

Министерство образования и науки Российской Федерации  
Федеральное государственное бюджетное образовательное  
учреждение высшего образования

**ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ  
УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ**

**Кафедра автоматизированных систем управления (АСУ)**

**М. Ю. Катаев**

# **ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ**

**Методические указания  
по выполнению курсовой работы**

**Томск 2017**

Корректор: А. Н. Миронова

**Катаев М. Ю.**

Объектно-ориентированное программирование : методические указания по выполнению курсовой работы / М. Ю. Катаев. – Томск : ФДО, ТУСУР, 2017. – 48 с.

© Катаев М. Ю., 2017

© ФДО, ТУСУР, 2017

## СОДЕРЖАНИЕ

Введение .....	4
1 Общие требования к выполнению курсовой работы.....	5
2 Порядок выполнения курсовой работы .....	6
2.1 Конкретизация задания на курсовую работу .....	6
2.2 Разработка блок-схемы решения задач.....	6
2.3 Разработка программы .....	6
2.4 Подготовка ответа на теоретические вопросы.....	7
3 Подготовка отчета о проделанной работе .....	7
3.1 Содержание пояснительной записки .....	8
4 Рекомендуемая литература.....	8
5 Проверка курсовой работы.....	9
Приложение А Язык блок-схем как язык записи алгоритмов.....	10
Приложение Б Варианты заданий .....	26
Приложение В Теоретические вопросы.....	47

## ВВЕДЕНИЕ

Целью курсовой работы по дисциплине «Объектно-ориентированное программирование» (ООП) является получение навыков самостоятельной разработки программного продукта на языке программирования C++.

Полученные знания могут быть использованы при проведении практических занятий, выполнении курсовых по изучаемым предметам, контрольных и аудиторных работ, а также при самостоятельном изучении данных дисциплин.

Одним из наиболее интенсивных способов изучения дисциплины ООП является самостоятельная реализация различных алгоритмов, в виде программного кода. При этом вырабатываются навыки, необходимые при разработке сложных программных средств.

Предлагаемые задания позволят глубже освоить теоретические и практические вопросы объектно-ориентированной парадигмы программирования, понять принципы написания объектно-ориентированных (основанных на классах) программ, научиться грамотно применять эти теоретические знания при разработке программных средств различной сложности.

Для выполнения курсовой работы студент получает индивидуальное задание – описание задачи, для решения которой разрабатывается и реализуется программный продукт. Варианты заданий курсовой работы приведены в приложениях настоящих методических указаний.

### **Задание:**

- 1) выбрать вариант решаемой задачи;
- 2) выбрать теоретический вопрос, в соответствии с вариантом;
- 3) разработать блок-схему решаемой задачи;
- 4) разработать программу на языке программирования C++;
- 5) провести тестирование полученной программы;

6) подготовить ответ на теоретический вопрос, используя пособие «Объектно-ориентированное программирование», другие книги и интернет-издания;

7) описать в пояснительной записке полученные результаты при составлении блок-схемы задачи, тестировании программы и ответ на теоретический вопрос.

## 1 ОБЩИЕ ТРЕБОВАНИЯ К ВЫПОЛНЕНИЮ КУРСОВОЙ РАБОТЫ

Выбор варианта курсовой работы (индивидуального задания) осуществляется по общим правилам с использованием следующей формулы:

$$V = (N \times K) \text{ div } 100,$$

где  $V$  – искомый номер варианта,

$N$  – общее количество вариантов (= 25),

div – целочисленное деление,

при  $V = 0$  выбирается максимальный вариант,

$K$  – код варианта.

Студент самостоятельно разрабатывает структуру программы – определяет типы входных и выходных данных, разрабатывает алгоритмы решения предложенных задач, определяет функциональные блоки будущего программного продукта и реализует предложенную задачу на языке Си/Си++.

Для программирования можно выбрать свободно распространяемые среды программирования:

1) Visual C++ для Visual Studio 2015 (или другие версии) можно скачать по ссылке: <https://www.microsoft.com/ru-ru/download/details.aspx?id=48145> ;

2) DEV C++ можно скачать по ссылке <http://www.bloodshed.net/dev/devcpp.html> .

В приложении А представлены элементы, необходимые для разработки блок-схемы структуры программы. В приложении Б представлены задачи, в приложении В представлены теоретические вопросы.

## **2 ПОРЯДОК ВЫПОЛНЕНИЯ КУРСОВОЙ РАБОТЫ**

### **2.1 Конкретизация задания на курсовую работу**

На первом этапе выполнения работы исходя из рассчитанного варианта необходимо выбрать:

- А) задачу из приложения Б;
- Б) теоретический вопрос из приложения В.

### **2.2 Разработка блок-схемы решения задач**

На втором этапе требуется описать алгоритм работы будущего программного продукта в виде блок-схемы. Алгоритм может быть представлен блок-диаграммой. В приложении А представлены основные элементы решения поставленной задачи в виде блок-схемы. Для представления прототипа алгоритма программы в виде блок-схемы можно воспользоваться средствами рисования Microsoft Word.

### **2.3 Разработка программы**

На третьем этапе необходимо разработать программу на языке программирования C++. При выполнении этого этапа необходимо спроектировать вид основного консольного меню программы и описать все варианты выбора элементов меню, необходимых для решения задачи, сформулированной в задании на курсовую работу. В отчете должно присутствовать описание правил пользования программным средством, правил формирования входных данных.

## 2.4 Подготовка ответа на теоретические вопросы

На четвертом этапе необходимо составить письменный ответ на теоретический вопрос. Ответ должен быть не менее 3 страниц и содержать не только основные определения, но и примеры программного кода.

## 3 ПОДГОТОВКА ОТЧЕТА О ПРОДЕЛАННОЙ РАБОТЕ

Оформление пояснительной записки должно соответствовать требованиям образовательного стандарта вуза ОС ТУСУР 01–2013 «Работы студенческие по направлениям подготовки и специальностям технического профиля. Общие требования и правила оформления». Текст стандарта можно найти по ссылке: <https://regulations.tusur.ru/documents/70>

Рекомендуется следующее содержание отчета:

- титульный лист,
- задание на курсовую работу,
- введение,
- основная часть,
- заключение,
- список использованных источников.

Введение должно содержать цель работы, описание ее значения.

Основная часть пояснительной записки должна отражать выполнение всех этапов работы, составляющих содержание курсовой работы, описанных в п. 2.

Пример содержания пояснительной записки приведен ниже.

### **3.1 Содержание пояснительной записки**

Введение

1 Блок-схема решения задачи

2 Описание программы

2.1 Описание структуры входных и выходных данных

2.2 Функциональная структура программы

2.3 Тестирование программы

2.4 Руководство пользователя

3 Теоретический вопрос

Заключение

Список литературы

Заключение должно содержать краткие выводы по результатам выполненной работы. Список использованных источников оформляется согласно стандарту.

### **4 РЕКОМЕНДУЕМАЯ ЛИТЕРАТУРА**

1. Катаев М. Ю. Объектно-ориентированное программирование : учеб. пособие / М. Ю. Катаев. – Томск : Эль Контент, 2013. – 212 с.

2. Павловская Т. А. С/С++. Программирование на языке высокого уровня для магистров и бакалавров : учебник для вузов / Т. А. Павловская. – СПб. : ПИТЕР, 2012. – 461 с.

3. Хабибуллин И. Ш. Программирование на языке высокого уровня С/С++ : учеб. пособие для вузов / И. Ш. Хабибуллин. – СПб. : БХВ-Петербург, 2006. – 485 [13] с.

## **5 ПРОВЕРКА КУРСОВОЙ РАБОТЫ**

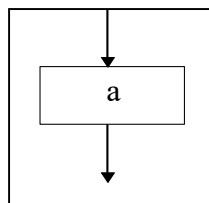
На проверку присылается исходный код программы (.cpp и .h файлы), исполняемый файл (\*.exe) и пояснительная записка, в которой по главам расположены: теоретическая часть, описание структуры программы и описание результатов работы программы.

## ПРИЛОЖЕНИЕ А

### Язык блок-схем как язык записи алгоритмов

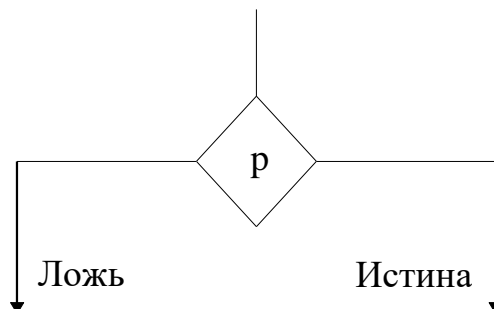
Выше мы говорили, что для записи алгоритма используется язык блок-схем. Рассмотрим подробнее грамматику этого языка. Прежде всего заметим, что блок-схема представляет собой двухмерный рисунок, построенный из управляющих структур. При рисовании этих структур используются специальные обозначения.

