

Владимирский государственный университет

М. В. ШИШКИНА

УЧЕБНАЯ ПРАКТИКА

Практикум

Владимир 2024

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Владимирский государственный университет
имени Александра Григорьевича и Николая Григорьевича Столетовых»

М. В. ШИШКИНА

УЧЕБНАЯ ПРАКТИКА

Практикум

Электронное издание



Владимир 2024

ISBN 978-5-9984-1815-0

© ВлГУ, 2024

УДК 004.42
ББК 32.973-018

Рецензенты:

Кандидат технических наук
генеральный директор ООО «ФС Сервис»
Д. С. Квасов

Кандидат физико-математических наук, доцент
зав. кафедрой функционального анализа и его приложений
Владимирского государственного университета
имени Александра Григорьевича и Николая Григорьевича Столетовых
В. Д. Бурков

Издаётся по решению редакционно-издательского совета ВлГУ

Шишкина, М. В.

Учебная практика [Электронный ресурс] : практикум /
М. В. Шишкина ; Владим. гос. ун-т им. А. Г. и Н. Г. Столетовых. –
Владимир : Изд-во ВлГУ, 2024. – 128 с. – ISBN 978-5-9984-1815-0. –
Электрон. дан. (1,61 Мб). – 1 электрон. опт. диск (CD-ROM). – Си-
стем. требования: Intel от 1,3 ГГц ; Windows XP/7/8/10 ; Adobe Reader ;
дисковод CD-ROM. – Загл. с титул. экрана.

Приведены теоретические аспекты основ программирования с демонстрацией достаточного количества примеров, задания для выполнения во время учебной практики студентами первого курса, упражнения для практической и самостоятельной работы, а также контрольные вопросы для закрепления материала.

Предназначен для студентов бакалавриата направления подготовки 01.03.02 «Прикладная математика и информатика».

Рекомендовано для формирования профессиональных компетенций в соответствии с ФГОС ВО.

Ил. 24. Табл. 4. Библиогр.: 7 назв.

ISBN 978-5-9984-1815-0

© ВлГУ, 2024

ОГЛАВЛЕНИЕ

ПРЕДИСЛОВИЕ.....	4
ВВЕДЕНИЕ	5
1. ОСНОВНЫЕ АЛГОРИТМИЧЕСКИЕ СТРУКТУРЫ.....	6
2. БАЗОВЫЕ ТИПЫ ЯЗЫКА C++	11
3. ОПЕРАТОРЫ ВЕТВЛЕНИЯ ЯЗЫКА C++	18
4. ОПЕРАТОРЫ ОРГАНИЗАЦИИ ЦИКЛОВ ЯЗЫКА C++	26
5. УКАЗАТЕЛИ И ССЫЛКИ.....	35
6. МАССИВЫ.....	42
7. ПОЛЬЗОВАТЕЛЬСКИЕ ТИПЫ ДАННЫХ. СТРУКТУРЫ.....	51
8. ФУНКЦИИ	57
9. ЛИНЕЙНЫЕ ДИНАМИЧЕСКИЕ СТРУКТУРЫ ДАННЫХ.....	64
10. ВВЕДЕНИЕ В ЯЗЫК ПРОГРАММИРОВАНИЯ <i>PYTHON</i>	76
11. ОПЕРАТОР ВЕТВЛЕНИЯ В ЯЗЫКЕ <i>PYTHON</i>	82
12. ОПЕРАТОРЫ ОРГАНИЗАЦИИ ЦИКЛОВ ЯЗЫКА <i>PYTHON</i>	89
13. СТРОКИ.....	97
14. МАССИВЫ. СПИСКИ.....	103
15. ФАЙЛЫ	110
16. ПРОЦЕДУРЫ. ФУНКЦИИ	114
ЗАКЛЮЧЕНИЕ.....	120
РЕКОМЕНДАТЕЛЬНЫЙ БИБЛИОГРАФИЧЕСКИЙ СПИСОК	121
ПРИЛОЖЕНИЯ	122

ПРЕДИСЛОВИЕ

Учебная практика студентов, обучающихся по направлению подготовки 01.03.02 «Прикладная математика и информатика», – один из важных этапов подготовки к научно-исследовательской, проектной и производственной деятельности.

Цель практики – закрепление пройденного материала теоретического курса дисциплин основной профессиональной образовательной программы (ОПОП), получение навыков практического решения прикладных задач, а также первичных профессиональных навыков и умений.

Задачи учебной практики: 1) приобретение студентами навыков решения практических задач в области разработки, алгоритмического решения и программного кода, формирование первичных навыков математического и компьютерного моделирования; 2) приобретение навыков самостоятельной и коллективной работы при решении поставленных задач; 3) закрепление теоретических знаний, полученных во время обучения на первом курсе, а также умений, необходимых для подготовки и оформления отчетов, статей, рефератов на базе современных средств редактирования в соответствии с установленными требованиями; 4) приобретение навыков применения современных информационных технологий.

Выполнение заданий практикума позволяет углубить и закрепить знания, полученные в ходе теоретического обучения.

Практикум состоит из 16 разделов, которые содержат теоретические сведения, достаточное количество примеров, задания для практической и самостоятельной работы, вопросы для закрепления материала.

В приложениях представлены образцы титульного листа отчёта по практике и листа задания на практику (прил. 1 и 2), а также справочная информация.

ВВЕДЕНИЕ

Цель любого обучения – формирование у обучающегося умения применять свои знания на практике. Практика играет важную роль в освоении студентами общепрофессиональных и профессиональных компетенций. Решение практических задач помогает закрепить теоретические сведения, глубже проникнуть в суть изучаемых дисциплин, а также познакомиться с нюансами выбранной профессии, повторить забытые теоретические аспекты и применить их для решения реальных задач.

Полноценное овладение любой профессией базируется на умении применять на практике теоретические знания. Для получения соответствующих навыков важно не только решение практических задач параллельно с получением и расширением теоретических знаний, но и умение решать практические задачи после освоения теоретических курсов. Такие задачи должны быть комплексными, т. е. для их решения нужно владеть знаниями нескольких дисциплин. Решение достаточного количества таких задач в ходе учебной практики позволит не только усовершенствовать навыки, полученные за период обучения, но и выявить слабые места, которым стоит уделить больше внимания в ходе дальнейшего обучения. Решение практических задач дает возможность понять необходимость вдумчивого и тщательного изучения теоретических основ профессии, что в свою очередь закладывает прочный фундамент для дальнейшего обучения. Студенты, успешно справившиеся с обучением на первом курсе, могут проявить самостоятельность, инициативу и творческий подход при решении задач учебной практики. Кроме того, решение практических задач помогает выявить сильные стороны студента, область его профессиональных интересов и определиться с выбором места дальнейшего трудоустройства. Освоившие курс могут принимать самостоятельное обоснованное решение при выборе средств разработки программного продукта.

1. ОСНОВНЫЕ АЛГОРИТМИЧЕСКИЕ СТРУКТУРЫ

Алгоритм – четкое предписание исполнителю выполнить определенную последовательность действий, направленных на достижение определённой цели.

От любой другой последовательности действий алгоритм отличаются его *свойства*:

1. *Дискретность* – разбиение алгоритма на последовательность отдельных законченных действий, шагов. Каждый такой шаг должен быть закончен до выполнения следующего.

2. *Точность* – однозначность указаний.

Состояние объектов среды исполнителя однозначно определено на каждом шаге. На каждом шаге однозначно определён шаг, который нужно выполнить следующим. Таким образом, при применении алгоритма к одному набору входных данных на выходе каждый раз будет получен один и тот же результат.

3. *Понятность* – алгоритм должен быть изложен на языке, понятном для исполнителя; таким образом, каждый шаг алгоритма будет трактован однозначно, т. е. состоять из команд, входящих в систему команд исполнителя.

4. *Конечность* (результативность) – обязательное получение результата за конечное число шагов.

Работа алгоритма должна быть завершена за конечное число шагов в любом случае, даже если решение не найдено. Теоретические аспекты бесконечных алгоритмов в рамках учебной практики не рассматриваются.

5. *Массовость* – применение алгоритма к решению всего класса однотипных задач.

Способы представления алгоритма

Выделяются следующие формы записи алгоритмов:

- 1) графическая запись (блок-схемы);
- 2) на естественном языке (словесная запись, псевдокод);
- 3) код на языке программирования;
- 4) в виде математической формулы.

Алгоритм в виде блок-схемы представляет собой последовательность связанных между собой функциональных блоков, соответствующих шагам алгоритма. Блоки соединены между собой линиями, определяющими действие, которое должно быть выполнено следующим.

Представление алгоритма в виде блок-схемы строго формализовано. Инструкции к представлению алгоритма таким образом содержатся в ГОСТ 19.701-90. ЕСПД. Схемы алгоритмов, программ, данных и систем. Обозначения условные и правила выполнения.

Требования к форме и размерам блоков приведены в ГОСТ 19.003-80. Схемы алгоритмов и программ. Обозначения условные и графические.

При изображении блоков размер стороны a выбирается из ряда 15, 10, 20 мм, допускается увеличение значения параметра a на число, кратное 5. Сторона b соотносится со стороной a таким образом, что $b = 1,5a$.

Основные блоки представлены в прил. 3.

Любой алгоритм может быть представлен с использованием трех основных алгоритмических структур: следования, ветвления и цикла. Алгоритмы, содержащие несколько алгоритмических структур, называют комбинированными.

Следование – алгоритмическая структура, в которой все команды выполняются последовательно, одна за другой. Алгоритмы, в которых используется только алгоритмическая конструкция «следование», называются *линейными*.

Ветвление – алгоритмическая структура, в которой в зависимости от значения логического выражения будет выполнено либо одно, либо другое действие.

Существует две формы ветвления: полная и неполная. В полной форме ветвление содержит два действия (последовательности команд), одно из этих действий будет выполнено при значении условия «истина», а второе – при значении условия «ложь». В неполной форме ветвление содержит только одно действие или последовательность команд, которые будут выполнены при значении условия «истина». В прил. 4 приведены структурные схемы полной и неполной форм ветвления.

Алгоритмы, в основе которых лежит алгоритмическая конструкция «ветвление», называют *разветвляющимися*.

Повторение (цикл) – алгоритмическая конструкция, с помощью которой определенная последовательность действий выполнится необходимое число раз. Алгоритмы, основой которых служит конструкция «повторение», называют циклическими, или *циклом*.

Под *телом цикла* понимают действия, многократно повторяющиеся в процессе выполнения цикла.

Выделяют два типа циклов (по взаимному расположению тела цикла и условиям продолжения):

- 1) цикл с *постусловием*;
- 2) цикл с *предусловием*.

При использовании конструкции «цикл с предусловием» проверка условия происходит до выполнения действий тела цикла. Возможна ситуация, в которой тело цикла не выполнится ни разу.

При использовании конструкции «цикл с постусловием» тело цикла будет выполнено как минимум один раз, так как проверка условия происходит после выполнения тела цикла и в зависимости от результата этой проверки будет осуществлён выход из цикла или переход на следующую итерацию.

Для успешной организации алгоритмической конструкции «повторение» следует до входа в цикл задать начальные значения переменных, используемых в цикле. В теле цикла необходимо предусмотреть изменение переменных, анализируемых в условии продолжения цикла.

При решении конкретных задач алгоритм может содержать более одной конструкции «повторение». В зависимости от их взаимного расположения говорят о вложенных или последовательных циклах.

Если один цикл является частью тела другого цикла, то первый цикл называют вложенным, второй – внешним.

Перед решением задачи и написанием кода на языке программирования необходимо представить графическое решение в виде блок-схемы.

Задания для практической работы

1. Даны длины сторон треугольника A , B , C . Найдите площадь треугольника S .
2. Даны координаты вершин треугольника ABC . Найдите его площадь. Составьте блок-схему алгоритма решения поставленной задачи.

3. В квадратной комнате шириной A и высотой B есть окно и дверь с размерами $C \times D$ и $M \times N$ соответственно. Вычислите площадь стен для оклеивания их обоями. Составьте блок-схему алгоритма решения поставленной задачи.

4. Дана величина A , выражающая объем информации в байтах. Переведите A в более крупные единицы измерения информации. Составьте блок-схему алгоритма решения поставленной задачи.

5. Дано значение температуры T в градусах Цельсия. Определите значение этой же температуры в градусах Фаренгейта.

Задания для самостоятельной работы

1. Вычислите путь, пройденный лодкой, если ее скорость в стоячей воде v (км/ч), скорость течения реки v_1 (км/ч), время движения по озеру t_1 (ч), а против течения реки – t_2 (ч). Составьте блок-схему алгоритма решения поставленной задачи.

2. По двум введенным пользователем катетам вычислите длину гипотенузы.

3. Пользователь вводит две буквы. Определите, на каких местах алфавита они стоят и сколько между ними находится букв.

4. Пользователь вводит номер буквы в алфавите. Определите, какая это буква.

5. Пользователь вводит значения двух целочисленных переменных. Необходимо поменять значения переменных так, чтобы значение первой оказалось во второй переменной, а второй – в первой.

6. Выведите на экран вещественные числа с заданным количеством знаков после запятой.

7. Пользователь вводит длину и ширину прямоугольника. Вычислите и выведите на экран его площадь и периметр.

8. Пользователь вводит радиус круга. Вычислите и выведите на экран его площадь и длину дуги окружности.

9. Рассчитайте ежемесячные выплаты по кредиту и суммарную выплату. Данные о кредите вводит пользователь. Кредит составляет n (руб.), берется на y лет, под p %.

10. Пользователь вводит трёхзначное число. Найдите сумму и произведение цифр этого числа.

11. Пользователь вводит координаты двух точек. Выведите уравнение прямой вида: $y = kx + b$.

12. Пользователь вводит два числа. Выполните над ними логические побитовые операции.

13. Даны длины ребер a , b , c прямоугольного параллелепипеда. Найдите его объем и площадь поверхности.

14. Дана длина ребра куба a . Найдите его объем и площадь поверхности.

15. Найдите значение функции $y = 3x^6 - 6x^2 - 7$ при любом введенном значении x .

16. Найдите значение функции $y = 4(x - 3)^6 - 7(x - 3)^3 + 2$ при заданном значении x .

Вопросы для закрепления материала

1. Что такое алгоритм?
2. Какие существуют способы представления алгоритмических решений?
3. Перечислите достоинства и недостатки описания алгоритма на естественном языке.
4. Каковы свойства алгоритма?
5. Какие алгоритмы называют линейными?
6. Для чего в блок-схемах используют комментарии?
7. Для чего при графическом представлении алгоритма используют блок «терминатор»?
8. Для чего в блок-схемах используют блок «соединитель»?
9. Что в блок-схеме означают стрелки?
10. Каково соотношение сторон a и b основных блоков блок-схемы, оформленной в соответствии с ГОСТ 19.003-80?
11. Каково назначение блока «процесс»?
12. Для каких целей используют блок «предопределённый процесс»?
13. Что нужно сделать при необходимости перенести оформление блок-схемы на другую страницу?

2. БАЗОВЫЕ ТИПЫ ЯЗЫКА C++

Перед объявлением переменной или константы необходимо определить, какого она будет типа. Для этого нужно понимать цели, для которых она будет использована, какие значения она может принимать.

Тип данных определяет количество выделяемой памяти под переменную или константу, правила хранения данных этого типа, допустимые операции для данных этого типа.

Количество памяти, выделяемое под переменную, и правила хранения данных определяют диапазон значений данных конкретного типа данных.

Типы данных языка C++ можно разделить на две группы: базовые и пользовательские.

К базовым типам языка относят: `void`, `int`, `char`, `float`, `double`, `bool` (прил. 5).

К целым типам `int`, `short`, `long`, `char` применимы спецификаторы `signed` (знаковый), `unsigned` (беззнаковый).

Использование спецификаторов `short` и `long` перед типом данных `int` ведёт к уменьшению или увеличению количества выделяемой памяти.

Целые беззнаковые числа хранятся в прямом двоичном коде. Диапазон таких данных – от нуля до числа, двоичное представление которого состоит из ряда единиц, длина ряда – количество бит, отводимых под переменную соответствующего типа.

Таким образом, диапазон беззнаковых целых типов данных будет определён следующим образом: $0 - 2^n - 1$, где n – число разрядов, отводимых под число:

0	0	0	0	0	0	0	0	0 – минимальное значение
1	1	1	1	1	1	1	1	255 – максимальное значение

Знаковые целые числа хранятся в дополнительном коде. Старший бит хранит знак: единица – минус, ноль – плюс.

Незначащие разряды, следующие за знаковым, заполняются нулями. Например, если под переменную отведено восемь разрядов, то двоичное число 1001 в памяти будет представлено так:

0	0	0	0	1	0	0	1
---	---	---	---	---	---	---	---

Дополнительный код отрицательного числа m равен $2^k - |m|$, где k – количество разрядов в ячейке.

Для получения дополнительного кода отрицательного числа необходимо:

- модуль отрицательного числа представить в прямом коде;
- инвертировать значение всех бит числа, т. е. все нули заменить на единицы, а единицы – на нули;

– к полученному обратному коду прибавить единицу.

Получим 8-разрядный дополнительный код числа -48 :

00110000 – $|-48| = 48$ – в прямом коде;

11001111 – $|-48|$ – в обратном коде;

11010000 – -48 – в дополнительном коде.

Диапазон целых знаковых чисел – от -2^{n-1} до $2^{n-1} - 1$, где n – число разрядов, отводимых под переменную.

Для хранения вещественных типов данных в C++ определены типы `float`, `double`, `long double`, отличающиеся количеством выделяемой памяти.

При хранении этих типов данных часть разрядов отводится для записи порядка числа, остальные разряды – для записи мантиссы.

Логические переменные могут принимать значения *true* и *false*. Тип таких переменных – `bool`.

Тип данных `void` не имеет значений, он нужен в ряде ситуаций: например, когда правила языка требуют указания типа данных, но необходимости в таком типе нет (если отсутствует возвращаемое значение функции). Переменную типа `void` объявить нельзя.

При работе с символами используют типы `char` и `wchar_t`.

Для хранения символов набора из 256 символов ASCII применяют тип `char`. Для работы с символами, код которых занимает более одного байта, используют тип `wchar_t`.

Для объявления переменной необходимо указать её тип и имя, завершив объявление точкой с запятой:

```
<имя типа> <имя переменной>;  
int i;
```

Можно совместить объявление переменной с инициализацией:

```
<имя типа имя переменной> = <значение>;
```

```
int i = 25;
```

При необходимости объявить несколько однотипных переменных их разделяют запятой:

```
int i, j;
```

При объявлении константы перед её именем необходимо указать ключевое слово `const`. Значение константы нужно задать при объявлении:

```
<имя типа> const <имя константы> = <значение>;
```

```
int const N=7;
```

Правила именования в языке C++

Любой идентификатор (имя) должен начинаться с латинской буквы. Кроме того, имя может содержать цифры и символ подчёркивания. Язык программирования C++ регистрочувствительный. Это означает, что `int z;` и `int Z;` – объявление двух разных переменных. Имена переменных должны нести смысловую нагрузку, т. е. из названия переменной должно следовать, для каких целей она используется.

Место описания идентификатора задаёт его область действия. Переменная, описанная внутри блока, будет локальной, т. е. видна только в этом блоке от точки объявления и ниже. Блоком называют часть кода, ограниченную фигурными скобками, например, тело цикла. Переменную, объявленную вне блока, называют глобальной. Обращение к такой переменной возможно в модуле, в котором она объявлена, от точки объявления и ниже. Если имя локальной и глобальной переменных совпадают, то локальная перекрывает глобальную, т. е. обращение внутри блока будет вестись к локальной переменной.

При использовании в выражениях разнотипных операндов применяют явное или неявное приведение типов. При неявном приведении типа в операции присваивания тип правого операнда приводится к типу левого. При приведении вещественного типа к целочисленному дробная часть отбрасывается. При приведении целого типа к вещественному добавляется нулевая дробная часть. Например:

```
float r = 9.57;
```

```
int i = r; //i=7
```

```
i=w; //r=7.00
```

При использовании явного приведения типа перед операндом необходимо указать тип данных, к которому приводят исходный тип операнда. При этом важно понимать, что приведение типа осуществляется только в указанной точке, сама переменная, её значение и способ хранения данных при этом не изменяются.

Явное приведение возможно двумя способами. Если есть уверенность, что приведение пройдёт без потери данных, используют подход языка C. Перед идентификатором, тип которого приводится, указывают нужный тип в круглых скобках. Например:

```
float w = 4.35;  
int r = (int)w;
```

Если пользователь уверен в том, что переполнение не произойдёт, то применяется оператор `static_cast`:

```
static_cast<тип> (<выражение>)  
int a = 54;  
char ch = i; // неявное преобразование
```

Такой подход приведет к предупреждающему сообщению во время компиляции. Для того чтобы избежать этого, лучше сделать так:

```
int a = 54;  
char ch = static_cast<char>(a);
```

Оператор `%` возвращает остаток от деления, оператор `%=` присваивает левому операнду остаток от деления левого операнда на правый. Соответственно, в результате выполнения выражения `m %= n`; целочисленная переменная `m` примет значение частного текущего значения этой переменной и `n`.

Задания для практической работы

1. Определите количество памяти, отводимое под переменные базовых типов, используя функцию `sizeof()`.

2. Дано n клеток и m зайцев, рассаженных по этим клеткам. Рассчитайте максимальное количество зайцев, которое гарантированно окажется в одной клетке (по принципу Дирихле). Натуральные числа n и m вводит пользователь.

3. В книге на одной странице помещается k строк. Таким образом, на первой странице печатаются строки с 1-й по k -ю, на второй – с $(k + 1)$ -й по $(2k)$ -ю и т. д. Напишите программу, определяющую по

номеру строки в тексте номер страницы, на которой будет напечатана эта строка, и порядковый номер этой строки на странице.

Необходимо ввести: k – количество строк на странице и число n – номер строки в тексте ($1 \leq k \leq 200$, $1 \leq n \leq 20\,000$).

4. Объявите две переменные символьного типа данных $s1$ и $s2$, проинициализируйте их значениями 65 и 48, выведите значения этих переменных, используя функцию `printf()`. Значение каждой переменной необходимо вывести со спецификаторами для символьного типа данных, для целого знакового и беззнакового типов. Объясните результаты, выведенные на экран.

5. Наберите приведённый ниже фрагмент кода. Объясните полученные результаты.

```
a = 5; b = 7;
float y1, y2, f = 1.5;
y1 = f * a / b; y2 = f * (a / b);
std::cout<<"y1="<<y1<< " y2=" << y2 << "\n";
```

Задания для самостоятельной работы

1. Напишите код, который выводит на консоль строку "Hello student!" и переводит курсор на следующую строку. Используйте функцию стандартной библиотеки C++.

2. Объявите последовательно следующие переменные: b – логическая переменная, c – символ, w – широкий символ, i – целое число, f – число с плавающей точкой, d – число с плавающей точкой двойной точности. Каждую переменную объявите в отдельной строке. Проинициализируйте переменные следующими значениями: `true`, `'A'`, 655, 10, `1.1f`, `1.21e22`. Выведите значения переменных на экран в формате: *имя переменной=значение*;

3. Напишите программу «Калькулятор», которая складывает два целых числа a и b . Результат запишите в переменную c и выведите на консоль.

Пример результата работы программы:

Введите первое число: 3

Введите второе число: 3

Сумма чисел = 6

Найдите и исправьте ошибку в представленном ниже коде.


```

int main()
{
int a = 15;
int b = 2;
float c = a/b;
cout << c << endl;
return 0;
}

```

4. Запрограммируйте вычисление следующего выражения:

$x = (a + b - f/a) + fa^2 - (a + b)$. Числа a , b и f должны быть введены с клавиатуры, результат вычислен без объявления и использования переменной x сразу при выводе.

```
cout << "x = " <<...место для формулы...<<
```

Пример работы программы:

Введите числа a, b и f:

a = 15

b = 10

f = 3

x = 675

5. Вычислите и выведите на экран сумму цифр введенного с клавиатуры целого трёхзначного числа.

6. Найдите число десятков целого неотрицательного числа, введенного с клавиатуры, т. е. вторую справа цифру его десятичной записи.

7. Идёт k -я секунда суток. Определите, сколько целых часов h и целых минут m прошло с начала суток.

8. Определите, сколько сейчас целых часов h и целых минут m , если известно, что часовая стрелка повернулась с начала суток на d градусов. Пользователь вводит d такое, что $0 \leq d < 360$.

9. Даны значения двух моментов времени, принадлежащих одним и тем же суткам: часы, минуты, секунды для каждого из моментов времени. Известно, что второй момент времени наступил не раньше первого. Определите, сколько секунд прошло между двумя моментами времени. В первой строке входных данных находятся три целых числа – часы, минуты и секунды первого момента времени, во второй строке – три числа, характеризующие второй момент времени. Число часов лежит в диапазоне от 0 до 23, число минут и секунд – от 0 до 59.

10. Вычислите общую сумму покупки нескольких товаров, например плитки шоколада, кофе и пакета молока.

