с матрицами согласно индивидуальному варианту и добавить файл в созданный проект.

5. Написать функцию `main()`, тестирующую написанные функции для матриц произвольной размерности.

7. Защитить работу.

Пример заголовочного файла

Задание: Напишите библиотеку функций для работы с матрицами. В функции main продемонстрируйте вызовы функций для произвольной матрицы.

Создание матрицы.

Печать матрицы.

Поиск максимального элемента.

Печать элементов заданного столбца.

Печать элементов заданной строки.

Поиск количества строк матрицы, «похожих», на первую. «Похожими» будем называть строки, множества элементов которых совпадают.

```
// создание матрицы
int * Create (int *x, int n, int m);
// печать матрицы
void Print (int *x, int n, int m);
// поиск максимального элемента
int  Max (int *x, int n, int m);
// печать элементов заданного столбца с номером k
void PrintS (int *x, int n, int m, int k);
// печать элементов заданной строки с номером k
void  PrintR (int *x, int n, int m, int k);
// Поиск количества строк матрицы, "похожих", на первую.
int  Search (int *x, int n, int m);
```

2.16 Лабораторная работа «Структурные переменные»

Цель работы: закрепление навыков работы со структурированными данными.

Форма проведения: выполнение индивидуального задания.

Рекомендации по подготовке к лабораторной работе: для выполнения лабораторной работы необходимо изучить теоретический материал, в [1]. В главе 4 пособия содержится справочная информация об объявлении такого типа данных, как структура (стр. 67 — 69).

Порядок выполнения работы

1. Получить индивидуальный вариант.
2. Создать в папке проекта текстовый файл, содержащий описанную в задании информацию.
3. Написать программу, решающую поставленную задачу с использованием функций.
4. Отладить и протестировать программу.
5. Защитить работу.

Пример выполнения индивидуального варианта

Дан массив записей, содержащих информацию о сдаче студентами одной группы экзаменов по математике, физике и программированию. Расположить записи в массиве по убыванию оценки по математике. Вывести отсортированный массив на экран.

Программная реализация

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
int main(int argc, char *argv[]) {
  system("chcp 1251");
  typedef struct {
  struct Name{
     char surname[30];
     char name[20];
     char patronymic[30];
    } nstudent;
  int m,f,p;
} Student;
int n = 5,i,vs,j;
Student *array, S_vs;
array = (Student*) malloc(sizeof(Student)*n);
for ( i=0;i<n;i++){
    printf("Вводите информацию о студенте: \n");
    printf("Фамилия: ");
    scanf("%s", array[i].nstudent.surname);
    printf("Имя: ");
    scanf("%s", array[i].nstudent.name);
    printf("Отчество: ");
    scanf("%s", array[i].nstudent.patronymic);
    printf("Математика: ");
    scanf("%d",&array[i].m);
    printf("Физика: ");
    scanf("%d",&array[i].f);
    printf("Программирование: ");
    scanf("%d",&array[i].p);
 }
```

```
printf("Исходные данные: \n");
for(i=0;i<n;i++)
 printf("%15s%15s%3d%3d%3d\n",array[i].nstudent.surname,
                              array[i].nstudent.name,
                      array[i].m,  array[i].f,
                      array[i].p);
for (i=1;i<n;i++)
  {  S_vs = array[i];
     vs = array[i].m;
     j = i-1;
    while(vs>array[j].m&&j>=0)
       {array[j+1]=array[j];
         j--;}
    array[j+1] = S_vs;
  }
printf("\n После сортировки:\n");
for(i=0;i<n;i++)
  printf("%15s%15s%3d%3d%3d\n",array[i].nstudent.surname,
                               array[i].nstudent.name,
                           array[i].m, array[i].f,
                       array[i].p);

return 0;}
```

2.17 Лабораторная работа «Текстовые файлы»

Цель работы: закрепление навыков работы с текстовыми файлами.

Форма проведения: выполнение индивидуального задания.

Рекомендации по подготовке к лабораторной работе: для выполнения лабораторной работы необходимо изучить теоретический материал, изложенный в [1]. В главе 9 пособия описаны стандартные функции языка Си для работы с текстовыми файлами (стр. 144 — 155).

Порядок выполнения работы

1. Получить индивидуальный вариант.
2. Создать в папке проекта текстовый файл, содержащий описанную в задании информацию.
3. Написать программу, решающую поставленную задачу с использованием функций.
4. Отладить и протестировать программу.
5. Защитить работу.

Пример выполнения индивидуального варианта

В текстовом файле расположено произвольное количество чисел. Не считывая все числа в массив найти минимальное значение в файле и количество четных чисел в файле. Дописать найденную информацию в исходный файл

Программная реализация

```
int main(int argc, char *argv[]) {
    FILE *f = fopen("p2.txt", "r");
    if (!f) { printf ("Файл не найден...");
    system("pause");
    return 0;
    }
    int min, k = 0, i = 0, a;
    while (!feof(f)) {
        if (fscanf(f,"%d",&a)!=1) break;
        printf("%3d", a);
        i++;
        if (i==1) min = a;
          else {
            if (a<min) min = a;
          }
        if (a%2 == 0) k++;
    }
    fclose(f);
    f = fopen("p2.txt", "a");
    printf("\nМинимальный элемент: %d\n", min);
    printf("Количество четных чисел: %d\n", k);
    fprintf(f,"\nМинимальный элемент: %d\n", min);
    fprintf(f,"Количество четных чисел: %d\n", k);
    fclose(f);
    return 0;
}
```

2.18 Лабораторная работа «Двоичные файлы»

Цель работы: освоить навыки работы с двоичными файлами.

Форма проведения: выполнение индивидуального задания.

Рекомендации по подготовке к лабораторной работе: перед проведением занятия необходимо изучить теоретический материал, изложенный в [1]. Глава 9 пособия, стр. 156 — 166.

Порядок проведения работы

1. Получить индивидуальный вариант.
2. Написать, отладить и выполнить программу, создающую двоичный файл.
3. Написать программу, выполняющую задание, сформулированное в индивидуальном варианте.
4. Отладить и протестировать программу.
5. Защитить работу.

Пример программы, заполняющей двоичный файл