#### *Элементы управляющих структур*

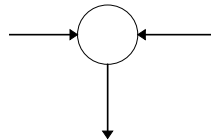


*Обозначение обработки.* Действие, которое необходимо выполнить, обозначается прямоугольником, в который входит и из которого выходит ровно одна линия управления. Этот прямоугольник называется узлом обработки, или функциональным узлом.

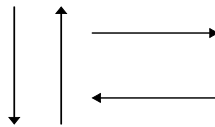
Действие, указываемое в этом прямоугольнике, может быть отдельным оператором, вызовом некоторой подпрограммы, произвольной управляющей структурой.



*Обозначение проверки.* Операция проверки обозначается символом, который называется условным (предикатным) узлом. Он представляет собой ромб, в который входит одна линия управления, а выходят две. В результате проверки помещенного внутри ромба условия (предиката)  $p$  выбирается один из выходов (но не оба сразу).



*Обозначение слияния.* Символ слияния – это кружок, где соединяются пути управления.



*Соединительные линии.* Изображают передачи управления от одного из вышеперечисленных обозначений к другому в направлении, указанном стрелкой.



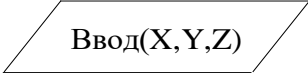
*Начало алгоритма* (вход в блок-схему) и *конец алгоритма* (выход из блок-схемы) изображаются очевидным образом.

Изображения любой из управляющих структур представляют собой комбинации этих обозначений.

### *Основные управляющие структуры*

Далее мы проанализируем специфику преобразования данных и выделим базисные классы преобразований. Для каждого из них предложим соответствующую управляющую структуру. Безальтернативные вычисления (управляющая структура «следование»).

Прежде всего, мы уточним понятие действия по обработке данных, подразделив ее на три группы: ввод исходных данных в оперативную память, изменение данных в оперативной памяти, вывод данных из оперативной памяти.

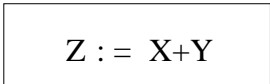


Ввод(X,Y,Z)

*Ввод данных.* Предписание на ввод данных содержит указание устройства ввода и имя переменной, значение которой надо ввести. Чаще всего мы будем определять это действие в виде оператора ввода с клавиатуры:

Ввод(<список переменных>)

Допускается неформальное описание ввода. Например: «читать очередную запись входного файла».



$Z := X+Y$

*Изменение данных.* Чаще всего предписание на изменение данных мы будем представлять в виде оператора присваивания (последовательности

операторов присваивания). В языке программирования есть специфическое обозначение оператора присваивания:

$\langle \text{переменная} \rangle := \langle \text{арифметическое выражение} \rangle$

Допускается неформальное описание преобразования данных. Например: «Учесть сумму в строке баланса и в итоговой строке».

Вывод(Z)

*Вывод данных.* Предписание на вывод данных содержит указание устройства вывода и имя переменной, значение которой следует вывести.

Чаще всего мы будем определять это действие в виде оператора вывода на монитор:

Вывод( $\langle \text{список переменных} \rangle$ )

Допускается неформальное описание вывода. Например: «печатать результат вычислений». Самая простая и очевидная управляющая структура – следование (рис. 1). Допускается запись последовательности действий по преобразованию данных в одном прямоугольнике.

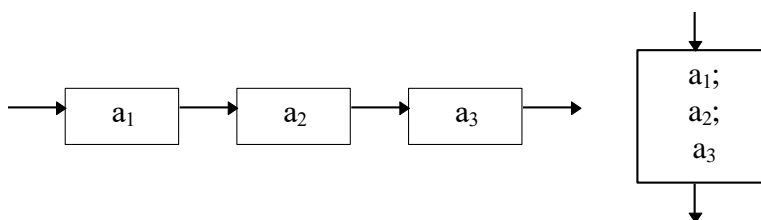


Рис. 1 – Управляющая структура «следование»

Управляющая структура «следование» используется в случае, когда алгоритм представляется как последовательность, элементами которой

служат только действия по преобразованию данных. В этом случае мы имеем дело с линейным алгоритмом.

*Прикладная задача 1.* Вычисление площади треугольника. Для хранения размеров сторон треугольника используем переменные a,b,c. Преобразование данных задается двумя операторами присваивания:

$$P=(a+b+c)/2; \quad S=\sqrt{P*(P-a)*(P-b)*(P-c)}$$

Результат получается в переменной S. Переменная P используется для хранения промежуточного результата – полупериметра треугольника.

### *Альтернативные вычисления (управляющая структура «выбор»)*

К сожалению, линейных алгоритмов практически не бывает. Дело в том, что большинство вычислительных процессов обладают следующим свойством: способ вычисления результата зависит от конкретного исходного набора исходных данных. Например, квадратное уравнение, когда способ вычисления его корней зависит от дискриминанта, т. е. в конечном итоге от значений коэффициентов этого уравнения. Поэтому при реализации алгоритма так или иначе приходится анализировать исходные и промежуточные данные и выстраивать альтернативный процесс преобразования исходных данных в результирующие.

Для реализации альтернативного двоичного выбора используется управляющая структура выбор в двух ее модификациях (рис. 2). Условие p (логическая функция) принимает два значения: {True – да, False – нет }. Если структура выбора «полная», то в зависимости от значения условия выполняется либо действие S1, либо действие S2. Если структура выбора

«укороченная», то в случае истинности условия выполняется действие S1. В противном случае его выполнение игнорируется.

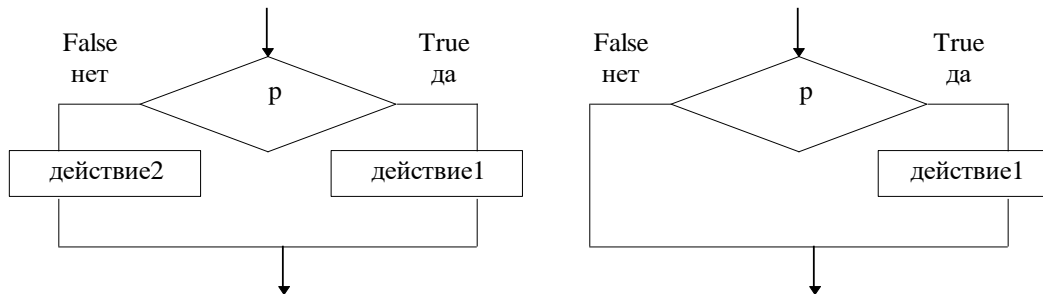


Рис. 2 – Полная и укороченная управляющая структура «выбор»

В языках программирования есть оператор, обозначающий эту управляющую структуру:

if p then S1 else S2    или    if p then S1

Действием является либо простой оператор, либо составной оператор. Составной оператор – запись на языке программирования произвольной управляющей структуры, заключенная в операторные скобки.

*Прикладная задача 2.* Напечатать наибольшее из двух заданных чисел.

Для хранения сравниваемых чисел отводим переменные a,b. Результат хранится в переменной Max.

Как уже говорилось, внутри блока действия может размещаться любая управляющая структура. Характерный пример, когда внутри блоков S1 и/или S2 размещается управляющая структура выбора.

*Прикладная задача 3.* Напечатать наибольшее из трех заданных чисел.

Повторяющиеся вычисления (управляющая структура «цикл пока»).

Большинство вычислительных процессов реализуются как длинные и очень длинные последовательности действий. Возникает проблема компактной записи длинной последовательности действия. К счастью, вычислительные процессы обладают замечательным свойством: в последовательности действий можно выделить подпоследовательности, в каждой из которых выполняются однородные действия. И эти подпоследовательности однородных действий можно обозначить управляющей конструкцией «ЦИКЛ».

*Прикладная задача 4.* Вычисление вещественной функции  $a \cdot \exp\{b \cdot x - c \cdot x^2\}$  в заданном интервале с заданным шагом. Представим предписание на вычисление этой функции в виде оператора присваивания:

$$z := a \cdot \exp(b \cdot x - c \cdot x \cdot x).$$

Пусть интервал вычисления  $(-1, 1)$ , шаг 0.5. Тогда вычисление выполняется в следующих точках:  $(-1, -0.5, 0, 0.5, 1)$ . Такое вычисление можно записать в виде последовательности пяти действий, каждое из которых состоит из оператора присваивания и оператора вывода. А если вычисления выполняются в тысяче точек: интервал вычислений  $(1, 1000)$ , шаг вычислений 1? А если сто тысяч точек? В этих случаях, теоретически рассуждая, можно представить алгоритм в виде управляющей структуры следования, состоящей из большого количества блоков.

