**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 «Программная инженерия»
(уровень бакалавриата) заочной формы обучения

2018

**Пермякова Наталья Викторовна**
Информатика и программирование: Методические указания к лабораторным работам, курсовой работе и организации самостоятельной работы для студентов направления «Программная инженерия» (уровень бакалавриата) заочной формы обучения / Н.В. Пермякова. — Томск, 2018. — 54 с.

# Содержание

# 1 Введение

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

**Целью** проведения лабораторных работ, практических занятий и организации самостоятельной работы является формирование и развитие навыков структурного программирования.

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

— **знать** основные факты, концепции, принципы и теории, связанные с информатикой; основные принципы и конструкции структурного программирования; 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;

— **владеть навыками** реализации и отладки программ на алгоритмических языках программирования.

Цикл лабораторных работ по дисциплине направлен на закрепление навыков использования конструкций структурного программирования и навыков разработки программ на языке Си.

Самостоятельная работа студентов по дисциплине содержит несколько видов деятельности — проработка лекционного материала, подготовка и выполнение контрольных работ, самостоятельное изучение тем, подготовка к зачету и экзамену, выполнение курсовой работы.

Все виды самостоятельной активности студентов тесно взаимосвязаны. Проработка лекционного материала и тем, вынесенных на самостоятельное изучение, подразумевает изучение законспектированного материала и одновременно является подготовкой к выполнению предусмотренных учебным планом контрольных работ. В свою очередь, знания и практические навыки, полученные при выполнении лабораторных и контрольных работ, создают необходимый базовый уровень знаний для выполнения курсовой работы.

# 2 Методические указания к проведению лабораторных работ

## 2.1 Общие положения

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

Основной формой проведения лабораторных работ является разработка алгоритма решения индивидуальной задачи и его программная реализация на языке 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 организация входного контроля знаний студентов по теоретическому материалу дисциплины, практическое применение которого осуществляется в ходе выполнения лабораторной работы и индивидуальная защита выполненной работы.

Для успешной защиты лабораторной работы 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 получения зачета по дисциплине и допуском к итоговой форме контроля — 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;`<br>`float y;` |
| 3.2 | Ввод данных с клавиатуры | `printf("Введите значение переменной x: ");`<br>`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.

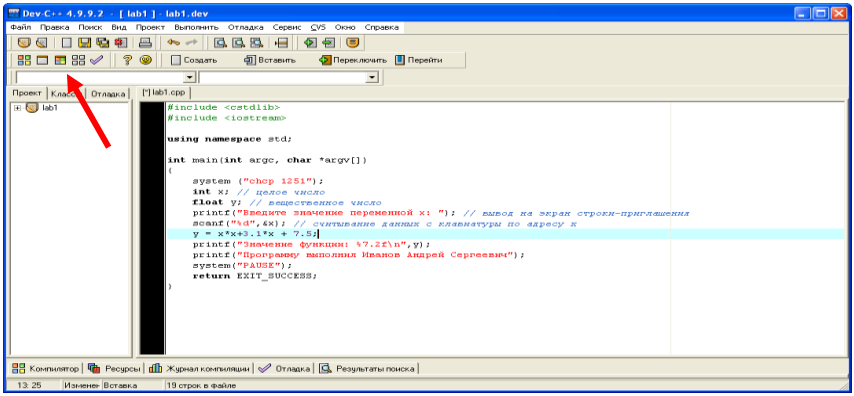



Рисунок 1 — Проект выполнения индивидуального задания

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

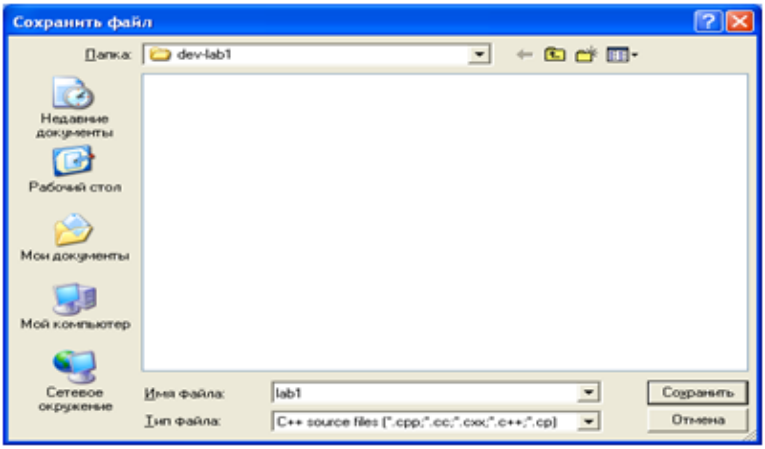



Рисунок 2 — Определение имени файла

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.

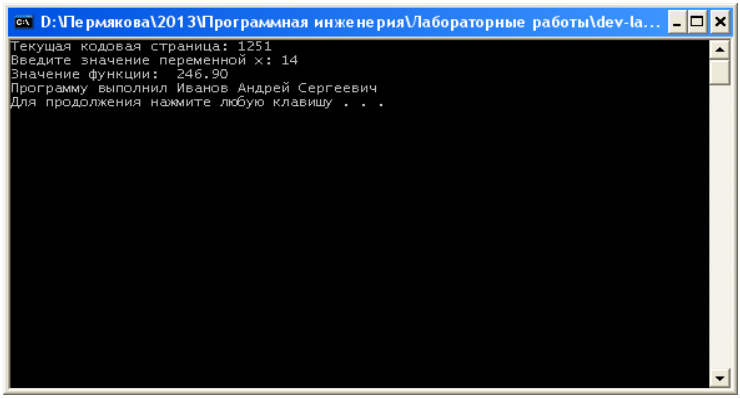



Рисунок 3 — Тестирование программы при $x$ =14

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;<br>$z - x = 0$— ошибка деления на 0;<br>$x \leq 0$— ошибка извлечения квадратного корня, ошибка деления на 0;<br>$x - \frac{y}{\sqrt{x}} < 0$— ошибка извлечения корня четвертой степени<br>$a = 0$ — ошибка деления на 0;<br>$a = z$— ошибка деления на 0. |
| 3. | Алгоритм | алг Лабораторная работа 2 нач<br>ввод x,y,z<br>если x или y или z — не число, то<br>рез “Введены не числа”<br>если z-x=0 или x = 0, то<br>рез “Ошибка деления на 0”<br>если$x - \frac{y}{\sqrt{x}} < 0$или $x < 0$ то<br>рез “Ошибка извлечения корня”<br>$a = (\frac{2y}{z-x} - \lvert x \rvert)(\sqrt[4]{x - \frac{y}{\sqrt{x}}})$<br>рез a<br>если a = 0 или a = z то<br>рез “Ошибка деления на 0”<br>$b = a - \frac{x}{2a} + \frac{x^2}{a-z}$<br>рез b<br>конец |
| 4. | Составление программы | Для использования математических функций и констант подключите библиотеку математических функций:<br>`#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 кла- | `printf(“Введите значения переменных: ”);` |