```
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[]) {
    system("chcp 1251");
    int i, z, n = 100;
    FILE *f = fopen("1.bin","wb");
    if (f==NULL) {
        printf("Ошибка создания файла. \n");
        system("pause");
        return;     }
// Запись n чисел в файл
    for(i=0;i<n;i++)
      {
        z = rand()%200/(rand()%100+1.)-rand()%70;
        fwrite(&z,sizeof(float),1,f);
      }
   fclose(f);
}
```

2.19 Лабораторная работа «Введение в объектно-ориентированное программирование. Работа с классом Vector»

Цель работы: ознакомиться с основными принципами объектно-ориентированного программирования (ООП) и синтаксисом C++ для создания объектно-ориентированных приложений на примере класса Vector.

Форма проведения: выполнение индивидуального задания.

Рекомендации по подготовке к лабораторной работе: перед проведением занятия необходимо изучить теоретический материал, изложенный в [3], глава 6 пособия, стр. 105 — 112, справочный материал по теме работы можно найти в [4], стр. 130 — 133.

Порядок проведения работы

1. Получить индивидуальный вариант.
2. Самостоятельно рассмотрите реализацию методов класса VECTOR.
3. Написать программу, выполняющую задание, сформулированное в индивидуальном варианте.
4. Отладить и протестировать программу.
5. Защитить работу.

Описание класса Vector представлено в приложении 1.

2.20 Лабораторная работа «Введение в ООП. Создание класса»

Цель работы: формирование навыков реализации классов в Си++.

Форма проведения: выполнение индивидуального задания.

Рекомендации по подготовке к лабораторной работе: изучите синтаксис языка Си++. Изучите теоретический материал пособий [3] (стр. 105 — 109) и [4] (стр. 130 — 139). Обратите внимание на следующие вопросы темы: описание класса; поля класса; методы класса; атрибуты доступа к элементам класса.

Порядок проведения работы

1. Получить индивидуальный вариант.
2. Разработать структуру класса по выбранному варианту.
3. Написать программу, демонстрирующую создание объектов реализованного класса.
4. Отладить и протестировать программу.

5. Защитить работу.

2.21 Лабораторная работа «Конструкторы и деструкторы»

Цель работы: формирование навыков реализации классов в Си++.

Форма проведения: выполнение индивидуального задания.

Рекомендации по подготовке к лабораторной работе: изучите синтаксис языка Си++, необходимый для реализации специальных методов класса. Изучите теоретический материал пособий [3] (стр. 125 — 137) и [4] (стр. 134 — 139). Обратите внимание на следующие вопросы темы: использование конструктора и деструктора по умолчанию; перегрузка конструкторов; связь конструктора и деструктора с операторами `new` и `delete`.

Порядок проведения работы

1. Для индивидуального варианта, полученного на предыдущей лабораторной работе, разработать структуру дополнительного конструктора и деструктора.
2. Написать программную реализацию разработанных элементов класса.
3. Написать программу, демонстрирующую создание объектов реализованного класса разными способами.
4. Отладить и протестировать программу.
5. Защитить работу.

2.22 Лабораторная работа «Методы класса»

Цель работы: формирование навыков реализации классов в Си++.

Форма проведения: выполнение индивидуального задания.

Рекомендации по подготовке к лабораторной работе: Изучите теоретический материал пособий [3] (стр. 108 — 111) и [4] (стр. 142 — 145). Обратите внимание на следующие вопросы темы: атрибуты доступа к элементам класса; статические поля и методы; указатель this.

Порядок проведения работы

1. Получить индивидуальный вариант.
2. Разработать структуру и алгоритмы реализуемых методов.
3. Написать программу, демонстрирующую применение написанных методов класса к объектам класса.
4. Отладить и протестировать программу.

5. Защитить работу.

2.23 Лабораторная работа «Массивы объектов»

Цель работы: формирование навыков реализации классов в Си++.

Форма проведения: выполнение индивидуального задания.

Рекомендации по подготовке к лабораторной работе: при подготовке к лабораторной работе необходимо повторить справочный материал по организации работы с массивами в языке Си — учебное пособие [1], стр. 114 — 123.

Организация работы с массивами объектов в С++

Статические массивы статических объектов

Синтаксис описания:

<Имя класса> <Имя массива> [Размер массива]

Такое описание может быть выполнено только при наличии в классе конструктора без параметров.

Если конструктор без параметров в классе не определен, то в программе можно описать статический массив ссылок на объекты.

Синтаксис описания:

<Имя класса> *<Имя массива> [Размер массива]

В этом случае необходимо выделить память под все объекты массива.

Динамические массивы объектов

Динамические массивы объектов могут быть организованы двумя способами: динамические массивы объектов и динамические массивы ссылок на объекты.

Динамические массивы объектов можно объявить в программе следующим образом:

<Имя класса> *<Имя массива> ;

Выделение памяти под массив:

<Имя массива> = new <Имя класса>[количество элементов массива];

Такой синтаксис выделения памяти может быть использован только в случае наличия в классе конструктора без параметров.

Пример, создающий массивы различных типов представлен в приложении 2.

Порядок проведения работы

1. Получить индивидуальный вариант.

2. Написать программу, в которой создается массив объектов указанного типа для класса, реализованного в предыдущих лабораторных работах.

3. Отладить и протестировать программу.

4. Защитить работу.

2.24 Лабораторная работа «Перегрузка операторов»

Цель работы: формирование навыков реализации перегруженных операторов в Си++.

Форма проведения: выполнение индивидуального задания.

Рекомендации по подготовке к лабораторной работе: при подготовке к лабораторной работе необходимо повторить справочный материал по организации перегрузки стандартных операторов в Си++ — учебное пособие [3], стр. 138 — 144, учебное пособие [4], стр. 140 — 142, учебное пособие [5], стр. 212 — 228.

Порядок проведения занятия

1. Получить индивидуальный вариант.

2. Разработать структуру перегружаемых операторов.

3. Написать программу, демонстрирующую применение перегруженных операторов к объектам класса.

4. Отладить и протестировать программу.

5. Защитить работу.

2.25 Лабораторная работа «Наследование и полиморфизм»

Цель работы: формирование навыков проектирования и реализации системы классов.

Форма проведения: выполнение индивидуального задания.

Рекомендации по подготовке к лабораторной работе: при подготовке к лабораторной работе необходимо ознакомиться с двумя основными принципами ООП: наследование — учебное пособие [4], стр. 171 — 175, учебное пособие [5], стр. 229 — 238; полиморфизм — учебное пособие [4], стр. 177 — 182, учебное пособие [5], стр. 239 — 247. При подготовке к лабораторной работе обратите внимание на следующие вопросы темы: типы наследования, синтаксис объявления классов-наследников, типы полиморфизма, виртуальные функции, позднее и ранее связывание.

Порядок проведения занятия

1. Самостоятельно разработать класс-наследник для ранее реализованного класса.
2. Реализовать класс-наследник программно.
3. Написать программу, демонстрирующую принципы динамического и статического полиморфизма.
4. Отладить и протестировать программу.
5. Защитить работу.

2.26 Лабораторная работа «Обработка исключений»

Цель работы: знакомство с механизмом обработки исключительных ситуаций.

Форма проведения: выполнение индивидуального задания.

Рекомендации по подготовке к лабораторной работе: при подготовке к лабораторной работе необходимо изучить теоретический материал по теме — учебное пособие [4], стр. 163 — 165, учебное пособие [5], стр. 271 — 274. При изучении материала особое внимание уделите следующим вопросам: понятие исключительной ситуации, синтаксис обработки исключительной ситуации, этапы обработки исключительных ситуаций.