Управляющая структура «цикл» позволяет «свернуть» длинную последовательность действий и представить ее в компактном виде. Основным типом цикла является «цикл пока» (рис. 3). Он предписывает выполнять тело цикла до тех пор, ПОКА истинно условие  $p$ . В качестве такого условия выступает логическое выражение. Тело цикла – действие, как для управляющей структуры ВЫБОР.

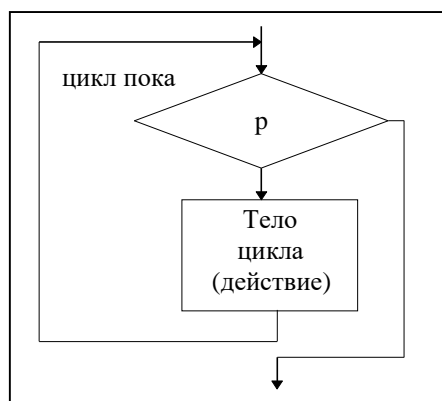


Рис. 3 – Управляющая структура «цикл пока»

При конструировании циклов следует соблюдать обязательное условие результативности алгоритма (т. е. его окончания за конечное число шагов). Практически это означает, что в логическом выражении  $p$  должна быть переменная, значение которой изменяется в теле цикла. Причем изменяется таким образом, чтобы условие  $p$  в конечном итоге стало ложным. Такая переменная называется управляющей переменной цикла или параметром цикла. Условие окончания цикла  $p$ , совместно со спецификой изменения параметра цикла, определяет число повторения тела цикла.

В языках программирования есть оператор, обозначающий эту управляющую конструкцию:

```
while p do
  тело цикла
```

Использование «цикла пока» позволяет записать алгоритм вычисления функции в заданном интервале  $(x_1, x_2)$  с шагом  $dx$  (прикладная задача 4). Управляющая структура «цикл» обладает замечательным свойством: количество образующих цикл элементов не зависит от числа повторения тела цикла. Так, одна и та же блок-схема (алгоритм 4) позволяет задать вычисление функции и в десяти точках и в десяти тысячах точек.

Доказано, что алгоритм решения любой прикладной задачи можно сконструировать только из структур следование, двоичный выбор и цикл ПОКА. В результате такого конструирования получается структурная блок-схема алгоритма. Заметим, что управляющие конструкции структурной блок-схемы являются двухполюсниками, т. е. имеют один вход и один выход. Это позволяет, подставляя одни двухполюсники в другие, конструировать сложную блок-схему, которая также получается двухполюсной и является структурной. В конечном итоге программу любой сложности можно представить в виде одного двухполюсника.

Современные языки программирования являются также структурными, т. е. включают в себя операторы, обозначающие эти управляющие структуры – двухполюсники. Оператор (составной оператор) обозначает «узел обработки», линейный участок программы – «следование», оператор if – «выбор», оператор while – «цикл пока». Поэтому, запись структурной блок-схемы на современном языке программирования сводится, по сути дела, к перекодировке и не является проблемой.

### ***Вспомогательные управляющие структуры***

Считается целесообразным при конструировании блок-схем использовать вспомогательные управляющие структуры, которые, не нарушая структурного представления алгоритма, обеспечивают более краткое его описание.

#### *Повторяющиеся вычисления (управляющая структура «цикл for»)*

*Прикладная задача 5.* Вычислить факториал для заданного n. Известно, что факториал вычисляется по следующей формуле:

$$F=n!=1 \times 2 \times 3 \times \dots \times n$$

Здесь используется характерная и часто встречающаяся комбинация управляющих конструкций, предписывающая выполнить тело цикла вполне определенное число раз (в данном случае  $n$  раз). Общий вид такой конструкции приведен на рис. 4, а. Она предписывает выполнить тело цикла при следующих значениях параметра цикла  $i$ :

нач, нач+шаг, нач+2×шаг, нач+3×шаг..., конец.

Здесь, нач – переменная (константа), декларирующая начальное значение параметра цикла; конец – переменная (константа), декларирующая конечное значение параметра цикла; шаг – переменная (константа), декларирующая приращение параметра цикла.

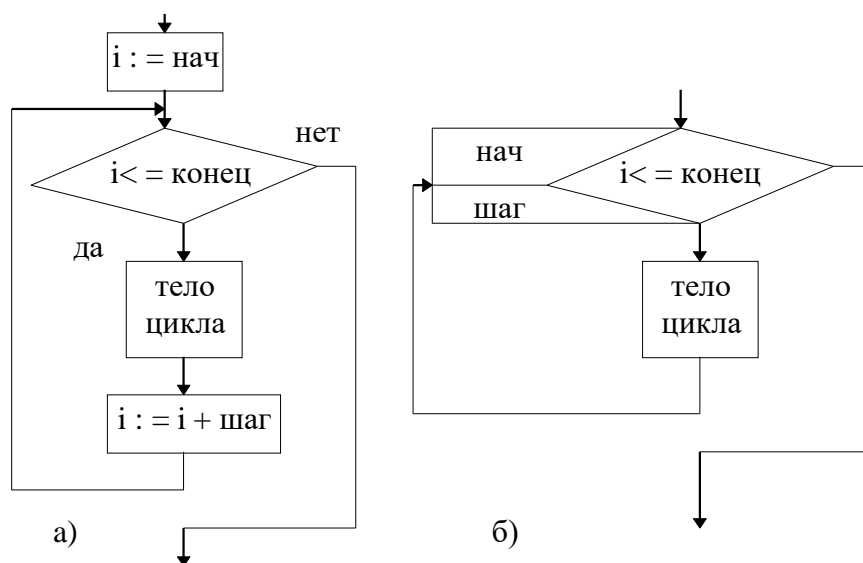


Рис. 4 – Управляющая конструкция «цикл for» (а) и ее обозначение (б)

Такая управляющая конструкция получила специальное название «цикл for» или «цикл для». В языках программирования есть соответствующий оператор для записи подобной управляющей конструкции:

for  $i := \text{нач}$  to  $\text{конец}$  step  $\text{шаг}$  do  
 тело цикла

Приращение параметра цикла называется «шаг цикла». В некоторых языках шаг цикла произвольный, в других языках может иметь только значение «единица» и в операторе `if` не указывается. Имеется редко используемое, но удобное обозначение «цикла `for`» на языке блок-схем (рис. 4, б).

### *Повторяющиеся вычисления (управляющая структура «цикл до»)*

Факториал существует только для положительного числа  $n$ . В алгоритме допускается ввод числа  $n$  любого знака, т. е. не контролируется допустимость исходных данных. Для такого контроля целесообразно использовать еще один вид управляющей конструкции – «цикл до». Она предписывает выполнять тело цикла до выполнения условия  $p$ . Так же как и для «цикла пока», условие  $p$  является логическим выражением, включающим в себя параметр цикла. Последний должен изменяться в теле цикла таким образом, чтобы условие  $p$  в конечном итоге стало истинным.

В языках программирования есть специальный оператор, обозначающий управляющую конструкцию «цикл до»:

```
repeat
тело цикла
until p
```

**Замечание.** Если тело «цикла пока» может не выполняться ни одного раза, то тело «цикла до» выполняется хотя бы один раз.

### *Альтернативные вычисления*

#### *(управляющая структура «множественный выбор»)*

Основная управляющая структура «выбор» включает в себя условие  $p$ , которое имеет два возможных значения:  $\{\text{True}, \text{False}\}$ . Однако при разработке алгоритмов возникают ситуации, когда условие  $p$  имеет  $k$  значений.

Конечно, алгоритмизировать обработку такого многозначного условия можно посредством вложенных конструкций выбора (рис. 5).

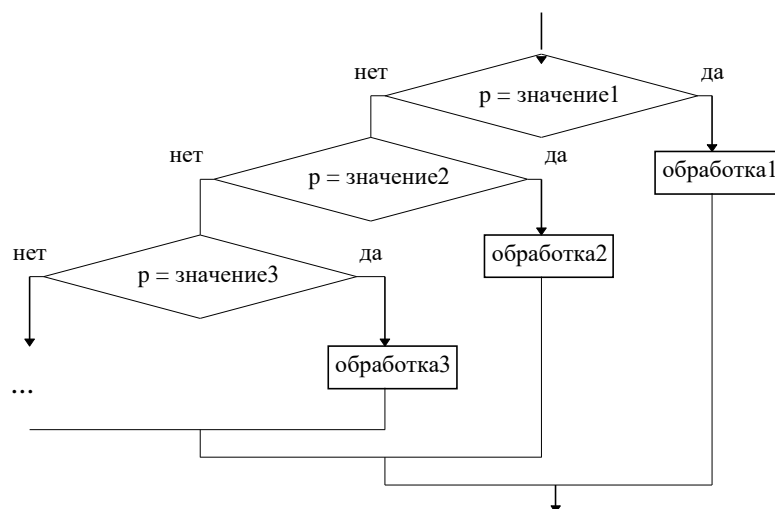


Рис. 5 – Алгоритмизация обработки многозначного условия вложенными конструкциями двоичного выбора

Однако чем больше  $k$ , тем больше глубина вложенности конструкций выбора и сложнее понимание и программная запись получающейся блок-схемы. Можно предложить алгоритмизацию обработки многозначного условия без вложенности конструкций выбора. Плата за такую простоту – большее время обработки условия.