| | | |
|---|---|---|
| | виатуры | `int m = scanf(“%f%f%f”,&x,&y,&z);` |
| 4.3. | Проверка входных данных | `if (m!=3) {`<br>`printf (“Введены не числа\n”);`<br>`system(“pause”);`<br>`return 0;}`<br>`if (x==0 ||z-x==0) {`<br>`printf (“Ошибка деления на 0\n”);`<br>`system(“pause”);`<br>`return 0;}`<br>`float temp = x – y/sqrt(x);`<br>`if (temp<0||x<0) {`<br>`printf (“Ошибка извлечения корня\n”);`<br>`system(“pause”);`<br>`return 0;}` |
| 4.4 | Вычисление значения функции | `a = 2*y/(z-x)-fabs(x);`<br>`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) {`<br>`printf (“Ошибка деления на 0\n”);`<br>`system(“pause”);`<br>`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: $y_3 = \mathrm{a}x_3 + b; y_2 > \mathrm{a}x_2 + b; y_1 < \mathrm{a}x_1 + b$.

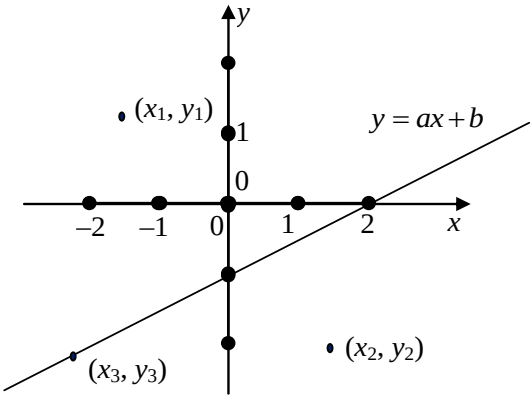



Рисунок 4 — Расположение точки относительно прямой

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:

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

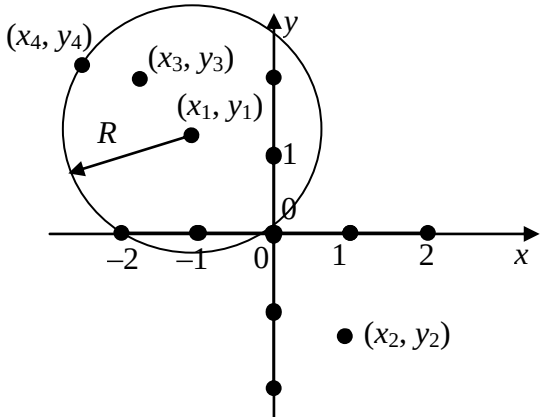



Рисунок 5 — Расположение точки относительно окружности

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.

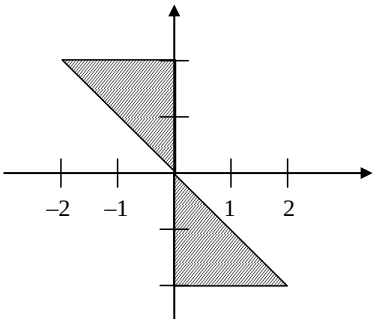



Рисунок 6 — Пример 1

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 $(1 > (x - 1)^2 + (y - 1)^2)$.

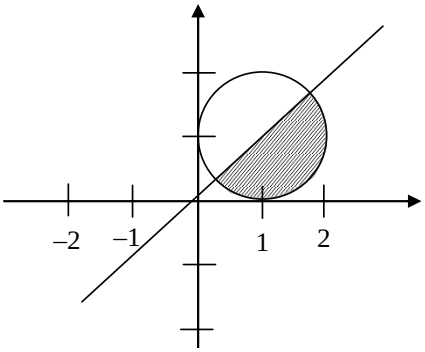



Рисунок 7 — Пример 2

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 «Вычисление суммы бесконечного ряда»

**Цель работы:** закрепление навыков программирования итерационных процессов.

**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 конструкций циклов в языке Си (стр. 91 — 95) в главе 6 пособия. Математические основы вычисления суммы бесконечного ряда и логика процесса вычисления суммы бесконечного ряда описана на стр. 95 — 97.

**Порядок выполнения работы**

1. Получить индивидуальный вариант.
2. Если необходимо, преобразовать исходную формулу ряда в итерационную формулу.
3. Написать и протестировать программу вычисления суммы бесконечного ряда.
4. Защитить работу.

## 2.6 Лабораторная работа «Обработка статического одномерного массива»

**Цель работы:** закрепление навыков обработки статического одномерного массива.

**Форма проведения:** выполнение индивидуального задания.

**Рекомендации по подготовке к лабораторной работе:** для выполнения лабораторной работы необходимо изучить теоретический материал, изложенный в [1]. Справочная информация об организации работы с массивами описана в главе 8 пособия (стр. 114 — 120). Графические способы представления алгоритмов описываются в главе 1 (стр. 15 — 20).

**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**

Вводится последовательность из n целых чисел. Сохранить все введенные значения в массиве и найти сумму всех отрицательных чисел (*индивидуальное задание*).

Блок-диаграмма задачи представлена на рисунке 8.