Порядок проведения занятия

1. Самостоятельно определить 2 — 3 исключительные ситуации, которые могут возникнуть при работе с объектами ранее разработанных классов.
2. Реализовать программно механизм обработки исключительных ситуаций.
3. Отладить и протестировать программу.
4. Защитить работу.

2.27 Лабораторная работа «Потоки»

Цель работы: знакомство с файловыми потоками языка Си++.

Форма проведения: выполнение индивидуального задания.

Рекомендации по подготовке к лабораторной работе: при подготовке к лабораторной работе необходимо изучить теоретический материал по теме — учебное пособие [3], стр. 159 — 179, учебное пособие [5],

стр. 248 — 262. При изучении материала особое внимание уделите следующим вопросам: типы потоков, иерархия потоковых классов, организация прямого доступа в потоках

Порядок проведения занятия

1. Получить индивидуальный вариант.
2. Разработать систему классов для решения задачи индивидуального варианта.
3. Программно реализовать спроектированную систему классов.
4. Отладить и протестировать программу.
5. Защитить работу.

3 Методические указания к выполнению курсовой работы

3.1 Общие положения

Курсовая работа является завершающим этапом в изучении блока дисциплины, рассматривающего структурное программирование. Выполнение курсовой работы должно способствовать закреплению теоретических знаний, полученных во время изучения дисциплины и применения этих знаний для решения поставленной задачи.

Можно определить следующие цели выполнения курсовой работы:

- систематизация, закрепление и расширение теоретического материала по структурному подходу к программированию и основам языка Си;
- формирование у студента навыков научно-исследовательской работы; формирование навыков самостоятельной разработки алгоритмов для решения вычислительных задач;
- приобретение практических навыков реализации компьютерных программ.

Во время выполнения курсовой работы студент должен приобрести и закрепить навыки:

- работы со специальной литературой фундаментального и прикладного характера;
- самостоятельной работы над поставленной задачей; оформления пояснительной записки к курсовой работе;
- представлению и защите результатов курсовой работы.

Курсовая работа выполняется и защищается в сроки, определенные учебным графиком. Выполнение курсовой работы предполагает проведение аудиторных занятий в форме консультаций и самостоятельное развитие тематики курсовой работы студентом в течение семестра. Задание на курсовую работу утверждаются и выдаются в начале семестра.

Основной формой контроля уровня формирования компетенций при выполнении курсовой работы является публичная защита выполненной работы.

Работа выполняется студентом самостоятельно. Руководитель курсовой работы формирует задание на курсовую работу, осуществляет мониторинг процесса выполнения работы студентом и предоставляет консуль-

тативную помощь во время аудиторных занятий по курсовой работе согласно расписанию занятий.

Общие требования к выполнению работы

Программная реализация задания на курсовую работу должна быть выполнена на языке программирования Си.

Тестовые данные для программной реализации студент формирует самостоятельно.

Программа должна представлять собой законченный продукт, который может использовать сторонний пользователь.

К защите работы студент обязан предоставить: протестированное откомпилированное приложение, исходные коды программ на CD-диске; отчет по курсовой работе.

3.2 Примерная тематика курсовых работ

В соответствии с вариантом задания необходимо реализовать два численных метода и исследовать эффективность реализованных методов. При выполнении курсовой работы исследуется эффективность методов численного интегрирования, оптимизации функций одного аргумента, методов решения нелинейных уравнений. Критерием эффективности является время выполнения метода при заданной точности вычислений.

Примеры заданий на курсовую работу

1. Численное интегрирование. Метод трапеций. Метод Уэддля.
2. Оптимизация функций. Метод золотого сечения. Метод дихотомии.
3. Численное интегрирование. Метод прямоугольников. Метод Симпсона.
4. Решение нелинейных уравнений. Метод итераций. Метод касательных.
5. Оптимизация функций. Метод сканирования. Общий поиск.
6. Решение нелинейных уравнений. Метод секущих. Метод половинного деления.
7. Оптимизация функций. Метод Фибоначчи. Метод сканирования.
8. Численное интегрирование. Метод трапеций. Метод Симпсона.
9. Численное интегрирование. Метод трапеций. Метод модифицированных прямоугольников.
10. Решение нелинейных уравнений. Метод итераций. Метод половинного деления.

11. Численное интегрирование. Метод трапеций. Метод прямоугольников.

12. Решение нелинейных уравнений. Метод касательных. Метод половинного деления.

3.3 Порядок выполнения курсовой работы

Конкретизация задания на курсовую работу

На первом этапе выполнения работы необходимо выделить задачи, которые требуется решить при выполнении курсовой работы и оформить часть отчета, содержащую задание на курсовую работу. В приложении 3 приведен пример выполнения данного этапа для одного из вариантов курсовой работы.

Изучение численных методов

На этом этапе курсовой работы студент должен самостоятельно разобрать алгоритмы работы численных методов, согласно теме задания на курсовую работу. При изучении методов рекомендуется обратиться к следующим источникам:

Волков, Е.А. Численные методы [Электронный ресурс] : учебник / Е.А. Волков. — Электрон. дан. — Санкт-Петербург : Лань, 2008. — 256 с. — Режим доступа: https://e.lanbook.com/book/54. — Загл. с экрана. (стр. 103 — 118).

Киреев, В.И. Численные методы в примерах и задачах [Электронный ресурс] : учебное пособие / В.И. Киреев, А.В. Пантелеев. — Электрон. дан. — Санкт-Петербург : Лань, 2015. — 448 с. — Режим доступа: https://e.lanbook.com/book/65043. — Загл. с экрана. (стр. 70 — 117).

Денежкина, И. Е. Численные методы: Курс лекций[Электронный ресурс] : Учебное пособие / И. Е. Денежкина. — М.: Финансовая академия, 2004. — 112 с. — Режим доступа: http://znanium.com/bookread2.php?book=497545 . (стр. 32 — 47, 73 — 78).

В процессе изучения рекомендуется вести заметки и определить основные этапы выполнения численного метода. Для понимания принципов работы алгоритма рекомендуется визуализировать этапы выполнения метода в виде схем или графиков. Если после самостоятельного изучения остались вопросы по работе метода, можно обсудить сложные для понимания элементы метода во время аудиторных занятий по курсовой работе.

Проектирование алгоритмов численных методов

На этом этапе курсовой работы студент должен представить работу метода в виде алгоритма. Для представления алгоритма допускается использовать любой из графических способов представления алгоритмов или воспользоваться псевдокодом. При разработке алгоритма обратите особое внимание на определение входных и выходных данных, на структуру реализуемого алгоритма.

Программная реализация численных методов

На этом этапе курсовой работы студент выполняет представление разработанных ранее алгоритмов в виде программ на языке высокого уровня. Программная реализация включает этапы написания кода, отладки программы и тестирование программы. Тестирование программы рекомендуется выполнять на функциях, для которых значения, вычисленные с помощью реализованных методов можно проверить, определив эти значения аналитическим способом.

Подготовка тестовых материалов для исследования эффективности реализованных методов