11. Объявите три переменные типа *int* и присвойте первой числовое значение, вторая переменная должна быть равна первой переменной, увеличенной на 3, а третья переменная – сумме первых двух.

12. Подсчитайте общее количество предметов для сервировки стола, например чашек, блюдец и ложек.

13. Напишите программу для перевода сантиметров в дюймы. *Для справки:* в одном дюйме 2,54 сантиметра.

14. Код, представленный ниже, вычисляет сумму первых N натуральных чисел по формуле суммы арифметической прогрессии:

```
int n;  
cin >> n;  
cout << n * (n + 1) / 2 << "\n";
```

Программа должна работать для всех $N \leq 4\,000\,000\,000$, но для некоторых N она работает неправильно. Найдите причину проблемы и исправьте программу.

Вопросы для закрепления материала

1. Какие базовые типы языка программирования C++ вам известны?

2. Какие типы данных используют для хранения целочисленных значений в языке программирования C++?

3. Каким образом хранятся целые знаковые числа?

4. Что такое переменная?

5. Чем необходимо руководствоваться при выборе типа данных для переменной?

6. Что такое переполнение? В каком случае может возникнуть переполнение типа? К каким ошибкам это может привести?

7. Какие типы данных используют в языке программирования C++ для хранения вещественных переменных?

8. Поясните способ хранения вещественных переменных.

9. Какая операция будет произведена при использовании в выражении разнотипных переменных?

10. По какому правилу осуществляется приведение типов операндов в операторе присваивания?

3. ОПЕРАТОРЫ ВЕТВЛЕНИЯ ЯЗЫКА C++

Условный оператор *if*

Оператор `if` позволяет реализовать алгоритмическую конструкцию «ветвление» на языке программирования C++. Оператор имеет полную и сокращённую формы. Последняя используется при отсутствии действий на ветке «иначе», т. е. при невыполнении условия.

Синтаксис оператора:

```
if (выражение) оператор1; [else оператор 2;]
```

Выполнение оператора начинается с вычисления выражения. Выражение может быть любым: арифметическим, логическим или сложным. В любом случае результат вычисления выражения будет интерпретирован как логическое значение. При этом ноль – «ложь», любое отличное от нуля значение – «истина».

Если результат вычисления выражения – «истина», то выполняется *оператор1*. В противном случае выполняется *оператор2* – при использовании полной формы оператора `if`, или ничего не выполняется, если используется неполная форма оператора ветвления и *оператор2* отсутствует. *Оператором1* и *оператором2* могут быть любые операторы языка программирования C++. Если на месте одного из операторов нужно выполнить несколько действий, используют составной оператор. Для этого эти действия заключают в фигурные скобки. Таким образом не нарушаются правила использования оператора `if`. Рассмотрим несколько примеров.

Полная форма оператора `if`:

```
int c = 7, d = 9, max = 0;
if (c > d) max = c; else max = d;
```

Неполная форма оператора `if`:

```
if (c > d) max = c;
```

Использование составного оператора:

```
if (c < d && (c > d || c == 0)) d++; else { d*= c;
c++;
}
```

Так как *оператор1* и *оператор2* могут быть любыми, значит, на их месте может быть использован оператор `if`. Такое ветвление называют вложенным, например:

```
int c = 6, d = 7, f = 9, max = 0;
if (c > d) { if (c > f) max = c; else max = f; }
    else if (d > f) max = d; else max=f;
cout << "max="<<max;
```

Оператор множественного ветвления *switch*

В ситуациях, когда выражение является целочисленным и возможны более чем два варианта ответа, целесообразнее использовать оператор множественного ветвления `switch`.

Правило использования оператора следующее:

```
switch (<целочисленное выражение>)
{
case константное выражение1: оператор1;
case константное выражение2: оператор2;
case константное выражение3: оператор3;
...
case константное выражениеN-1: операторN-1;

[default: <оператор N>]
}
```

После вычисления выражения его значения последовательно сравнивают со значениями констант. Все константы, используемые в операторе, должны быть уникальными, т. е. не должно быть повторов. Сравнение происходит до первого совпадения. Если совпадение найдено, то последовательно выполняют все операторы, без проверки на совпадение, включая оператор, указанный после ключевого слова `default`. Если необходимо выполнить только действия, относящиеся к определённой константе, следует использовать составной оператор. Последним действием этого оператора должен быть оператор `break`, передающий управление оператору, следующему за оператором `switch`.

Если значение выражения не совпало ни с одним из значений указанных констант, будет выполнен оператор, указанный после ключевого слова `default`. Ключевое слово `default` может отсутствовать: в этом случае никакие действия не будут выполнены, если значение выражения не совпало ни с одной константой.

Рассмотрим простой и наглядный пример:

```
int i = 1;
switch (++i)
{
case 1:printf("1\n");
case 2:printf("1+1=2\n");
case 3:printf("2+1=3\n");
case 4:printf("3+1=4\n");
default:printf("No!\n");
}
```

Результат выполнения этого фрагмента кода будет следующим:

1 + 1 = 2

2 + 1 = 3

3 + 1 = 4

No!

Если в рассмотренном выше примере после каждого оператора использовать оператор `break`, то результат будет другим:

```
int i = 1;
switch (++i)
{
case 1: {printf("1\n"); break; }
case 2: {printf("1+1=2\n"); break; }
case 3: {printf("2+1=3\n"); break; }
case 4: {printf("3+1=4\n"); break; }
default:printf("No!\n");
}
```

В результате выполнения этого фрагмента на экране появится следующая строчка: *1 + 1 = 2*.

Если значение переменной *i* задать отрицательным, то на экране появится строчка «*No!*».

Тернарная операция

Тернарная операция имеет три операнда. Эту операцию используют, когда необходимо не только реализовать ветвление, но и вернуть некоторое значение в точку вызова. Синтаксис операции следующий:

```
(<операнд1>)?<операнд2> :<операнд3>;
```

Операнд1 – выражение, результат которого интерпретируется логически. Если результат вычисления выражения – «истина», будет выполнен *операнд2*, иначе – *операнд3*.

Операнд2 и *операнд3* представляют собой операторы. Результат выполнения соответствующего оператора – результат тернарной операции, это значение будет возвращено в точку вызова.

Рассмотрим примеры применения тернарной операции.

```
float y = 3.2;
float x = -2.9;
float z = (y < x) ? y++ : x++;
cout << "z =" << z << endl;
cout << "x=" << x << endl;
cout << "y=" << y;
```

В результате выполнения этого фрагмента кода на экране появятся следующие строки:

```
z = -2.9
x = -1.9
y = 3.2
```

Рассмотрим менее очевидный пример:

```
int m=1, i, j, k; i = 6; j = 9;
k=i<j?i++:++j?m=i+j, printf("i=%i\n", i) : i;
cout<<"i="<<i<<endl;
cout<<"j="<<j<<endl;
cout<<"m="<<m<<endl;
cout<<"k="<<k<<endl;
```

В результате выполнения на экран будут выведены следующие значения:

```
i = 7
j = 9
m = 1
k = 6
```

Так как первый операнд тернарной операции имеет значение «истина», выполнится второй операнд *i++*, возвращаемое значение – 7 – будет сохранено в переменной *k*. Третий операнд выполнен не будет, поэтому значения переменных *j* и *m* не будут изменены.

Задания для практической работы

1. Пользователь вводит параметр x . Вычислите значение переменной y , зависящей от x :

$$y = \begin{cases} x - 12, & x > 0 \\ 5, & x = 0 \\ x^2, & x < 0 \end{cases}$$

2. Если погода будет хорошая, школьник пойдёт гулять, если плохая, то будет делать уроки. Введите информацию о погоде, выведите сообщение о действиях школьника.

3. При покупке товара на сумму более 1000 руб. предоставляется скидка 15 %. Введите цену товара, вычислите и выведите сумму покупки.

4. Ракета запускается с Земли со скоростью V (км/ч) в направлении движения Земли по орбите вокруг Солнца. Определите результат запуска космической ракеты с Земли в зависимости от скорости V . Известно, что при $V < 7,8$ ракета упадет на Землю; при $7,8 < V < 11,2$ ракета станет спутником Земли; при $11,2 < V < 16,4$ ракета станет спутником Солнца; при $V > 16,4$ ракета покинет Солнечную систему. Значение скорости вводит пользователь.

5. Напишите программу, которая по введенному числу K выводит на экран фразу «Мы нашли в лесу K грибов», причем согласовывает окончание слова «гриб» с числом K . Количество грибов может быть любым целым числом. Окончание фразы определяется значением последней цифры.

Задания для самостоятельной работы

1. Найдите наибольшее целое число из трех введенных с клавиатуры.

2. Определите, является ли введенное пользователем число чётным.

3. Проверьте, делится ли введенное число на 3.

4. Определите, лежит ли точка с указанными координатами в круге радиусом R с центром в начале координат. Значение радиуса вводит пользователь.

5. Пользователь вводит длину стороны квадрата и радиус круга. Определите, какое из утверждений верно: «Круг может быть вписан в квадрат» или «Квадрат может быть вписан в круг».

6. Напишите программу, анализирующую возраст человека и относящую его к одной из четырех групп: дошкольник, ученик, работающий, пенсионер. Возраст человека вводится с клавиатуры.

7. Пользователь вводит два целых числа. Необходимо определить наименьшее число и вывести его на экран. Если числа равны, то выводится сообщение «Числа равны» (без точки).

8. На вход подаётся целое число. Требуется проверить, является ли число четным или нечетным, а также отрицательным или положительным. Нужно вывести следующее сообщение: « N четное/нечетное и отрицательное/положительное.» (с точкой в конце предложения). Если на вход подаётся 0, то нужно вывести: «Вы ввели ноль.» (тоже с точкой).

9. На вход программы подается целое число от 1 до 7 – значения дня недели от понедельника до воскресенья. Нужно вывести, является ли день будним или выходным («Будний»/«Выходной», без точки). Если на вход подаётся число не в диапазоне от 1 до 7, необходимо вывести «Ошибка. Такого дня недели не существует!».

10. На вход поступают две строки – значение символа у игрока 1 и игрока 2 в игре «Камень, ножницы, бумага». Необходимо определить, кто из игроков выиграл, в формате: «Игрок N выиграл» (без точки в конце); N – номер игрока. Если они выбросили одинаковый символ, то на выход подаётся строка «Ничья» (без точки).

11. На вход подается строка – значение месяца, с прописной буквы. Необходимо определить, какому времени года принадлежит месяц, и вывести время года на экран. Если при этом на вход поступила строка, не соответствующая ни одному из месяцев, вывести: «Ошибка. Такого месяца не существует!».

12. На вход подаётся два целых положительных числа – цена на товар до скидки и скидка в процентах. Скидка на товар пробивается только в том случае, если цена до скидки превышает 150 руб., а сама скидка не должна быть больше 50 %. Выведите вещественное число – значение цены на товар после того, как проббили скидку. Если одно из условий скидки не совпадает, необходимо вывести одну из ошибок:

а) «Ошибка 1. Низкая цена» – если цена не превышает 150 руб., но скидка меньше или равна 50 %;

б) «Ошибка 2. Большая скидка» – если скидка больше 50 %, но цена превышает 150 руб.

в) «Ошибка 3. Оба условия» – если оба условия не соблюдаются.

13. Ученик вводит в программу два вещественных числа: сначала свой балл на экзамене, а затем – балл, который он получил в полугодии. Разбалловка в его школе следующая:

$A - 90,1 - 100$

$B - 74,4 - 90$

$C - 60,1 - 74,3$

$FX - < 60,1$

Общая оценка считается как сумма баллов за экзамен и за полугодие. Вычислите общий балл и выведите на экран сначала его, а затем – ту оценку, которую получил ученик.

14. На вход подаются целые числа a , b и c – коэффициенты квадратного уравнения. Необходимо вычислить и вывести на экран вещественные числа – дискриминант и квадратные корни в формате:

Уравнение имеет два корня: « $D = N$. Первый корень = M , второй корень = L », $M > L$, M – бóльший корень.

Уравнение не имеет корней: « $D = N$. Уравнение не имеет корней».

Уравнение имеет один корень: « $D = N$. Единственный корень = X ».

15. На вход программы подаётся два целых числа – расстояние в километрах и вес посылки в килограммах. Рассчитайте стоимость доставки исходя из следующих условий:

Расстояние до 10 км: 100 руб./кг.

Расстояние от 10 до 50 км (включительно): 300 руб./кг.

Расстояние более 50 км: 750 руб./кг.

Скорость доставки – 60 км/ч.

Выведите предложение в формате: «Доставка будет ехать меньше часа/больше часа/ровно час, стоимость – N руб.», где N – целое число.

16. На вход программы подаётся четыре значения, два набора «строка + целое число» – начальная и конечная координаты фигуры (т. е., например, « e » $2 = e2$ – начальная координата, а « f », $4 = f4$ – конечная координата).

Осталось всего два вида фигур – пешки и конь. Определите, возможно ли сделать предложенный ход одной из фигур.

Выведите фразы «Ход допустим для коня», или «Ход допустим для пешки», или «Ход недопустим ни для одной фигуры».

Для справки: конь в шахматах ходит в форме буквы «Г» в любом направлении.

17. На вход подаётся целое число – время в минутах. Необходимо определить, какому часу соответствует время, а также какому времени суток соответствует этот час, в формате: «Время: *N* часов, время суток: "время_суток"». Время должно считаться в 24-часовой системе (т. е. 4 часа дня – это 16 часов). При этом слово «час» должно склоняться, т. е. 21 час, но 12 часов.

Принято считать, что утро продолжается с 4.00 до 11.00, день – с 12.00 до 16.00, вечер – с 17.00 до 23.00, ночь – с 00.00 до 3.00.

Вопросы для закрепления материала

1. Какие два вида ветвления существуют?
2. Приведите структурную схему полного ветвления.
3. Перечислите операторы, с помощью которых можно реализовать ветвления в языке программирования C++.
4. Запишите синтаксис и приведите пример использования условного оператора `if`.
5. Запишите синтаксис и приведите пример использования оператора множественного ветвления `switch case`.
6. В какой ситуации предпочтительнее использовать оператор `if`, а в какой – оператор `switch`?
7. В каком случае будет выполнен оператор, указанный после ключевого слова `default` в операторе множественного выбора?
8. В каком случае ключевое слово `default` в операторе множественного выбора можно опустить?
9. Каким образом осуществляют досрочный выход из оператора `switch`?
10. Запишите синтаксис и приведите пример использования тернарного оператора.
11. Что является возвращаемым значением тернарного оператора?

4. ОПЕРАТОРЫ ОРГАНИЗАЦИИ ЦИКЛОВ ЯЗЫКА C++

Алгоритмическую конструкцию «повторение» реализуют при помощи операторов организации цикла.

В языке программирования C++ существует три оператора организации циклов: `while`, `do while` и `for`.

Независимо от того, каким оператором реализован цикл, необходимо соблюсти три этапа работы с ним: задание начальных значений и/или параметров цикла, выполнение тела цикла, проверка условия продолжения цикла.

Телом цикла называют многократно повторяющиеся действия.

Итерацией называют однократное выполнение тела цикла.

Условие продолжения цикла – выражение, в зависимости от значения которого будет выполнена следующая итерация и осуществлён выход из цикла.

В зависимости от взаимного расположения условия продолжения цикла и тела цикла различают цикл с *предусловием* (условие расположено перед телом цикла) и цикл с *постусловием* (условие расположено после тела цикла).

Структурные схемы цикла с предусловием и постусловием представлены в прил. 4.

Если из условия задачи ясно, что тело цикла необходимо выполнить как минимум один раз, используют цикл с постусловием; если же возможна ситуация, в которой тело цикла не выполнится ни разу, используют цикл с предусловием.

Параметрами цикла называют переменные, используемые в условии продолжения цикла, *счётчиком* цикла – целочисленные параметры цикла, изменяющиеся с определённым шагом.

Для того чтобы цикл работал корректно, в его теле обязательно должно быть предусмотрено изменение параметров цикла. Начальное значение всех обрабатываемых в цикле переменных, включая параметры цикла, должно быть задано до тела цикла.

Цикл с предусловием *while*

Оператор `while` позволяет реализовать цикл с предусловием. Синтаксис оператора:

`while (выражение) оператор;`

Если результат вычисления выражения «истина», будет выполнено тело цикла. Если результат вычисления выражения «ложь», то управление будет передано оператору, следующему за телом цикла. В случаях, когда тело цикла должно содержать более одного оператора, используют фигурные скобки, объединяющие последовательность операторов в один оператор, называемый *составным*. Таким образом, требование к тому, чтобы после выражения стоял оператор «тело цикла», не нарушается.

Пример использования оператора `while` для вычисления произведения пяти введенных с клавиатуры чисел:

```
int p = 1;
int const n = 5;
int a = 0, i = 1;
while (i <= n)
{   printf("a=");
    scanf_s("%i", &a);
    p = p * a; i++;
}
printf("p=%i\n", p);
```

Цикл с постусловием *do...while()*

Оператор `do...while` позволяет реализовать цикл с постусловием. Использовать этот оператор следует в ситуациях, когда из условия задачи понятно, что тело цикла необходимо выполнить как минимум один раз. Синтаксис оператора:

```
do <оператор> while (выражение);
```

После выполнения оператора «тело цикла» вычисляется выражение. В зависимости от результата вычисления выражения будет осуществлён переход на следующую итерацию цикла, если результат «истина», или выход из цикла, если результат – «ложь».

В качестве примера вычислим произведение пяти вводимых с клавиатуры элементов с помощью оператора `do...while`:

```
int const n = 5;
int p = 1, a = 0, i = 1;
do {printf("a=");
    scanf_s("%i", &a);
    p = p * a; i++;
} while (i < n);
printf("p=%i\n", p);
```

Цикл с параметром *for*

Оператор `for` позволяет реализовать цикл с предусловием и условием продолжения. Синтаксис оператора:

```
for (выражение1; выражение2; выражение3;) оператор;
```

Выражение1 называют *инициализатором*. Инициализатор содержит объявления и инициализацию переменных – параметров цикла, выполняется один раз до начала выполнения цикла. Инициализатор может содержать несколько операторов, разделённых запятой.

Выражение2 – условие продолжения цикла. Если результат вычисления выражения – «истина», то выполняется тело цикла, в противном случае управление передаётся оператору, следующему за оператором `for`.

Выражение3 – *итератор*, обычно содержит оператор, изменяющий значение параметра цикла. Итератор будет выполнен после всех операторов тела цикла. Далее снова осуществляется проверка условия – *выражения2*.

Если тело цикла содержит более одного оператора, то его заключают в фигурные скобки.

Вычислим произведение пяти введенных с клавиатуры чисел с использованием оператора `for`:

```
int const n = 5;
int p = 1, a = 0, i = 1;
for (int i = 0; i < n; i++)
{printf("a="); scanf_s("%i", &a);
p*=a; }
printf("p=%i\n", p);
```

Любое из выражений, указанных в круглых скобках после ключевого слова `for`, может быть пропущено. При этом возможна ситуация, в которой все три выражения отсутствуют. Какое бы из выражений ни было опущено, наличие точки с запятой, завершающей это выражение, обязательно.

Если отсутствует первое выражение, инициализация параметра цикла должна быть выполнена до ключевого слова `for`. Если отсутствует второе выражение, выход из цикла необходимо предусмотреть в теле цикла. Обычно для этого используют оператор `break`, передающий управление оператору, следующему сразу за циклом. Если от-

существует третье выражение, необходимо включить итератор в тело цикла.

Внесём изменения в рассмотренный выше пример, опустив все три выражения:

```
int p = 1;
int const n = 5;
int a= 0;
int i = 1;
for(;;)
{if (i>n) break;
    printf("a=");
    scanf_s("%i",&a);
    p = p * a; i++;
}
printf("s=%i\n", s);
```

При необходимости осуществить досрочный выход из цикла используют оператор `break`. После того как оператор `break` будет встречен в теле цикла, управление будет передано оператору, следующему за циклом. Таким образом, операторы, стоящие в теле цикла после ключевого слова `break`, выполнены не будут.

Если в любом из рассмотренных операторов цикла необходимо выполнить досрочный выход, то используют операторы `break` и `return`.

Разберём пример с использованием в теле цикла оператора `break`. Будем вычислять сумму пяти вводимых с клавиатуры чисел. Если на i -м шаге сумма превысит некоторое значение N , то будет выполнен досрочный выход из цикла:

```
int s = 0;
int const n = 5;
int const N = 15;
int a= 0;
int i = 1;
while (i<=n)
{
    printf("a=");
    scanf_s("%i",&a);
    s = s + a; i++;
}
```

```

    if (s>N) break;
}
printf("s=%i\n", s);

```

Использование оператора `return` приводит к досрочному выходу из функции, в теле которой он был использован.

Использование в теле цикла оператора `continue` позволяет перейти на следующую итерацию, минуя оставшиеся операторы тела цикла:

```

for (int count = 0; count <= 30; count++)
    {if ((count % 5) != 0) continue;
      std::cout << count << std::endl; /* Это действие
будет пропущено, если число не делится нацело на 5*/
    }

```

В результате выполнения этого цикла на экран будет выведен столбец чисел от 0 до 30, делящихся нацело на 5.

Задания для практической работы

1. Выведите на экран в одну строку все натуральные числа, не превосходящие 250, которые кратны 13 и не оканчиваются на 8.

2. Пользователь вводит целое положительное число N . Выведите все целые степени четверки, не превосходящие N , в порядке возрастания.

3. На вход программы пользователь вводит целое число N . Выведите факториал этого числа.

4. Пользователь вводит целое число N , $N \geq 3$. Выведите первые N чисел последовательности Фибоначчи.

Для справки: числа Фибоначчи – элементы числовой последовательности 0, 1, 1, 2, 3, 5, 8, 13, 21, ..., в которой первые два числа равны 0 и 1, а каждое последующее число равно сумме двух предыдущих чисел.

5. Пользователь вводит целое положительное число K . Напишите программу, которая будет находить сумму цифр заданного числа. В случае если на вход подаётся отрицательное число, необходимо выйти из программы и вывести текст: "Ошибка!"

Задания для самостоятельной работы

1. С клавиатуры вводится последовательность целых чисел, завершающаяся числом -1 (1 в последовательность не входит). Введенные числа гарантированно находятся в диапазоне от $-10\,000$ до $10\,000$. Выведите на экран минимальное и максимальное числа последовательности через пробел.

2. Пользователь вводит целое положительное число. Проверьте, является ли введенное пользователем число палиндромом (т. е. читается одинаково слева направо и справа налево). В случае если число является палиндромом, необходимо вывести текст « N является палиндромом», где N – число. Иначе – « N не является палиндромом» (без точки в конце).

3. Паша отправился в путешествие на остров, взяв с собой определенное количество бутылок воды. Он знает, что на острове каждый день ему нужно выпить *половину* оставшейся воды и ещё ровно 1 л, чтобы поддерживать себя в форме. Рассчитайте максимальную длительность путешествия (количество полных дней, на которое хватит запасов воды).

4. Пользователь вводит целое число N , $N > 0$. Напишите программу, которая находит и выводит на экран все делители заданного пользователем числа, включая единицу и само число, разделяя делители пробелами.

5. Пользователь вводит целое положительное число в десятичной системе счисления и основание системы счисления от 2 до 9 . Переведите первое число в систему с этим основанием.

6. Пользователь вводит целое положительное число. Выведите два числа: произведение четных и произведение нечетных цифр числа, через пробел. В случае, если четных или нечетных цифр в числе нет, на месте произведения выводится 1 .

7. Рассчитайте, сколько времени потребуется, чтобы вырастить и убрать урожай. Введите целое число N – количество дней, которое необходимо для созревания урожая, $N > 1$. Выведите N строк в следующем формате:

а) если урожай ещё не созрел, то в строке должно быть записано:

Урожай растет. Прошло 1 д.

Урожай растет. Прошло 2 д.

...

и так далее. За первый день считать день посева;

б) при наступлении N дня вывести строку:

«Урожай созрел!»

8. С клавиатуры вводится натуральное число N – число Фибоначчи, имеющее порядковый номер M . $N \neq 1$, так как имеет два порядковых номера последовательности. Найдите и выведите число M – порядковый номер числа Фибоначчи. Последовательность Фибоначчи начинается с 0, 1, 1, ...

9. Реализуйте простой калькулятор, выполняющий операции сложения, вычитания, умножения и деления двух целых чисел.

Калькулятор должен:

а) запрашивать у пользователя выбор операции: сложение (1), вычитание (2), умножение (3) или деление (4);

б) запрашивать два числа для выполнения выбранной операции, каждое в новой строке;

в) выполнять выбранную операцию над этими числами;

г) выводить результат операции.

Как только пользователь при выборе операции вводит 0, калькулятор завершает работу с текстом «Программа завершена.»

Программа должна обрабатывать ошибки ввода. При выборе операции «деление», если делитель равен нулю, выдать текст: «Делить на ноль нельзя!». При вводе неверной операции выдать сообщение: «Неверная команда!».

После вывода сообщения об ошибке программа должна снова сделать запрос на ввод данных.

10. С клавиатуры вводится число N . Выведите на экран N строк, где на четных строках выводится «Я сдам все экзамены», а на нечетных – «Я успешно защищу диплом». Строки на экране должны быть без кавычек.