Для удобства алгоритмизации обработки множественного выбора используется специальное обозначение – вспомогательная управляющая конструкция «множественный выбор» (рис. 6).

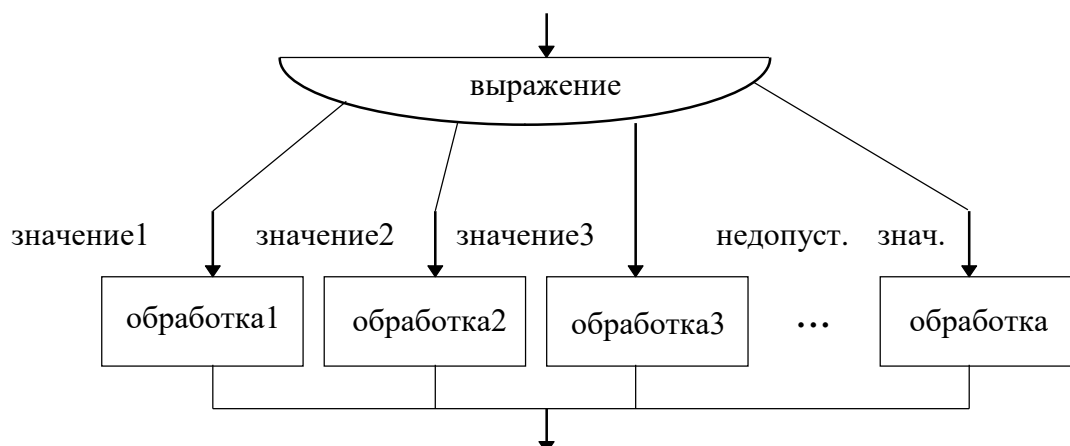


Рис. 6 – Вспомогательная управляющая конструкция «множественный выбор»

**Замечание.** Допускается объединение нескольких значений в группы для реализации одной и той же обработки.

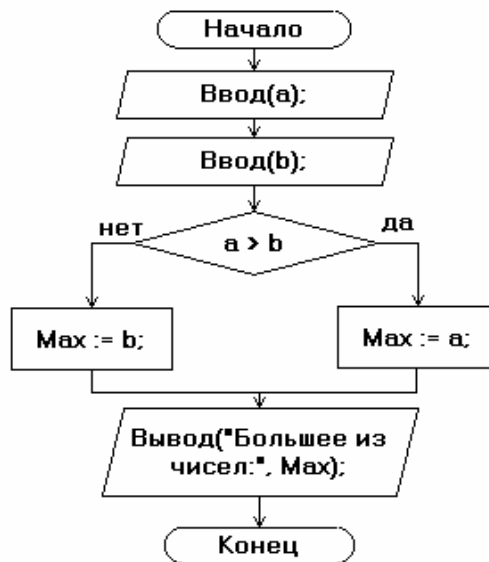
В языках программирования есть специальный оператор, обозначающий эту управляющую конструкцию:

```

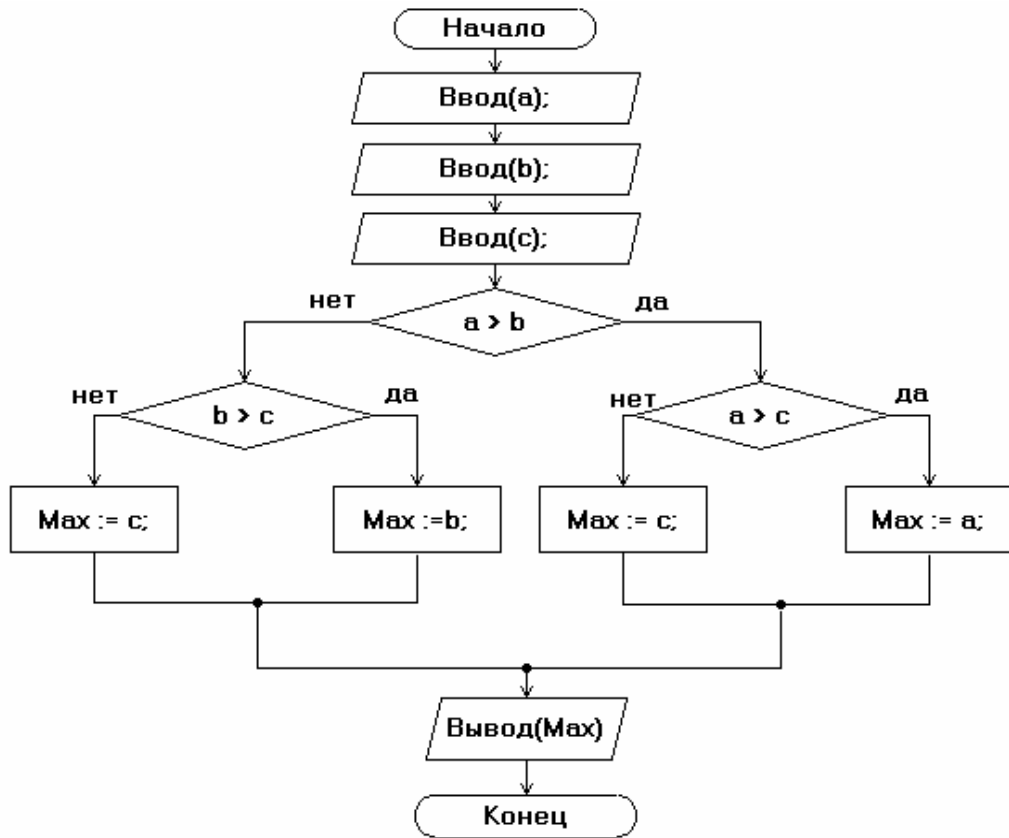
case p of
  группа значений1: обработка1;
  группа значений2: обработка2;
  группа значений3: обработка3;
  ...
else обработка недопустимого варианта
end

```

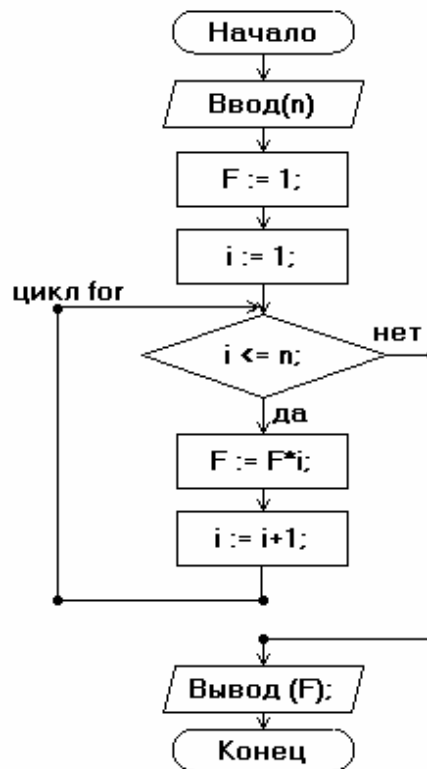
***Примеры блок-схем решения задач без описания их работы***