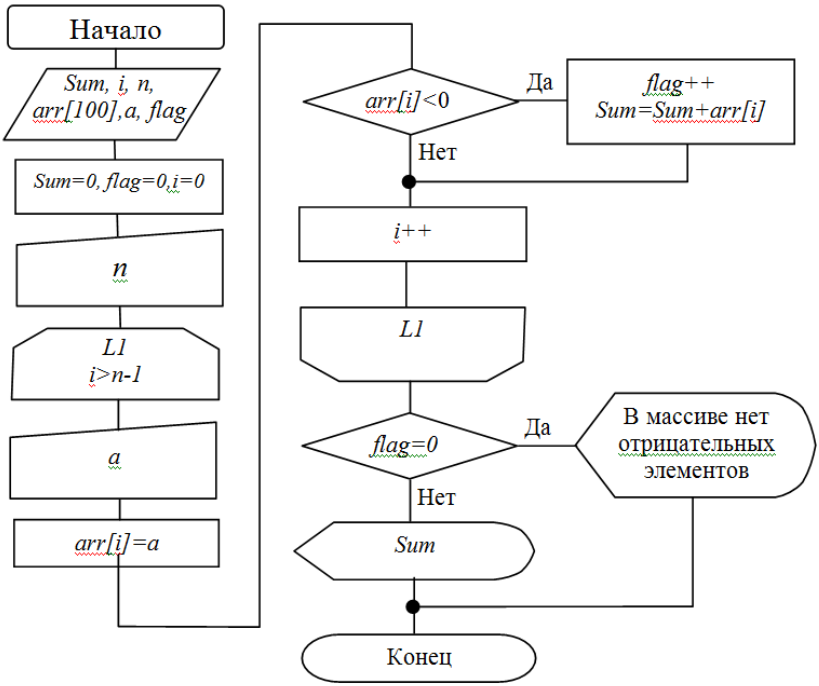



Рисунок 8 — Алгоритм поиска суммы отрицательных элементов

Реализация задания на языке Си может быть выполнена следующим образом:

```
   int main(int argc, char *argv[])
    {system("chcp 1251");
     int Sum=0, a, n, arr[100];
     int flag = 0;
     printf("Введите количество членов последовательности
не более 100 ");
     scanf("%d",&n);
     for(int i=0; i<n; i++){
     printf("Введите значение члена последовательности");
     scanf("%d", &a);
      arr[i]=a;
      if (arr[i]<0)   {Sum+=arr[i]; flag++;} }
   if (!flag) printf("В массиве нет отрицательных элемен-
тов\n");
```

```
    else
    printf("Значение суммы отрицательный членов последова-
тельности равна %6d", Sum);}
```

## 2.7 Лабораторная работа «Обработка двумерных массивов»

**Цель работы:** изучение способов работы с динамическими двумерными массивами; реализация простых алгоритмов сортировки и 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 двумерными массивами в языке Си (стр. 114 — 120), способы сортировки наборов данных рассмотрены (стр. 121 — 123). Руководство по использованию встроенного отладчика в IDE описано в главе 2 (стр. 37 — 42).

**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 матрицей [5x5].

## 2.8 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.

## 2.9 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.

## 2.10 Лабораторная работа «Линейные динамические списки»

**Цель работы:** формирование навыков хранения обрабатываемой информации в динамических линейных списках.

**Форма проведения:** выполнение индивидуального задания.

**Подготовка к выполнению лабораторной работы**

Для выполнения лабораторной работы необходимо изучить теоретический материал, изложенный ниже.

**Динамические структуры данных**

При решении прикладных задач часто приходится использовать данные, размер и структура которых должны меняться в процессе работы. В этом случае значение объема выделяемой памяти выясняется только в процессе работы. В таких случаях применяют данные, которые представляют собой отдельные элементы, связанные с помощью ссылок.

Каждый элемент (узел) состоит из двух областей памяти: информационного поля и ссылок (рис. 9).

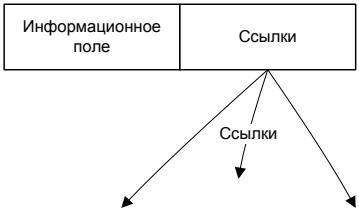



Рисунок 9 — Структура узла списка

Ссылочные данные хранят адреса других узлов этого же типа. В языке Си для организации ссылок используются переменные-указатели. При добавлении нового узла в такую структуру выделяется новый блок

памяти и устанавливаются связи этого элемента с уже существующими. Для обозначения конечного элемента в списке используются нулевой указатель — NULL.

**Линейный список**

В простейшем случае каждый узел содержит всего одну ссылку на следующий элемент. У последнего в списке элемента поле ссылки содержит нулевой указатель NULL. Чтобы не потерять информацию о списке, необходимо организовать хранение адрес первого узла списка в отдельной переменной. Первый узел списка будем называть «головой» списка (рис. 10).

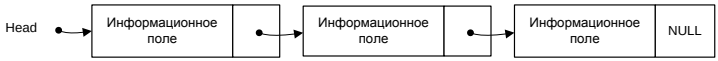



Рисунок 10 — Структура линейного списка

Для программной реализации необходимо объявить два новых типа данных — узел списка *Node* и указатель на него *PNode*. Узел представляет собой структуру, которая содержит два поля — целое число и указатель на такой же узел. Правилами языка Си допускается объявление:

```
struct Node {

int count; // информационное поле
Node *next;  // ссылка на следующий узел
};

typedef  Node *PNode; // тип данных: указатель на узел
```

В дальнейших приведенных примерах указатель *Head* указывает на начало списка, то есть, объявлен в виде: `PNode Head = NULL.` Такое определение первого элемента говорит о том, что в начале работы список пуст.

**Создание элемента списка**

Для добавления узла к списку необходимо выделить память под хранение узла и запомнить адрес выделенного блока памяти. Функция, создающая новый узел в памяти и возвращает его адрес может быть реализована следующим образом:

```
PNode CreateNode (int NewData)
{
PNode NewNode = new Node;  // указатель на новый узел
```

```
NewNode->count = NewData; // запись информационного поля
                          //нового узла
NewNode->next = NULL; // следующего узла нет
return NewNode; // результат функции — адрес созданного узла
}
```

После выполнения этой функции, узел необходимо добавить к списку (в начало, в конец или в середину).

**Добавление узла**

***Добавление узла в начало списка***

При добавлении нового узла `NewNode` в начало списка ссылка узла `NewNode` устанавливается на голову существующего списка (рис. 11)

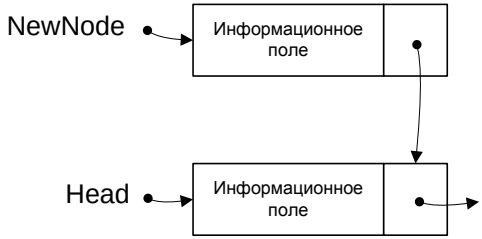



Рисунок 11 — Связь нового узла с «головой» списка

и изменяется переменная, в которой хранится «голова» (рис. 12).

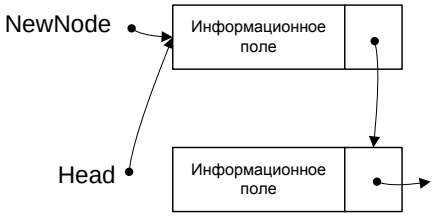



Рисунок 12 — Изменение адреса «головы» списка

По такой схеме работает функция `AddFirst`. Предполагается, что адрес начала списка хранится в `Head`. Обратите внимание, что здесь и далее адрес начала списка передается по ссылке, так как при добавлении нового узла он изменяется внутри процедуры.

```
void AddFirst (PNode *Head, PNode NewNode)
{
NewNode->next = Head;
*Head = NewNode;
}
```

***Добавление узла после заданного***

Предположим, что по условию задачи узел `NewNode` необходимо вставить после узла с заданным адресом `p`. Такая операция выполняется в два этапа:

1) установить ссылку нового узла на узел, следующий за данным;
2) установить ссылку данного узла `p` на `NewNode` (рис. 13).