11. Пользователь вводит целое положительное число N . Напишите программу, которая выводит числа от N до 0 включительно, т. е. выводит числа в порядке убывания. Каждое число выводится в новой строке.

12. Пользователь вводит целое положительное число N . Выведите кубы чисел от 1 до N включительно. Каждое число выводится в новой строке.

13. Пользователь вводит целое неотрицательное число N . Выведите все нечетные числа от 1 до N включительно в одну строку. Оператор ветвления в задаче не использовать.

14. Пользователь вводит целое положительное число N . Вычислите и выведите на экран сумму нечетных чисел от 1 до N включительно.

15. Пользователь вводит целое положительное число N . Вычислите и выведите на экран произведение чисел от 1 до N не включительно, которые делятся на 6, но не делятся на 9. В случае если такие числа не найдены, выведите единицу.

16. Пользователь вводит целое положительное число N . Вычислите и выведите на экран сумму чисел от 1 до N включительно, которые делятся на 3 и оканчиваются на 9. В случае если такие числа не найдены, выведите ноль.

17. Пользователь вводит целое неотрицательное число N . Далее вводит N целых чисел. Найдите количество чисел среди введенных, которые оканчиваются на 763.

18. Пользователь вводит целое неотрицательное число N . Далее вводит N трёхзначных целых чисел. Найдите количество чисел среди введенных, которые начинаются на 6. Выведите сумму цифр этих чисел.

19. Пользователь вводит два целых положительных числа N и M , при этом $N < M$. Найдите и выведите все числа от N до M включительно, такие, где две последние цифры числа одинаковые.

20. Пользователь вводит целое положительное число N . Выведите N строк таких, что в первой строке выведена единица, а в каждой следующей строке по возрастанию через пробел, начиная с единицы и заканчивая числом на единицу большим, чем в предыдущей строке. Таким образом, в последней строке будут числа от единицы до N включительно.

21. Пользователь вводит целое положительное число D . Найдите и выведите на экран все трехзначные числа от 100 до 300 не включительно, кратные D .

22. Пользователь вводит семь целых чисел – температуру воздуха на каждый день недели. Выведите среднюю температуру за неделю. Результат округлите до двух цифр после точки.

23. Пользователь вводит целое неотрицательное число N . Выведите все простые числа от 2 до $(N + 5)$ включительно. Число является

простым, если оно отлично от 1 и делится без остатка только на 1 и на само себя.

24. Пользователь вводит целое положительное число N . Рассматривая все возможные пары натуральных чисел, не превышающих N , найдите, сколько существует таких пар, произведение чисел которых дает полный квадрат, и выведите эти пары на экран.

Пары из одинаковых чисел, стоящих в разном порядке, считать разными. Пары одинаковых чисел считать одинаковыми и вывести один раз.

25. Пользователь вводит целое положительное число N . Найдите и выведите на экран все пары натуральных чисел (x, y) , находящиеся в диапазоне от 1 до 1000 включительно, такие, что разность их квадратов равна заданному числу n .

26. Пользователь вводит два целых положительных числа – N и S . Найдите и выведите на экран все пары натуральных чисел (x, y) от 1 до 500 включительно, такие, что произведение их чисел равно заданному числу N , а сумма чисел равна S . Пары чисел могут повторяться, если при этом числа поменялись местами. Пары одинаковых чисел должны выводиться один раз.

Вопросы для закрепления материала

1. Что называют циклом в программировании?
2. Какой цикл называют циклом с предусловием?
3. Какой цикл называют циклом с постусловием?
4. Что такое итерация?
5. Что называют телом цикла?
6. Перечислите операторы организации цикла в языке программирования C++.
7. Напишите синтаксис и приведите пример использования оператора `for`.
8. Напишите синтаксис и приведите пример использования оператора `while`.
9. Напишите синтаксис и приведите пример использования оператора `do...while`.
10. Каким образом можно организовать досрочный выход из цикла?

5. УКАЗАТЕЛИ И ССЫЛКИ

Указатель – производный тип данных, хранящий адрес какой-либо ячейки памяти. Нельзя объявить просто переменную – указатель. Указатель всегда связан с определённым типом данных, типом указуемого.

Синтаксис объявления указателя:

```
<тип данных> * <имя указателя>;
```

Пример: `int *p;` `p` – указатель на целое число.

Для инициализации указателя в правой части оператора присваивания нужно указать адрес конкретной области данных.

Если указатель настраивают на объявленные ранее переменную или константу, то перед соответствующим идентификатором ставят символ `&` – операция взятия адреса:

```
int q =7;
```

```
int* p;
```

```
p=&q;
```

Объявление указателя может быть совмещено с инициализацией.

Пример:

```
int a =7;
```

```
int* p=&a;
```

Результат работы приведённого примера можно схематично представить, как показано на рис. 1.

Стрелка на рис. 1 означает, что указатель `p` содержит адрес переменной `a`, т. е. указывает на переменную `a`.

Через указатель возможна работа как с самим указателем, так и со значением указуемого.

Для работы с указуемым необходимо совершить переход по адресу. Для этого используют операцию разадресации (разыменовывание). Для разадресации перед уже объявленным и проинициализированным указателем ставят символ `*`.

Пример:

```
int a =7;
```

```
int* p=&a;
```

```
*p=5;
```

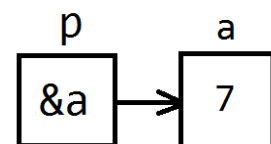


Рис. 1. Схематичное представление связи указателя с указуемым

В приведённом примере в указуемое – переменную `a` – записано значение 5.

Указатель может быть константным. Значение константного указателя нельзя изменить, т. е. его нельзя перенастроить, а изменить значение указуемого (если указуемое не константа) можно. Для объявления константного указателя перед его именем указывают ключевое слово `const`.

Пример:

```
int b=10;
int * const p =&b;
```

При необходимости объявить указатель на константу ключевое слово `const` следует указать до или после типа указуемого, но строго до символа `*`.

Пример:

```
const int b=10;
int const *p =&b;
const int c = 30;
p = &c;
printf("*p=%d", *p); // *p=30
```

В рассмотренном примере `p` – указатель на целочисленную константу, значение `p` можно изменить. В примере указатель `p` перенастроен на другую константу того же типа – `c`.

Так как указатель – это тип данных, значит, он может также быть типом указуемого, т. е. возможно объявление указателя на указатель.

Пример:

```
int a = 7;
int* p = &a;
int** pp = &p;
**pp = 10;
```

На рис. 2 схематично представлены результаты работы написанных выше строк кода.

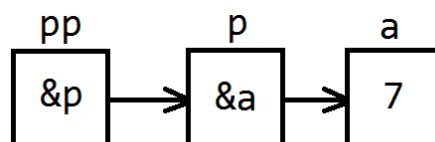


Рис. 2. Схематичное представление инициализации цепочки указателей

В рассмотренном примере переменная `pp` – указатель на указатель на целое – настроена на указатель на целое. После выполнения приведённого выше фрагмента кода значение переменной `a` будет равно 10.

Операции над указателями

На указателях определены следующие операции: сравнения, вычитания, сложения с константой, инкремента и декремента. Для получения наиболее наглядного результата эти операции удобно продемонстрировать, настраивая указатели на элементы массива.

Под массивом понимают поименованную последовательность элементов, расположенных в памяти подряд. Ниже приведен пример объявления и инициализации массива из пяти целых чисел:

```
int mas[5] = {1,2,3,4,5};
```

В языке `C++` нумерация элементов массива начинается с нуля.

Объявим два указателя на целое `p1` и `p2` и настроим их на элементы массива `mas`. Настроим `p1` на нулевой элемент массива, `p2` – на первый элемент массива:

```
int *p1 = &mas[0]; // указатель настроен на нулевой элемент  
int *p2 = &mas[1]; // указатель настроен на первый элемент
```

В языке программирования `C++` имя массива рассматривается как константный виртуальный указатель; с учётом сказанного настройку указателей правильнее выполнять следующим образом:

```
int *p1 = mas;  
int* p2 = mas+1;
```

На рис. 3 схематично представлен результат работы приведённых примеров.

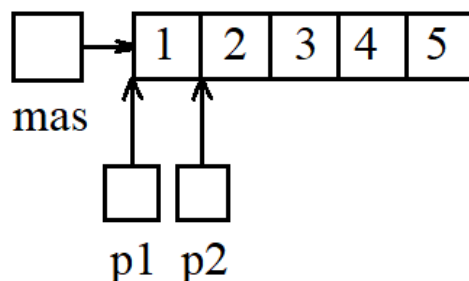


Рис. 3. Схематичное представление настройки указателей на элементы массива

Если, используя операции сравнения, сравнить два указателя, то будут сравниваться значения указателей, т. е. адреса указуемых.

Пример:

```
if (p1 < p2)printf("p1\n");else printf("p2\n");
```

В данном случае ($p1 < p2$) – «истина», так как указатели настроены на последовательно расположенные элементы, адрес указуемого $p2$ старше.

Операция присвоения одного указателя другому. `int* p3 = p2;` // указатель $p3$ будет настроен туда же, куда настроен указатель $p2$.

Унарные инкремент и декремент, сложение указателя с целочисленной константой. Операции постфиксного и префиксного инкремента и декремента изменяют значение указателя на единицу. За единицу принимается размер типа указуемого. Таким образом, указатель будет перенастроен:

```
p3++; // значение p3 увеличится на один размер типа указуемого.
```

```
(*p3)++; // увеличение на единицу значения указуемого.
```

```
p3+=4; // указатель будет перенастроен, его значение увеличится на четыре единицы, под единицей в данном случае понимают размер типа указуемого.
```

Вычитание указателей. Возвращаемое значение операции вычитания указателей – целое число. Это число – количество ячеек указуемого типа между указуемыми:

```
int d = p3-p1;
```

Переменная d получит значение 2, что наглядно видно из рис. 4.

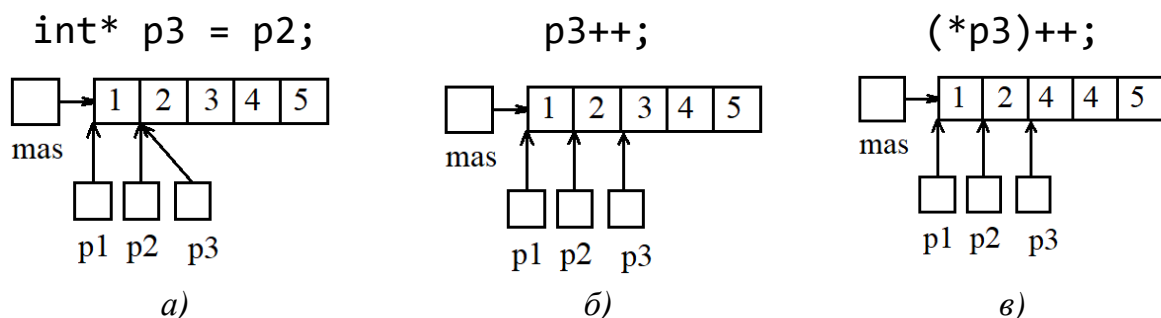


Рис. 4. Схематичное представление операций на указателях: а – настройка указателя $p3$ на ячейку `mas[2]`; б – сдвиг $p3$ на одну ячейку размера указуемого; в – изменение значения указуемого

Ссылки

Ссылка, как и указатель, содержит адрес переменной, с которой связана. Инициализация ссылки обязательна при объявлении:

```
<тип данных> & <имя ссылки> = <переменная>;
```

Наибольшее применение ссылки находят при передаче параметров в функции. При использовании ссылки операции производятся с переменной, с которой она связана. Ссылку нельзя перенастроить.

Пример:

```
float F=9.5;
float H=7.43;
int &ref=F; // объявление ссылки ref и связь её с переменной.
ref = H; /*переменной F присвоено значение переменной H,
ссылка ref остаётся связанной с переменной F*/
ref++; // изменение значения переменной, связанной с ref
```

Задания для практической работы

1. Объявите и проинициализируйте указатели и ссылки на переменные всех базовых типов данных языка программирования C++. Выведите значения переменных (указуемых) на экран, обращаясь к ним через указатели.

2. Объявите и проинициализируйте массив из десяти вещественных чисел. Объявите два указателя на вещественный тип данных. Настройте эти указатели на элементы объявленного выше массива. Продемонстрируйте на объявленных указателях все операции, применимые к указателям. Поясните смысл этих операций.

3. Объявите и проинициализируйте следующие переменные и константы:

- константный указатель на `int`;
- указатель на целочисленную константу;
- указатель на указатель на `int`;
- константный указатель на указатель на `int`;
- указатель на константный указатель на целое.

Поясните схематично организованные цепочки данных.

4. Объявите и проинициализируйте целочисленную переменную. Объявите указатель на целое и настройте его на эту переменную.

Обратившись к указуемому через указатель, увеличьте значение первого втрое и выведите его на экран.

5. Объявите и проинициализируйте целочисленную переменную. Объявите указатель на целое и настройте на эту переменную. Обратившись к указуемому через указатель, выведите на экран увеличенное вдвое значение указуемого, не изменяя его.

Задания для самостоятельной работы

1. Объявите и проинициализируйте вводом с клавиатуры вещественные переменные m и h , объявите и настройте на эти переменные указатели. Объявите вещественную константу, задайте её значение равным приближённому значению ускорения свободного падения. Свяжите с этой константой указатель соответствующего типа. Вычислите значение потенциальной энергии, обратившись к множителям через соответствующие указатели, $E_p = mgh$.

2. Объявите и проинициализируйте вещественную переменную k . Измените и выведите на экран значение этой переменной, обратившись к ней через cp – константный указатель на указатель на вещественную переменную. Выполните все необходимые объявления и инициализацию таким образом, чтобы обращение к k через cp было корректным.

3. Объявите символьную переменную, выведите на экран её значение, предварительно изменив его. Обращение к переменной выполните через указатель на указатель на `char`. Предварительно необходимо объявить и проинициализировать все переменные, входящие в цепочку связанных данных.

4. Выведите на экран значение вещественной константы, обратившись к ней через указатель на указатель. Предварительно выполните все необходимые объявления и инициализацию.

5. Объявите три целочисленные переменные. Проинициализируйте две из них произвольными значениями. Объявите указатель на целое и настройте его на первую переменную. Объявите ссылку и свяжите её со второй переменной. В третью переменную сохраните сумму первых двух, обратившись к ним через указатель и ссылку соответственно.

6. Объявите и проинициализируйте три целочисленные переменные. Объявите и свяжите с этими переменными три ссылки на це-

лое. Найдите наименьшее из трёх значений переменных, обратившись к ним через ссылки.

7. Объявите и проинициализируйте три целочисленные переменные. Объявите указатель на целое и настройте его на переменную с наименьшим значением.

8. Объявите массив из десяти целых чисел. Проинициализируйте массив вводом с клавиатуры. Объявите указатель на целое и настройте его на элемент массива с максимальным значением.

9. Объявите и проинициализируйте две целочисленные переменные. Объявите ссылки на целое и свяжите с этими переменными. Объявите третью целочисленную переменную, проинициализируйте её разностью квадратов первых двух. К переменным следует обращаться по ссылке.

10. Объявите и проинициализируйте две целочисленные переменные. С первой свяжите ссылку, на вторую настройте указатель. Удвойте значения переменных, обратившись к ним через указатель и ссылку соответственно.

Вопросы для закрепления материала

1. Что в программировании называют указателем?
2. Каким образом реализуется связь указателя и указуемого?
3. Возможно ли изменение значения указуемого, если обратиться к нему через указатель?
4. Какой указатель называют константным?
5. В чем отличие константного указателя от указателя на константу?
6. Что значит «перенастроить указатель»?
7. Всегда ли возможно перенастроить указатель?
8. Поясните значение операции разыменовывания.
9. Перечислите операции, применимые к указателям, и поясните их смысл.
10. Что в программировании называют ссылкой?
11. Возможно ли перенастроить ссылку?
12. Возможно ли изменить значение переменной, связанной со ссылкой?

6. МАССИВЫ

Массивом в программировании называют поименованную последовательность однотипных элементов, расположенную в памяти подряд.

Для объявления массива нужно указать тип данных его элементов, имя массива и количество элементов в квадратных скобках:

```
<тип элементов массива> <имя массива>[<количество элементов>];
```

Нумерация ячеек массива в языке программирования C++ начинается с нуля, следовательно, номер последнего элемента массива на единицу меньше количества элементов в массиве. Если в массиве N элементов, то номер последнего элемента $N - 1$.

Объявим массив из пяти целых чисел:

```
int mas [5];
```

Количество элементов массива при его объявлении должно быть обязательно задано целочисленной *константой*. Это связано с тем, что при выделении памяти под массив в момент компиляции необходимо иметь точную информацию о количестве его элементов и это количество не должно изменяться во время работы. При этом константа может быть как именованной, так и не именованной, как в приведённом выше примере.

Предпочтительнее вариант с именованной константой:

```
int const N=5;  
int mas [N];
```

Такой подход ведёт к минимуму изменений в коде программы при необходимости создания массива из другого количества элементов. Для этого будет достаточно изменить значение константы при объявлении.

Задать значение элементов массива можно тремя способами: при объявлении, в цикле и поэлементно. При объявлении это будет выглядеть следующим образом:

```
int const N=5;  
int mas [N]={1,2,3,4,5};
```

Если список значений содержит меньше элементов, чем их количество, указанное в квадратных скобках, то указанными значениями будут проинициализированы первые элементы массива. Оставшиеся элементы получают значение по умолчанию для соответствующего типа данных.

Пример:

```
int const N=5;
int mas [N]={1,2};
```

В случае когда список инициализации содержит больше необходимого числа значений, на шаге компиляции возникает ошибка:

```
int mas1 [3]={1,2,3,4,5,6,7,8,9}; // ошибка
```

Для обращения к элементам массива используют два подхода:

1. Через индексное выражение, например:

```
mas[3]=45;
```

Значение элемента с индексом 3 будет изменено на 45.

2. Использование имени массива как константного указателя (рис. 5).

При инициализации элементов массива в цикле организуют перебор элементов – обычно от наименьшего индекса к наибольшему.

Объявим массив из пяти целых чисел и проинициализируем его в цикле значениями индексов соответствующих элементов:

```
*mas = 9;
```

Таким образом, нулевой ячейке массива будет присвоено значение 9.

Объявим массив из пяти целых чисел и проинициализируем его в цикле каждым из описанных способов обращения к элементам:

```
int mas1[5]; // объявление массива
for (int i = 0; i < 5; i++)
{mas1[i] = i*i; cout << mas1[i]<<" "}; // обращение к
```

элементам через индексное выражение.

```
int const M = 5;
int mas2[M]; // объявление второго массива
for (int i=0; i<M; i++)
{ // обращение к элементу массива через указатель
*(mas+i)=i*i;
printf("%i ",mas[i]);
}
```

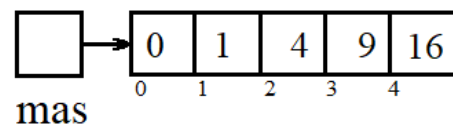


Рис. 5. Схематичное представление одномерного массива с точки зрения языка C++ (после инициализации)

Размерность массива определяется количеством квадратных скобок после его имени:

```
<тип элементов имя массива> [<целочисленная константа1>][<целочисленная константа2>];
```

Пример объявления двумерного массива:

```
int const M = 5;  
int const N = 3;  
int mas2[N][M];
```

Пример объявления трёхмерного массива:

```
int mas3[N][M][N];
```

Элементы многомерных массивов в памяти располагаются подряд построчно (рис. 6). При переходе к следующему элементу первым изменяется последний индекс:

```
mas[0][0]=2; // запись значения в ячейку с индексами 0, 0.
```

Для задания значений многомерного массива при объявлении значения указывают в фигурных скобках списком через запятую.

Пример:

```
int mas2[3][2] = {1,2,3,4,5,6 };
```

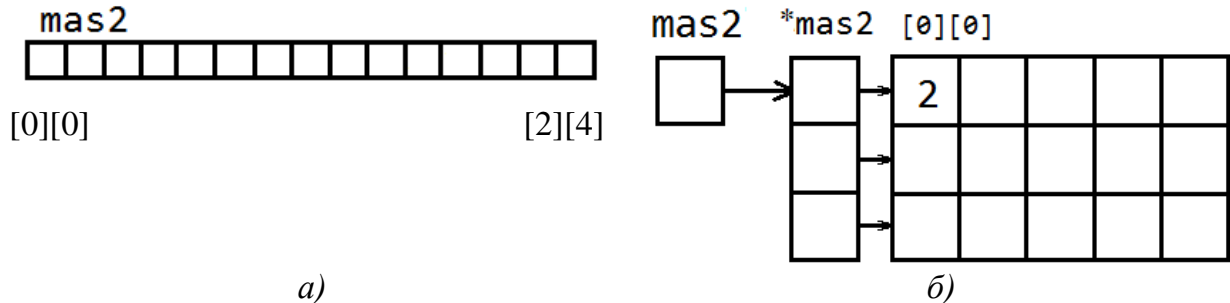


Рис. 6. Схематичное представление двумерного массива: а – расположение двумерного массива в памяти; б – представление массива с точки зрения языка C++

Пример вывода элементов двумерного массива в цикле:

```
for (int i = 0; i < 3; i++)  
{  
  for (int j = 0; j < 2; j++)  
    printf("%i ", mas2[i][j]);  
  printf("\n");  
}
```

Пример обращения к элементам двумерного массива:

```
mas2[1][1] = 2;
*(*(mas2+1)+2) = 3;
```

Динамическое распределение памяти

До настоящего момента речь шла о статических переменных. Так называют переменные, память под которые выделяется на этапе компиляции.

Переменные, память под хранение которых выделяется во время работы программы, называют *динамическими*.

Динамически распределяемую память называют *кучей*. Работа с динамическими переменными происходит через указатели.

Для выделения динамической памяти используют оператор `new`:
`new <тип данных> [<количество элементов>];`

или так:

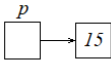
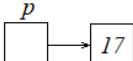

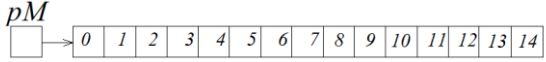
`new <тип данных> (<инициализатор>);`

В точку вызова оператор `new` возвращает указатель на начало области захваченной памяти.

В табл. 1 приведены примеры работы с динамическими данными, память под которые захвачена с помощью оператора `new`, и схематичное пояснение произведённых действий.

Таблица 1

Работа с динамически захваченной памятью

Фрагмент кода	Схематичное представление
<code>int* p = new int(15);</code>	
Объявлен указатель на целочисленную переменную, выделена динамическая память под переменную типа <code>int</code> , в выделенную ячейку записано значение 15. Адрес начала захваченной области памяти записан в указатель <code>p</code> .	
<code>*p += 2;</code>	
Значение указуемого увеличено на 2/	
<code>int* pM = new int[15];</code>	
Объявлен указатель на целочисленную переменную, выделена динамическая память под массив из 15 целых чисел. Адрес его записан в указатель <code>pM</code> .	
<code>for (int i = 0; i < 15; j++) {pM[i] = i;}</code>	
Инициализация в цикле элементов динамического массива.	

Важно помнить, что динамическая память, захваченная при помощи `new`, должна быть освобождена. Для этого используют `delete`:

```
delete[] <указатель>;  
delete p;  
delete []pM;
```

Задания для практической работы

1. Объявите одномерный целочисленный массив из десяти элементов, проинициализируйте массив при объявлении.

2. Объявите одномерный целочисленный массив из десяти элементов, проинициализируйте элементы объявленного массива в цикле значениями их удвоенных индексов.

3. Объявите одномерный целочисленный массив из десяти элементов, проинициализируйте его в цикле вводом с клавиатуры. Объявите указатель на целое. Настройте этот указатель на максимальный элемент массива.

4. Объявите и проинициализируйте вводом с клавиатуры массив из 30 целочисленных элементов. Найдите пять соседних элементов, сумма значений которых максимальна. Выведите на экран сумму и границы последовательности (индекс первого и последнего элементов).

5. Объявите и проинициализируйте вводом с клавиатуры массив из 20 целочисленных элементов. Уменьшите вдвое все элементы, значение которых оканчивается цифрой 4.

6. Объявите и проинициализируйте любым способом массив из 20 целочисленных элементов. Найдите число пар соседних элементов массива, являющихся четными числами.

7. Объявите и проинициализируйте любым способом массив из 20 целочисленных элементов. Определите количество элементов, которые больше суммы всех элементов массива. Выведите на экран найденные элементы и их номера.

8. Объявите и проинициализируйте любым способом массив из 20 целочисленных элементов. Найдите и выведите на экран элемент и его номер, наиболее близкий по значению к среднему значению элементов массива.

9. По заданным вещественным числам a_0, a_1, \dots, a_{10} (a – массив) вычислите значение многочлена $a_{10}x_{10} + a_9x_9 + \dots + a_1x_1 + a_0$ и его производной в точке t .