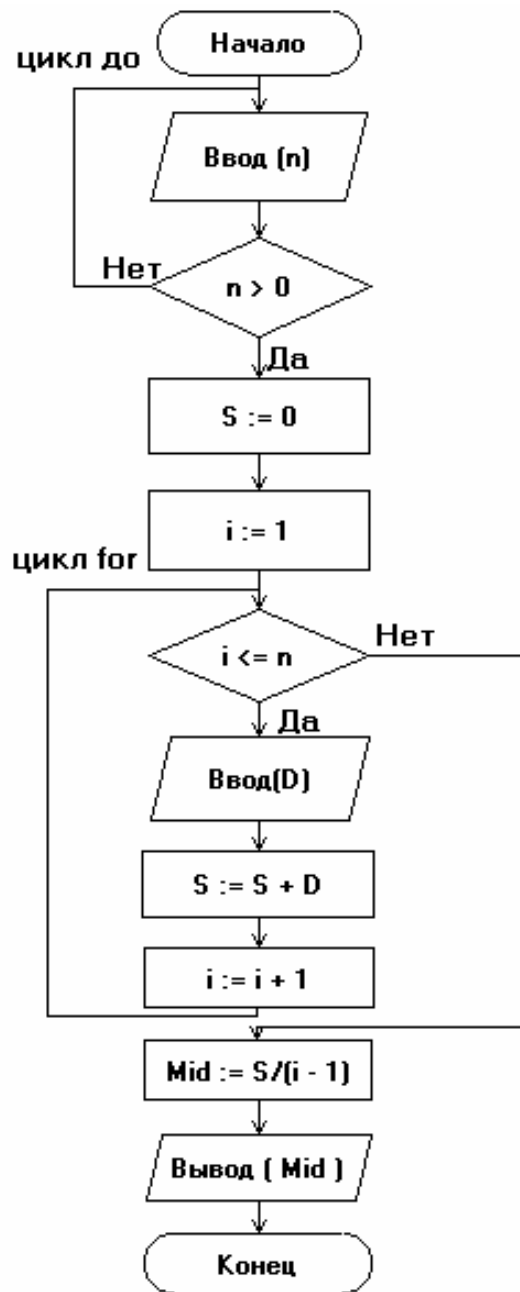
Блок-схема 1 – Печать наибольшего из двух чисел



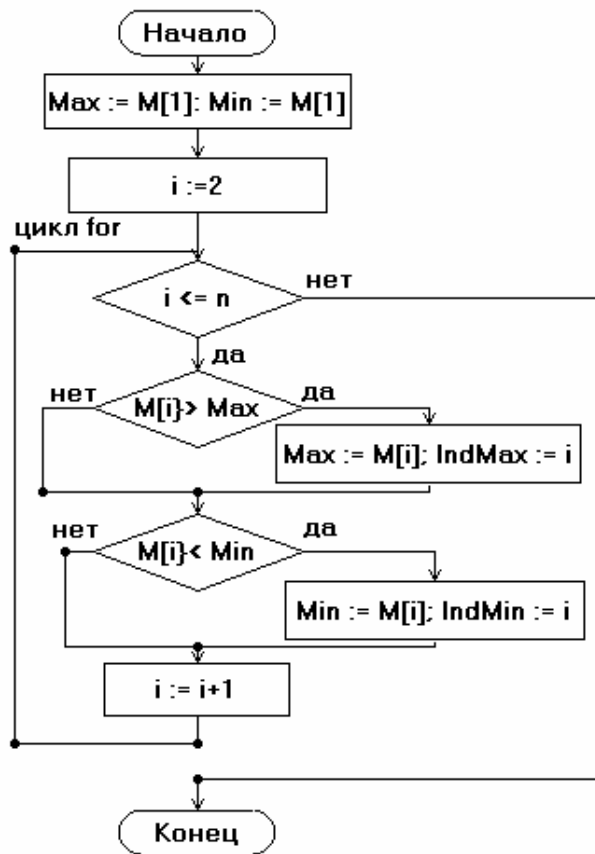
Блок-схема 2 – Печать наибольшего из трех чисел



Блок-схема 3 – Вычисление факториала



Блок-схема 4 – Обработка статической последовательности данных известной длины



Блок-схема 5 – Вычисление минимакса для последовательности, хранящейся в массиве М

## ПРИЛОЖЕНИЕ Б

### Варианты заданий

1. В текстовом файле, содержащем текст программы на языке C++, проверить соответствие открывающихся и закрывающихся фигурных скобок { и }. Результат проверки вывести на экран и записать в виде фразы в текстовый файл. Результат работы программы (вывод) поместить в отдельный текстовый файл (например, «out . txt »), продублировав на экране.

Введите имя файла  
Баланс скобок не нарушен!  
Для закрытия данного окна нажмите «ENTER»

2. Вычислить цепную дробь, используя показатели функции. Количество элементов дроби надо задавать с клавиатуры.  $1 + 1 / (1 + 1 / (1 + 1 / (1 + 1 / \dots)))$ . Количество элементов дроби должно быть равно не менее 5. Рассмотреть три случая, например, 5, 7 и 10.

Введите число элементов цепной дроби: 6  
Решение – 1.61538

3. Нужно вычислить произведение матрицы на вектор. В полученном векторе найти максимальный элемент. Необходимо решить данную задачу тремя способами: с использованием статических массивов, использовать только динамические массивы с явным разыменованием указателя, использовать только динамические массивы, адресацию к элементам массива выполнять с помощью индексов. То есть у вас должно получиться три отдельных исходника – три программы. Пример работы программы смотрите ниже.

```

Введите количество строк матрицы: 2
Введите количество столбцов матрицы: 4
matrix[1][1] = 1
matrix[1][2] = 2
matrix[1][3] = 3
matrix[1][4] = 4
matrix[2][1] = 5
matrix[2][2] = 6
matrix[2][3] = 7
matrix[2][4] = 8
Введите элементы вектора:
vector[1] = 1
vector[2] = 0
vector[3] = 1
vector[4] = 0
Введенная матрица:
1 2 3 4
5 6 7 8
Введенный вектор:
1
0
1
0
Результирующий вектор:
4
12
Максимальное значение = 12

```

4. Задача на динамическое выделение памяти. Изначально есть указатель на массив с одним элементом. Пользователь вводит число. Если оно больше 0, записываем его в массив. Далее пользователь вводит второе число, если оно больше 0, надо перевыделять память для 2 элементов массива и записать в массив второе число. И так далее...для 3 элементов, для 4... пока пользователь не введет отрицательное число.

```

Введите целое число: 3
3
Введите целое число: 4
3 4
Введите целое число: 5
3 4 5
Введите целое число: 6
3 4 5 6
Введите целое число: 7

```

```

3 4 5 6 7
Введите целое число: 8
3 4 5 6 7 8
Введите целое число: 5
3 4 5 6 7 8 5
Введите целое число: 4
3 4 5 6 7 8 5 4
Введите целое число: 0
3 4 5 6 7 8 5 4 0
Введите целое число: 0
3 4 5 6 7 8 5 4 0 0
Введите целое число: 7
3 4 5 6 7 8 5 4 0 0 7
Для выхода введите целое отрицательное число: -1

```

5. Необходимо создать структуру, которая содержит элемент типа `int` и строку. Объявить указатель типа структуры (объект структуры) и выделить память для хранения данных одной структуры. Предложить пользователю внести данные и записать их в элементы объекта структуры. Далее пользователю необходимо сделать выбор: внести еще данные (создать еще один объект структуры) либо выйти из программы. Если пользователь продолжает ввод, необходимо выделить новую память для указателя на объект структуры и дописать в нее введенные пользователем новые данные. Цикл выделения новой памяти продолжается, пока пользователь не выберет выход из программы.

```

Имя: Alina
Возраст: 22
Элементы структуры объекта P: Alina 22;
Внести еще данные – 1, выход – 0: 1
Имя: Danila
Возраст: 30
0-е элементы структуры объекта P: Alina 22;
1-е элементы структуры объекта P: Danila 30;
Внести еще данные – 1, выход – 0: 1
Имя: Fagot
Возраст: 34
0-е элементы структуры объекта P: Alina 22;
1-е элементы структуры объекта P: Danila 30;
2-е элементы структуры объекта P: Fagot 34;
Внести еще данные – 1, выход – 0: 0

Для продолжения нажмите любую клавишу

```

6. Требуется написать программу, которая заполняет массив размерности  $n \times n$  по заданному правилу:

```
/*  1  3  4 10 11
    2  5  9 12 19
    6  8 13 18 20
    7 14 17 21 24
    15 16 22 23 25 */
```

То есть заполнение массива должно быть по диагонали, сверху вниз, слева направо. Причем заполнение еще и зигзагообразное.

```
1  3  4 10 11 21 22 36 37 55
2  5  9 12 20 23 35 38 54 56
6  8 13 19 24 34 39 53 57 72
7 14 18 25 33 40 52 58 71 73
15 17 26 32 41 51 59 70 74 85
16 27 31 42 50 60 69 75 84 86
28 30 43 49 61 68 76 83 87 94
29 44 48 62 67 77 82 88 93 95
45 47 63 66 78 81 89 92 96 99
46 64 65 79 80 90 91 97 98 100
```

7. По заданному вещественному  $x$  вычислить корень кубический из  $x$  по следующей итерационной формуле:  $y_{i+1} = 0.5 (y_i + 3x / (2y_i^2 + x / y_i))$ .

Начальное приближение  $y_0 = x$ . Итерации прекратить при  $|y_{i+1} - y_i| < 10^{-5}$ . Смысл данного задания – найти корень кубический без использования специальных функций.