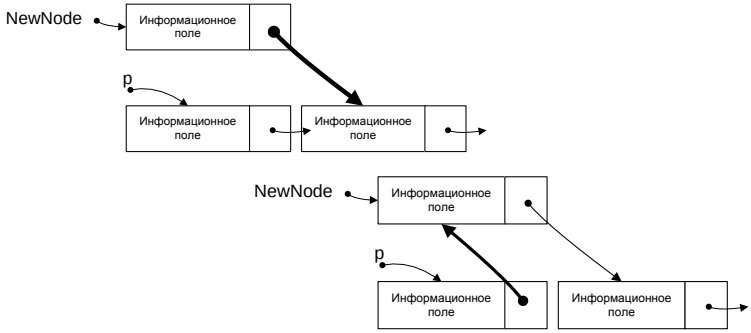



Рисунок 13 — Добавление узла перед заданным

```
void AddAfter (PNode p, PNode NewNode)
{
NewNode->next = p->next;
p->next = NewNode;
}
```

***Добавление узла перед заданным***

Такая схема добавления самая сложная. Проблема заключается в том, что в простейшем линейном списке (он называется односвязным, потому что связи направлены только в одну сторону) для того, чтобы получить адрес предыдущего узла, нужно пройти весь список сначала.

Задача сводится либо к вставке узла в начало списка (если заданный узел — первый), либо к вставке после заданного узла.

```
void AddBefore(PNode *Head, PNode p, PNode NewNode)
{
PNode q = *Head;
if (Head == p) {
AddFirst(Head, NewNode); // вставка перед первым узлом
return;
}
while (q && q->next!=p) // ищем узел, за которым следует p
q = q->next;
if (q) // если нашли такой узел,
AddAfter(q, NewNode); // добавить новый после него
}
```

***Добавление узла в конец списка***

Для решения задачи необходимо сначала найти последний узел, а затем воспользоваться процедурой вставки после заданного узла. Отдельно необходимо обработать случай, когда список пуст.

```
void AddLast(PNode *Head, PNode NewNode)
{
PNode q = *Head;
if (*Head == NULL) { // если список пуст,
AddFirst(Head, NewNode); // вставляем первый элемент
return;
}
while (q->next) q = q->next; // ищем последний элемент
AddAfter(q, NewNode);}
```

**Проход по списку**

Для обхода всего списка необходимо организовать проход с «головы» и, используя указатель `next`, продвигаться к следующему узлу.

```
PNode p = Head; // начали с головы списка
while ( p != NULL ) { // пока не дошли до конца
p = p->next; // переходим к следующему узлу
}
```

**Поиск узла в списке**

Часто требуется найти в списке нужный элемент (его адрес или данные). Надо учесть, что требуемого элемента может и не быть, тогда про-

смотр заканчивается при достижении конца списка. Такой подход приводит к следующему алгоритму:

1) начать с головы списка;

2) пока текущий элемент существует (указатель — не равен NULL), проверить нужное условие и перейти к следующему элементу;

3) закончить, когда найден требуемый элемент или все элементы списка просмотрены.

Например, следующая функция ищет в списке элемент, соответствующий заданному значению (для которого поле count совпадает с заданным значением `NewNode`), и возвращает его адрес или NULL, если такого узла нет.

```
PNode Find (PNode Head, char NewWord[])
{
PNode q = Head;
while (q && strcmp(q->word, NewWord))
q = q->next;
return q;
}
```

**Удаление узла**

Эта процедура связана с поиском заданного узла по всему списку, так как необходимо поменять ссылку у предыдущего узла, а перейти к нему непосредственно невозможно. Если найден узел, за которым идет удаляемый узел, необходимо просто переставить ссылку (рис. 14).

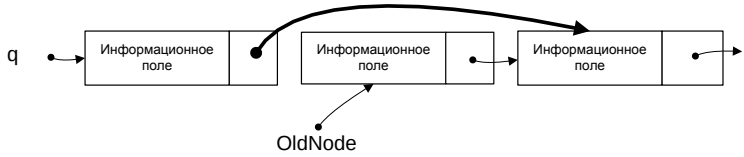



Рисунок 14 — Удаление узла

Отдельно обрабатывается случай, когда удаляется первый элемент списка.

```
void DeleteNode(PNode *Head, PNode OldNode)
{
PNode q = Head;
if (*Head == OldNode)
```

```
Head = OldNode->next; // удаляем первый элемент
else {
while (q && q->next != OldNode) // ищем элемент
q = q->next;
if ( q == NULL ) return; // если не нашли, выход
q->next = OldNode->next;
}
delete OldNode; // освобождаем память
}
```

**Порядок выполнения работы**

1. Самостоятельно разработать алгоритм индивидуального варианта.
2. Реализовать построенный алгоритм на языке Си.
3. Отладить и протестировать программу.
4. Защитить работу.

# 3 Методические указания к выполнению курсовой работы

## 3.1 Общие положения

Курсовая работа является завершающим этапом в изучении дисциплины. Выполнение курсовой работы должно способствовать закреплению теоретических знаний, полученных во время изучения дисциплины и формирования навыков самостоятельной разработки программного продукта в соответствии с принципами структурного программирования, рассмотренными в процессе изучения дисциплины.

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. Задания 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 консультативную 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

Во время выполнения курсовой работы формируются практические навыки использования теоретического материала по следующим темам дисциплины:

- сложные и производные типы данных;
- динамические массивы;
- текстовые файлы;
- функции;
- базовые алгоритмы.

**Примеры заданий на курсовую работу**

Напишите программу, реализующую систему меню для выполнения заданий индивидуального варианта.

**Вариант 1**

Исходные данные хранятся в файле:

| Фамилия | Группа | Оценки | | | |
|---|---|---|---|---|---|
| | | Физика | Операционные системы | Математика | Программирование |

Реализовать возможность формирования следующих запросов информации:

- список студентов с задолженностями, упорядоченный по убыванию количества несданных экзаменов,
- список студентов заданной группы, сдавших экзамены без троек;
- результаты сессии по запрошенной фамилии;
- результаты сессии по запрошенной группе;
- выборка студентов, сдавших заданный экзамен на определенную оценку.

**Вариант 2**

Сведения о парке легковых автомобилей города хранятся в текстовом файле:

| Номер | Цвет | Марка | ФИО | | |
|---|---|---|---|---|---|
| | | | Фамилия | Имя | Отчество |

Реализовать возможность формирования следующих запросов информации:

- список владельцев по заданной марке автомобиля и цвету;
- список автомобилей с заданной цифрой в номере;
- список владельцев автомобилей заданной марки;
- информация об автомобиле по заданной фамилии владельца;
- список владельцев автомобилей заданного цвета.