10. Объявите и проинициализируйте любым способом массив из 20 целочисленных элементов. Определите, имеются ли в массиве одинаковые элементы.

Задания для самостоятельной работы

1. Дан массив, такой, что его значения образуют неубывающую последовательность. Несколько элементов, идущих подряд, могут быть равны между собой. Найдите самую длинную серию повторяющихся элементов.

2. Дан одномерный целочисленный массив. Переменной t присвойте значение «истина» или «ложь» в зависимости от того, есть ли среди элементов массива число, являющееся степенью двойки. Выведите на экран сообщение об этом в зависимости от значения переменной t .

3. Даны два целочисленных массива. Найдите наименьшее значение среди значений элементов первого массива, не входящих во второй массив (по условию задачи хотя бы одно такое число есть).

4. Дан одномерный целочисленный массив. Удалите из него первый отрицательный элемент (если отрицательные элементы есть). Под удалением элемента следует понимать исключение этого элемента из массива путем смещения всех следующих за ним элементов влево на одну позицию и присваивание последнему элементу массива значения 0.

5. Дан одномерный целочисленный массив. Определите и выведите на экран минимальный элемент массива и его номер, а также номер и значение элемента, являющегося минимальным без учета первого найденного элемента.

6. Дан одномерный целочисленный массив. Измените знак максимального по модулю элемента массива.

7. Дан одномерный массив целых положительных чисел. Рассматривая его элементы как последовательность цифр десятичной записи некоторого неотрицательного целого числа, получите это число, сохраните в переменную и выведите на экран. (Например, массив {3, 2, 1, } представляет число 321.)

8. Дан целочисленный массив, элементы которого образуют неубывающую последовательность. Найдите количество различных чисел в массиве.

9. Дан одномерный целочисленный массив. Если в нём есть хотя бы одна пара соседних неотрицательных элементов, выведите на экран все элементы, следующие за первой такой парой.

10. Дан одномерный целочисленный массив. Вычислите и выведите на экран разность сумм его отрицательных и положительных элементов.

11. Дан одномерный целочисленный массив. Вставьте в него заданное число перед последним четным элементом (если четные элементы есть). Под вставкой числа n в массив после k -го элемента понимается смещение всех элементов, начиная с $(k + 1)$ -го вправо на одну позицию, и присваивание $(k + 1)$ -му элементу массива значения n . При этом значение последнего элемента массива теряется.

12. Вычислите $y = x_1 + x_1x_2 + x_1x_2x_3 + \dots + x_1x_2\dots x_m$, где x_i – элементы одномерного целочисленного массива; m – либо номер первого отрицательного элемента в этом массиве, либо номер последнего элемента, если в массиве не нашлось отрицательных элементов.

13. Найдите сумму элементов массива, расположенных между максимальным и минимальным элементами (включительно).

14. Дан одномерный целочисленный массив. Сдвиньте циклически на две позиции влево его элементы.

15. Поясните схематично следующие строки кода:

```
int i = 10;  
int *pi = &i;  
int *pi2 = int(10);  
int *pi3 = new int[10];
```

16. Объявите и проинициализируйте динамическую переменную.

17. Объявите и проинициализируйте динамический одномерный массив целых чисел из десяти элементов.

18. Объявите и проинициализируйте вводом с клавиатуры одномерный массив вещественных чисел. Вычислите и выведите на экран разность квадратов максимального и минимального значения элементов этого массива.

19. Объявите одномерный целочисленный массив. Вычислите среднее арифметическое элементов, обратившись к ним через имя массива, используя его как указатель.

20. Объявите указатель на указатель на целое, настройте его на последовательность из пяти указателей на целое, каждый из которых настройте на последовательность из пяти целых чисел.

21. Объявите массив указателей на вещественный тип данных. Настройте элементы массива указателей на минимальные элементы в каждой строке двумерного массива вещественных чисел. В созданной структуре данных найдите: максимальное значение среди сумм элементов в строках; минимальное значение среди произведений элементов в столбцах.

Во всех задачах ниже необходимо объявить и проинициализировать двумерный целочисленный динамический массив размером n на m , где n – число строк, m – число столбцов.

22. Вычислите и выведите на экран сумму элементов двумерного массива.

23. Найдите и выведите на экран минимальный элемент двумерного целочисленного массива и его индексы.

24. Определите максимальное из чисел, встречающихся в массиве более одного раза.

25. Расположите строки двумерного массива в соответствии с ростом суммы положительных элементов в этих строках.

26. Расположите строки двумерного массива в соответствии с ростом суммы положительных четных элементов в этих строках.

27. Подсчитайте количество столбцов, не содержащих ни одного отрицательного элемента.

28. Подсчитайте и выведите на экран количество строк, не содержащих ни одного нулевого элемента.

29. Замените все отрицательные элементы на значение минимального элемента массива.

30. Выведите на экран минимальные элементы строк массива, обратившись к ним через настроенные на них элементы массива указателей.

31. Найдите и выведите на экран максимальный элемент двумерного массива и его индексы.

32. Расположите строки двумерного массива в соответствии с ростом суммы элементов в этих строках.

33. Объявите два указателя на целое, настройте их на максимальный и минимальный элементы массива соответственно. Выведите на экран разность максимального и минимального элементов массива, обратившись к ним через указатели. Дополнительные переменные не использовать.

34. Найдите и выведите на экран индексы минимального элемента двумерного динамического массива.

35. Вычислите квадрат разности максимального и минимального элементов двумерного динамического массива.

Вопросы для закрепления материала

1. Что называют массивом в программировании?
2. С какого значения начинается индексация элементов массива в языке программирования C++?
3. Чем является имя массива с точки зрения языка C++?
4. Каким образом необходимо задать количество элементов статического массива?
5. Запишите синтаксис и приведите пример создания одномерного статического массива на языке C++.
6. Какие существуют способы инициализации элементов массива в языке C++?
7. Какие существуют способы обращения к элементам массива в языке C++?
8. Если одномерный массив состоит из N элементов, то каким будет номер последнего элемента?
9. Объявите одномерный целочисленный массив из десяти элементов, проинициализировав его при объявлении.
10. Объявите одномерный целочисленный массив из десяти элементов, проинициализировав его элементы в цикле значениями их удвоенных индексов.

7. ПОЛЬЗОВАТЕЛЬСКИЕ ТИПЫ ДАННЫХ. СТРУКТУРЫ

Тип данных, который описывает программист, называют пользовательским. Описание пользовательского типа происходит по определённым правилам исходя из условий конкретной задачи.

Структура – пользовательский тип данных, объединяющий под одним именем разнотипные переменные, называемые полями.

Описание типа «структура» начинается с ключевого слова `struct` и указания имени описываемого типа. Далее в фигурных скобках следует описание полей. Завершается описание типа точкой с запятой.

Пример:

```
struct <имя типа> {<описание полей>};
```

```
struct TypeExample
{
    char pole1;
    float pole2;
    int pole3[10];
};
```

Объявление переменной описанного типа подчиняется тем же правилам, что и объявление переменной любого другого типа:

```
<имя типа> <имя переменной>;
```

Пример:

```
TypeExample Ex1, Ex2;
```

Объявить переменную типа «структура» можно только после описания типа, однако во время описания возможно объявление поля – указатель на описываемый тип.

Объявление переменных может быть совмещено с описанием типа данных. В этом случае, после закрывающей фигурной скобки, до точки с запятой перечисляют объявляемые переменные:

```
struct TypeExample
{
    int pole1;
    float pole2;
    char pole3[10];
} Ex; // Ex – переменная типа TypeExample
```

Поля структуры можно задать списком или обращаясь к каждому полю через уточняющие идентификаторы.

При инициализации списком в фигурных скобках через запятую перечисляют необходимые значения полей в порядке их объявления:

```
struct TypeExample
{
    int pole1;
    float pole2;
    char pole3[10];
} ST{1, 2.3, "name"}; # используя копирование
ST_Example ST1{1, 3.14, "name1"},
ST2={10, 2.4, "name1"};
```

Для обращения к полю структуры необходимо уточнить идентификатор – имя структуры, указав через точку имя поля:

<имя переменной – структуры>.<имя поля>

Пример:

```
ST1.pole1 = 17;
```

Допустимо копирование одной структуры в другую через оператор присваивания:

```
ST1 = ST2;
```

То же самое можно сделать, обратившись к каждому полю:

```
Ex1.pole1 = Ex2.pole1;
```

```
Ex1.pole2 = Ex2.pole2;
```

Так как структура – это тип данных, возможно объявление массива структур и указателя на структуру:

```
ST_Example massiv[5];
```

```
ST_Example* pST = &Ex1;
```

Задания для практической работы

1. Опишите структуру *Point*, хранящую координаты точки. Введите количество точек *n* и последовательность их координат. Выведите на экран координаты точки, наиболее удалённой от начала координат.

2. Есть информация по итогам экзаменов (в группе всего по списку *N* человек). По каждому из студентов имеются следующие сведения: фамилия, оценка по математике, оценка по информатике и оценка по физике. Опишите соответствующую структуру, объявите и

проинициализируйте массив из N элементов. Выведите на экран количество и фамилии круглых двоечников.

3. Есть информация по итогам экзаменов (в группе всего по списку N человек). По каждому из студентов имеются следующие сведения: фамилия, оценка по математике, оценка по информатике и оценка по физике. Опишите соответствующую структуру, объявите и проинициализируйте массив из N элементов. Выведите на экран количество и фамилии отличников.

4. Есть информация по итогам экзаменов в институте (всего в списке N человек). По каждому из студентов имеются следующие сведения: фамилия, оценка по математике, оценка по информатике и оценка по физике. Опишите соответствующую структуру, объявите и проинициализируйте массив из N элементов. Выведите на экран количество и фамилии хорошистов.

5. Есть информация по итогам экзаменов в институте (всего в списке N человек). По каждому из студентов имеются следующие сведения: фамилия, оценка по математике, оценка по информатике и оценка по физике. Опишите соответствующую структуру, объявите и проинициализируйте массив из N элементов. Выведите на экран количество и фамилии обучающихся без троек.

Задания для самостоятельной работы

1. Опишите структуру *Zavod*, содержащую следующую информацию: фамилия, средний возраст, специальность, средний оклад. Объявите и проинициализируйте массив из N таких структур. Выведите на экран информацию о количестве слесарей и токарей.

2. Опишите структуру *Zavod*, содержащую следующую информацию: фамилия, средний возраст, специальность, средний оклад. Объявите и проинициализируйте массив из N таких структур. Выведите на экран номера заводов, где средний возраст выше 35 лет.

3. Опишите структуру *Zavod*, содержащую следующую информацию: фамилия, средний возраст, специальность, средний оклад. Объявите и проинициализируйте массив из N таких структур. Найдите количество заводов, где средний оклад по заводу выше среднего по всем заводам. Выведите на экран это значение и всю информацию по этим заводам.

4. Считая количество вещей и общий вес характеристиками багажа пассажира, опишите соответствующую структуру и создайте массив из N таких структур. Проинициализируйте массив. Определите, есть ли среди пассажиров такой, у которого самый большой багаж по количеству и наименьший – по весу.

5. Считая количество вещей и общий вес характеристиками багажа пассажира, опишите соответствующую структуру и создайте массив из N таких структур. Проинициализируйте массив. Выведите на экран информацию о пассажирах с самым большим багажом по числу вещей и весу.

6. Считая количество вещей и общий вес характеристиками багажа пассажира, опишите соответствующую структуру, создайте массив из N таких структур. Проинициализируйте массив. Определите, есть ли среди пассажиров такой, у которого самый маленький багаж по числу вещей и весу.

7. Опишите структуру с именем *WORKER*, содержащую следующие поля:

- Фамилия и инициалы работника;
- Название занимаемой должности;
- Год поступления на работу.

Объявите и проинициализируйте вводом с клавиатуры массив из десяти элементов типа *WORKER*, упорядочите элементы массива по алфавиту. Выведите на экран фамилии работников, чей стаж работы в организации превышает значение, введенное с клавиатуры. Если таких работников нет, выдайте на экран соответствующее сообщение.

8. Опишите структуру с именем *TRAIN*, содержащую следующие поля:

- Название пункта назначения;
- Номер поезда;
- Время отправления.

Объявите и проинициализируйте вводом с клавиатуры массив из десяти элементов типа *TRAIN*. Упорядочите элементы массива по алфавиту названий пунктов назначения. Выведите на экран информацию о поездах, отправляющихся после введенного с клавиатуры времени. Если таких поездов нет, выдайте на экран соответствующее сообщение.

9. Опишите структуру с именем *TRAIN*, содержащую следующие поля:

- Название пункта назначения;
- Номер поезда;
- Время отправления.

Введите с клавиатуры данные в массив, состоящий из шести элементов типа *TRAIN*. Упорядочите элементы массива по времени отправления поезда. Выведите на экран информацию о поездах, направляющихся в пункт, название которого введено с клавиатуры. Если таких поездов нет, выдайте на экран соответствующее сообщение.

10. Опишите структуру с именем *TRAIN*, содержащую следующие поля:

- Название пункта назначения;
- Номер поезда;
- Время отправления.

Введите с клавиатуры данные в массив, состоящий из восьми элементов типа *TRAIN*. Упорядочите элементы массива по номерам поездов. Выведите на экран информацию о поезде, номер которого введен с клавиатуры. Если таких поездов нет, выдайте на экран соответствующее сообщение.

11. Опишите структуру с именем *MARSH*, содержащую следующие поля:

- Название начального пункта маршрута;
- Название конечного пункта маршрута;
- Номер маршрута.

Введите с клавиатуры данные в массив, состоящий из восьми элементов типа *MARSH*. Упорядочите элементы массива по номерам маршрутов. Выведите на экран информацию о маршруте, номер которого введен с клавиатуры. Если таких маршрутов нет, выдайте на экран соответствующее сообщение.

12. Опишите структуру с именем *MARSH*, содержащую следующие поля:

- Название начального пункта маршрута;
- Название конечного пункта маршрута;
- Номер маршрута.

Введите с клавиатуры данные в массив, состоящий из восьми элементов типа *MARSH*. Упорядочите элементы массива по номерам маршрутов. Выведите на экран информацию о маршрутах, которые начинаются или оканчиваются в пункте, название которого введено с клавиатуры. Если таких маршрутов нет, выдайте на экран соответствующее сообщение.

13. Опишите структуру с именем *NOTE*, содержащую следующие поля:

- Фамилия, имя;
- Номер телефона;
- Дата рождения (массив из трех чисел).

Введите с клавиатуры данные в массив, состоящий из восьми элементов типа *NOTE*. Упорядочите элементы массива по датам рождения. Выведите на экран информацию о человеке, номер телефона которого введен с клавиатуры. Если такого нет, выдайте на экран соответствующее сообщение.

14. Опишите структуру с именем *NOTE*, содержащую следующие поля:

- Фамилия, имя;
- Номер телефона;
- Дата рождения (массив из трех чисел).

Введите с клавиатуры данные в массив, состоящий из восьми элементов типа *NOTE*. Упорядочите элементы массива по алфавиту. Выведите на экран информацию о людях, чьи дни рождения приходятся на месяц, значение которого введено с клавиатуры. Если таких нет, выдайте на экран соответствующее сообщение.

15. Опишите структуру с именем *NOTE*, содержащую следующие поля:

- Фамилия, имя;
- Номер телефона;
- Дата рождения (массив из трех чисел).

Введите с клавиатуры данные в массив, состоящий из восьми элементов типа *NOTE*. Упорядочите элементы массива по трем первым цифрам номера телефона. Выведите на экран информацию о человеке, чья фамилия введена с клавиатуры. Если человека с такой фамилией в массиве нет, выдайте на экран соответствующее сообщение.

Вопросы для закрепления материала

1. Какие типы данных называют пользовательскими?
2. Напишите синтаксис описания типа данных «структура».
3. Приведите пример объявления структуры.
4. Каким образом можно проинициализировать объявленную структуру?
5. Что такое поля структуры?
6. Каким образом можно задать значение полей структуры при объявлении?
7. Запишите синтаксис и приведите пример обращения к полям структуры.
8. Объявите и проинициализируйте массив структур.
9. Чем тип данных «структура» отличается от типа данных «массив»? Что между ними общего?

8. ФУНКЦИИ

Если в рамках решения некоторой задачи необходимо несколько раз выполнить определённую подзадачу, то эту подзадачу рекомендуется оформить в виде функции.

Код, оформленный в виде функции, должен полностью решать одну определённую задачу и удовлетворять определению и свойствам алгоритма.

После написания кода функции к ней можно обратиться в любом месте программы, что позволяет избежать дублирования кода.

Итак, *функция* – это именованная последовательность описаний и операторов, направленных на решение некой задачи.

`main()` – это главная функция. С этой функции начинается выполнение программы на языке C++. Функции, созданные пользователем (программистом), должны быть предварительно объявлены и определены. Таким образом, при работе с функцией выделяют три этапа: объявление, определение и вызов.

Объявление функции – это её заголовок. Определение функции – описание действий, которые выполняет функция. Вызов функции – это выполнение описанных действий с конкретными параметрами.

Объявление функции (или заголовок функции) содержит имя функции, тип возвращаемого значения, список формальных парамет-

ров в круглых скобках. Наличие круглых скобок обязательно, даже если список параметров отсутствует. Объявление функции заканчивается точкой с запятой.

Объявление функции должно быть выполнено до её вызова.

Синтаксис объявления функции:

[<класс памяти>] *тип возвращаемого значения* <имя функции>
([*список формальных параметров*]);

Определение может быть совмещено с объявлением функции или вынесено в отдельный модуль. Определение функции содержит заголовок функции и тело функции. Телом функции называют последовательность действий, заключённых в фигурные скобки.

Класс памяти – необязательная часть заголовка функции, задающая область видимости функции. Если класс памяти не указать при объявлении функции, то будет использован класс памяти по умолчанию – `extern`. Такие функции доступны для вызова во всех модулях программы.

Если функция объявлена с классом памяти `static`, то она доступна для вызова только в том модуле, в котором определена.

Тип возвращаемого значения – это тип значения, которое будет возвращено из функции в точку её вызова. Тип возвращаемого значения в заголовке функции должен присутствовать обязательно. В случае если возвращать значение из функции не нужно, то в качестве типа возвращаемого значения указывают `void`. Имя функции может быть любым идентификатором, отвечающим требованиям именования в C++. Наличие круглых скобок после имени функции обязательно. Скобки могут быть пустыми или содержать список формальных параметров. Если параметров несколько, их перечисляют через запятую. При объявлении функции можно указать только тип параметров; при определении функции нужно указывать и тип, и имя параметров. Параметры, указываемые при объявлении и определении функции, называют формальными. При вызове функции указывают фактические параметры, которые будут подставлены на место формальных.

Пример объявления функции, не принимающей параметры и не возвращающей значение:

```
void f1 ();
```

Пример объявления функции, принимающей два вещественных параметра типа `float` и возвращающей в точку вызова значение типа `float`:

```
float f (float, float);
```

В приведённом примере объявления имена параметров отсутствуют. При определении функции имена должны быть указаны, например, так:

```
float f (float c, float d){ ... };
```

Объявленная функция должна возвращать значение типа `float`. Для возвращения значения из функции используют оператор `return`. После ключевого слова `return` нужно указать возвращаемое значение соответствующего типа.

Пример:

```
float f (float c, float d){ return c+d;};
```

Для вызова функции, исполнения её кода с заданными при необходимости параметрами нужно указать имя функции и в круглых скобках – список фактических параметров. Тип фактических параметров обязательно должен соответствовать типу формальных параметров.

Вызов функций `f1` и `f`:

```
f();
```

```
f1 (1.2, 3.4);
```

```
или float S= f1 (1.2, 3.4);
```

При втором варианте вызова функции `f1` возвращаемое значение будет сохранено в переменную `S`. В первом случае возвращаемое значение будет потеряно.

Если возвращаемое значение не планируется использовать в дальнейших вычислениях и достаточно его вывода на экран, возможен такой подход:

```
cout << f1 (1.2, 3.4);
```

Передача параметров в функцию

Существует три способа передачи параметров в функцию:

- 1) по значению;
- 2) указателю;
- 3) ссылке.

При передаче параметров в функцию по значению в области памяти функции создаётся локальная копия фактических параметров. Изменение этих значений внутри функции не влечёт за собой изменения внешних, переданных значений. Изменения затрагивают только локальную копию, недоступную после выхода из функции.

Пример:

```
int f(int b)
{
    b++;
    return ++b;
}

int main()
{
    int b = 3;
    int c = f1(b);
}
```

В результате выполнения этого участка кода в переменную `c` будет сохранено значение, возвращаемое функцией `f`. Переменная `c` примет значение, равное 5, значение переменной `b`, объявленной в функции `main`, останется неизменным.

При передаче значений по указателю и ссылке в область памяти функции будут переданы адреса соответствующих переменных. Обращение к параметрам в теле функции означает переход по адресу. Соответственно, изменение значений в теле функции приведёт к изменению значений, переданных в функцию. Для работы со значением через указатель его необходимо разыменовать.

Пример:

```
int f2(int *b)
{
    (*b)++;
    return ++(*b);
}

int main()
{
    int b = 3;
    int* pb = &b;
    int b1 = f2(pb);
}
```

В приведённом примере значение переданной в функцию переменной будет увеличено на 2. Таким образом, значения переменных `b` и `b1` будут равны 5.

При передаче параметра по ссылке нет необходимости в разыменовывании, такой подход уменьшает вероятность возникновения ошибок.

Пример:

```
int f3(int& c)
{
    (++c)++;
    return ++c;
}
int main()
{ int c = 3;
  int c1 = f3(c);
}
```

Опишем и вызовем функцию, принимающую параметры, переданные всеми тремя способами:

```
int f4(int a, int* b, int& c)
{a++; (*b)++; c++;
return a + *b + c;
}
```

Здесь `int a` – передан по ссылке, `int* b` – по указателю, `int& c` – по значению.

Объявим соответствующие переменные и вызовем функцию:

```
int main()
{int a = 3;
 int b = 3;
 int* pb = &b;
 int c = 3;
 int S = f4(a,pb,c);
}
```

В теле функции значения всех переданных параметров увеличатся на единицу. Таким образом, возвращаемое значение равно 12. При этом значения внешних переменных `b` и `c` увеличатся на единицу, значение переменной `a` останется неизменным.

Передача массива в функцию реализуется через адрес его первого элемента. Таким образом, для организации передачи массива в функцию необходимо передать два параметра: адрес его начала и количество элементов. Так как массив передают в функцию через указатель, то любые действия, произведённые с элементами массива в теле функции, выполняются с элементами внешнего массива и возвращать массив из функции нет необходимости.

При работе с многомерными массивами необходимо передавать в функцию информацию о всех размерностях. Внутри функции многомерный массив рассматривается как одномерный.

Пример передачи массива в функцию:

```
void Sort(int* mas, int N) // объявление функции
{ . . . }; // тело функции
. . .
int const n = 10;
int mas[n] = {1, -2, 3, -4, 5, -1, 2, -3, 4, -5};
Sort(mas, n, ); // вызов функции
```

Параметры со значениями по умолчанию

Параметры по умолчанию – это параметры, значение которых задано предварительно при объявлении функции. При вызове функции такие параметры можно не указывать. В этом случае будет использовано значение по умолчанию.

Параметры со значением по умолчанию обязательно должны находиться в конце списка формальных параметров. Если таких параметров несколько и один из них при вызове опущен, то и все следующие за ним параметры должны быть опущены. Таким образом, все эти параметры примут значение по умолчанию.

Пример:

```
void f (int a, int b=1, int c=0 ); // объявление ф-ции
. . .
f(1); // параметры b и c принимают значение по умолчанию
f(1,2); /* параметр b принимает значение «2», параметр c
принимает значение по умолчанию.*/
```

Задания для практической работы

Напишите и приведите несколько различных примеров вызова следующих функций:

1. Сложение двух целочисленных переменных.

Функция получает два целочисленных параметра и возвращает их сумму.

2. Поиск минимального значения.

Функция получает три целочисленных параметра и возвращает наименьшее из них.

3. Сложение элементов целочисленного одномерного массива.

Функция получает на вход адрес первого элемента массива и количество элементов в массиве, возвращает целое число – сумму элементов массива.

4. Сортировка одномерного числового массива.

Функция получает на вход данные о массиве. В качестве возвращаемого значения укажите `void`.

5. Вывод элементов одномерного целочисленного массива на экран.

Функция получает на вход данные о массиве. В качестве возвращаемого значения укажите `void`.

Задания для самостоятельной работы

Напишите следующие функции и приведите несколько примеров вызова (с различными значениями входных параметров).

1. Инициализация элементов одномерного целочисленного массива.

2. Вывод на экран значений двумерного массива в табличном виде.

3. Вывод на экран столбца таблицы умножения на число, заданное в качестве параметра.

4. Вычисление суммы минимальных значений в столбцах двумерного массива.

5. Вычисление суммы максимальных значений в строках двумерного массива.

6. Сортировка методом выбора одномерного целочисленного массива.

7. Сортировка методом простой вставки двумерного целочисленного массива.
8. Вычисление факториала заданного числа.
9. Вычисление суммы первых n натуральных чисел, где n – параметр.
10. Вычисление разности квадратов максимального и минимального значений в одномерном целочисленном массиве.