Введите число  $X = 27$

Ответ = 3

8. Программа должна выполнять преобразования строки (длина строки 255 символов): СТРОКА ЗАГЛАВНЫМИ БУКВАМИ строка

в нижнем регистре С Заглавной Буквы (Первый Символ Каждого Слова В Строке) пЕРВЫЙ сИМВОЛ в НИЖНЕМ РЕГИСТРЕ Как в предложении (с заглавной буквы). Символ 'f' – выход из программы. Организовать в программе меню, через которое можно удобно выбирать любое действие. Программа должна выполняться, пока пользователь не введет символ 'f'. Чтобы преобразовать строчные буквы в прописные, в C++ есть функция toupper. Для преобразования строки в нижний регистр используем функцию tolower. Остальные преобразования можно выполнить также с помощью функций библиотеки ctype.

Введите строку

This program is written Dmitry Kozharsky

Для преобразования строки в заглавные буквы нажмите : 1

Для преобразования строки в нижний регистр нажмите : 2

Для преобразования строки с заглавной буквы нажмите : 3

Для преобразования строки в первый символ в нижнем регистре нажмите : 4

Для преобразования строки как в обычном предложении нажмите : 5

Для выхода нажмите : f

Введите необходимое действие : 1

THIS PROGRAM IS WRITTEN DMITRY KOZHARSKY

Для преобразования строки в заглавные буквы нажмите : 1

Для преобразования строки в нижний регистр нажмите : 2

Для преобразования строки с заглавной Буквы нажмите : 3

Для преобразования строки в первый символ в нижнем регистре нажмите : 4

Для преобразования строки как в обычном предложении нажмите : 5

Для выхода нажмите : f

Введите необходимое действие : 2

this program is written dmitry kozharsky

Для преобразования строки в заглавные буквы нажмите : 1

Для преобразования строки в нижний регистр нажмите : 2

Для преобразования строки с заглавной буквы нажмите : 3

Для преобразования строки в первый символ в нижнем регистре нажмите : 4

Для преобразования строки как в обычном предложении нажмите : 5

Для выхода нажмите : f

Введите необходимое действие : 3

This Program Is Written Dmitry Kozharsky

Для преобразования строки в заглавные буквы нажмите : 1  
 Для преобразования строки в нижний регистр нажмите : 2  
 Для преобразования строки с заглавной буквы нажмите : 3  
 Для преобразования строки в первый символ в нижнем регистре нажмите : 4  
 Для преобразования строки как в обычном предложении нажмите : 5  
 Для выхода нажмите : f

Введите необходимое действие : 4

tHIS pROGRAM iS wRITTEN dMITRY kOZHARSKY

Для преобразования строки в заглавные буквы нажмите : 1  
 Для преобразования строки в нижний регистр нажмите : 2  
 Для преобразования строки с заглавной буквы нажмите : 3  
 Для преобразования строки в первый символ в нижнем регистре нажмите : 4  
 Для преобразования строки как в обычном предложении нажмите : 5  
 Для выхода нажмите : f

Введите необходимое действие : 5

This program is written dmitry kozharsky

Для преобразования строки в заглавные буквы нажмите : 1  
 Для преобразования строки в нижний регистр нажмите : 2  
 Для преобразования строки с заглавной буквы нажмите : 3  
 Для преобразования строки в первый символ в нижнем регистре нажмите : 4  
 Для преобразования строки как в обычном предложении нажмите : 5  
 Для выхода нажмите : f

Введите необходимое действие : f

9. Составить программу, которая будет генерировать случайные числа в интервале [a;b], и заполнять ими двумерный массив размером 10 на 10. В массиве необходимо найти номер строки с минимальным элементом. Поменять строки массива местами, строку с минимальным элементом и первую строку массива. Организовать удобный вывод на экран.

Кроме создания двумерных динамических массивов вам понадобится генератор случайных чисел.

Введите число a: 3  
 Введите число b: 9  
 Сгенерированный массив вещественных чисел  
 4.331 3.406 5.992 7.247 4.044 7.214 5.284 8.227 8.799 3.479  
 3.458 6.928 8.264 7.817 3.798 7.376 8.919 3.802 7.556 6.523  
 5.332 6.940 5.754 8.261 3.766 5.582 4.351 3.658 6.841 3.788

```

3.602 8.172 4.194 6.593 6.419 5.238 4.808 8.703 4.466 4.606
3.182 4.924 8.534 8.445 3.741 3.332 6.821 3.660 4.134 5.377
7.183 6.466 3.318 3.937 5.726 4.084 6.519 7.078 4.742 4.360
7.865 5.344 3.531 3.060 8.937 6.950 5.298 4.745 6.653 6.764
6.352 6.835 8.688 5.886 6.280 3.429 6.218 4.101 4.089 7.351
6.479 8.273 4.817 6.797 3.210 7.543 7.880 6.729 5.621 3.623
8.089 4.486 5.967 8.620 4.546 5.904 6.570 6.844 7.649 4.223
Номер строки с минимальным элементом: 7
Минимальный элемент: 3.060
Массив с перестановленными строками
7.865 5.344 3.531 3.060 8.937 6.950 5.298 4.745 6.653 6.764
3.458 6.928 8.264 7.817 3.798 7.376 8.919 3.802 7.556 6.523
5.332 6.940 5.754 8.261 3.766 5.582 4.351 3.658 6.841 3.788
3.602 8.172 4.194 6.593 6.419 5.238 4.808 8.703 4.466 4.606
3.182 4.924 8.534 8.445 3.741 3.332 6.821 3.660 4.134 5.377
7.183 6.466 3.318 3.937 5.726 4.084 6.519 7.078 4.742 4.360
4.331 3.406 5.992 7.247 4.044 7.214 5.284 8.227 8.799 3.479
6.352 6.835 8.688 5.886 6.280 3.429 6.218 4.101 4.089 7.351
6.479 8.273 4.817 6.797 3.210 7.543 7.880 6.729 5.621 3.623
8.089 4.486 5.967 8.620 4.546 5.904 6.570 6.844 7.649 4.223

```

10. Подсчитать количество повторений элементов, заданного множества символов, во введенной строке. Суть задачи такова: вводим строку символов, после чего подсчитываем количество повторений каждого символа в строке. Таким образом, количество повторений любого символа будет равно как минимум единице. Например, если строка состоит из 10 различных символов, значит в выводе программы должно быть 10 чисел, каждое из которых характеризует количество вхождений данного символа в строку.

Ввести строку:

Nightwish

N	1
g	1
h	2
i	2
s	1
t	1
w	1

11. Дан массив размера n, заполнить его случайными числами. Найти все нечётные числа массива.

Введите размер массива = 20

Массив = 60 78 78 77 75 94 49 7 8 81 35 94 8 38 29 31 76 42 12 67

Результат = 77 75 49 7 81 35 29 31 67

12. Разработать программу, в которой будет организовано меню. Выбор функций меню должен быть организован по функциональной клавише. Вся информация должна храниться в массиве структур, с возможностью их записи в файл. Организовать сортировку данных различными методами (быстрая, Шелла, пузырьковая), вывод результатов сортировки должен быть в табличной форме. Функции программы: функция для ввода данных пользователем; функция для записи данных в файл; чтение данных из файла; вывод данных на экран; дополнение данных; удаление данных; поиск информации по атрибуту; сортировка различными методами; состав и структура исходных данных. Таблица «штатное расписание» с полями: «ФИО работника», «кол-во отработанных часов», «должность», «оплата за час работы», «присваиваемый id», вычисляемое поле «зарплата». Примечание: зарплата = количество отработанных часов умножить на оплату за час работы.

Выберите действие:

N New worker.

E Edit data.

F Find worker.

D Delete data.

S Save to file.

R Load from file.

V Show all workers.

C Sort data.