**Вариант 3**

Исходная информация о штатах поликлиник города хранится в текстовом файле.

| Номер | Штаты | | | | | | |
|---|---|---|---|---|---|---|---|
| | Хирург | Окулист | Невролог | Лор | Кардиолог | Эндокринолог | Терапевт |

Каждая поликлиника города должна быть укомплектована врачами различной специализации (окулист, хирург, невропатолог и т.д.), по одному врачу каждой специальности.

Реализовать возможность формирования следующих запросов информации:

- список поликлиник с укомплектованным штатом;
- список фамилий заданных узких специалистов с указанием номеров поликлиник;

- список поликлиник с наличием заданного узкого специалиста с указанием фамилии узкого специалиста;
- номера поликлиник, сотрудником которых является заданный узкий специалист с заданной фамилией;
- список поликлиник с неукомплектованным штатом, отсортированный по убыванию показателя неукомплектованности.

**Вариант 4**

Исходные данные хранятся в файле:

| Пункт назначения | № поезда | Время отправления | Стоимость билета | Общее количество мест | Проданное количество билетов |
|---|---|---|---|---|---|

Реализовать возможность формирования следующих запросов информации:

- список поездов до заданного пункта назначения;
- список поездов, цены на билеты которых, находятся в заданном ценовом интервале;
- список поездов по заданному интервалу времени отправления, отсортированный по убыванию количества свободных мест;
- количество свободных мест поезда с заданным номером;
- наличие свободных мест на поезда с заданным пунктом назначения.

**Вариант 5**

Исходные данные хранятся в файле:

| Страна | Наименование товара | Объем экспорта |
|---|---|---|

Реализовать возможность формирования следующих запросов информации:

- список стран по заданному наименованию товара и объему экспорта, не меньшему, чем заданный;
- список стран по заданному наименованию товара, отсортированный по возрастанию объема экспорта;
- список стран, занимающих *n* первых мест по объему экспорта;
- список наименований товаров, упорядоченных по алфавиту, экспортирующихся из заданной страны;

- список стран, одновременно занимающихся экспортом товаров с заданными наименованиями.

**Вариант 6**

Исходные данные хранятся в файле:

| Страна | Дата | Наличие путевок | Цена |
|---|---|---|---|

Реализовать возможность формирования следующих запросов информации:

- список стран по заданной дате и приемлемой цене (при наличии путевок);
- список туристических предложений по наличии путевок на заданную дату, отсортированный по возрастанию цены;
- список дат туристических предложений по заданной стране, упорядоченный по возрастанию дат;
- список стран, упорядоченный по алфавиту, в которые нет возможности купить путевку;
- список цен, упорядоченный по возрастанию, по заданной стране.

**Вариант 7**

Исходные данные хранятся в файле:

| Наименование товара | Цена | Срок реализации |
|---|---|---|

Реализовать возможность формирования следующих запросов информации:

- список наименований товара, срок реализации которого истекает через неделю или менее;
- список наименований товара с непросроченным сроком реализации, упорядоченный по возрастанию цены;
- список из *n* товаров, имеющих максимальные цены;
- список наименований товаров, упорядоченных по алфавиту, с просроченным сроком реализации;
- список наименований товаров, с заданным сроком реализации.

**Вариант 8**
Исходные данные хранятся в файле:

| Наименование препарата | Цена | Отметка о наличии | Список заменителей |
|---|---|---|---|

Реализовать возможность формирования следующих запросов информации:

- сведения о наличии заданного препарата, при его отсутствии — список заменителей, имеющихся в наличии;
- список имеющихся в наличии препаратов;
- сведения о цене заданного препарата и его возможных заменителей.

**Вариант 9**
Исходные данные хранятся в файле

| Фамилия | Город | Адрес | | |
|---|---|---|---|---|
| | | Улица | Дом | Квартира |

Реализовать возможность формирования следующих запросов информации:

- список жителей, живущих в разных городах по одному адресу;
- список жителей, проживающих на одной улице;
- адрес, по заданной фамилии;
- список жителей, проживающих в одном доме с заданным жителем, упорядоченный по алфавиту;
- список однофамильцев, проживающих в одном городе.

**Вариант 10**
Исходные данные хранятся в файле

| Фамилия | Пол | Дата рождения | | |
|---|---|---|---|---|
| | | Число | Месяц | Год |

Реализовать возможность формирования следующих запросов информации:

- список людей, старше заданного возраста;
- список людей с заданным днем рождения;
- список всех мужчин, родившихся зимой;
- фамилию самой младшей женщины;

- список однофамильцев, родившихся в один день.

## 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 1 приведен пример выполнения данного этапа для одного из вариантов курсовой работы.

**Разработка алгоритма решения**

На втором этапе требуется описать алгоритм работы будущего программного продукта. Алгоритм может быть представлен блок-диаграммой или записан с использованием псевдокода. В приложении 2 приведен пример выполнения этого этапа.

**Разработка прототипа программы**

На третьем этапе необходимо разработать прототип (примерные экранные формы) будущей программы. При выполнении этого этапа нужно спроектировать вид основного меню программы и описать все варианты меню второго уровня, необходимых для решения задачи, сформулированной в задании на курсовую работу. Для представления прототипа программы можно воспользоваться средствами рисования LibreOffice или Microsoft Office Visio.

**Определение формы представления входных и выходных данных**

Согласно поставленной задаче необходимо определить и описать структуру данных, которые будут обрабатываться в программе. Для описанного в приложении 1 варианта курсовой работы может быть предложен следующий вид входных данных:

```
typedef struct {char surname [30];
                char name[30];
                char patronymic[30];
                } fio;
typdef struct{  fio Name;
                int physics;
                int math;
```

```
                int comp_sc;
                } Abiturient;
Abiturient A[100];
```

Выходные данные для всех решаемых задач будут иметь аналогичную структуру. Выборки данных, полученные после выполнения задач, будут сохранены в текстовых файлах.

**Определение функциональной структуры программы**

На этом этапе работы необходимо определить основные функции разрабатываемой программы и выполнить их описание в виде:

<Возвращаемое значение> имя функции <Входные параметры>.
<Краткое описание работы функции>

Например:

`Abiturient * Poisk1 (Abiturient* X, int n)` — функция, определяющая абитуриентов, общий балл которых выше среднего. Входные параметры функции — массив `X`, содержащий информацию обо всех абитуриентах, `n` — размерность массива.

`void WriteToFile(Abiturient X, int n, char *FileName )` — функция, сохраняющая массив, содержащий информацию об абитуриентов размерности `n` в текстовый файл с именем `FileName`.