При выполнении этого этапа студент самостоятельно или с помощью преподавателя подбирает наборы тестовых функций, на которых впоследствии будет проводиться анализ эффективности численных методов. Реализованная программа дополняется маркерами, с помощью значений которых можно выполнить сравнение методов. Такими элементами могут быть подсчет количества сравнений, выполняемых алгоритмом, количество вычислений значения целевой функции, количество итераций, выполняемых методом для получения значения, время выполнения метода. Все полученные во время тестирования данные документируются.

Анализ эффективности реализованных методов

Студент самостоятельно анализирует полученные результаты и делает выводы об эффективности реализованных численных методов. Анализ результатов может содержать визуализацию полученных после тестирования данных в виде разнообразных графиков изменения полученных маркеров.

Подготовка отчета по курсовой работе

Пояснительная записка к курсовой работе должна включать:

- титульный лист;
- задание на курсовую работу;

- содержание;
- введение;
- основную часть;
- заключение;
- список литературы;
- приложения.

Титульный лист, содержание и список литературы оформляется согласно ОС ТУСУР 01-2013.30 [6]. Шаблон листа задания приведен в приложении 3.

Содержание раздела «Введение»: определение цели; формулировка задач; краткая характеристика предметной области (вычислительная математика).

Основная часть отчета должна содержать: математическое описание используемых численных методов; описание алгоритмов численных методов; описание тестовых данных, процесса тестирования и результатов тестирования, анализ результатов тестирования.

Заключение должно содержать краткие выводы о проделанной работе, практическое приложение, перспективы использования результатов работы.

В список литературы входят те источники литературы, на которые есть ссылки в пояснительной записке к курсовой работе. В качестве приложений к пояснительной записке помещают листинги программ.

Подведение итогов выполнения курсовой работы

Подведение итогов выполнения курсовой работы включает следующие этапы: предварительная проверка отчета и реализованных численных методов руководителем; доработка курсовой работы с учетом замечаний руководителя; сдача готового и подписанного отчета по курсовой работе; подготовка презентации и доклада.

Защита курсовой работы

Курсовая работа, допускается к защите, при условии отсутствия замечаний со стороны руководителя по выполненной работе и по содержанию и оформлению работы, о чем руководитель делает запись на титульном листе. Защита курсовой работы проводится публично в присутствии группы.

4 Методические указания для организации самостоятельной работы

4.1 Общие положения

Самостоятельная работа является важной составляющей в изучении дисциплины и состоит из следующих видов деятельности: проработка лекционного материала для подготовки к тестированию и контрольным работам, подготовка к лабораторным работам, выполнение домашних заданий, выполнение контрольных работ.

Самостоятельная работа над теоретическим материалом направлена систематизацию и закрепление знаний, полученных на лекционных занятиях и на получение новых знаний по дисциплине, путем самостоятельного изучения тем.

Самостоятельная работа по подготовке к лабораторным работам направлена на изучение методического и теоретического материала по теме лабораторной работы.

Выполнение домашних заданий — полностью самостоятельная работа, направленная на получение навыков самостоятельного составления алгоритмов, реализацию программ, их дальнейшей отладки и тестирования.

4.2 Проработка лекционного материала, подготовка к контрольным работам и лабораторным работам

Проработка лекционного курса является одной из важных активных форм самостоятельной работы. Этот вид самостоятельной работы может быть организован следующим образом:

- прочитайте конспект лекции, согласуя Ваши записи с информацией на слайдах лекции;
- попробуйте выполнить самостоятельно примеры программ, разобранных на лекции;
- если в лекции рассматривался какой-либо алгоритм, попытайтесь выполнить этот алгоритм на тестовых данных без использования компьютерной программы; такой способ проработки материалов лекции покажет, правильно ли Вы поняли идею алгоритма;
- изучите дополнительные учебные материалы, рекомендованные преподавателем;

- попытайтесь ответить на контрольные вопросы, которыми, как правило, заканчиваются разделы учебных пособий или учебников;
- если после выполненной работы Вы считаете, что материал освоен не полностью, сформулируйте вопросы и задайте их преподавателю.

Методические указания к ведению конспектов лекций. Лекции по дисциплине проводятся с использованием слайдов. Но это не означает, что лекцию можно просто слушать. Ведение конспектов значительно повышает качество последующей проработки лекционного материала. В силу специфики дисциплины на слайдах лекций очень много алгоритмов, кодов программ, примеров демонстрации работы изучаемых алгоритмов. Но этот материал может быть бесполезен, если Вы не делаете записи в течение лекции, потому что большинстве случаев, комментарии по представленным на слайдах примерам, лектор выполняет в устной форме.

Можно рекомендовать распечатывать слайды перед лекцией и вести конспект непосредственно на бумажном варианте слайд-презентации.

Одной из форм текущего мониторинга уровня знаний по дисциплине являются контрольные работы. Во время изучения дисциплины проводятся контрольные работы двух типов: тестовые опросы на лекции и контрольные работы, в которых студентам необходимо применить полученные знания на практике. Выполнение выше перечисленных действий поможет подготовиться и к выполнению контрольных работ. В приложении 4 указаны темы контрольных работ и приведены примерные варианты.

Самостоятельная работа по подготовке к лабораторным работам по дисциплине состоит в изучении методических материалов по темам соответствующих видов аудиторных занятий.

Рекомендуется перед выполнением лабораторной работы изучить лекционный и методический материал по теме занятия, ознакомиться с алгоритмами, реализацию которых необходимо выполнить во время проведения занятия. Обратите особое внимание на порядок выполнения работы. Поскольку конечным результатом всех лабораторных работ является компьютерная программа, самостоятельно разработайте структурную схему будущей программы, выполните заготовку проекта, подготовьте самостоятельно тестовые данные. Если при подготовке к занятию остались нерешенные вопросы, обратитесь за консультацией к преподавателю.

4.3 Выполнение домашних заданий

4.3.1 Общие положения

Выполнение домашних заданий — это полностью самостоятельный вид деятельности студента. Темами домашних заданий и контрольных работ являются темы дисциплины, не охваченные циклом лабораторных работ и практических занятий.

Выполнение домашнего задания состоит в написании программы по индивидуальному варианту. Обучающиеся самостоятельно разрабатывают алгоритм решения задания, реализуют разработанный алгоритм на языке программирования Си, отлаживают и тестируют написанную программу.

Защита домашнего задания проходит в сроки, установленные преподавателем. Процедура защиты представляет собой индивидуальное собеседование с преподавателем, примерный сценарий которого представлен ниже:

- комментирование студентом кода и логики написанной программы;
- ответы на вопросы преподавателя по коду и логике программы;
- комментирование студентом тестовых данных;
- демонстрация работы программы и пояснение полученных результатов.

4.3.2 Домашнее задание «Подготовка к лабораторной работе «Создание консольного приложения в среде DEV-C++. Ввод-вывод информации»