Вопросы для закрепления материала

1. Что в программировании называют функцией?
2. Запишите синтаксис и приведите пример заголовка функции.
3. Что определяет класс памяти?
4. Что называют возвращаемым из функции значением?
5. Что нужно указать в качестве типа возвращаемого значения, если значение из функции возвращать не нужно?
6. Что называют телом функции?
7. В каком случае указывают формальные параметры, в каком – фактические?
8. Возможно ли объявление функции без параметров?
9. Перечислите и поясните способы передачи параметров в функцию.
10. Каким образом осуществляют вызов функции?

9. ЛИНЕЙНЫЕ ДИНАМИЧЕСКИЕ СТРУКТУРЫ ДАННЫХ

Под структурой данных понимают множество данных и множество связей между ними.

Динамические структуры данных – это структуры, в которых количество элементов и связи между этими элементами могут изменяться во время выполнения программы. По типу организации связей между элементами структуры могут быть линейными и нелинейными.

Динамические структуры данных используют в ситуациях, когда заранее неизвестен объём данных, с которыми предстоит работать, и предполагается изменение (увеличение или уменьшение) их количества во время исполнения программы.

Связь элементов динамической структуры реализуют через указатели.

Списки, очереди, стеки, деки, деревья – примеры динамических структур данных.

Динамическую структуру данных, элементы которой связаны между собой определённым образом и не обязательно расположены в памяти подряд, называют *списком*.

Линейные списки могут быть однонаправленными и двунаправленными; и те и другие могут быть кольцевыми (закольцованными, замкнутыми).

Список называют *кольцевым*, если его последний элемент указывает на первый. Если кольцевой (или закольцованный) список – двунаправленный, то каждый элемент указывает на следующий и предыдущий; соответственно, последний элемент настроен на первый и предпоследний, а первый элемент указывает на второй и последний.

Для организации динамической структуры данных каждый элемент списка должен содержать два вида полей: информационные поля и поля-указатели, служащие для связи между элементами. Количество информационных полей и их типы определяют исходя из условий конкретной решаемой задачи. Поля-указатели должны иметь тип данных – указатель на элемент списка. Количество полей-указателей определяется видом списка: один указатель – для однонаправленных и два – для двунаправленных.

Элемент списка объединяет в себе разнотипные поля, поэтому в языке программирования C++ его описывают типом данных «структура», при этом под словом «структура» понимают тип данных, под словосочетанием «структура данных» – подход к организации данных.

Рассмотрим пример описания элемента динамического списка. Опишем структуру с двумя полями: информационным и полем-указателем на следующий элемент. Информационное поле будет целочисленным:

```
struct elem
{int inf;
elem * p;};
```

Поле *p* – это указатель на элемент (где элемент – это описываемая структура). Схематично элемент списка показан на рис. 7.



Рис. 7. Схематичное отображение элемента динамического списка

Создадим и проинициализируем элемент описанного выше типа:
`elem E1 = { 2, NULL };`

На рис. 8 схематично показана структура данных «линейный список», который можно организовать, используя в качестве элементов описанный тип `elem`.

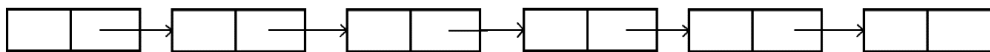


Рис. 8. Схематичное представление линейного списка

Из рис. 8, понятно, что для обращения к элементам такой структуры необходим как минимум один указатель на её начало для обращения к элементам.

Поэтому работа по созданию динамического списка начинается с объявления указателя на элемент списка. Для дальнейшей корректной работы может понадобиться указатель на текущий элемент. Результат объявления настройки таких указателей показан на рис. 9.

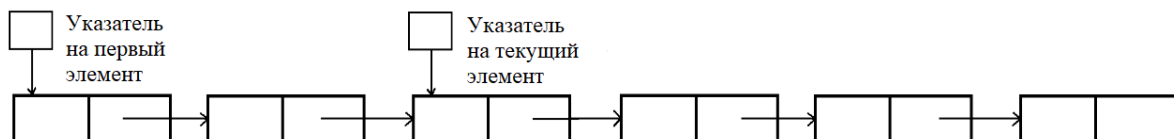


Рис. 9. Схематичное представление линейного списка и указателей, настроенных на первый и текущий элементы

Если список кольцевой, то последний элемент необходимо настроить на первый (рис. 10).

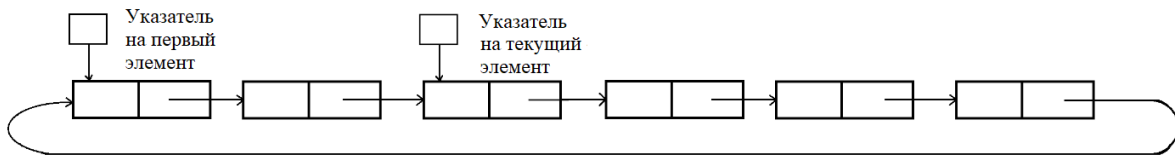


Рис. 10. Однонаправленный кольцевой список

Если список двунаправленный, то каждый его элемент должен содержать два поля-указателя, настроенные на следующий и на предыдущий элементы:

```
struct elem
{
    elem* prev;
    int data;
    elem* next;
};
```

Схематично элемент типа `struct elem` представлен на рис. 11.

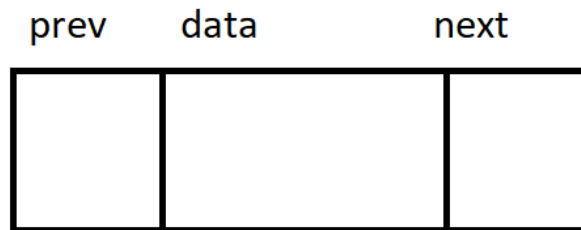


Рис. 11. Элемент двунаправленного списка

На рис. 12 наглядно показан кольцевой двунаправленный список. Для первого элемента такого списка предыдущим элементом является последний элемент, следующим элементом для последнего – первый.

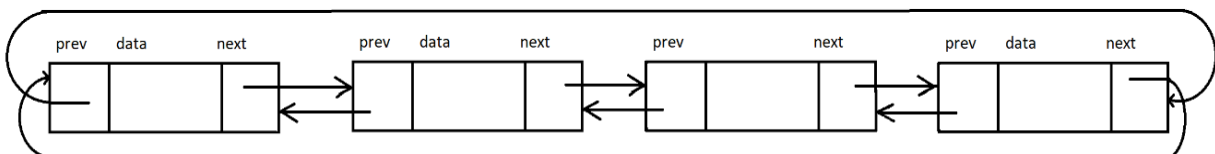


Рис. 12. Двунаправленный кольцевой список

Для работы с динамическими структурами данных необходимо определить следующие операции:

- создание первого элемента;
- добавление элемента в начало или конец списка;
- поиск элемента по ключу;
- вставка элемента по ключу;
- упорядочивание по ключу;
- удаление элемента из начала или конца списка;
- удаление элемента по ключу.

При добавлении и исключении элементов из списка необходимо проверять список на пустоту следующим образом:

```
head != NULL
```

где `head` – указатель на первый элемент списка (иначе называемый головой, или вершиной, списка).

Понятно, что алгоритм добавления элемента в список будет зависеть от того, первый это элемент списка или нет. То же касается и удаления. Предварительно нужно убедиться в том, есть ли элементы в списке.

Добавление элемента в однонаправленный линейный список

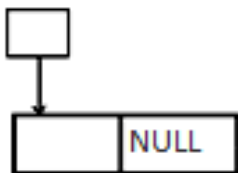


Рис. 13. Созданный первый элемент списка

Добавление первого элемента:

- захватите память под элемент;
- заполните соответствующим образом информационные поля, поле-указатель настройте на NULL (рис. 13).

Добавление последующих элементов:

- захватите память под элемент (рис. 14);

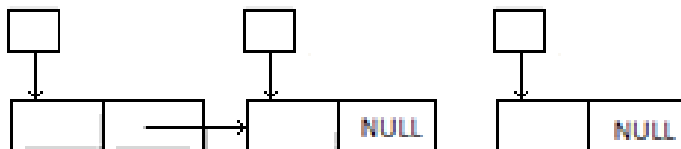


Рис. 14. Захват памяти под новый элемент, который будет добавлен к существующему списку

- настройте соответствующим образом поля-указатели данного и предыдущего элементов списка; настройка полей-указателей будет

зависеть от того, куда будет добавлен новый элемент, в голову или хвост списка (рис. 15).

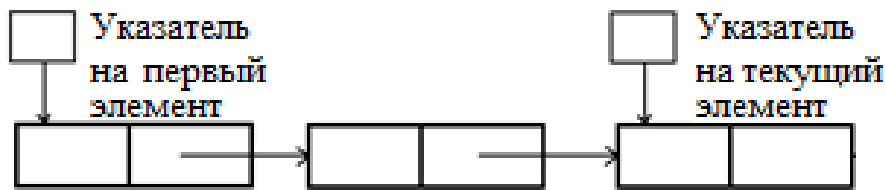


Рис. 15. Добавление элемента в хвост линейного динамического списка

Вставка элемента по ключу

Перед тем как вставить элемент, следует реализовать поиск по ключу нужного элемента.

Например, необходимо вставить новый элемент за элементом с искомым значением. Для этого нужно настроить вспомогательный указатель на голову списка и перемещать его в цикле на следующий элемент до тех пор, пока не будет достигнуто искомое значение либо конец списка.

Для того чтобы найти искомое значение, нужно сравнить ключ с информационным полем элемента списка, на который настроен вспомогательный указатель:

`if (H->inf==k) H=H->p; // фрагмент реализации поиска,`
 где H – указатель на найденный элемент, k – ключ.

Результат поиска элемента схематично представлен на рис. 16.

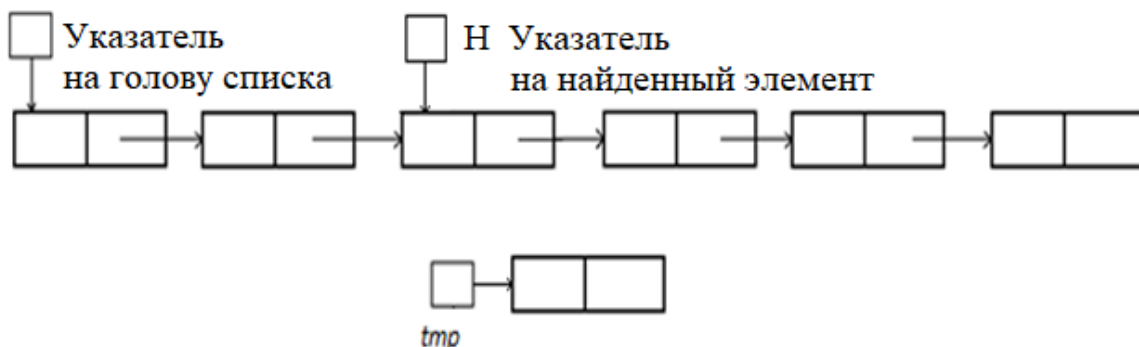


Рис. 16. Поиск элемента по ключу

Далее необходимо настроить поле-указатель нового элемента списка на элемент, на который указывает элемент с совпавшим ключом:

$tmp \rightarrow p = H \rightarrow p;$

Поле-указатель найденного элемента следует настроить на новый элемент:

$H \rightarrow p = tmp;$

Результат такой перенастройки представлен на рис. 17.

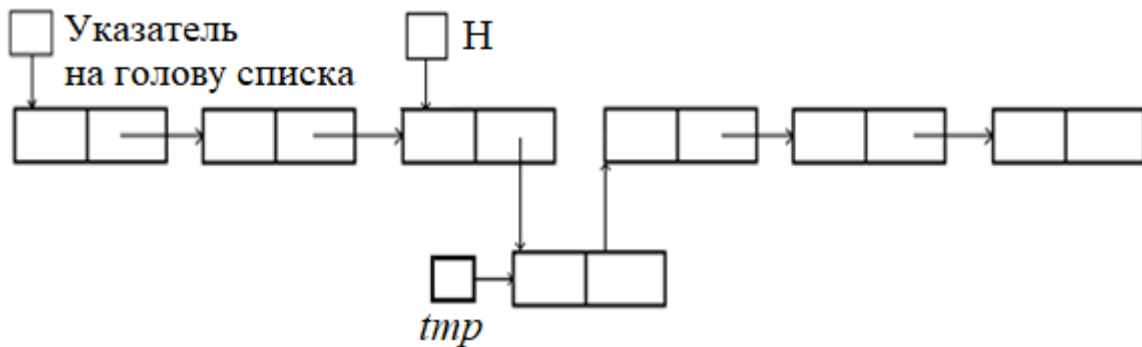


Рис. 17. Элемент вставлен в список

Если необходимо вставить новый элемент перед элементом с заданным ключом, то проверяют на совпадение с ключом элемент, следующий за элементом, на который настроен вспомогательный указатель.

Отдельно рассмотрим ситуацию, когда нужно вставить элемент перед первым. На рис. 18 представлена следующая ситуация: новый элемент уже создан, но ещё не вставлен в список.



Рис. 18. Подготовка к добавлению элемента

В этом случае поле-указатель нового элемента настраивают на первый элемент, после чего перенастраивают указатель на голову списка на вновь добавленный элемент. Либо поле-указатель первого элемента настраивают на новый в зависимости от организации направления списка, таким образом, если на схеме стрелки будут

направлены в другую сторону, это означает, что поля-указатели настроены на предыдущий элемент, а не на следующий. На рис. 19 схематично представлен результат добавления элемента в голову списка.



Рис. 19. Добавленный в голову списка элемент

Исключение элемента из списка

Прежде чем освободить память, занимаемую элементом, необходимо выполнить перенастройку указателей.

Алгоритм удаления первого элемента:

- настройте вспомогательный указатель на голову (рис. 20);

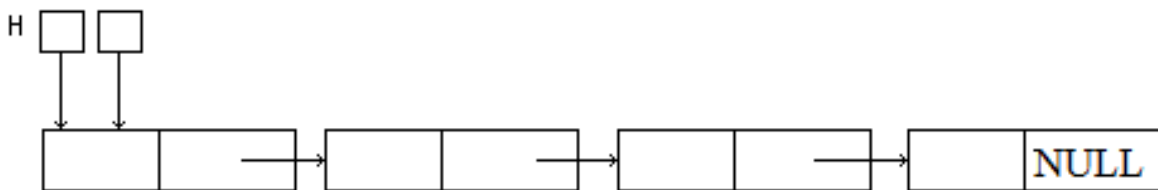


Рис. 20. Настройка дополнительного указателя на первый элемент динамического списка

- перенастройте указатель на голову на следующий элемент (рис. 21);

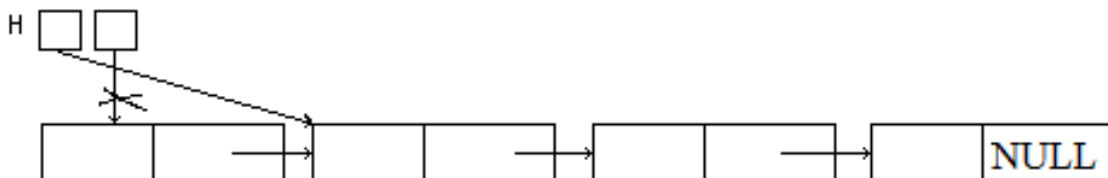


Рис. 21. Удаление элемента из головы динамического списка

- освободите память, занимаемую первым элементом.

Удаление элемента по ключу

Для реализации этой задачи необходимо два вспомогательных указателя – указатель на удаляемый элемент и указатель на предыдущий элемент.

Поле-указатель элемента, предшествующего удаляемому, настраивают на элемент, следующий за удаляемым, после чего освобождают память, занимаемую удаляемым из списка элементом.

Линейный список, организованный по принципу «первый вошёл, первый вышел» (*FIFO – first in, first out*), называют **очередью**.

Для наглядности целесообразно представить очередь в бытовом понимании, например очередь в магазин: кто первый в очередь встал, тот первым товар и получит. В случае динамического списка элемент, который был добавлен в такой список первым, будет извлечён из очереди первым.

Для реализации такого подхода необходимо объявить указатель на голову (вершину) списка и указатель на хвост (последний элемент) очереди для возможности добавления элементов в конец очереди. Эти два указателя можно объединить в одной переменной – структуре. Эта структура будет не того же типа, что и элементы списка; это другой пользовательский тип данных, который должен быть описан отдельно после описания структуры «элемент списка». Например, так:

```
struct Q
{ elem* head;
  elem* tail;
};
```

Если планируется работа не более чем с одним списком, то поля `head` и `tail` можно объявить отдельными переменными вне структуры.

Для добавления элемента в очередь потребуется дополнительный указатель:

```
elem* tmp;
tmp = new elem;
```

Алгоритм добавления элемента в очередь:

– захватите память под новый элемент списка:

```
elem* tmp;
tmp = new elem;
```

– заполните его поля:

```
tmp->inf = a;
```

– если это первый элемент списка (т. е. указатель на голову ни на что не указывает, `head == NULL`), то настройте его на новый элемент:

```
head = tmp;
```

Этот же элемент будет последним в силу единственности, т. е. указатель на хвост нужно также настроить на него;

– если элемент не первый, то указатель на голову не трогают. Перенастройте указатель на хвост на вновь захваченный элемент:

```
tail = tmp;
```

– предварительно настройте на него же поле-указатель предыдущего элемента очереди:

```
tail->p = tmp;
```

Линейный список, организованный по принципу «последний вошёл, первый вышел» (*LIFO – last in, first out*), называют **стеком**.

Для понимания структуры такой организации данных и способов работы с ними стек представляют в виде узкой трубки, закрытой с одного конца и заполненной теннисными мячиками (рис. 22). Из рис. 22 видно, что в противоположность очереди, элемент, добавленный в стек первым, будет извлечён из него последним.



Рис. 22. Наглядное представление стека

Алгоритм добавления элемента в стек:

– захватите память под элемент и заполните его информационное поле:

```
elem* tmp;
```

```
tmp = new elem;
```

```
tmp->inf = a;
```

– дальнейшая настройка зависит от того, первый это элемент или в стеке уже есть значения:

```
(head == NULL);
```

– если элемент первый, указатель на вершину стека настройте на этот элемент:

```
head = tmp;
```

```
tmp->p = NULL;
```

– если в стеке уже есть значения, перенастройте указатели таким образом, чтобы не потерять связь между элементами и при этом новый элемент стал вершиной стека:

```
tmp->p = head;
```

```
head = tmp;
```

При написании функций работы с динамическими списками параметры-указатели передают через ссылки; если указатель передать по значению, то изменённое значение не сохранится после выхода из функции.

Задания для практической работы

1. Элементы целочисленного массива запишите в очередь. Напишите функцию извлечения элементов из очереди до тех пор, пока первый элемент очереди не станет чётным.

2. Даны две очереди, содержащие целые числа. Объедините очереди в одну таким образом, чтобы первой в результирующей очереди оказалась та, сумма элементов которой больше.

3. Даны две непустые очереди одинаковой длины. Объедините очереди в одну такую, в которой элементы исходных очередей чередуются.

4. Даны две непустые очереди. Элементы каждой из очередей упорядочены по возрастанию. Объедините очереди в одну с сохранением упорядоченности элементов.

5. Пусть имеется файл действительных чисел и некоторое число A . Используя очередь, выведите на экран сначала все элементы, меньшие числа A , а затем – все остальные элементы.

Задания для самостоятельной работы

1. Напишите функцию добавления элемента в динамический список, организованный по принципу очереди.

2. Напишите функцию добавления элемента в динамический список, организованный по принципу стека.

3. Напишите функцию проверки линейного динамического списка на пустоту.

4. Напишите функцию извлечения элементов из линейного динамического списка.
5. Напишите функцию поиска элемента в списке по ключу.
6. Напишите функцию удаления по ключу элемента из линейного динамического списка.
7. Напишите функцию вставки по ключу элемента в линейный динамический список.
8. Напишите функцию создания/добавления элемента в однонаправленный кольцевой список.
9. Напишите функцию создания/добавления элемента в двунаправленный кольцевой список.
10. Напишите функцию удаления элемента по ключу из двунаправленного кольцевого динамического списка.

Вопросы для закрепления материала

1. Какие структуры данных называют динамическими?
2. Какие динамические структуры называют линейными?
3. Приведите примеры линейных динамических структур.
4. Каким образом должен быть описан элемент динамической структуры данных на языке программирования C++?
5. Назовите правило для организации линейной динамической структуры данных «очередь».
6. Назовите правило для организации линейной динамической структуры данных «стек».
7. Перечислите этапы добавления элемента в линейную динамическую структуру.
8. Перечислите этапы удаления элемента из линейной динамической структуры.
9. Каким образом нужно передать указатель на первый и указатель на последний элемент списка в функцию добавления элемента?
10. Сколько указателей необходимо использовать для создания и работы со стеком?

10. ВВЕДЕНИЕ В ЯЗЫК ПРОГРАММИРОВАНИЯ *PYTHON*

Как известно, переменная – это ячейка в памяти компьютера, имеющая имя и хранящая некоторое значение определенного типа (например, строка, целое число, вещественное число), тип значения переменной в языке программирования *Python* определяется автоматически. Имя переменной может состоять из заглавных и строчных латинских букв, цифр и знака подчеркивания, других символов в имени переменной быть не может. Имя переменной не может начинаться с цифры и не может содержать пробелы, знаки препинаний и символы арифметических операций. Рекомендуется давать переменным имена «со смыслом».

Для того чтобы переменная хранила значение, это значение нужно в нее «положить», т. е. сохранить. Для этого служит оператор присваивания =

Примеры: `a = 5; b = 3.14; name = 'N. A.'`

Для обозначения комментариев в языке программирования *Python* используют символ #.

Ввод значения с клавиатуры выполняется при помощи команды `input()`. При ее выполнении программа будет ожидать, пока пользователь введет значение и нажмет на клавиатуре кнопку *Enter*, введенное значение будет сохранено в переменную. Например, так:

```
name = input() # ввод строки
```

Для того чтобы ввести число, необходимо преобразовать строку, введенную командой `input()`, в нужный формат при помощи функций `int()` или `float()`

Например, строка `c1 = int(input())` реализует ввод целого числа и его запись в переменную в `c1`. Строка кода `c2 = float(input())` реализует ввод вещественного числа и его запись в переменную `c2`.

Внутри круглых скобок функции ввода `input()` можно написать параметр-подсказку для пользователя. Например, так:

```
a = int(input('Введите целое число: '))
```

Эта подсказка будет отображена на экране перед вводом значения переменной.

Для вывода на экран какой-либо информации используется команда (функция) `print()`.

После ключевого слова `print` в круглых скобках указывают то, что необходимо вывести: это может быть число, строка (в кавычках), переменная, арифметическое выражение и т. д. Функция `print` может содержать несколько аргументов, перечисляемых через запятую. Например: `print(5, 5+8), print('Answer:', (5+8)*14)`

Для того чтобы вывести на экран одинарные кавычки, их необходимо заключить в двойные кавычки, и наоборот, чтобы вывести двойные кавычки, их необходимо заключить в одинарные.

Например, строка кода `print('"Hello! "')` выведет на экран строку "Hello!"

Если внутри круглых скобок указать несколько аргументов через запятую, то они будут выведены через пробел. Если этот пробел не нужен, его можно убрать, указав в качестве последнего аргумента `sep=""`. Например, так:

```
print('My', 'Little', 'Friend', sep="")
```

В результате работы этой строки кода на экране появится строка: *MyLittleFriend*

Каждая новая команда `print()` выводит на экран текст с новой строки. Отменить переход на новую строку можно, указав в качестве последнего параметра `end=""`.

Таким образом, две строки кода:

```
print('My ', end="")
print('friend')
```

выведут на экран одну строку: *My friend*

Отступы (сдвиги относительно левой границы) используются в языке программирования *Python* для разделения блоков программы между собой. Для того чтобы операторы были отнесены к одному блоку, необходимо разместить их на одном уровне.

Арифметические операции

В табл. 2 приведены основные арифметические операции языка программирования *Python*.

Знаки операций могут быть использованы в простых или составных выражениях. Для повышения приоритета операции используют круглые скобки.

Таблица 2

Основные арифметические операции языка Python

Знак операции	Операция	Пример использования
+	Сложение	$a + b$
-	Вычитание	$a - b$
*	Умножение	$a * b$
/	Деление	a / b
**	Возведение в степень	$a ** b$
//	Деление нацело	$a // b$
%	Остаток от деления	$a \% b$

В табл. 3 приведены составные операторы присваивания. Использование таких операторов позволяет сократить запись.

Таблица 3

Составные операторы присваивания

Знак операции	Пример использования	Альтернатива
/=	$x /= 3$	$x = x / 3$
*=	$x *= 3$	$x = x * 3$
+=	$x += 3$	$x = x + 3$
-=	$x -= 3$	$x = x - 3$
%=	$x \% = 3$	$x = x \% 3$
//=	$x //= 3$	$x = x // 3$
**=	$x ** = 3$	$x = x ** 3$

Задания для практической работы

1. Выведите на экран через пробел первые десять заглавных латинских букв по три буквы в строке (последняя буква на отдельной строке).