Q Quit.

n

Enter data:

id: 1

Surname: Hume  
Name: Desmond  
Patronymic: ---  
Post: athlete  
Hr. pay: 40  
Hours: 4320

Выберите действие:

N New worker.  
E Edit data.  
F Find worker.  
D Delete data.  
S Save to file.  
R Load from file.  
V Show all workers.  
C Sort data.  
Q Quit.

n

Enter data:

id: 2

Surname: Shephard

Name: Jack

Patronymic: ---

Post: leader

Hr. pay: 100

Hours: 5235

Выберите действие:

N New worker.  
E Edit data.  
F Find worker.  
D Delete data.  
S Save to file.  
R Load from file.  
V Show all workers.  
C Sort data.  
Q Quit.

n

Enter data:

id: 3

Surname: Ford

Name: James

Patronymic: ---

Post: killer

Hr. pay: 43

Hours: 345

Выберите действие:

N New worker.  
E Edit data.

```

F Find worker.
D Delete data.
S Save to file.
R Load from file.
V Show all workers.
C Sort data.
Q Quit.
v
Worker data
ID   Surname  Name Patronymic  Post  h/pay Hours Salary
-----
1    Hume Desmond  --- athlete    40 4320 172800
2    Shephard  Jack   --- leader    100 5235 523500
3    Ford James   --- killer    43 345 14835
Выберите действие:
N New worker.
E Edit data.
F Find worker.
D Delete data.
S Save to file.
R Load from file.
V Show all workers.
C Sort data.
Q Quit.
q

```

13. Время задается в формате час, минута, секунда. Реализовать:

- 1) вычитание из времени указанного пользователем количества секунд;
- 2) подсчёт числа секунд между двумя моментами времени, лежащими в пределах одних суток.

```

Введите значения времени!

часы: 4
минуты: 32
секунды: 47

Введите количество секунд: 145
Оставшееся время: 4:30:22
Введите первый момент времени
Введите значения времени!
часы: 12
минуты: 10
секунды: 12

```

```

Введите второй момент времени
Введите значения времени!
часы: 18
минуты: 45
секунды: 0

Количество секунд между введенными моментами времени: 23688

```

14. Создать класс, описывающий понятие «работник» со свойствами: фамилия; стаж; часовая заработная плата; количество отработанных часов.

С помощью метода реализовать ввод данных работника с клавиатуры. Рассчитать с помощью методов заработную плату за отработанное время и премию, размер которой определяется в зависимости от стажа (при стаже до 1 года 0%, до 3 лет 5%, до 5 лет 8%, свыше 5 лет 15%). С помощью метода печати реализовать вывод информации о работнике на экран. Предусмотреть метод для записи в файл данных о работнике.

```

Введите имя рабочего: Denis
Введите стаж рабочего: 2
Введите почасовую заработную плату: 50
Введите отработанное число Denis: 2450

Denis
Стаж = 2
Почасовая зарплата = 50
Отработанные Denis часов = 2450
Зарплата = 122500
Премия = 6125
Информация о Denis представлена в файле 'Workers.txt'

```

15. Дана матрица  $A$  размерностью  $n \times m$ . Записать все элементы матрицы в одномерный массив.

Для решения этой задачи организовать заполнение массива случайными числами. Это сократит время на ввод элементов двумерного массива. Задача создать одномерный массив, в котором могут поместиться все элементы

двумерного. Для этого определяем размер двумерного массива, умножаем количество строк на количество столбцов. Далее динамически выделяем память под одномерный массив. Используя несколько циклов `for`, заполняем одномерный массив и выводим на экран.

```
Введите количество строк матрицы: 3
Введите количество столбцов матрицы: 5
4 46 83 2 79
78 49 65 52 73
94 98 61 4 82
4 46 83 2 79 78 49 65 52 73 94 98 61 4 82
```

16. Выполнить преобразование матрицы, а именно перестановку строк и столбцов. Для квадратной матрицы размером `n` переставляйте столбцы и строки таким образом, чтобы элемент матрицы с наибольшим значением по модулю располагался в нижнем правом углу матрицы. Заполнение исходной матрицы организовать с клавиатуры.

В этом задании приходится манипулировать только строками и столбцами двумерного массива. Заметьте, матрица квадратная, а значит количество строк и столбцов должно быть одинаково, с помощью функции `abs` находим модуль значения.

Например,

```
4 8 9 1
3 0 1 1 – это исходная матрица.
9 8 1 2
5 5 5 5
```

9 – наибольшее значение по модулю, находится в первой строке и в третьем столбце. Необходимо этот элемент переместить в правый нижний угол, но мы имеем право переставлять только строки и столбцы. Отдельно элемент перемещать нельзя.

Первая итерация:

5 5 5 5

3 0 1 1 – поменяли первую и последнюю строку местами.

9 8 1 2

4 8 9 1

Вторая итерация:

5 5 5 5

3 0 1 1 – поменяли последний и предпоследний столбец местами.

9 8 2 1

4 8 1 9

И теперь элемент со значением 9 стоит в правом нижнем углу. Задача решена.

17. Поменять согласные буквы на гласные во введенной пользователем строке, а гласные – на соответствующий ASCII код, используя функции.

Введите строку букв: cppstudio  
Результат: O E i o o 117 e 105 111

## Коды управляющих символов (0–31)

Код	Обозначение	Клавиша	Значение	Отображаемый символ
1	2	3	4	5
0	<i>nul</i>	$\wedge @$	Нуль	
1	<i>soh</i>	$\wedge A$	Начало заголовка	☺
2	<i>stx</i>	$\wedge B$	Начало текста	☹
3	<i>etx</i>	$\wedge C$	Конец текста	♥
4	<i>eot</i>	$\wedge D$	Конец передачи	♦
5	<i>enq</i>	$\wedge E$	Запрос	♣
6	<i>ack</i>	$\wedge F$	Подтверждение	♠
7	<i>bel</i>	$\wedge G$	Сигнал (звонок)	•
8	<i>bs</i>	$\wedge H$	Забой (шаг назад)	▣
9	<i>ht</i>	$\wedge I$	Горизонтальная табуляция	○
10	<i>lf</i>	$\wedge J$	Перевод строки	▣
11	<i>vt</i>	$\wedge K$	Вертикальная табуляция	♂
12	<i>ff</i>	$\wedge L$	Новая страница	♀
13	<i>cr</i>	$\wedge M$	Возврат каретки	♪
14	<i>so</i>	$\wedge N$	Выключить сдвиг	♪
15	<i>si</i>	$\wedge O$	Включить сдвиг	☀
16	<i>dle</i>	$\wedge P$	Ключ связи данных	▶
17	<i>dc1</i>	$\wedge Q$	Управление устройством 1	◀
18	<i>dc2</i>	$\wedge R$	Управление устройством 2	↕
19	<i>dc3</i>	$\wedge S$	Управление устройством 3	!!
20	<i>dc4</i>	$\wedge T$	Управление устройством 4	¶
21	<i>nak</i>	$\wedge U$	Отрицательное подтверждение	§
22	<i>syn</i>	$\wedge V$	Синхронизация	—
23	<i>etb</i>	$\wedge W$	Конец передаваемого блока	↕
24	<i>can</i>	$\wedge X$	Отказ	↑
25	<i>em</i>	$\wedge Y$	Конец среды	↓
26	<i>sub</i>	$\wedge Z$	Замена	→
27	<i>esc</i>	$\wedge [$	Ключ	←
28	<i>fs</i>	$\wedge \backslash$	Разделитель файлов	└
29	<i>gs</i>	$\wedge ]$	Разделитель группы	↔
30	<i>rs</i>	$\wedge \wedge$	Разделитель записей	▲
31	<i>us</i>	$\wedge \_$	Разделитель модулей	▼

32	пробел	56	8	80	P	104	H
33	!	57	9	81	Q	105	I
34	"	58	:	82	R	106	J
35	#	59	;	83	S	107	K
36	\$	60	<	84	T	108	L
37	%	61	=	85	U	109	m
38	&	62	>	86	V	110	n
39	'	63	?	87	W	111	o
40	(	64	@	88	X	112	p
41	)	65	A	89	Y	113	q
42	*	66	B	90	Z	114	r
43	+	67	C	91	[	115	s
44	,	68	D	92	\	116	t
45	-	69	E	93	]	117	u
46	.	70	F	94	^	118	v
47	/	71	G	95	_	119	w
48	0	72	H	96	`	120	x
49	1	73	I	97	A	121	y
50	2	74	J	98	b	122	z
51	3	75	K	99	c	123	{
52	4	76	L	100	d	124	
53	5	77	M	101	e	125	}
54	6	78	N	102	f	126	~
55	7	79	O	103	g	127	del