Домашнее задание состоит в самостоятельном изучении среды программирования DEV-C++. Описание порядка действий при создании проекта представлено в [1] на стр. 28 — 41. При выполнении домашнего задания рекомендуется создать проект на языке Си, написать небольшую программу, выполнить компиляцию и запуск проекта. Выполнение этого домашнего задания поможет качественно и своевременно выполнить первую лабораторную работу дисциплины.

4.3.3 Домашнее задание «Целочисленная арифметика»

Домашнее задание формирует навыки работы с основными типами данных языка Си и знакомит обучающихся со способами вывода информации в языке Си. При выполнении рекомендуется изучить теоретический материал, изложенный в [1], стр. 77 — 78. Алгоритмы разбора целых чисел рассматриваются в [2], стр. 9 — 15.

Примерный вариант.

а) Вывести на экран число *e* (основание натурального логарифма) с точностью до десятых.

б) Дано двузначное число. Найти число единиц числа.

4.3.4 Домашнее задание «Условные алгоритмы»

Домашнее задание формирует навыки работы с конструкцией проверки условия в языке Си. При выполнении рекомендуется изучить теоретический материал, изложенный в [1], стр. 87 — 89 и [7], стр. 14 — 17.

Примерный вариант.

Даны две точки: $A(x_1, y_1)$ и $B(x_2, y_2)$. Напишите программу, определяющую, какая из точек находится ближе к началу координат.

4.3.5 Домашнее задание «Программирование итерационных алгоритмов»

Домашнее задание формирует навыки работы с конструкциями циклов в языке Си. При выполнении рекомендуется изучить теоретический материал, изложенный в [1], стр. 91 — 98.

Примерный вариант.

Дано натуральное n, действительное число x. Вычислить:

$$\sum_{i=1}^{n} \frac{x+\cos(ix)}{2^i}$$

4.3.5 Домашнее задание «Обработка матриц»

Домашнее задание формирует навыки работы с двумерными массивами языка Си. При выполнении рекомендуется изучить теоретический материал, изложенный в [1], стр. 124 — 132.

Примерный вариант.

Напишите программу, заполняющую матрицу nxn (значение n вводить с клавиатуры) по правилу, которое представлено на примере матрицы 7x7:

1	0	0	0	0	0	1
0	1	0	0	0	1	0
0	0	1	0	1	0	0
0	0	0	1	0	0	0
0	0	1	0	1	0	0
0	1	0	0	0	1	0
1	0	0	0	0	0	1

4.3.6 Домашнее задание «Машинное представление графов»

Цель домашнего задания — показать взаимосвязь двух дисциплин, информатики и дискретной математики. Для выполнения домашнего задания будет полезен справочный материал из разделов «Массивы» и «Матрицы» пособия [1], стр. 114 — 132, материал по теории графов пособия [7], стр. 250 — 251 и пособия [8], стр. 6 — 20.

Примерный вариант.

Запишите программу получения матрицы смежности орграфа по заданной матрице инцидентности.

4.3.7 Домашнее задание «Динамические списки»

Цель домашнего задания — формирование навыков работы с динамическими структурами данных. При выполнении задания может быть полезен материал пособия [7], стр. 224 — 228.

Примерный вариант

Пусть L динамический однонаправленный список. Напишите функцию, которая в списке L удаляет все элементы между первым и последним вхождением элемента E, если E входит в L не менее двух раз.

4.4 Подготовка к экзамену

Студенты дневной формы обучения сдают экзамен по дисциплине в первом и втором семестре изучения. Выполнение всех видов самостоятельных работ, лабораторных работ, посещение практических и лекционных занятий — гарантия успешной сдачи экзамена.

Для подготовки к экзамену рекомендуется повторить темы, вынесенные на экзамен. При подготовке обращайтесь не только к конспектам лекций, но и к рекомендованным преподавателем источникам. Организуйте план повторения материала таким образом, чтобы каждый день прорабатывать примерно одинаковый по объему материал. Изучая учебники и учебные пособия, отвечайте на контрольные вопросы. Прорешайте задачи примерного билета. Если после изучения материала Вы не смогли найти ответы на какие-либо вопросы — посетите консультацию перед экзаменом, кроме ответов на вопросы по теме экзамена на консультации освещаются организационные вопросы проведения экзамена время начала экзамена, время проведения экзамена, план проведения экзамена и т.д..

Пример экзаменационного билета (1 семестр)

Билет 1

Цикл `for` в языке Си. Синтаксис. Принцип работы. Запишите с помощью цикла for фрагмент программы, выводящий на экран значения 2.2 2.4 2.6 2.7 2.8 2.10

Запишите алгоритм, проверяющий, есть ли среди цифр заданного произвольного натурального числа цифра 5. Для записи алгоритма используйте блок-диаграмму.

Напишите программу, которая запрашивает с клавиатуры размерность массива, задает элементы массива случайным образом и выводит их на экран. Найдите и выведите на экран значение максимального элемента.

Пример экзаменационного билета (2 семестр)

Билет 1

Написать программу, которая создает динамическую матрицу размерности $A[n \times n]$, и заполняет ее следующим образом:

1	5	9	13
2	6	10	14
3	7	11	15
4	8	12	16

.

Написать функцию поиска суммы положительных элементов массива. В функции main создать три массива X[7], Y[12], Z[100], используя написанную функцию найти суммы положительных элементов всех массивов.

В текстовом файле хранится произвольное количество чисел. Написать программу, которая считывает информацию из файла и находит максимальное число (текстовый файл предварительно создать в блокноте).

В текстовом файле хранится информация о сдаче экзаменов студентов одной группы. Используя структурные переменные, прочитайте информацию из текстового файла и выведите на экран информацию о студентах, первый экзамен у которых сдан на «хорошо» или «отлично». Текстовый файл – students.txt.

Пример экзаменационного билета (3 семестр)

Билет 1

Принципы объектно-ориентированного программирования. Свойство инкапсуляции. Приведите пример инкапсуляции.

Понятие класса в Си++. Конструкторы.

Дописать в класс Matrica

а) перегрузку оператора << вывод матрицы на экран с указанием размерности

б) метод, возвращающий номер столбца с максимальной суммой.

5 Рекомендуемые источники

1. Пермякова, Н. В. Информатика и программирование: Учебное пособие [Электронный ресурс] / Н. В. Пермякова — Томск: ТУСУР, 2016. — 188 с. — Режим доступа: https://edu.tusur.ru/publications/7678 .

2.Златопольский, Д.М. Подготовка к ЕГЭ по информатике. Решение задач по программированию [Электронный ресурс] : учебное пособие / Д.М. Златопольский. — Электрон. дан. — Москва : ДМК Пресс, 2017. — 252 с. — Режим доступа: https://e.lanbook.com/book/100911. — Загл. с экрана.

3. Егоров, И.М. Объектно-ориентированное программирование на C++ [Электронный ресурс] : учебное пособие / И.М. Егоров. — Электрон. дан. — Москва : ТУСУР, 2007. — 180 с. — Режим доступа: https://e.lanbook.com/book/5482. — Загл. с экрана.

4. C++ как второй язык в обучении приемам и технологиям программирования: учебное пособие / Я.М. Русанова. — Ростов-на-Дону: Издательство ЮФУ, 2010. — 200 с. — Режим доступа: http://znanium.com/bookread2.php?book=550811

5. Ашарина, И.В. Объектно-ориентированное программирование в C++: лекции и упражнения [Электронный ресурс] : учебное пособие / И.В. Ашарина. — Электрон. дан. — Москва : Горячая линия-Телеком, 2012. — 320 с. — Режим доступа: https://e.lanbook.com/book/5115. — Загл. с экрана.

6. Образовательный стандарт вуза ОС ТУСУР 01-2013. Работы студенческие по направлениям подготовки и специальностям технического профиля. Общие требования и правила оформления — Томск, ТУСУР, 2013. — 57 с. Режим доступа: https://regulations.tusur.ru/documents/70 .

7. Потопахин, В. Искусство алгоритмизации [Электронный ресурс] / В. Потопахин. — Электрон. дан. — Москва : ДМК Пресс, 2011. — 320 с. — Режим доступа: https://e.lanbook.com/book/1269. — Загл. с экрана.

8. Асанов, М.О. Дискретная математика: графы, матроиды, алгоритмы [Электронный ресурс] : учебное пособие / М.О. Асанов, В.А. Баранский, В.В. Расин. — Электрон. дан. — Санкт-Петербург : Лань, 2010. — 368 с. — Режим доступа: https://e.lanbook.com/book/536. — Загл. с экрана.

ПРИЛОЖЕНИЕ 1

Описание класса Vector