2. С клавиатуры введите два целых числа a и b . Вычислите и выведите на экран сумму квадратов этих чисел. Напишите подсказки для пользователя.

3. Выведите на экран свои имя и фамилию. Имя и фамилия должны находиться на одной строке, между ними – пробел, в конце – точка.

4. Выведите на экран свою дату рождения (число, название месяца, год) в одной строке через пробел. Число, месяц и год должны быть выведены тремя различными операторами `print`.

5. Выведите на экран три числа: число пи (его приближительное значение), число дней в високосном году, число – телефон вызова экстренных служб. Числа должны быть выведены на разных строках. Для справки: числа в *Python* выводятся без использования кавычек и апострофов.

6. Дано трёхзначное число. Найдите сумму его цифр.

Задания для самостоятельной работы

1. С клавиатуры введите два числа. Программа должна вычислить их сумму, разность, произведение и частное.

2. На вход программы подаются два целых числа: l – длина прямоугольника, d – ширина. Вычислите и выведите на экран площадь и периметр данного прямоугольника (с подсказками).

3. На вход программы подается целое число x . Выведите на экран остаток от деления этого числа на 3.

4. Пользователь вводит с клавиатуры целое двузначное число N и целое трёхзначное число M . Выведите на экран произведения цифр каждого из этих чисел.

5. На вход программы подаётся три целых числа: l – длина параллелепипеда, d – ширина и h – высота параллелепипеда (по одному в строке). Вычислите и выведите на экран объем и площадь полной поверхности параллелепипеда (в одну строку через пробел).

6. На вход программы подаётся целое число x . Выведите на экран последнюю цифру этого числа в 9СС.

7. Пользователь вводит с клавиатуры целое пятизначное число N . Выведите на экран произведения цифр этого числа.

8. Ребята участвуют в командной олимпиаде по информатике, где им предстоит решить k задач. Было принято решение разделить задачи поровну, а оставшиеся задачи решить вместе. Сколько задач предстоит решить каждому ученику самостоятельно? Сколько задач ребята решат вместе? Программа получает на вход числа n – количество учеников и k – количество задач (по одному в строке) и должна вывести искомые числа (через пробел).

9. Выведите на экран единичную матрицу 5×5 (между числами должны стоять пробелы).

10. Выведите на экран первые десять натуральных чисел (> 0) по два в строке, где разделителем является " _".

11. С клавиатуры введите три натуральных числа a, b, c (по одному в строке). Выведите на экран результат выражения $5a^2 + 3b^2 - c$.

12. С клавиатуры введите три натуральных числа x, y, z (по одному в строке). Выведите на экран остатки от деления чисел x, y на z . Результаты деления запишите через пробел в одной строке.

13. С клавиатуры введите площадь основания пирамиды и её высоту (по одному значению в строке). Выведите на экран объем пирамиды.

14. В школе решили набрать три новых класса. Так как занятия по математике у них проходят в одно и то же время, было решено выделить кабинет для каждого класса и купить для них новые парты. За каждой партой может сидеть не больше двух учеников. Известно количество учащихся в каждом из трёх классов. Сколько всего нужно закупить парт, чтобы их хватило на всех учеников? Программа получает на вход три натуральных числа (по одному в строке): количество учащихся в каждом из трех классов.

15. Студентка зашифровала свой пароль, состоящий из шести цифр, следующим образом:

1) умножила его на 2;

2) прибавила 7;

3) перевернула запись числа (например, число 123 превратилось бы при таком преобразовании в 321), гарантируется, что последняя цифра не равна 0;

4) отняла 8.

После этого она стала считать, что выкладывать пароль в общий доступ стало безопасно. Докажите, что это не так, и расшифруйте пароль. Гарантируется, что пароль – число, меньшее 450 000 450 000. На вход подается зашифрованный пароль, а вывести необходимо исходный пароль.

16. В некоторой школе расписание составлено таким образом, что на неделе пять учебных дней, в которые суммарно проходит n уроков. При этом количество уроков в любые два дня отличается не более чем

на один. Определите минимальное число уроков в день и количество дней, в которых уроков больше всего.

17. В магазине батарейки продаются поштучно или упаковками по четыре штуки. Выгоднее покупать батарейки упаковками, чем поштучно. Вам необходимо купить ровно n батареек. Определите, сколько упаковок батареек и батареек поштучно нужно купить, чтобы покупка была максимально выгодной.

18. Скорость первого автомобиля V_1 (км/ч), второго – V_2 (км/ч), расстояние между ними S (км). Определите расстояние между ними через N часов, если автомобили удаляются друг от друга, двигаясь в противоположных направлениях. *Для справки:* данное расстояние равно сумме начального расстояния и общего пути, проделанного автомобилями; общий путь равен времени, умноженному на суммарную скорость.

Программа получает на вход четыре числа в разных строках: скорости двух автомобилей, начальное расстояние и время пути и должна вывести одно число – ответ на задачу.

19. Известно, что один пирожок стоит a руб. b коп. Напишите программу, которая выводит стоимость покупки (в рублях и копейках) n пирожков. Программа получает на вход три числа: цену пирожка в рублях и копейках и количество пирожков. Программа должна вывести два числа: стоимость покупки в рублях и копейках.

20. Напишите программу для решения задачи. Известно, что X кг конфет стоит A руб. Определите, сколько стоит 1 кг и Y кг этих же конфет. Программа получает на вход три целых числа: первое количество килограммов конфет, их цену и второе количество килограммов конфет. Программа должна вывести два числа: стоимость одного килограмма конфет и стоимость Y кг конфет.

Вопросы для закрепления материала

1. Каким образом можно ввести с клавиатуры строку и сохранить ее значение в переменную в языке программирования *Python*?

2. Какие действия необходимо выполнить для того, чтобы преобразовать введенную строку в целое число?

3. С помощью какой функции в языке программирования *Python* осуществляется вывод значения на экран?

4. Что необходимо сделать для вывода на экран строки, заключённой в кавычки?

5. Каким образом можно убрать пробел между выводимыми на экран значениями?

6. Каким образом можно отменить перевод на новую строку после вывода значений на экран?

7. Какой оператор нужно использовать для получения остатка от деления?

8. Какой оператор нужно использовать для возведения числа в степень?

9. Какой оператор нужно использовать для получения целой части от деления?

11. ОПЕРАТОР ВЕТВЛЕНИЯ В ЯЗЫКЕ PYTHON

Условный оператор (оператор ветвления) используется при необходимости выполнить определённые команды в зависимости от значения некоторого условия. Оператор ветвления имеет две формы – полную и сокращённую. Структурные схемы конструкций представлены на рис. 23 и в прил. 4.

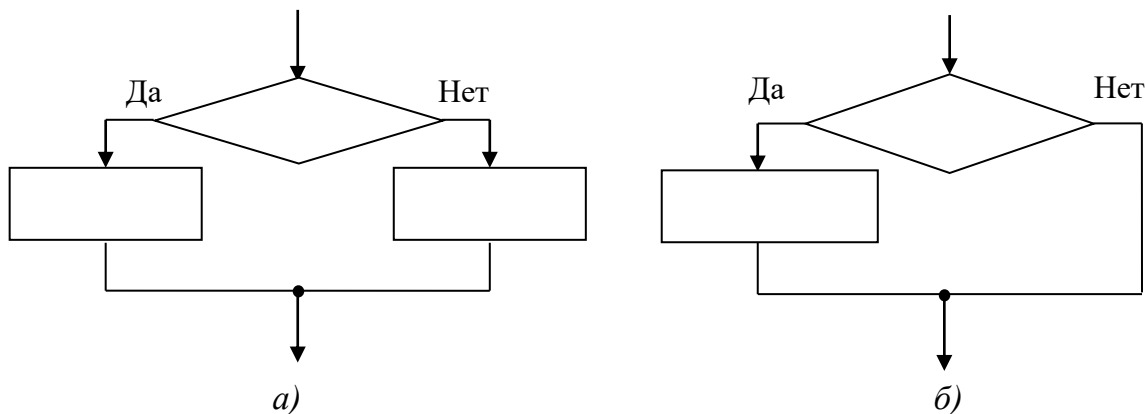


Рис. 23. Алгоритмическая структура «ветвление»:
а – полное ветвление; б – неполное ветвление

В любом случае для реализации ветвления необходимо указать ключевое слово `if`, после которого следует логическое выражение, завершающееся двоеточием. Например:

```
if cash >= 150:
```

Далее следует блок инструкций, который будет выполнен, если логическое выражение истинно. Сокращённая конструкция на этом заканчивается. В случае полной конструкции далее следует указать ключевое слово `else`, после которого нужно поставить двоеточие, и блок инструкций, который будет выполнен, если условие не выполняется.

Ключевое слово `else` должно быть расположено строго под ключевым словом `if`, команды, входящие в блок инструкций, – с отступом относительно расположения ключевых слов `if` и `else`. Рекомендуется отступ в четыре пробела, но не менее одного.

Синтаксис и пример использования оператора «условие» в языке программирования *Python* представлены ниже.

Синтаксис:

```
if <условие>:  
    <блок инструкций 1>  
else:  
    <блок инструкций 2>
```

Пример:

```
if d==1:  
    print("true")  
else:  
    print ("false")
```

В примере, приведённом выше, на экран будет выведено слово `true` при равенстве значения переменной `d` единице, в противном случае, если значение переменной не равно единице, на экран будет выведено слово `false`.

Логическое выражение, используемое в условной конструкции, может быть простым или составным. В простых условиях используются следующие знаки операций: `==`, `!=`, `>`, `>=`, `<=`.

Например: `a > 0` или `x != y*z`.

Составным называют условие, состоящее из двух или более простых условий, соединённых логическими операциями: `and`, `or`, `not`. Например, так: `a < b and b < 10` или так: `x >= 0 or x < 10`, `not (a == 0)`.

Операция `or` (логическое ИЛИ) требует выполнения хотя бы одного из условий. Конструкция `<условие 1> or <условие 2> or <условие 3>` будет принимать значение «ложь», только если все про-

стые условия ложны. Операция `and` (логическое И) возвращает значение «истина» при выполнении всех условий: `<условие 1> and <условие 2> and <условие 3>` будет принимать истинное значение, только если все простые условия истинны. Причем если условие 1 ложно, то условие 2 проверяться уже не будет.

Операция `not <условие 1>` (логическое НЕ) будет принимать ложное значение («ложь»), если условие 1 истинно, и наоборот. Например, следующие два условия равносильны: `A>B` и `not (A<=B)`.

Логические операции можно комбинировать между собой, но важно помнить о приоритете: первой выполняется операция `not`, затем `and`, последней выполняется операция `or`.

При необходимости изменить порядок выполнения операций используют круглые скобки.

Пример: `a > 0 or a < 100 and a % 2 == 0`

Равносильно: `(a > 0 or a < 100) and a % 2 == 0`

Вложенные условные конструкции

Внутри условных конструкций можно использовать любые инструкции языка *Python*, в том числе и условную конструкцию. В этом случае говорят о вложенном ветвлении. При таком подходе вложенные блоки имеют больший размер отступа.

На рис. 24 представлен пример структурной схемы вложенного ветвления.

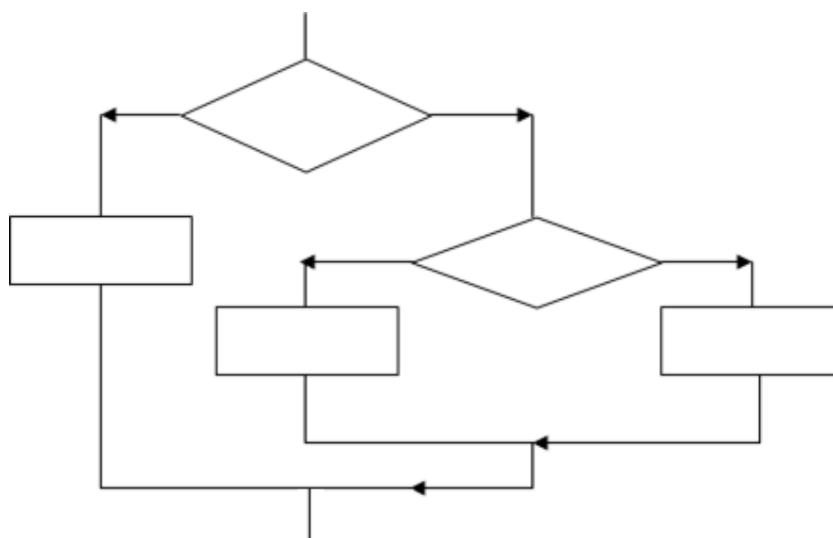


Рис. 24. Структурная схема вложенного ветвления

Множественные вложенные условные операторы можно сократить, используя каскадную конструкцию `if-elif-else`. Ключевое слово `elif` заменяет конструкцию `else: if`. При этом ключевые слова располагаются строго друг под другом, а блоки инструкций расположены с одинаковым отступом. Пример использования такой конструкции:

```
d=int(input())
if d==0:
    print("zero")
elif d>0:
    print ('plus')
else:
    print('minus')
```

В языке *Python* существуют встроенные функции `max()` и `min()`. Функция `max(<последовательность>)` возвращает максимальное значение из последовательности, функция `min(<последовательность>)` – минимальное значение из последовательности. Например:

```
m = max(a, b, c)
n = min(a+b, c // 3)
```

Задания для практической работы

1. Даны три различных натуральных числа. Найдите и выведите на экран среднее число. Если среди чисел есть равные, выведите сообщение "Ошибка".

2. Введите с клавиатуры два различных целых числа. Определите, кратно ли первое число второму. Выведите на экран сообщение об этом, а также остаток от деления, если первое число не кратно второму.

3. Напишите программу, определяющую по координатам точки, в какой четверти она находится. Координаты точки вводятся с клавиатуры.

4. Введите с клавиатуры три вещественных числа. Определите, может ли существовать треугольник с такими сторонами. Треугольник существует только тогда, когда длина ни одной из его сторон не превышает сумму длин двух других.

5. Введите с клавиатуры координаты некоторой точки и радиус круга. Определите, принадлежит ли точка кругу с центром в начале координат.

Задания для самостоятельной работы

1. Автомат продаёт кофе по цене 150 руб. за чашку. Принцип работы: пользователь вводит денежную сумму, а автомат выдаёт напиток и сдачу либо надпись с суммой, которой не хватает. Напишите программу, запрашивающую на вход сумму и выдающую информацию о сумме сдачи или нехватке денежных средств.

2. Модифицируем задачу про автомат с кофе. Теперь в автомате есть три вида кофе: эспрессо (100 руб.), капучино (150 руб.) и латте (200 руб.). Пользователь выбирает напиток, вводит денежную сумму. Напишите программу, выдающую сообщение о сдаче или о том, что введенной суммы недостаточно.

3. Пользователь вводит с клавиатуры число. Напишите программу, выдающую сообщение о том, является ли введённое число пятизначным.

4. Пользователь вводит с клавиатуры два числа. Напишите программу, выдающую меньшее из них на экран.

5. Пользователь вводит с клавиатуры два числа. Напишите программу, выдающую сообщение о том, делится первое число на второе или нет. Также выводится частное и остаток, если он есть.

6. Пользователь вводит с клавиатуры два числа x и y . Напишите программу, которая должна определить, в какой из четвертей координатной плоскости находится точка (x, y) .

7. Перепишите задачу 6, используя каскадную последовательность операций.

8. Даны числа x и y . Вычислите число z , равное $x + y$, если $x \leq y$, и $1 - x + y$ в противном случае. Выведите его на экран.

9. Пользователь вводит натуральное число, меньшее 10 000. Необходимо вывести все его цифры через пробел в обратном порядке.

10. Решите квадратное уравнение $ax^2 + bx + c = 0$. На вход даются коэффициенты уравнения a , b , c . Необходимо вывести корни уравнения в порядке возрастания через пробел, если уравнение имеет два корня; один корень, если уравнение имеет лишь один корень; не выводить ничего, если у уравнения нет корней.

11. Напишите программу, которая определяет, существует ли треугольник с заданными длинами сторон. Если такой треугольник существует, то программа должна определить его тип (равносторонний, равнобедренный, разносторонний).

На вход даются три числа – длины сторон треугольника. Программа должна вывести:

- *NO*, если треугольника с такими сторонами не существует;
- *equilateral*, если треугольник равносторонний;
- *isosceles*, если треугольник равнобедренный;
- *versatile*, если треугольник разносторонний.

12. Пользователь вводит с клавиатуры два числа (по одному в строке). Программа выводит сообщение о том, является ли первое число квадратом второго: *YES*, если является, иначе – *NO*.

13. Пользователь вводит число – год. Необходимо определить, является ли год високосным, вывести на экран соответствующую надпись, а также количество дней в году. Високосным год является, если его номер кратен 4, но не кратен 100, а также если он кратен 400.

14. Вводится четырехзначное число. Напишите программу, считающую сумму первых двух и последних двух цифр в четырехзначном числе. Если их суммы равны, выведите *true*, в противном случае выведите *false*.

15. С клавиатуры вводятся два целых числа. Программа должна вывести:

- *first*, если первое число больше второго;
- *second*, если второе больше первого;
- *equal*, если они равны.

16. Поле шахматной доски определяется парой чисел (a, b), каждое от 1 до 8, первое число задает номер столбца, второе – номер строки. Требуется определить, бьет ли ферзь, стоящий на клетке с указанными координатами (номер строки и номер столбца), фигуру, стоящую на другой указанной клетке.

Входные данные: четыре числа – координаты ферзя и координаты другой фигуры. Координаты – целые числа в интервале от 1 до 8.

Требуется вывести слово *YES*, если ферзь может побить фигуру за один ход, в противном случае вывести слово *NO*.

17. Пользователь вводит сумму (в рублях) от 1 до 9999. Необходимо записать введенную сумму словами. Например, если на вход подано 1900, на выходе должна быть строка:

«одна тысяча девятьсот рублей».

18. Пользователь вводит номер месяца. Выведите название месяца и времени года.

19. Пользователь вводит номер дня недели. Выведите соответствующее название дня недели.

20. Пользователь вводит сумму. Определите и выведите соответствующее сообщение, сможет ли ученик купить на эту сумму n мороженоых по 120 руб.

Вопросы для закрепления материала

1. В каких случаях используют оператор ветвления?
2. Какие формы оператора ветвления существуют?
3. Запишите синтаксис и приведите пример полной формы оператора ветвления.
4. Запишите синтаксис и приведите пример сокращённой формы оператора ветвления.
5. Представьте структурную схему полной и сокращенной конструкции ветвления.
6. В каком случае говорят о вложенном ветвлении?
7. Приведите пример использования вложенного ветвления.
8. В каком случае говорят о каскадном ветвлении? Приведите пример его использования.
9. Каким образом в языке программирования *Python* последовательность команд объединяют в один блок?
10. Какие функции языка *Python* позволяют определить максимальное/минимальное значение в определённой последовательности?

12. ОПЕРАТОРЫ ОРГАНИЗАЦИИ ЦИКЛОВ ЯЗЫКА PYTHON

Цикл – конструкция языка программирования, использующаяся при необходимости выполнить определённую последовательность действий несколько раз. В языке *Python* существуют два оператора, позволяющих реализовать цикл.

Оператор *while*

При применении цикла *while* последовательность команд – тело цикла – выполняется до тех пор, пока условие истинно. Принцип работы оператора следующий: проверяется условие, если оно истинно, то выполняются команды из тела цикла, затем условие проверяется снова. Действия повторяются до тех пор, пока условие не станет ложным.

Синтаксис оператора *while*:

```
while <логическое выражение>:  
  <операторы тела цикла>
```

Команды тела цикла записываются с одинаковым отступом относительно ключевого слова *while*. Ниже приведён пример организации цикла при помощи оператора *while*:

```
n=int(input())  
i=0  
while i<n:  
    print(i)  
    i+=1
```

Способы прерывания циклов

Оператор *break* используют при необходимости досрочно завершить цикл. Когда *break* будет выполнен в теле цикла, последний будет досрочно прерван, управление будет передано оператору, следующему за оператором цикла.

Оператор *continue* используют при необходимости в одном из случаев пропустить часть тела цикла и перейти к следующей итерации. Когда оператор *continue* выполнен, текущая итерация будет прервана и снова проверено условие.

Ниже приведены примеры использования операторов прерывания цикла:

```

x=0
while x<10:
    x+=1
    if x==5:
        break
    x+=1
    print('x=',x)
print('Out of loop')

```

В результате работы этого фрагмента кода на экран будут выведены следующие строки:

```

x= 2
x= 4
Out of loop
x=0
while x<10:
    x+=1
    if x==5:
        continue
    x+=1
    print('x=',x)
print('Out of loop')

```

Результат на экране будет таким:

```

x=0
while x<10:
    x+=1
    if x==5:
        continue
    x+=1
    print('x=',x)
print('Out of loop')

```

В языке *Python* в циклах так же, как и в условном операторе, можно использовать ключевое слово **else**. В этом случае блок внутри **else** исполняется один раз, как только условие цикла станет ложным. Обычно такой подход используют для проверки способа выхода из цикла. Блок внутри **else** выполнится только в том случае, если выход из цикла произошел без помощи **break**:

```
x=0
while x<5:
    print(x)
    x+=1
else:
    print('The End')
```

Результат работы:

```
0
1
2
3
4
The End
```

Модернизируем тело цикла, используя условие и оператор прерывания:

```
x=0
while x<5:
    if x==3:
        break
    print(x)
    x+=1
else:
    print('The End')
```

Результат работы будет таким:

```
0
1
2
```

Оператор *for*

Оператор `for` реализует цикл с параметром. Используют этот оператор в ситуациях, когда количество повторений цикла заранее известно.

Синтаксис оператора `for`:

```
for <переменная-параметр> in <множество значений переменной-параметра>:
```

```
    <тело цикла>
```

Переменная-параметр принимает начальное значение из множества значений, далее выполняется первая итерация, после чего значение переменной-параметра автоматически изменяется на следующее значение и снова выполняется тело цикла, затем значение переменной-параметра будет вновь изменено и так далее до достижения переменной-параметром конечного значения.

Рассмотрим пример. Следующий фрагмент кода выведет на экран строку «я – студент» три раза:

```
for i in 1,2,3:  
    print ("я – студент")
```

Оператор цикла `for` так же, как и оператор `while`, может прерываться при помощи `break` и `continue`, содержать часть `else`, выполняющуюся только если не был выполнен досрочный выход из цикла.

Значения переменной-параметра удобно задавать с помощью функции `range`, позволяющей генерировать ряд чисел в рамках заданного диапазона.

Функция `range` может быть вызвана тремя способами: с одним, двумя или тремя аргументами.

В первом случае функция генерирует ряд чисел от нуля до конечного числа, не включая его: `range(10)` – будет сгенерирован ряд чисел от 0 до 9.

Во втором случае функция принимает два аргумента – начальное и конечное значения; будет сгенерирован ряд чисел от начального числа до конечного, не включая конечное, например: `range(1, 10)` – будет сгенерирован ряд чисел от 1 до 9.

В третьем случае список аргументов дополняется третьим параметром, шагом, т. е. значением, на которое будет изменено предыдущее значение аргумента за один шаг. Шаг может быть как положительным, так и отрицательным, например: `range(1, 10, 2)` – будет сгенерирован ряд нечётных чисел от 1 до 9 включительно. С учётом вышесказанного предыдущий пример можно переписать следующим образом:

```
for i in range(3):  
    print ("я – студент")
```

или так:

```
for i in range(1,4):
```

```
print ("я - студент")
```

или так:

```
for i in range(1,4,1):  
    print ("я - студент")
```

Результат работы приведённых фрагментов кода в любом случае будет таким же, как и в предыдущем примере.

Задания для практической работы

Все задания необходимо решить двумя способами: используя оператор *while* и с помощью оператора *for*.

1. С клавиатуры вводится натуральное число. Выведите количество значащих разрядов в этом числе.

2. Выведите на экран (в одну строку) все натуральные числа, не превосходящие 150, которые кратны 3 и не оканчиваются на 3.

3. С клавиатуры вводится натуральное число N . Выведите на экран кубы всех натуральных четных чисел от 1 до N .

4. С клавиатуры вводится последовательность целых чисел, завершающаяся числом 0. Выведите на экран среднее арифметическое всех элементов последовательности.

5. Выведите на экран все натуральные числа от 10 до 50, кроме чисел, кратных 8, используя оператор *continue*.

Задания для самостоятельной работы

1. Пользователь вводит с клавиатуры целые числа до тех пор, пока не введется 0. Проверьте, встретится ли в введенной последовательности число 1 и выведите сообщение об этом, используя *else*.

2. С клавиатуры вводятся натуральные числа n и d . Выведите все натуральные числа, меньшие n , квадрат суммы цифр которых равен d .

3. Введите с клавиатуры целое число. Выведите сумму цифр в этом числе.