**Программная реализация**

На этом этапе выполняется программная реализация поставленной задачи — кодирование описанных выше алгоритма решения и функций программы. Этап программной реализации включает в себя разработку кода программы и ее тестирование.

**Подготовка тестовых данных**

Для тестирования программы создается текстовый файл, содержащий не менее 20 записей, соответствующих поставленной задаче. Для рассматриваемого примера необходимо создать файл с данными об абитуриентах. При этом могут быть созданы файлы, использование которых приведет к исключительным ситуациям, например, неверно заполненные или пустые файлы.

**Разработка программной документации**

***Описание структуры программы***

На этом этапе описывается взаимодействие ранее описанных функций программы. Фрагмент структуры программы приведен в приложении 3.

***Руководство пользователя***

Описание правил пользования программным средством и правил формирования входных данных.

**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[2]. Шаблон листа задания приведен в приложении 4.

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

Основная часть отчета должна содержать: алгоритмы решения поставленных задач, описание прототипа программы, описание структур входных и выходных данных, описание функциональной структуры программы, результаты тестирования программы и руководство пользователя.

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 реализованного программного приложения 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 выполнению контрольных работ, подготовка к лабораторным работам, самостоятельное изучение тем курса, выполнение контрольных работ и курсовой работы.

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 контрольных работ  и курсового проекта — полностью самостоятельная работа, направленная на получение навыков самостоятельного составления алгоритмов, реализацию программ, их дальнейшую отладку и тестирования.

## 4.2 Проработка лекционного материала, подготовка к лабораторным работам и выполнению контрольных работ

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. Выполнение выше перечисленных действий поможет подготовиться и к выполнению контрольных работ.

**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 контрольных работ

### 4.3.1 Общие положения

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

Контрольные работы выполняются самостоятельно, во время семестра, а защищаются перед преподавателем во время экзаменационных сес-

сий, как это предусмотрено учебным планом дисциплины. Допускается общение с преподавателем во время семестра посредством электронной почты — выполненные контрольные работы могут быть высланы на предварительную проверку. Выполнение контрольной работы 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 контрольной работы 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 Контрольная работа «Конструкции структурного программирования»

Контрольная работа «Конструкции структурного программирования» состоит из четырех заданий, которые необходимо выполнить в форме программы (или отдельных программ) на языке Си. Задания контрольной работы соответствуют темам «Функции», «Целочисленная арифметика», «Условные алгоритмы» и «Программирование итерационных процессов». При решении контрольной работы приветствуется создание минимального пользовательского интерфейса — при написании программы используйте строки приглашения, оформляйте вывод полученных значений с соответствующими пояснениями. Перед каждым выполненным заданием из контрольной работы в тексте программы обязательно должны быть написаны комментарии, содержащие следующую информацию — фамилия, имя, отчество студента, формулировка задания, словесное описание алгоритма решения задания. Методические материалы, которые могут быть полезны для решения заданий:

**Задание 1.** При выполнении рекомендуется изучить теоретический материал, изложенный в [1], стр. 77 — 78. Алгоритмы разбора целых чисел рассматриваются в [3], стр. 9 — 15.

**Задание 2.** При выполнении рекомендуется изучить теоретический материал, изложенный в [1], стр. 87 — 89 и [3], стр. 14 — 17.

**Задание 3.** При выполнении рекомендуется изучить теоретический материал, изложенный в [1], стр. 91 — 98.

**Задание 4.** При выполнении рекомендуется изучить теоретический материал, изложенный в [1], стр. 59 — 72, 103 — 113.

**Примерный вариант**

**Задание 1.**

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

б) Дано двузначное число. Найти число единиц числа.

**Задание 2.** 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.

**Задание 3.** Дано натуральное *n*, действительное число *x*. Вычислить: $\sum_{i=1}^{n} \frac{x+\cos(ix)}{2^i}$

**Задание 4.** Заданы массивы чисел *X* [0.. n] и *Y*[0..m]. Написать программу, определить значение переменной *z*. Исходные данные и результат напечатать с пояснительным текстом. Решение задачи оформить с использованием функций. Значения *n* и *m* задавать с клавиатуры, значения элементов массива задавать случайным образом.

В формулах расчета у использованы следующие условные обозначения:

A1(X) — сумма элементов массива *X*;

A2(X) — сумма положительных элементов массива *X*;

A3(X) — сумма отрицательных элементов массива *X*;

A4(X) — количество нулевых элементов массива *X*;

A5(X) — сумма максимального и минимального элементов массива *X*;

A6(X) — среднее арифметическое значение элементов массива *X*;

A7(X) — произведение абсолютных значений элементов массива *X*;

A8(X) — корень квадратный из суммы положительных элементов массива *X*;

A9(X) — натуральный логарифм из суммы абсолютных значений элементов массива *X*;

A10(X) — сумма корней квадратных из положительных элементов массива *X*;

M1(X) — количество элементов массива *X*, значения которых меньше A1;

M2(X) — количество отрицательных элементов массива *X*;

M3(X) — количество элементов массива *X*, значения которых больше A6;

M4(X) — количество элементов массива *X*, значения которых меньше A6;

M5(X) — количество элементов массива *X*, значения которых больше A8.

$$z = \begin{cases} \dfrac{A4(x) + 2.8 * 10^{-3} * A4(x)}{M1(y) + A4(y)}, & \text{если } M1(x) < 2 \\ 0, & \text{в противном случае} \end{cases}$$

### 4.3.3 Контрольная работа «Сложные и производные типы данных. Файлы»

Контрольная работа состоит из четырех заданий, которые необходимо выполнить в форме программы (или отдельных программ) на языке Си. Задания контрольной работы соответствуют темам «Функции», «Структурные переменные», «Обработка строк» и «Двоичные файлы». При решении контрольной работы приветствуется создание минимального пользовательского интерфейса — при написании программы используйте строки приглашения, оформляйте вывод полученных значений с соответствующими пояснениями. Перед каждым выполненным заданием из контрольной работы в тексте программы обязательно должны быть написаны комментарии, содержащие следующую информацию — фамилия, имя, отчество студента, формулировка задания, словесное описание алгоритма решения задания.

**Задание 1.** При выполнении рекомендуется изучить теоретический материал, изложенный в [1]. В главе 4 пособия содержится справочная информация об объявлении такого типа данных, как структура (стр. 67 — 69).

**Задание 2.** При выполнении рекомендуется изучить теоретический материал, изложенный в [1], стр. 156 — 166.

**Задание 3.** Для выполнения задания 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).

**Задание 4.** При выполнении задания может быть полезен материал пособия [1], стр. 110 — 113 и пособия [4], стр. 131 — 142.