```
class VECTOR
    {
    protected :
     int n; // количество элементов массива
     int *vector; // массив целочисленных элементов
    public :

     VECTOR(int size, const flag = 0);
     /* конструктор, создает объект массив,
     размерностью size, константный параметр
     flag задает способ за-полнения массива:
     значение по умолчанию, равное 0  - элементы
     массива задаются случайным образом, при
     других значениях элементы массива задаются
     с клавиатуры */
     void ENTER(int number);
     /* метод, осуществляющий ввод нового
     элемента в позицию number*/
     void PRINT_ALL();
     /*  метод печати элементов массива */
     int DELETE(int number);
     /* метод удаления элемента с номером
     number*/
     int GETN();
     /* метод, возвращающий
     количество элементов массива */
     int GETELEM(int number);
     /*метод, возвращающий значение
     элемента с номером number*/
     void SAVEFILE(const char* filename);
     /*метод, сохраняющий элементы массива
     в текстовом файле с именем filename*/
     ~VECTOR();
     /* деструктор класса*/
    };
```

ПРИЛОЖЕНИЕ 2

Массивы объектов

```
#include <iostream>
#include <cstdlib>
using namespace std;
// Описание класса Pixel
class Pixel{
    private:
        int x,y;
    public:
    Pixel(){
    };
    Pixel(int x,int y)  {
        this->x = x;
        this->y = y;
    }
    void Show(){
        cout << "Pixel (" << x << ","
                          << y<< ")" << endl;
    }
    void PutX(int x){
        this->x = x;
    }
    void PutY(int y){
        this->y = y;
    }
};
```