4. Выведите на экран (в одну строку) все натуральные числа, не превосходящие 550, которые кратны 5 и не оканчиваются на 0.

5. Введите с клавиатуры натуральное число N . Выведите на экран сумму всех квадратов натуральных нечетных чисел на полуинтервале $[1; N)$.

6. Введите с клавиатуры натуральные числа a и b . Выведите разность между суммой квадратов всех чисел на отрезке $[a; b]$ и суммой всех чисел на этом же отрезке.

7. Введите с клавиатуры целое число n ($n > 0$). Если число является степенью числа 7, выведите на экран слово «YES», в противном случае – слово «NO».

8. Введите с клавиатуры натуральное число n . Выведите на экран $n!$ (факториал числа n).

9. Введите с клавиатуры целое число n ($n \geq 2$). Выведите его наименьший натуральный делитель, отличный от 1.

10. Вклад в банке составляет x руб. Ежегодно вклад увеличивается на p процентов, после чего дробная часть копеек отбрасывается. Каждый год сумма вклада становится больше. Определите, через сколько лет вклад составит не менее y руб.

Необходимо ввести с клавиатуры три натуральных числа x , p , y , вывести – целое число.

11. Введите с клавиатуры натуральное число N . Выведите представление этого числа в двоичной системе счисления, но в обратном порядке.

12. Введите с клавиатуры натуральное число n . Выведите все нетривиальные делители этого числа (т. е. делители, не равные самому числу n и 1) в порядке возрастания через пробел. Если таких делителей нет, выведите слово «Prime».

13. Введите с клавиатуры последовательность целых чисел, завершающуюся числом 0. Вычислите, сколько было введено положительных чисел и сколько отрицательных, выведите два этих числа через пробел.

14. Введите с клавиатуры последовательность целых чисел, завершающуюся числом 0. Определите и выведите на экран количество чисел, равных значению максимального числа в последовательности.

15. Элемент последовательности называют локальным максимумом, если он строго больше предыдущего и последующего элементов последовательности. Первый и последний элементы последовательности не являются локальными максимумами.

Введите с клавиатуры последовательность целых чисел, завершающуюся числом 0. Определите количество строгих локальных максимумов в этой последовательности.

16. Введите с клавиатуры последовательность целых чисел, завершающуюся числом 0. Определите наименьшее расстояние между двумя локальными максимумами последовательности. Если в последовательности нет двух локальных максимумов, выведите -1 . Начальное и конечное значения локальными максимумами не являются. Расстоянием между максимумами назовём количество элементов между ними.

17. Число называют палиндромом, если оно не изменяется при перестановке его цифр в обратном порядке. Напишите программу, которая по заданному числу K выводит количество натуральных палиндромов, не превосходящих значение K .

18. Введите с клавиатуры последовательность целых чисел, завершающуюся числом 0. Определите, какое наибольшее число подряд идущих элементов этой последовательности равно друг другу.

19. Введите с клавиатуры натуральное число b . Выведите на экран строку "я – программист" b раз.

20. Введите с клавиатуры неотрицательное число b . Найдите сумму первых b неотрицательных чисел.

21. Введите с клавиатуры два натуральных числа a и b (по одному в строке), причем $b > a$. Выведите на экран все целые числа, кратные 3, от a до b включительно.

22. Введите с клавиатуры натуральное число n . Затем введите с клавиатуры n натуральных чисел. Определите количество введенных чисел, которые кратны 5 и не оканчиваются на 0.

23. Введите с клавиатуры натуральное число n . Выведите квадраты натуральных чисел от 1 до n в порядке убывания.

24. Выведите на экран все возможные трехзначные натуральные числа, в записи которых первые две цифры одинаковые.

25. Рассматриваются все возможные пары различных натуральных чисел меньше 50. Выведите все пары (по одной паре в строке, числа в одной строке разделите пробелом), сумма чисел которых оканчивается на 2.

Для справки: под парой чисел подразумевается их определенная последовательность, т. е. (10, 43) и (43, 10) – две различные пары чисел.

26. На вход подаются два числа n и x . Необходимо посчитать сумму $x^1 + x^2 + x^3 + \dots + x^n$ и вывести ее на экран.

27. Введите с клавиатуры целые числа a и b . Гарантируется, что a не превосходит b . Выведите (через пробел) все четные числа от a до b (включительно).

28. Введите с клавиатуры натуральное число N , а затем N целых чисел. Необходимо подсчитать и вывести на экран, сколько среди этих N чисел нулей, положительных чисел, отрицательных чисел.

29. Введите с клавиатуры четыре числа a , b , c и d . Найдите все целые решения уравнения $ax^3 + bx^2 + cx + d = 0$ на отрезке $[0, 1000]$ и выведите их в порядке возрастания. Если на данном отрезке нет ни одного решения, ничего выводить не нужно.

30. Введите с клавиатуры натуральное число n . Выведите на экран таблицу умножения (от 1 до 10) введенного числа.

31. Напишите программу, которая принимает на вход строку и печатает ее на экран в рамочке из символов $+$, $-$ и $|$. Слева и справа текст должен отделяться от рамки пробелом.

Пример вывода:

```
+-----+
| Harry cat |
+-----+
```

32. Введите с клавиатуры положительные числа A , B , C . Найдите максимально возможное количество квадратов со стороной C , размещенных (без наложений) на прямоугольнике размером $A \times B$. Операции умножения и деления при решении задачи не использовать.

33. Напишите программу, которая выводит на экран числа от 1 до 100. При этом вместо чисел, кратных 3, программа должна выводить слово «*Fizz*», а вместо чисел, кратных 5, – слово «*Buzz*». Если число кратно и 3, и 5, то программа должна выводить слово «*FizzBuzz*».

Вопросы для закрепления материала

1. Что называют циклом в программировании?
2. С помощью каких операторов реализуют цикл на языке программирования *Python*?
3. Чем необходимо руководствоваться при выборе оператора, которым будет организован цикл?
4. В каком случае при организации цикла используют ключевое слово *else*?
5. Что такое досрочный выход из цикла?

6. Каким оператором в языке программирования *Python* можно реализовать досрочный выход из цикла?

7. Куда будет передано управление, если в теле цикла встречен оператор досрочного выхода из цикла?

8. Каким образом можно досрочно прервать выполнение текущей итерации цикла и перейти на следующую?

9. Каким образом можно задать список значений, которые будет принимать переменная-параметр в цикле *for*?

10. Поясните назначения параметров функции *range*.

13. СТРОКИ

Строки – объекты, состоящие из последовательности символов. Для того чтобы присвоить строку переменной, ее надо заключить в одинарные кавычки: `s = 'I like study'`. Строку можно считать со стандартного ввода функцией `input ()`. Любой другой объект в *Python* можно привести к строке. Для этого нужно использовать функцию `str()`, передав ей в качестве параметра объект, переводимый в строку. И наоборот, при необходимости строку можно привести к другому классу, например к `int`:

```
s='1234' # строка
c=int(s) # c – целое число, хранящее значение 1234
a=567 # a – целое число, хранящее значение 567
b=str(a) # b – строка
```

Операции для работы со строками

`+` – оператор сложения строк (конкатенации). Возвращает строку, состоящую из двух строк-параметров:

```
s='abcd'
c='efg'
d=s+c
```

В переменной `d` будет содержаться строка `abcd`.

`*` – оператор умножения строки на число. Создает несколько копий строки-параметра:

```
s='abc'
d=s*3
```

Переменная `d` примет значение `abcabcabc`.

Срезы – извлечение из строки подстроки; это может быть один символ или некоторый фрагмент строки. Существуют три типа срезов:

1. Срез с одним параметром. `s[i]` – это срез, состоящий из одного символа, который имеет индекс `i`. Нумерация символов строки начинается с нуля. Например, если `s == 'Python'`, то `s[0] == 'P'`, `s[1] == 'y'`, `s[2] == 't'` и т. д. Каждый объект, который получается в результате среза `s[i]`, – это тоже строка типа `str`. Если указать отрицательное значение индекса, то номер будет отсчитываться с конца, начиная с номера `-1`, т. е. `-1` это номер последнего символа.

2. Срез с двумя параметрами. `s[a:b]` возвращает подстроку из `b-a` символов, начиная с символа с индексом `a`, до символа с индексом `b`, не включая его. Например, `s[1:4] == 'yth'`. То же самое получится, если написать `s[-4:-1]`. Можно использовать как положительные, так и отрицательные индексы в одном срезе, например: `s[1:-1]` – это строка без первого и последнего символов (срез начинается с символа с индексом `1` и заканчивается индексом `1`, не включая его).

Если опустить второй параметр, но поставить двоеточие, то срез берется до конца строки. Например, чтобы удалить из строки первый символ (его индекс равен `0`), можно взять срез `s[1:]`. Если опустить первый параметр, то можно взять срез от начала строки, т. е. удалить из строки последний символ можно при помощи среза `s[:-1]`. Срез `s[:]` совпадает с самой строкой `s`.

3. Срез с тремя параметрами. `s[a:b:step]` – третий параметр задает шаг, таким образом, в срез будут включены символы с индексами `a`, `a + step`, `a + 2step` и т. д. до `b`, не включая `b`. При задании значения третьего параметра, равного `2`, в срез попадет каждый второй символ, а если взять шаг `-1`, то символы будут идти в обратном порядке. Например, можно перевернуть строку срезом `s[::-1]`.

Функция `len()` возвращает количество элементов строки, переданной в функцию в качестве аргумента, т. е. длину этой строки. Например: `len('abcdef') == 6`

Функция `len()` может использоваться не только для строк, но и для любых последовательностей.

Оператор принадлежности `in` возвращает `True`, если подстрока входит в строку, и `False`, если подстрока не входит в строку.

Оператор `not in` возвращает *True*, если подстрока не входит в строку, и *False* в противном случае.

Для фрагмента кода:

```
s='qwerty'  
print('wer' in s)  
print('rwe' in s)  
print('ty' in s)  
print('try' in s)
```

Результат будет таким:

True

False

True

False

Рассмотрим некоторые методы работы со строками.

Method – это функция, применяемая к объекту (в данном случае – к строке). Для вызова метода следует указать после имени объекта через точку имя метода с круглыми скобками, содержащими необходимые параметры:

```
<имя объекта>.<имя метода>(<параметры>)
```

Метод `count()` возвращает количество вхождений одной строки в другую строку. Например, `s.count(t)` возвращает число вхождений строки `t` внутри строки `s`. При этом подсчитываются только непересекающиеся вхождения (т. е. вхождения, которые не перекрывают друг друга).

Метод `find()` находит в строке данную подстроку, которая передается в качестве параметра, и возвращает индекс первого вхождения искомой подстроки. Если же подстрока не найдена, то метод возвращает значение `-1`.

Метод `rfind()` возвращает индекс последнего вхождения данной подстроки в строке.

Метод `replace(old, new[, k])` заменяет старую подстроку на новую. У метода есть третий необязательный параметр: с его помощью можно указать количество вхождений (первых), которые нужно заменить. Если его не писать, заменятся все вхождения.

Применим к строке `S="баобаb"` описанные методы. Результат применения каждого метода приведён в комментариях:

```
print(S.count('ба')) // на экране – 2
```

```
print(S.count('б')) ## на экране – 3
print(S.find('ба')) ## на экране – 0
print(S.rfind('ба')) ## на экране – 3
print(S.replace('ба', '*')) ## на экране – *о*б
print(S.replace('ба', '*',1)) ## на экране – *обаб
```

Задания для практической работы

Дана строка из десяти символов. Выведите на экран:

1. В первой строке выведите пятый символ этой строки. Во второй строке выведите предпоследний символ этой строки.
2. В третьей строке выведите первые семь символов этой строки. В четвертой строке выведите всю строку, кроме последних трех символов.
3. В пятой строке выведите все символы с нечетными индексами. В шестой строке выведите все символы с четными индексами.
4. В седьмой строке выведите все символы в обратном порядке. В восьмой строке выведите все символы строки через один в обратном порядке, начиная с последнего.
5. В девятой строке выведите длину данной строки.

Задания для самостоятельной работы

1. Напишите программу, которая принимает на вход три строковые переменные, а на выходе объединяет их в одну строку, разделяя символом "#".
2. С клавиатуры вводится строка произвольной длины. Выведите новую строку, состоящую только из символов, индекс которых не кратен 2.
3. Дана строка, состоящая из одного предложения. Определите, сколько в предложении слов.
4. С клавиатуры вводится строка. Если в этой строке буква *x* встречается только один раз, выведите её индекс. Если она встречается два и более раз, выведите индекс её первого и последнего вхождения. Если буква в данной строке не встречается, выведите *NO*.
5. С клавиатуры вводится строка, в которой буква *z* встречается минимум два раза. Удалите из этой строки первое и последнее вхож-

дение буквы z , а также все символы, находящиеся между ними. Выведите получившуюся строку на экран.

6. С клавиатуры вводится строка, состоящая из неповторяющихся символов. Удалите из нее все символы, индексы которых кратны 3. Выведите получившуюся строку на экран.

7. С клавиатуры вводится строка. Замените в этой строке все появления буквы a на букву A , кроме первого и последнего вхождения.

8. С клавиатуры вводится строка. Если в этой строке буква x встречается только один раз, выведите её индекс. Если она встречается два и более раза, выведите индекс её первого и последнего появления. Если буква в данной строке не встречается, выведите *NO*.

9. Дана строка, состоящая из одного предложения. Определите, сколько в предложении слов, начинающихся с буквы "Б" или "б".

10. С клавиатуры вводится произвольная строка. Если в этой строке буква "а" (русская) встречается только один раз, выведите её индекс. Если она встречается два и более раза, выведите индекс её первого и последнего вхождения. Если буква "а" в данной строке не встречается, выведите *FALSE*.

11. С клавиатуры вводится строка, в которой буква S встречается минимум три раза. Удалите из этой строки первое и последнее вхождение буквы S . Выведите получившуюся строку на экран.

12. С клавиатуры вводится строка, состоящая из неповторяющихся символов. Удалите из нее все символы, индексы которых кратны 2 и не кратны 3. Выведите получившуюся строку на экран.

13. С клавиатуры вводится строка. Замените в этой строке все появления букв a, b на букву G .

14. Дана строка. Разрежьте ее на две равные части, если длина строки четная; если длина строки нечетная, то длина первой части должна быть на один символ больше. Переставьте эти две части местами, результат запишите в новую строку и выведите на экран.

15. Дана строка, состоящая из двух слов, разделенных пробелом. Переставьте эти слова местами. Результат запишите в строку и выведите получившуюся строку.

16. Дана строка. Вставьте символ "*" между каждой парой символов исходной строки и выведите получившуюся последовательность на экран.

17. Дана строка, содержащая пробелы. Найдите, сколько в ней слов (слово – это последовательность непробельных символов, слова разделены одним пробелом, первый и последний символы строки – не пробел).

18. С клавиатуры вводится строка. Замените в ней все гласные строчные буквы латинского алфавита на прописные, кроме первого и последнего вхождения.

19. По заданной строке определите, является ли она палиндромом (т. е. читается одинаково как слева направо, так и справа налево). Необходимо вывести *YES*, если строка является палиндромом, и *NO* в противном случае.

20. С клавиатуры вводится строка, состоящая только из символов "(" и ")". Выведите *True*, если строка является правильной скобочной последовательностью, в противном случае выведите *False*.

21. Пользователь вводит строку, включающую цифры. Замените все нечетные цифры на их название и выведите полученную строку.

Вопросы для закрепления материала

1. Что такое строка с точки зрения языка программирования *Python*?

2. Какую операцию называют операцией конкатенации?

3. Какой результат будет получен при умножении строки на целое число?

4. Какую функцию нужно использовать для получения длины строки?

5. Какой номер имеет первый символ строки?

6. Каким образом можно получить последний символ строки?

7. Что такое срезы в языке программирования *Python*?

8. Что такое подстрока?

9. Каким образом можно определить вхождение подстроки в строку?

10. Каким образом можно заменить одну подстроку в строке на другую?

14. МАССИВЫ. СПИСКИ

Для создания массива в языке программирования *Python* необходимо указать ключевое слово `array` с пустыми круглыми скобками или со скобками, содержащими список значений.

Массив может содержать элементы только одного типа данных (например, только целые числа). Списки в *Python* имеют более широкий функционал, поэтому их используют чаще.

Список – именованный упорядоченный изменяемый набор объектов произвольных типов. Нумерация элементов начинается с 0.

Для создания списка существует несколько способов. Рассмотрим каждый из них.

1. Перечисление элементов списка в квадратных скобках. Например:

```
d=[1,2,3,4,-9,5]
d1=[]
d2=[1,'w',123,'Python',3.14]
```

где *d* – список целых чисел, *d1* – пустой список, *d2* – список, содержащий разнотипные данные, в том числе и другой список.

Для того чтобы обратиться к определенному элементу списка, необходимо написать имя списка, а затем индекс требуемого элемента в квадратных скобках: например, запись `a[0]` соответствует нулевому элементу списка *a*.

На экран можно вывести как список целиком (при этом список будет заключён в квадратные скобки), так и его отдельные элементы:

```
print(d2)
print(d2[0])
```

2. Приведение любого другого итерируемого объекта (например, строку) к списку при помощи функции `list()`. Например:

```
S="Python"
List_S=list(S)
```

В списке *List_S* каждый элемент является строкой. Если вывести этот список на экран `print(List_S)`, результат будет таким:

```
['P', 'y', 't', 'h', 'o', 'n']
```

3. Заполнение пустого списка или изменение готового списка при помощи цикла. Например:

```
a=[0]*10 # создали список из 10 нулей
```



```
for i in range(10):  
    a[i]=i # заполнили список значениями от 0 до 9
```

4. Генератор списков. Использование генератора списков очень похоже на цикл for, например:

```
a=[a for a in "Python"]  
print(a)
```

Список a принимает значение из шести строк, каждая из которых равна символу строки "Python". Вывод списка на экран даст следующий результат:

```
['P', 'y', 't', 'h', 'o', 'n']
```

При генерации элементов возможно задание повторения символа в рамках одного элемента, как показано в следующем примере:

```
a=[a*3 for a in "Python"]
```

Список a примет следующие значения:

```
['PPP', 'yuu', 'ttt', 'hhh', 'ooo', 'nnn']
```

Возможно наложение ограничений при генерации списка, например:

```
a=[a*2 for a in "Python" if a!="t"]
```

Список a будет заполнен следующим образом:

```
['PP', 'yy', 'hh', 'oo', 'nn']
```

Операции для работы со списками

Со списками можно выполнять те же операции, что и со строками.

Оператор конкатенации + возвращает список, состоящий из элементов других списков-операндов. Например:

```
d=[1,2,3,4,-9,5]  
d1=[1, 'w', 123, 'Python', [1,2,3], 3.14]  
d2=d+d1  
print(d2)
```

На экран будет выведено:

```
[1, 2, 3, 4, -9, 5, 1, 'w', 123, 'Python', [1, 2, 3], 3.14]
```

Оператор * умножения списка на число создает несколько дубликатов списка. Например:

```
d=[1,2,3, 'w']  
d2=d*3
```

Будет создан список d2:

```
[1, 2, 3, 'w', 1, 2, 3, 'w', 1, 2, 3, 'w']
```

Из списков можно извлекать срезы по тем же правилам, что и из строк:

```
d=['P','y','t','h','o','n']
```

```
d2=d[2:4]
```

Список d2 будет равен ['t', 'h']

```
print(d[2:-1:]) – на экран будет выведено ['t', 'h', 'o']
```

В случае среза `print(d[2::-1])` на экран будет выведено ['t', 'y', 'P'].

Методы для работы со списками

Для вызова метода необходимо указать имя этого метода после имени списка, разделив их точкой. После имени метода ставят круглые скобки, содержащие при необходимости список параметров:

`append(x)` – добавляет элемент x в конец списка

```
S=[1,2,3,4]
```

```
S.append(5)
```

Список S будет содержать следующие элементы: [1, 2, 3, 4, 5].

`extend(listname2)` – добавляет в конец списка все элементы списка listname2

```
S=[1,2,3,4,5]
```

```
S1=[6,7,8,9,]
```

Список S будет содержать следующие элементы: [1, 2, 3, 4, 5, 6, 7, 8, 9].

`insert(i, x)` – вставляет значение x на i-ю позицию списка:

```
S=['P','y','t','h','o','n']
```

```
S.insert(7, '!')
```

Список примет значение ['P', 'y', 't', 'h', 'o', 'n', '!'].

`remove(x)` – удаляет из списка первый элемент, равный x:

```
S.remove('!')
```

Список примет первоначальное значение.

```
S=['a','b','a','b','a','b']
```

```
S.remove('a')
```

Первый элемент будет удалён из списка. Список примет значение ['b', 'a', 'b', 'a', 'b']

`pop(i)` – удаляет элемент списка с индексом i. Если индекс не задан, то удаляется последний элемент списка:

```
S=['P','y','t','h','o','n']
```

```
S.pop(0)
```

```
S.pop()
```

Список *S* примет значение ['y', 't', 'h', 'o']

`sort()` – сортирует список в порядке возрастания по умолчанию и в порядке убывания, если задать значение параметра `reverse = True`:

```
S=[1,5,0,4,-3,-2,2,-1,6,3]
```

```
S.sort()
```

Список *S* примет значение [-3, -2, -1, 0, 1, 2, 3, 4, 5, 6]

```
S.sort(reverse = True)
```

Значение списка *S* [6, 5, 4, 3, 2, 1, 0, -1, -2, -3]

`count(x)` – возвращает количество элементов со значением *x* в списке:

```
S=['a','a','b','a','b','a','b']
```

```
c=S.count('a') ## c=4
```

`index(x, a, b)` – возвращает индекс первого элемента со значением *x* в списке в диапазоне индексов от *a* до *b*, не включая последний (диапазон можно не указывать, тогда поиск будет осуществлён по всему списку):

```
S=['a','a','b','a','b','a','b']
```

```
c=S.index('a',2,6) ##c=3
```

```
d=S.index('a') ## d=0
```

`copy()` – копирует список и возвращает копию в точку вызова:

```
S1=S.copy() ## Строка S1 равна строке S
```

`clear()` – очищает список:

```
S.clear()
```

`str.split(separator[, maxsplit])` – разбивает строку *str* на части по разделителю *separator* и возвращает эти части в виде списка. Если разделитель не задан, то в качестве разделителя используется пробел и символ новой строки. Необязательный параметр *maxsplit* определяет максимальное количество частей, на которые можно разбить строку. Например:

```
str='aaaaabaabaab'
```

```
str1='ab ab abbb'
```

```
s=str.split('a')
```

```
##s=['', '', '', '', '', 'b', '', '', 'b', '', 'b']
s1=str.split('a',2)
## ['', '', 'aaabaaabaab']
s2=str1.split()
##['ab', 'ab', 'abbb']
```

Функция `map(function, iterable_object)` применяет функцию `function` к каждому элементу итерируемого объекта, переданного вторым аргументом, т. е., по сути, заменяет цикл `for`, перебирающий элементы. `map()` может принимать несколько итерируемых объектов в качестве аргументов функции, отправляя в функцию по одному элементу каждого итерируемого объекта за раз. Функция `map()` возвращает не список, а объект `map` (итератор), поэтому для приведения его к списку нужно использовать `list()`. Например:

```
S=[1,2,3,4,5]
```

```
S1=list(map(float, S ))
```

Список `S1` примет значения `[1.0, 2.0, 3.0, 4.0, 5.0]`

Задания для практической работы

1. Создайте список из 15 целых чисел. Выведите на экран все его элементы, кратные 3, но не кратные 6.
2. Создайте два списка из натуральных чисел. Выведите список, объединенный из двух исходных, но оставьте в нем только элементы, оканчивающиеся на 2.
3. Создайте список целых чисел. Выведите элементы списка, кратные индексу минимального элемента.
4. Пользователь вводит с клавиатуры целые числа. Концом последовательности является 0. Выведите на экран список, состоящий из кубов введенных чисел.
5. С клавиатуры вводится 12 целых чисел. Выведите на экран сумму этих чисел, а затем все эти числа в порядке убывания.

Задания для самостоятельной работы

1. Пользователь вводит числа одной строкой через запятую. Выведите эти числа в порядке возрастания.

2. На вход программы подаётся список натуральных чисел, не превышающих 30, одной строкой через пробел. Программа должна вывести числа, которые встречаются в списке более одного раза.

3. С клавиатуры вводится натуральное число n , затем n целых чисел (каждое число с новой строки). Выведите на экран сумму квадратов этих чисел, а затем все эти числа в одну строку, разделяя их символом ";".

4. С клавиатуры вводится натуральное число n , затем n натуральных чисел (каждое число с новой строки). Выведите на экран все его элементы (по одному в строке), кратные 5, но не кратные 25. Элементы должны быть отсортированы в порядке возрастания.

5. С клавиатуры вводится натуральное число n , затем n натуральных чисел (каждое число с новой строки). Найдите сумму тех элементов, чей индекс оканчивается на последнюю цифру минимального числа в списке.

6. Пользователь вводит с клавиатуры целые числа. Концом последовательности является 0. Выведите на экран упорядоченный по убыванию список (по одному элементу в строке), состоящий из произведения цифр введенных чисел.