**Примерный вариант**

**Задание 1.** Исходные данные хранятся в файле

| № рейса | Пункт назначения | Дата | Количество свободных мест |
|---|---|---|---|

Составить список номеров рейсов и наличии свободных мест по входным данным: пункту назначения и дате. Использовать структурные переменные. Решение задания оформить с использованием функций.

**Задание 2.** Создать файл вещественных чисел, записать в него матрицу вещественных чисел A размерности $m * n$ ($m < n$) . Не считывая матрицу в память, реорганизовать файл, сделав матрицу квадратной ($n * n$) путем удаления "лишних" столбцов, начиная с заданного (номер столбца вводить с клавиатуры). Полученному файлу дать новое имя. Напечатать исходную и полученную матрицы с указанием имен файлов**.**

**Задание 3.** В текстовом файле хранится произвольный текст, напишите программу, которая находит в тексте слово с максимальной длиной. Выведите на экран найденное слово. Если таких слов несколько, выведите все найденные слова.

**Задание 4.** Напишите рекурсивную и не рекурсивную функции, реализующие алгоритм решения поставленной задачи — вычисление факториала натурального числа n.

## 4.4 Самостоятельное изучение тем теоретической части курса

### 4.4.1 Функции

**Перечень вопросов, подлежащих изучению**

1. Синтаксис написания функций в языке Си.
2. Объявление и вызов функций.
3. Локальные переменные.
4. Организация выхода из функции.
5. Передача параметров по ссылке.
6. Рекурсивные функции.

**Методические рекомендации по изучению**

При изучении темы старайтесь следовать списку вопросов, представленному выше. Процесс изучения будет более эффективным, если Вы параллельно с изучением темы будете выполнять задания практически.

Практическую часть самостоятельной работы можно начать с выполнения примеров, описанных в пособии. После реализации примеров попробуйте оформить в виде функций решенные ранее задачи.

Обратите особое внимание на такие понятия, как возвращаемое значение функции, аргументы (параметры функции).

При реализации рекурсивных функций помните — рекурсивная реализация всегда может быть заменена реализацией с использованием цикла.

**Рекомендуемые источники**

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, стр. 103 — 113.

Подбельский, В.В. Курс программирования на языке Си [Электронный ресурс] : учебник / В.В. Подбельский, С.С. Фомин. — Электрон. дан. — Москва : ДМК Пресс, 2012. — 384 с. — Режим доступа: https://e.lanbook.com/book/4148. — Загл. с экрана. Стр. 176 — 238.

### 4.4.2 Сложные типы данных

**Перечень вопросов, подлежащих изучению**

1. Структуры.
2. Объединения.
3. Перечисления.

**Методические рекомендации по изучению**

При изучении темы обратите особое внимание на семантику изучаемых типов данных. Определите для себя типы задач, в которых могут использоваться такие переменные. Параллельно с изучением теоретического материала попробуйте сформулировать задачи, решение которых невозможно без использования этих данных. Обязательно попробуйте набрать и выполнить компиляцию примеров, приведенных в учебном пособии.

**Рекомендуемые источники**

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, стр. 67 — 69.

Подбельский, В.В. Курс программирования на языке Си [Электронный ресурс] : учебник / В.В. Подбельский, С.С. Фомин. — Электрон. дан. — Москва : ДМК Пресс, 2012. — 384 с. — Режим доступа: https://e.lanbook.com/book/4148. — Загл. с экрана. Стр. 239 — 282.

### 4.4.3 Текстовые файлы

**Перечень вопросов, подлежащих изучению**

1. Описание файловых переменных.
2. Режимы доступа к файлу.
3. Открытие и закрытие файла.
4. Способы чтения информации из текстового файла.
5. Способы записи информации в текстовый файл.

**Методические рекомендации по изучению**

При изучении этой темы следуйте списку вопросов, представленному выше. Для более полного владения темой найдите описание структуры FILE. Запомните существующие режимы доступа к файлу. Обратите внимание, язык программирования Си предоставляет самые разнообразные варианты чтения информации из файла. Отметьте для себя, каким образом можно организовать проверку ошибок чтения или записи в файл. Не забывайте про практическую реализацию приведенных в учебных пособиях примеров.

**Рекомендуемые источники**

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, стр. 144 — 155.

Подбельский, В.В. Курс программирования на языке Си [Электронный ресурс] : учебник / В.В. Подбельский, С.С. Фомин. — Электрон. дан. — Москва : ДМК Пресс, 2012. — 384 с. — Режим доступа: https://e.lanbook.com/book/4148. — Загл. с экрана. Стр. 283 — 321.

### 4.4.4 Двоичные файлы

**Перечень вопросов, подлежащих изучению**

1. Способы чтения информации из двоичного файла.

2. Способы записи информации в двоичный файл.

3. Организация прямого доступа в двоичных файлах.

**Методические рекомендации по изучению**

Изучите функции языка Си, позволяющие считывать информацию из файла и записывать ее в файл. Обратите внимание на многообразие функций языка для организации прямого доступа к файлу. Выделите для себя различия между текстовыми и двоичными файлами. Параллельно с изучением справочного материала попробуйте написать функции записи заданной информации в двоичный файл. Попытайтесь открыть двоичный файл в текстовом редакторе. Найдите объяснение увиденной информации.

**Рекомендуемые источники**

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, стр. 156 — 166.

Подбельский, В.В. Курс программирования на языке Си [Электронный ресурс] : учебник / В.В. Подбельский, С.С. Фомин. — Электрон. дан. — Москва : ДМК Пресс, 2012. — 384 с. — Режим доступа: https://e.lanbook.com/book/4148. — Загл. с экрана. Стр. 322— 334.

## 4.5 Подготовка к зачету

Необходимым условием для получения зачета является выполнение всех лабораторных и контрольных работ, предусмотренных рабочей программой дисциплины. Зачет проводится в устной и письменной форме. Ниже приведен пример зачетного билета.

**Билет 2**

1. Синтаксис языка Си. Простая программа на языке Си. Функция `main`.
2. Напишите программу, которая запрашивает с клавиатуры размерность массива, заполняет массив случайными значениями из интервала [-10,15] и находит сумму положительных элементов.
3. Напишите программу, которая запрашивает с клавиатуры размерность матрицы $n$ и $m$, заполняет матрицу случайным образом. Отсортируйте строки матрицы по увеличению первого элемента. Используйте сортировку обменом. Сортировку строк матрицы оформите в виде функции. Параметры функции — матрица и ее размерность.

Самостоятельная подготовка к зачету состоит в изучении соответствующих тем и выполнении практических заданий (индивидуальных вариантов лабораторных и контрольных работ).

Для получения оценки *«зачтено»* необходимо выполнить два задания из трех.