```
int main(int argc, char** argv) {
    // Создание статического массива статических объектов
    Pixel A[5];
    for(int i=0;i<5;i++){
        A[i].PutX(rand()%20);
        A[i].PutY(rand()%20);
    }
    for(int i=0;i<5;i++)
        A[i].Show();
    // Создание статического массива динамических  объектов
    Pixel * B[5];
    cout << "Array B" << endl;
    for (int i=0;i<5;i++) {
        B[i] = new Pixel(rand()%20,rand()%20);
        B[i]->Show();
    }
    // Создание динамического массива статических объектов
    Pixel *C;
    int n;
    cout << "Enter n: ";
    cin >> n;
    C = new Pixel[n];
    for(int i=0;i<n;i++){
        C[i].PutX(rand()%20);
        C[i].PutY(rand()%20);}
        cout << "Array C" << endl;
    for (int i=0;i<n;i++) {
        C[i].Show();
    }
```

```
    // Создание динамического массива динамических объектов
    Pixel ** D;
    int n1;
    cout << "Enter n1: ";
    cin >> n1;
    D = new Pixel*[n1];
    for(int i=0;i<n1;i++){
        D[i] = new Pixel(rand()%20,rand()%20);
    }
        cout << "Array D" << endl;
    for (int i=0;i<n1;i++) {
        D[i]->Show();
    }
    // Освобождение выделенной памяти
    for (int i=0;i<5;i++)
        delete B[i];
    delete [] C;
    for (int i=0;i<n1;i++)
        delete D[i];
    delete [] D;
    return 0;
}
```

ПРИЛОЖЕНИЕ 3

Форма листа задания на курсовую работу

Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
**«ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ» (ТУСУР)**

Кафедра автоматизации обработки информации (АОИ)

УТВЕРЖДАЮ
Зав. кафедрой АОИ
д-р техн. наук, проф.
_____Ю.П. Ехлаков
«___»_____20__г.

ЗАДАНИЕ
на курсовую работу

студенту __*Петрову Андрею Васильевичу*_____

группа ___*405*_____факультет _____*СУ*_____

1. Задание __*Реализовать методы численного интегрирования — метод трапеций и метод прямоугольников. Сравнить методы по эффективности.*_____

2. Дата выдачи задания __*07.09.2016*_____

3. Исходные данные к проекту__*алгоритмы методов, тестовые функции для исследования эффективности.*_____

4. Содержание отчета (перечень подлежащих разработке вопросов)__*обзор методов численного интегрирования, описание реализуемых методов, алгоритмы реализуемых методов, описание результатов выполнения программы на тестовых данных, анализ эффективности реализованных методов.*_____

5. Срок сдачи законченного задания_*28.12.2016*_____

Руководитель _____

*старший преподаватель кафедры АОИ Пермякова Наталья Викторовна*_____

(должность, место работы, фамилия, имя, отчество)

ПРИЛОЖЕНИЕ 4

Темы контрольных работ

Темы и примерные варианты контрольных работ

1. Синтаксис и алфавит языка Си

Вариант 1 _____		

1. Выберите тип передачи управления, использующийся в структурном программировании:
• безусловная передача
• условная передача
• функционально-зависимая передача
	Посчитайте количество лексем в представленном фрагменте программы:
	`float x,y,z;`
	`printf(" -->");`
		Выберите ключевые слова Си:
		• `if`
		• `while`
		• `main`
		• `factorial`
		• `integer`

2. Основные типы данных. Условный оператор

Вариант 1 _____		

Опишите переменную x как указатель на тип `float`.	Что будет храниться по адресу y, если выполниться фрагмент программы:
	`int *y;`
	`int k = 12;`
	`y = &k;`
		Что будет выведено на экран при выполнении следующего фрагмента программы:
		`int x = 7;`
		`int y = 9;`
		`int z = 0;`
		`if (x>y) {`
		`z = y*2;`
		`y = x*4; }`
		`else { z = x*2;`
		`x = y+x;}`
		`printf (“ %d %d %d”, x,y,z);`

3. Циклы в языке Си

<table>
<tr><td colspan="3">Вариант 1 Фамилия _____ гр____</td></tr>
<tr><td>1. Используя цикл <code>while</code>, запишите фрагмент программы, который выводит на экран числа 2 5 8 11 14 17 20. Описание использованных переменных обязательно.</td><td>2. Что будет выведено на экран при выполнении следующего фрагмента программы:
<code>int i = 25;
do{
printf("%3d",i);
i-=2; }
while(i>=13);</code></td><td>3. Запишите фрагмент программы, решающей следующую задачу (используйте цикл <code>for</code>):

Вывести на экран числа от 0 до 12 с шагом 0.25. Фрагмент обязательно должен содержать описания использованных переменных.</td></tr>
</table>

4. Массивы в языке Си

Вариант 1.

С клавиатуры задается размерность массива. Элементы массива считываются с клавиатуры. Найти минимальный элемент массива.

С клавиатуры задается размерность массива. Элементы массива задаются случайным образом. Найти количество пар элементов, таких, что x[i] > x[i+1].

С клавиатуры задается размерность массива. Элементы массива задаются случайным образом. Поменять первый и последний элементы массива.

5. Функции

Напишите функцию, аргументами которой являются два целых числа a и b, вычисляющую значение $f = \begin{cases} 0, при\ a = b \\ 1 - \sin(a^2 + b^2), при\ a > b \\ \cos(a^2 + b^2) - 1, при\ a < b \end{cases}$

Напишите функцию, которая в целочисленном массиве X размерности n ищет количество пар элементов, таких, что x[i]>x[i+1].

Напишите функцию, которая в целочисленной матрице X размерности nxm ищет номер первой строки, в которой есть хотя бы один нулевой элемент.

6. Введение в объектно-ориентированное программирование

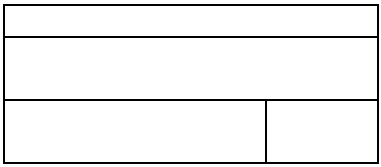
Существует описание классов Line,Str,Display1, реализующих экранную форму 1.