## Символы с кодами 128–255 (Кодовая таблица 866 – MS-DOS)

Код	Символ	Код	Символ	Код	Символ	Код	Символ
128	А	160	а	192	Љ	224	р
129	Б	161	б	193	Њ	225	с
130	В	162	в	194	Ћ	226	т
131	Г	163	г	195	Ќ	227	у
132	Д	164	д	196	—	228	ф
133	Е	165	е	197	†	229	х
134	Ж	166	ж	198	‡	230	ц
135	З	167	з	199	‡	231	ч
136	И	168	и	200	Ў	232	ш
137	Й	169	й	201	Ѓ	233	щ
138	К	170	к	202	Ќ	234	ъ
139	Л	171	л	203	Ћ	235	ы
140	М	172	м	204	Ќ	236	ь
141	Н	173	н	205	=	237	э
142	О	174	о	206	†	238	ю
143	П	175	п	207	—	239	я
144	Р	176	░	208	Ќ	240	Ё
145	С	177	▒	209	Ћ	241	ё
146	Т	178	▓	210	Ќ	242	Є
147	У	179		211	Ќ	243	е
148	Ф	180	└	212	Ќ	244	ї
149	Х	181	┘	213	Ѓ	245	і
150	Ц	182	┐	214	Ѓ	246	Ÿ
151	Ч	183	┌	215	†	247	ŷ
152	Ш	184	└	216	†	248	°
153	Щ	185	┘	217	┘	249	·
154	Ъ	186	┐	218	Ѓ	250	·
155	Ы	187	┌	219	▀	251	√
156	Ь	188	┐	220	▀	252	№
157	Э	189	┌	221	▀	253	¤
158	Ю	190	┘	222	▀	254	■
159	Я	191	└	223	▀	255	

## Символы с кодами 128–255 (Кодовая таблица 1251 – MS Windows)

Код	Символ	Код	Символ	Код	Символ	Код	Символ
128	Ъ	160		192	А	224	а
129	Ѓ	161	Ў	193	Б	225	б
130	,	162	ў	194	В	226	в
131	ѓ	163	Ј	195	Г	227	г
132	„	164	џ	196	Д	228	д
133	…	165	Ѓ	197	Е	229	е
134	†	166	Ї	198	Ж	230	ж
135	‡	167	Ѕ	199	З	231	з
136	€	168	Ё	200	И	232	и
137	‰	169	©	201	Й	233	й
138	Љ	170	Є	202	К	234	к
139	<	171	«	203	Л	235	л
140	Њ	172	¬	204	М	236	м
141	Ќ	173	–	205	Н	237	н
142	Ѧ	174	®	206	О	238	о
143	Ѣ	175	Ї	207	П	239	п
144	ђ	176	°	208	Р	240	р
145	`	177	±	209	С	241	с
146	’	178	І	210	Т	242	т
147	“	179	і	211	У	243	у
148	”	180	ѓ	212	Ф	244	ф
149	•	181	μ	213	Х	245	х
150	—	182	Ў	214	Ц	246	ц
151	—	183	·	215	Ч	247	ч
152		184	ё	216	Ш	248	ш
153	™	185	№	217	Щ	249	щ
154	љ	186	є	218	Ъ	250	ъ
155	>	187	»	219	Ы	251	ы
156	њ	188	ј	220	Ь	252	ь
157	ќ	189	ѕ	221	Э	253	э
158	ћ	190	ѕ	222	Ю	254	ю
159	џ	191	ї	223	Я	255	я

ASCII (American Standard Code for Information Interchange – Стандартный американский код обмена информацией) – это код для представления `ascii.org.ru` символов в виде чисел, в котором каждому символу сопоставлено число от 0 до 127. В большинстве компьютеров код ASCII используется для представления текста, что позволяет передавать данные с одного компьютера на другой. Стандартный набор символов ASCII использует только 7 битов для каждого символа. Добавление 8-го разряда позволяет увеличить количество кодов таблицы ASCII до 255. Коды от 128 до 255 представляют собой расширение таблицы ASCII. Эти коды используются для кодирования символов национальных алфавитов, а также символов псевдографики, которые можно использовать, например, для оформления в тексте различных рамок и текстовых таблиц.

18. Дан текст, который пользователь должен ввести в программу, найти наибольшее количество идущих подряд букв. Реализовать следующие возможности в программе: меню пользователя, состоящее как минимум из 4 пунктов: ввод данных (текст вводится пользователем); обработка данных (поиск наибольшего количества идущих подряд букв); вывод результата на стандартное устройство вывода (экран); выход из программы. Организовать промежуточное хранение результата, это относится к пункту меню «вывод результата»; разбить программу на функции.

```

Меню:
1-Введите строку
2-Преобразовать данные
3-Поглядеть результат и вывести в файл
0-Выход

1
Введите строку:
Аппетит приходит во время еды.
2
Преобразовать данные
3
Поглядеть результат

```

Ваш результат = 2

Для выхода нажмите 0

19. Необходимо разработать программу, которая предоставляет удобный функционал для работы с разреженной матрицей. Разреженная матрица – матрица с большим количеством нулевых элементов. Требуется написать следующие функции: Ввода матрицы; Печати матрицы; Суммирования двух матриц; Умножения двух матриц.

Представлять матрицы можно в виде списка или в виде динамического массива.

20. Разработать программу для построения графика функции  $y(x) = 2 \cdot \sin(x)^{4/5}$  или иной другой. Вывести в файл таблицу со значениями  $x$ ,  $y(x)$ . График функции построить в интервале от -50 до 50 с шагом 1. Результатом работы программы будут считаться образ функции на экране и файл, с элементами значений  $x$  и  $y(x)$  для интервала  $[-50; 50]$ .

21. Результаты соревнований по прыжкам в длину представлены в виде матрицы 5x3 (5 спортсменов по 3 попытки у каждого). Указать, какой спортсмен и в какой попытке показал наилучший результат. Результаты соревнований представляем в виде двумерного массива. После заполнения матрицы необходимо определить максимальное значение прыжка, т. е. нужно организовать поиск максимального значения в двумерном массиве.

Введите результаты соревнований:  
 для 1 спортсмена: 2.60 2.90 3.10  
 для 2 спортсмена: 2.90 2.95 3.00  
 для 3 спортсмена: 3.45 3.15 3.16  
 для 4 спортсмена: 3.00 3.10 2.90  
 для 5 спортсмена: 3.15 3.20 3.45  
 лучший результат у спортсменов:  
 № спортсмена – 3

№ попытки – 1  
 № спортсмена – 5  
 № попытки – 3

22. Организовать ввод строки, каждое слово в строке отделяется от других слов пробелами, их может быть неограниченное количество. Найти самое длинное и самое короткое слово в строке.

Введите строку:  
 Прокладывай себе дорогу в жизни с умом.

Результат минимальное слово:  
 Себе умом

Результат максимальное слово:  
 Прокладывай

23. Есть координаты двух клеток шахматной доски. Вывести Yes, если слон может за один ход перейти с одной клетки шахматной доски на другую [8x8]. В другом случае вывести No.

Enter x1: 3  
 Enter y1: 4  
 Enter x2: 1  
 Enter y2: 1  
 No

24. Есть координаты двух клеток шахматной доски. Вывести Yes, если конь может за один ход перейти с одной клетки шахматной доски на другую [8x8]. В другом случае вывести No.

Enter x1: 3  
 Enter y1: 4  
 Enter x2: 1  
 Enter y2: 1  
 No

25. Есть координаты двух клеток шахматной доски. Вывести Yes, если ладья может за один ход перейти с одной клетки шахматной доски на другую [8x8]. В другом случае вывести No.

```
Enter x1: 3
Enter y1: 4
Enter x2: 1
Enter y2: 1
No
```

## **ПРИЛОЖЕНИЕ В**

### **Теоретические вопросы**

1. Декларация структур. Инициализация и доступ к элементам структуры.
2. Вложенные структуры и массивы структур.
3. Объединения.
4. Указатели и адреса: указатели и массивы; связь указателей с массивами; элементы массива; символьные массивы; многомерные массивы; массивы указателей; массивы динамической памяти.
5. Определение класса. Public, Private и Protected части определения класса.
6. Функции – элементы класса и функции-друзья.
7. Объекты класса. Создание и уничтожение объектов класса. Конструкторы и деструкторы.
8. Статические члены объектов класса.
9. Вложенные и локальные классы.
10. Члены-данные и члены-функции. Клиенты класса. Способы задания доступа к членам класса.
11. Использование предопределенных и пользовательских типов, массивов и указателей.
12. Определение и объявление членов-функций в классе. Доступ членов-функций к членам класса.
13. Параметры членов-функций. Способы передачи значений в тело и из тела функции.
14. Встроенные функции: определение, семантика, использование.
15. Определение членов-функций вне класса: синтаксис и семантика.
16. Базовый и производный классы.

17. Функции-элементы и функции-друзья. Правила доступа к элементам производного класса.

18. Иерархия классов.

19. Одиночное и множественное наследование.

20. Виртуальные базовые классы. Особенности доступа при множественном наследовании.

21. Вызов объектов через указатели.

22. Перегрузка операций. Виртуальные функции. Таблицы виртуальных функций.

23. Чистые виртуальные функции и абстрактные базовые классы.

24. Шаблоны классов и функций. Наследование шаблонных классов. Правила отождествления параметров шаблона. Применение шаблонных классов для создания контейнерных классов.

25. Принципы объектно-ориентированного программирования.