## 4.6 Подготовка к экзамену

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**

**Билет 1**

1. В текстовом файле записан массив целых чисел. Считать массив из файла. Найти сумму чисел. В конец исходного файла дописать строку: «Сумма чисел = [найденная сумма]».

2. Написать:

функцию создания двоичного файла, содержащего матрицу целых чисел размерности nxm. Значения n, m задаются с клавиатуры. Элементы матрицы — целые случайные числа;

функцию печати содержимого двоичного файла;

функцию, выводящую на экран четные столбцы матрицы, не считывая при этом матрицу в память.

3. В текстовом файле хранится произвольное количество чисел. Считать данные из файла в однонаправленный динамический список, организованный по правилу очереди (первое число из файла должно оказаться в «голове», последнее в «хвосте» списка). Поменять местами первый и последний элементы списка.

## 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. 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 .

3.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.

4. 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.

## ПРИЛОЖЕНИЕ 1

**Конкретизация задания на курсовую работу**

***Задание:***

В текстовом файле хранится информация о сдаче абитуриентами экзаменационных испытаний по математике, физике и информатике. Информация структурирована следующим образом:

| Фамилия | Имя | Отчество | Номер специальности | Балл по физике | Балл по математике | Балл по информатике |
|---|---|---|---|---|---|---|
| | | | | | | |

По исходным данным необходимо вывести статистические данные о результатах экзаменов:

- информацию об абитуриентах, общий балл которых больше среднего балла;
- информацию об абитуриентах, набравших максимальный балл хотя бы по одному предмету;
- информацию об абитуриентах, подавших заявления на выбранную специальность, упорядоченную по возрастанию общего балла.

Для выполнения задания курсовой работы необходимо выполнить следующие задачи:

1. Разработать алгоритм решения задачи
2. Определить структуру входных данных
3. Написать программу, реализующую следующие функции:
   3.1. чтения информации из текстового файла;
   3.2. поиска информации о студентах, средний балл которых выше общего среднего балла;
   3.3. поиска информации об абитуриентах с максимальным баллом;
   3.4. поиска информации об абитуриентах заданной специальности.
4. Разработать и реализовать интерфейсную часть программы.
5. Выполнить тестирование программы.
6. Оформить отчет о проделанной работе.

## ПРИЛОЖЕНИЕ 2

### Алгоритмы решения задачи

***Общий алгоритм решения задачи***

1.Выполнить чтение данных из текстового файла

2.Найти студентов, средний балл которых выше общего среднего балла.

3.Вывести найденную информацию в текстовый файл.

4.Найти студентов, балл которых —максимальный.

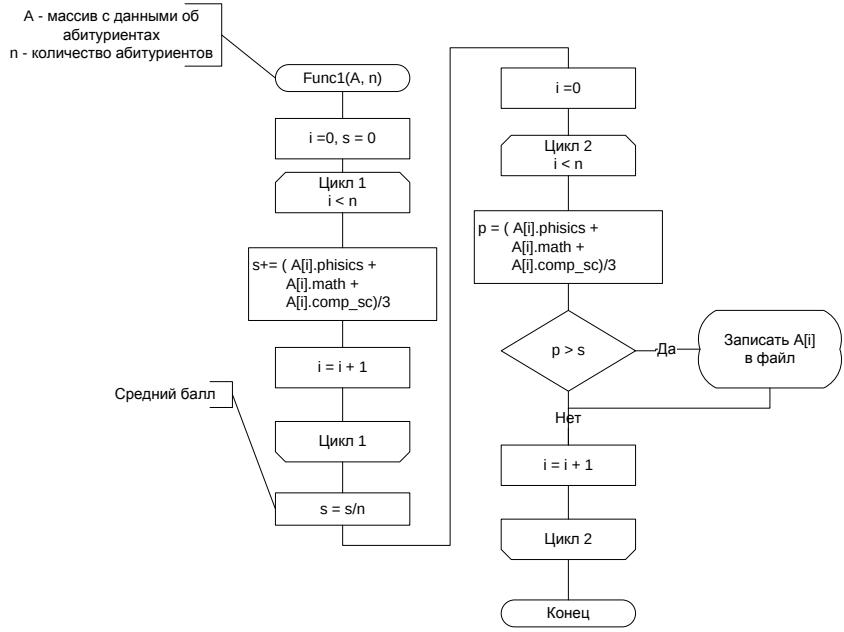
5.Вывести полученную информацию в текстовый файл.

6.Найти студентов, подавших заявление на заданную специальность.

7.Вывести информацию в текстовый файл.

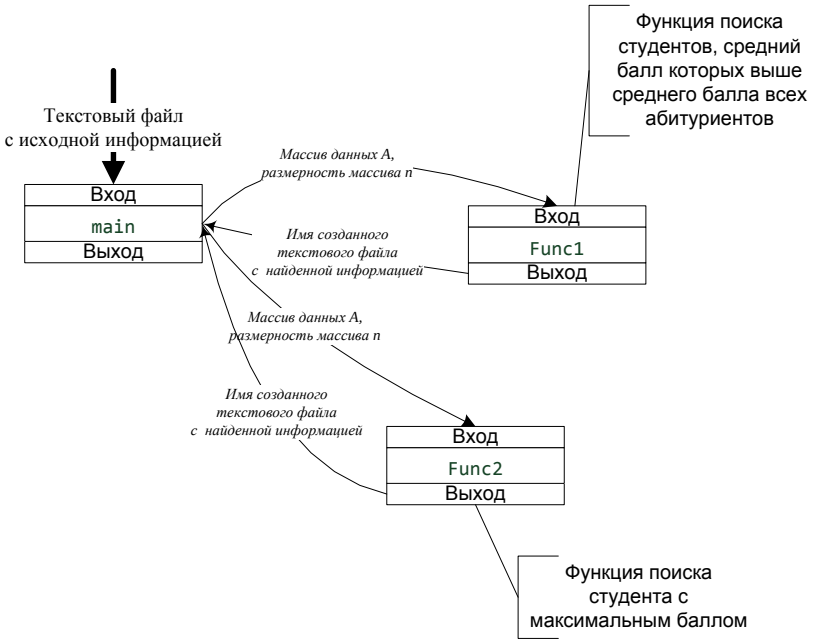
8.Выйти из программы.

***Алгоритм поиска студентов, средний балл которых выше общего среднего балла***



## ПРИЛОЖЕНИЕ 3

**Фрагмент структуры программы**



## ПРИЛОЖЕНИЕ 4

### 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 ____ _____факультет _____*СУ*_____

1. Задание _____

2. Дата выдачи задания __ _____

3. Исходные данные к проекту__ _____

_____

_____

_____

4. Содержание отчета (перечень подлежащих разработке вопросов)_____

_____

_____

_____

5. Срок сдачи законченного задания_ _____

Руководитель _____

_____

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