```
typedef  char Stroka [25];
   class Line {
//горизонтальные и
//вертикальные линии
        public:
        int x1,y1,x2,y2,color;
        Line (int xx1, int yy1,
          int xx2, int yy2, int col);
//x1,y1 - начальные
//координаты линии;
 //x2,y2-конечные координаты
//линии;
//color-цвет линии
        void Show();
 //линии изображаются
//в текстовом режиме с
//использованием
//символов псевдографики
        }
```

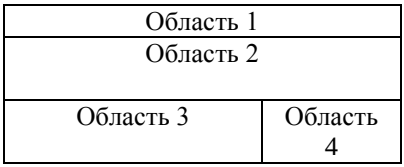
```
class  Str  { // строка символов
          int x,y,orient,color;
          Stroka int;
          Str(int x1, int y1, int or,
              int col;Stroka st1);
//x,y-координаты начала строки;
// orient-ориентация
//(0-горизонтальная
//1-вертикальная);
//color- цвет выводимых символов
          void Show();
          }
class Display1 { //экранная
                  //форма 1
          int ColorFon, ColorText,
              ColorRam;
Display1(int CFon, int CText,
            int CRam);
//ColorFon -цвет фона;
// ColorText-цвет текста;
//ColorRam-цвет рамок
void Show();
          }
```

Написать стартовый класс, изображающий экранную форму 2.

1



2



7. Реализация класса

Вариант 1 Фамилия _____ гр _____

Дан код:

```
class A {
int b;
public:
A(int i) {b =
i;}
A() {b = 111;}
void Show(){
cout<<b;}
}
```

```
void main(){
A* obj = new
A();
obj -
>show();
}
```

Что произойдет при компиляции и выполнении?

1. напечатается 0
2. напечатается 1
3. напечатается 111
4. ошибка компиляции
5. ошибка выполнения

8. Конструктор копирования

Вариант 1 Фамилия _____ гр _____

Класс для работы со строками описан следующим образом:

```
class String{
private:  char* str;
// строка объекта
public: String(char str1)
// конструктор
{ n = strlen (str1);
str = new char [n+1];
strcpy (str,str1);
void Print();
// вывод строки на экран
void Print_XY(int X, int Y);
// вывод строки в заданном //месте
экрана
~String();


    }
```

Дописать в класс конструктор копирования (внести все необходимые изменения в описание класса и написать сам конструктор копирования).

9. Форматирование ввода-вывода в Си++

Вариант 1 Фамилия _____ гр _____
Что будет выведено на экран при выполнении следующего фрагмента программы?

```
float m;
 for(int i=0;i<3;i++)
   {
   for(int j=0;j<3;j++)
     { m = i*j*1.;
          cout<<setw(4)<<setfill(':');
          cout << m;
     }
   cout << endl;
   }
```