7. Пользователь вводит числа одной строкой через точку с запятой. Выведите список отсортированных по возрастанию чисел (в одной строке).

8. С клавиатуры вводится натуральное число n , затем n натуральных чисел (каждое число с новой строки). Необходимо заменить на *NONE* числа, которые отличаются от максимального элемента не более чем на 3. Выведите получившийся список одной строкой, разделяя элементы пробелом.

9. На вход программы подаётся список натуральных чисел, не превышающих 50, одной строкой через пробел. Программа должна вывести числа, которые встречаются в списке менее четырёх раз, в порядке возрастания в одну строку через пробел.

10. На вход программы подаётся список натуральных чисел. Напишите программу, которая находит самое большое число, а затем делит его на длину списка.

11. Напишите программу, которая принимает два списка и выводит все элементы первого, которых нет во втором (в том же порядке, в котором они встречались в исходном списке).

12. Пользователь вводит список натуральных чисел в первой строке и число x во второй. Определите, содержит ли список данное число x . Если содержит, то выведите на экран *YES*, если не содержит, то выведите на экран *NO*.

13. Дан список натуральных чисел. Удалите из него первое четное число, имеющее нечетный индекс. Выведите измененный список.

14. С клавиатуры вводится натуральное число n , затем n целых чисел (каждое число с новой строки). Выведите все уникальные элементы списка (в одну строку) в порядке возрастания.

15. На вход программы подаётся список целых чисел. Напишите программу, которая переставит элементы списка так, чтобы все положительные элементы стояли в начале в порядке возрастания, а все отрицательные и нули – в конце в порядке убывания.

16. На вход программы подаётся список целых чисел. Напишите программу, которая выводит список чисел по убыванию их абсолютного значения.

17. На вход программы подаётся список целых чисел. Напишите программу, которая выводит список чисел по убыванию их абсолютного значения.

18. Создайте список целых чисел. Выведите максимальный элемент списка, являющийся полным квадратом. Если таких элементов нет, выведите *НЕТ*.

Вопросы для закрепления материала

1. Что называют массивом в программировании?
2. Что называют списком в языке программирования *Python*?
3. В чём отличие массивов от списков?
4. Каким образом можно создать список?
5. Какие операции применимы к спискам?
6. Какие методы работы со списками вы знаете?
7. Какой метод используют для очистки списка?
8. Какой метод используют для добавления в конец списка нового элемента?
9. Какой метод используют для добавления в конец списка другого списка?
10. Какой метод используют для сортировки списка?

15. ФАЙЛЫ

Для организации работы с файлом необходимо открыть его с помощью функции `open(имя файла)`, указав в круглых скобках имя открываемого файла, например:

```
f = open(«filename.txt»)
```

В качестве второго параметра можно указать режим открытия файла (табл. 4).

Таблица 4

Режим открытия файла

Режим	Обозначение
<i>r</i>	Открытие на чтение (является значением по умолчанию)
<i>w</i>	Открытие на запись, содержимое файла удаляется, если файла не существует, создается новый
<i>x</i>	Открытие на запись, если файла не существует, иначе исключение
<i>a</i>	Открытие на дозапись, информация добавляется в конец файла
<i>b</i>	Открытие в двоичном режиме
<i>t</i>	Открытие в текстовом режиме (является значением по умолчанию)
<i>+</i>	Открытие на чтение и запись

Режимы могут быть объединены, например `'rb'` – чтение в двоичном режиме. По умолчанию режим равен `'rt'`.

Для того чтобы прочитать одну строку из файла, используют функцию `readline()`:

`a = f.readline()` – из файла `f` будет считана строка и сохранена в переменной `a`

`a = int(f.readline())` – считанная строка будет переведена в целое число.

При этом файл с программой и файл с данными должны лежать в одной папке.

`f.readlines()` – возвращает список строк, считанных из файла.

Для записи данных в файл необходимо открыть файл для записи и использовать метод `write()`, возвращающий число записанных символов. После окончания работы с файлом его обязательно нужно закрыть с помощью метода `close()`:

```
f.close()
```

Задания для практической работы

1. В файле `file1.txt` находится строка. Посчитайте, сколько раз в ней встречается цепочка символов «1234».

2. В файле `file1.txt` находится строка из целых чисел, разделенных пробелами. Посчитайте, сколько раз в этой строке встречается число 0.

3. Файл `file2.txt` в первой своей строке содержит число N , а в остальных N строках – по одному числу. Выведите на экран последние три цифры суммы этих чисел, а также выведите все эти числа, отсортированные по убыванию, в одну строку через пробел.

4. Файл `file2.txt` в первой своей строке содержит число N , а в остальных N строках – по одному числу. Выведите, сколько раз в этой последовательности встречается минимальное число.

5. В файле электронной таблицы содержатся целые числа – координаты трёх точек в формате $X_1, X_2, X_3, Y_1, Y_2, Y_3$. Найдите количество таких строк, где все три точки лежат в одной четверти координатной плоскости.

Задания для самостоятельной работы

1. В файле электронной таблицы содержатся тройки натуральных чисел. Определите количество троек, в которых квадрат максимального из трёх чисел больше удвоенного произведения двух других чисел в строке.

2. В файле электронной таблицы в каждой строке содержатся четыре натуральных числа. Определите количество строк таблицы, в которых модуль куба разности максимального и минимального чисел в строке не превышает квадрат суммы двух оставшихся.

3. В файле электронной таблицы в каждой строке содержатся пять натуральных чисел. Определите количество строк таблицы, в ко-

торых квадратный корень произведения максимального и минимального чисел в строке больше кубического корня из произведения трех оставшихся.

4. В файле электронной таблицы в каждой строке содержатся четыре натуральных числа. Определите количество строк таблицы, содержащих числа, для которых выполнены оба условия:

- наибольшее из четырёх чисел меньше суммы трёх других;
- среди четырёх чисел есть только одна пара равных чисел.

5. В файле электронной таблицы в каждой строке содержатся шесть неотрицательных целых чисел. Определите количество строк таблицы, для которых выполнены оба условия:

- в строке только одно число повторяется дважды, остальные числа не повторяются;
- среднее арифметическое неповторяющихся чисел строки не больше суммы повторяющихся чисел.

6. В файле электронной таблицы содержатся целые числа – координаты трёх точек в формате $A_1, A_2, A_3, B_1, B_2, B_3$. Найдите, сколько таких строк, в которых две точки лежат в одной и той же четверти координатной плоскости.

7. В файле электронной таблицы содержатся четверки натуральных чисел. Определите количество четверок, в которых пятая степень минимального из четырех чисел меньше утроенного произведения трех других чисел в строке.

8. В файле электронной таблицы в каждой строке содержатся четыре натуральных числа. Определите количество строк, в которых утроенный модуль куба разности максимального и минимального чисел в строке превышает куб суммы двух оставшихся чисел.

9. В файле электронной таблицы в каждой строке содержатся шесть натуральных чисел. Определите количество строк, в которых удвоенный квадратный корень утроенного произведения максимального и минимального чисел в строке меньше кубического корня из удвоенного произведения четырех оставшихся.

10. В файле электронной таблицы в каждой строке содержатся пять натуральных чисел. Определите количество строк таблицы, содержащих числа, для которых выполнены оба условия:

- произведение наибольшего и наименьшего из пяти чисел меньше суммы трёх других;

– среди пяти чисел есть одна тройка равных чисел, а остальные числа различны.

11. В файле электронной таблицы в каждой строке содержатся четыре натуральных числа. Определите количество строк таблицы, содержащих числа, для которых выполнены оба условия:

– куб минимального числа не меньше, чем произведение двух других чисел, ни одно из которых не равно максимальному;

– в строке есть хотя бы два одинаковых числа и хотя бы одно число отлично от остальных.

12. В файле электронной таблицы в каждой строке содержатся пять целых чисел. Определите количество строк таблицы, для которых выполнены оба условия:

– в строке только одно число повторяется дважды, остальные числа не повторяются;

– квадрат среднего арифметического неповторяющихся чисел строки не меньше произведения повторяющихся чисел.

13. В файле электронной таблицы в каждой строке содержатся шесть неотрицательных целых чисел. Определите количество строк таблицы, для которых выполнены оба условия:

– в строке только одно число повторяется трижды, остальные числа не повторяются;

– утроенная сумма повторяющихся чисел строки не больше произведения неповторяющихся чисел.

14. В файле электронной таблицы в каждой строке содержатся три числа. Определите количество строк, где нет повторяющихся чисел и хотя бы одно произведение двух любых чисел равно четвертой степени целого числа.

15. В файле электронной таблицы в каждой строке содержатся десять целых чисел. Определите количество строк таблицы, для которых выполнены следующие условия:

– в строке одно число повторяется трижды (назовем это число α), другое число повторяется дважды (назовем его β), остальные числа различны;

– произведение чисел α и β не превосходит среднего геометрического модулей неповторяющихся чисел.

Вопросы для закрепления материала

1. Какие функции для работы с файлами на языке *Python* вы знаете?
2. Что необходимо выполнить перед началом работы с файлом?
3. Каким образом можно считать данные из файла?
4. Какая функция позволяет считать строку из файла?
5. Какая функция позволяет считать все строки из файла в список и возвращает его?
6. Какую функцию нужно использовать для записи в файл?
7. Какая функция закрывает файл?
8. Какие параметры нужно передать методу *open()* для открытия файла для чтения?
9. Какие параметры нужно передать методу *open()* для открытия файла для записи?
10. Какие параметры нужно передать методу *open()* для открытия файла для дозаписи?

16. ПРОЦЕДУРЫ. ФУНКЦИИ

Подпрограмма – часть программы, имеющая имя и решающая свою отдельную задачу. Все подпрограммы располагаются в начале основной программы и могут быть вызваны внутри основной программы по имени. Подпрограммы используют, когда в разных частях основной программы необходимо выполнять одни и те же действия несколько раз. В таком случае повторяемые операторы (решающие отдельную задачу) оформляются в виде подпрограммы, к которой можно обращаться и вызывать ее выполнение из разных частей программы. Подпрограммы позволяют избежать дублирования кода.

Каждая подпрограмма должна решать только одну задачу (что-то вычислять, выводить какие-либо данные и т. д.). Подпрограммы разделяют на два вида: процедуры и функции. Подпрограммы-процедуры выполняют какие-то действия (например, выводят что-то на экран). Подпрограммы-функции возвращают значение, т. е. результатом их работы является значение (число, строка, список и т. д.). В языке программирования *Python* подпрограммы – процедуры и функции – описываются одинаково.

Функция – это подпрограмма, принимающая аргументы и возвращающая значение. Функции могут быть встроенными, а могут быть пользовательскими, описанными программистом. Пример вызова встроенной функции:

```
l=[1,2,3,4,5]
s=sum(l)
```

Переменная S принимает значение суммы элементов списка l.

Для того чтобы создать собственную функцию, её необходимо объявить и описать. Для этого используется конструкция `def`, после которой указывают имя функции, круглые скобки, содержащие при необходимости список аргументов (параметров), и двоеточие. Далее после отступа следует описание тела функции. Для возвращения результата из функции необходимо использовать ключевое слово `return`, после которого указывают возвращаемое значение. `return` завершает выполнение функции.

Для вызова функции нужно написать ее имя в круглых скобках, содержащих при необходимости список передаваемых параметров.

Ниже приведено описание и вызов функции `Sum_ch`, принимающей на вход список и возвращающей сумму чётных элементов этого списка.

```
def Sum_ch(l):
    S=0
    for i in l:
        if i%2==0:
            S+=i
    return S
l=[1,2,3,4,5]
s=Sum_ch(l)
print(s)
```

В данном случае значение переменной s будет равно 6.

Если подпрограмма не возвращает, её называют *процедурой*.

Пример описания и вызова процедуры приведён ниже.

```
def Err():
    print("Ошибка! Повторите ввод")
n=-1
while n<0:
    n=int(input())
```

```
if n<0:  
    Err()
```

Переменную, объявленную внутри функции, называют локальной. Использовать такие переменные можно только внутри функции.

Глобальные переменные объявляют вне функции. В функциях такие переменные доступны, но изменять значение такой переменной внутри функции нельзя. Например:

```
x=1  
def F():  
    x=2  
    x+=1  
    print("x =",x) ## x=3  
x+=1  
F()  
print("x =",x) ## x=2
```

Функцию, которая внутри своего же тела вызывает сама себя, называют рекурсивной.

Приведём пример вычисления факториала целого неотрицательного числа с помощью рекурсивной функции:

```
n=int(input())  
def f(n):  
    if n >1:  
        return n*f(n-1)  
    else:  
        return 1  
ff= f(n)  
print("f(n) =",ff)
```

По умолчанию глубина рекурсии в языке *Python* ограничена 1000 вызовов.

Задания для практической работы

1. Напишите функцию, которая принимает в качестве аргументов шесть целых чисел и возвращает их сумму. Числа вводятся с клавиатуры одной строкой через пробел.

2. Напишите функцию *century*, принимающую один аргумент – год и возвращающую век, которому этот год принадлежит. Год вводится с клавиатуры.

3. Напишите функцию, принимающую три аргумента: b_1 , q и n . Функция должна вернуть сумму n первых членов геометрической прогрессии. Аргументы вводятся с клавиатуры.

4. Даны четыре отрезка. Выведите *YES*, если среди них найдутся три, из которых можно составить треугольник, и *NO* – в противном случае.

5. Для решения напишите функцию *triangle(a,b,c)*, которая будет возвращать *True*, если из трёх заданных отрезков можно составить треугольник, и *False* – иначе.

Задания для самостоятельной работы

1. Напишите функцию, которая принимает на вход натуральное число и возвращает *True*, если куб его минимального нетривиального делителя превосходит само число, и *False* – в противном случае. В случае если у числа нет нетривиальных делителей – выведите *False*.

2. На вход подается число n , а затем, в следующих n строках, записаны по три числа через пробел: a , b , c . Необходимо определить, какое количество троек чисел являются коэффициентами квадратного уравнения $ax^2 + bx + c = 0$ с двумя действительными корнями.

3. Напишите функцию *min(a, b)*, вычисляющую минимум двух чисел. Затем напишите функцию *min4(a, b, c, d)*, вычисляющую минимум четырех чисел с помощью функции *min*. Введите четыре целых числа и выведите их минимум.

4. Введите последовательность чисел, завершающуюся числом 0. Найдите сумму всех этих чисел, не используя цикл.

5. Алгоритм вычисления функции $F(n)$, где n – целое число, задан следующими соотношениями:

$$F(n) = n, \text{ при } n < 2,$$

$$F(n) = n + F(n // 3), \text{ когда } n > 1 \text{ и делится на } 3,$$

$$F(n) = n + F(n - 3), \text{ когда } n > 1 \text{ и не делится на } 3.$$

Пользователь вводит число n . Выведите на экран значение функции $F(n)$.

6. Напишите функцию *arithmetic*, принимающую три аргумента: первые два – числа, третий – операция, которая должна быть произве-

дена над ними. Если третий аргумент $+$, сложите их; если $-$, то вычтите; $*$ – умножьте; $/$ – разделите (первое на второе). В остальных случаях верните строку "Неизвестная операция".

7. Напишите функцию *is_year_leap*, принимающую один аргумент – год и возвращающую *True*, если год високосный, и *False* – иначе.

8. Напишите функцию *square*, принимающую один аргумент – сторону квадрата и возвращающую три значения (с помощью кортежа): периметр квадрата, площадь квадрата и диагональ квадрата.

9. Напишите функцию *season*, принимающую один аргумент – номер месяца (от 1 до 12) и возвращающую время года, которому этот месяц принадлежит (зима, весна, лето или осень).

10. Пользователь делает вклад в размере *a* руб сроком на *years* лет под 10 % годовых (каждый год размер его вклада увеличивается на 10 %. Эти деньги прибавляются к сумме вклада, и на них в следующем году тоже будут проценты). Напишите функцию *bank*, принимающую аргументы *a* и *years* и возвращающую сумму, которая будет на счету пользователя.

11. Напишите функцию *is_prime*, принимающую один аргумент – число от 0 до 1000 и возвращающую *True*, если оно простое, и *False* – иначе.

12. Напишите функцию *date*, принимающую три аргумента – день, месяц и год. Верните *True*, если такая дата существует в календаре, и *False* – иначе.

13. Напишите функцию *XOR_cipher*, принимающую два аргумента: строку, которую нужно зашифровать, и ключ шифрования и возвращающую строку, зашифрованную путем применения функции *XOR* (^) над символами строки с ключом. Напишите также функцию *XOR_uncipher*, которая по зашифрованной строке и ключу восстанавливает исходную строку.

14. Напишите функцию *multiply3*, которая принимает в качестве аргументов три числа и возвращает их произведение.

15. Напишите функцию *season*, принимающую один аргумент – номер месяца (от 1 до 12) и возвращающую время года, которому этот месяц принадлежит.

16. Напишите функцию, принимающую одно число и проверяющую, является ли это число полным квадратом (вернуть *True* или *False*).

17. Напишите функцию, возвращающую факториал введенного аргумента. *Для справки:* факториал – математическая функция для натурального числа n , обозначаемая $n!$ и равная произведению всех натуральных чисел от 1 до n включительно. Для вычисления факториала используйте цикл.

18. Напишите функцию, которая принимает два аргумента: натуральное десятичное число N и основание системы счисления X (от 2 до 9). Функция должна перевести число N в X СС.

19. Напишите функцию, которая принимает целое число и возвращает список всех его делителей.

20. Напишите рекурсивную функцию вычисления факториала.

21. Напишите функцию для вычисления n -го числа Фибоначчи. *Для справки:* числа Фибоначчи – это ряд чисел, в котором каждое следующее число равно сумме двух предыдущих. Первые два числа Фибоначчи равны 1, т. е. третье число Фибоначчи равно: $1 + 1 = 2$, четвертое: $1 + 2 = 3$ и т. д.

22. Напишите функцию, принимающую три аргумента: a , d и n . Функция должна вернуть сумму n первых членов арифметической прогрессии.

Вопросы для закрепления материала

1. Что такое подпрограмма?
2. Какие виды подпрограмм существуют в языке программирования *Python*?
3. Что такое процедура?
4. Что такое функция?
5. В чём отличие процедуры от функции?
6. Что такое параметры?
7. Какие переменные называют локальными?
8. Какие переменные называют глобальными?
9. Какую функцию называют рекурсивной?
10. Что такое глубина рекурсии?

ЗАКЛЮЧЕНИЕ

Любой вид практики играет ключевую роль при закреплении навыков, полученных студентами в ходе теоретического обучения. Учебная практика позволяет обучающимся ближе познакомиться с осваиваемой профессией, погрузиться в атмосферу командной работы. Это прекрасная возможность закрепить материал, изученный во время теоретической подготовки в течение учебного года, применить для решения одной задачи знания из нескольких дисциплин. Кроме того, учебная практика не столько нацелена на сближение студента с профессией, сколько на расширение его профессиональных компетенций, совершенствование полученных навыков, закрепление и углубление теоретических знаний. Решение большого количества типовых и исследовательских задач позволяет студенту наработать мощную базу, благодаря которой он легко войдет в профессию.

Таким образом, своевременное прохождение учебной практики, решение достаточного количества практических задач позволяют студентам усовершенствовать профессиональные компетенции, сделать большой шаг в освоении профессии.

РЕКОМЕНДАТЕЛЬНЫЙ БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Павловская, Т. А. С/С++. Программирование на языке высокого уровня : учеб. для вузов / Т. А. Павловская. – СПб. : Питер, 2009. – 432 с. – ISBN 978-5-91180-174-8.

2. Страуструп, Б. Программирование. Принципы и практика с использованием С++ / Б. Страуструп. – М. : Вильямс, 2016. – 1328 с. – ISBN 978-5-8459-1949-6.

3. Шилдт, Г. Полный справочник по С++ : пер. с англ. / Г. Шилдт. – 4-е изд., стер. – М. : Вильямс, 2015. – 800 с. – ISBN 978-5-8459-2047-8.

4. Воронова, Л. М. Типовые алгоритмические структуры для вычислений : учеб. пособие / Л. М. Воронова ; Владим. гос. ун-т. – 2-е изд., перераб. и доп. – Владимир : Ред.-издат. комплекс ВлГУ, 2004. – 96 с. – ISBN 5-89368-503-2.

5. Прата, С. Язык программирования С++. Лекции и упражнения : пер. с англ. / С. Прата. – М. : Вильямс, 2017. – 1248 с. – ISBN 978-5-8459-2048-5.

6. Шишкина, М. В. Основы программирования : практикум / М. В. Шишкина. – Владимир : Изд-во ВлГУ, 2021. – 112 с. – ISBN 978-5-9984-1408-4.

7. Шишкина, М. В. Программирование на языке высокого уровня С++ : практикум / М. В. Шишкина. – Владимир : Изд-во ВлГУ, 2022. – 104 с. – ISBN 978-5-9984-1462-6.

ПРИЛОЖЕНИЯ

Приложение 1

Образец титульного листа отчета по учебной практике

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Владимирский государственный университет
имени Александра Григорьевича и Николая Григорьевича Столетовых»
(ВлГУ)

КАФЕДРА ФИЗИКИ И ПРИКЛАДНОЙ МАТЕМАТИКИ

ОТЧЕТ ПО УЧЕБНОЙ ПРАКТИКЕ

Выполнил:

студент _____

группа _____

Принял:

руководитель от ВлГУ

должность _____

И. О. Фамилия _____

Владимир 20 ____

Приложение 2

Задание на учебную практику

Утверждаю

Зав. кафедрой _____

« ____ » _____ 20__ г.

З А Д А Н И Е

На учебную практику студента _____
(фамилия, имя, отчество)

__ курса, направления подготовки 01.03.02 – Прикладная математика
и информатика
группы _____

Предприятие _____

Последовательность прохождения практики _____

За время прохождения практики необходимо:

1. Изучить вопросы, предусмотренные программой по всем разделам.
2. Изучить технологический процесс _____
3. Изучить и исследовать _____

4. Выполнить эскиз _____

5. Задание по стандартизации _____

6. Задание по охране труда, технике безопасности и охране окружающей среды _____

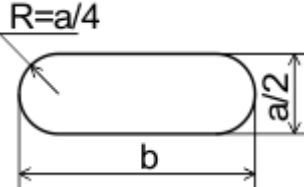
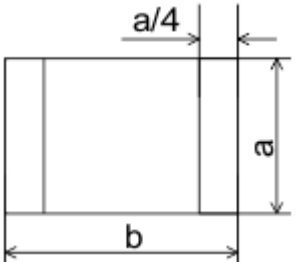
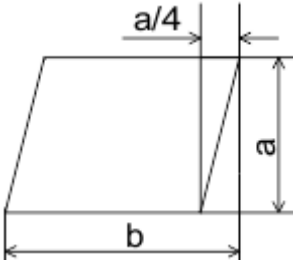
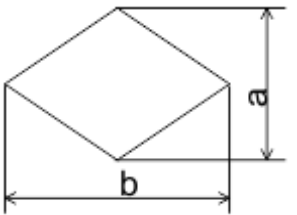
Отчет по практике составить к _____

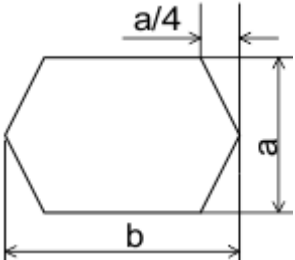
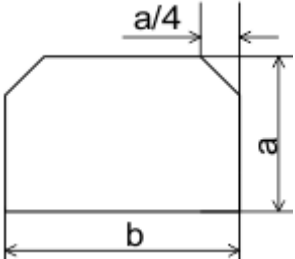
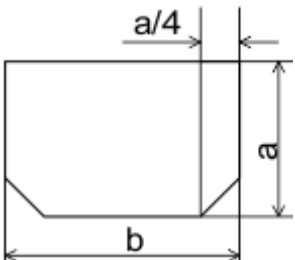
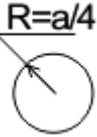
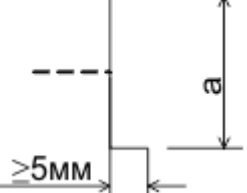
Задание выдал: _____
(фамилия, и., о. руководителя практики от университета)

Задание получил _____
(подпись студента, дата)

Примечание: задание должно быть приложено к отчету по практике (вторым листом после титульного листа)

Основные блоки

Форма блока	Назначение блока
	<p>Блок – терминатор Обозначает вход и выход во внешнюю среду. «Начало», «конец» либо прерывание обработки данных</p>
	<p>Блок – процесс Обозначает одно или несколько действий обработки данных. Выполнение этих действий ведет к изменению данных, их значения или формы хранения</p>
	<p>Блок – предопределённый процесс (функция) Использование ранее созданных и отдельно описанных алгоритмов или программ</p>
	<p>Блок – данные (ввод-вывод) Используют при вводе или выводе данных. Преобразование данных в форму, подходящую для обработки (ввод) или отображения результатов обработки (вывод)</p>
	<p>Блок – решение (условие), или переключатель Выбор направления выполнения алгоритма. Блок содержит некоторое условие, имеет один вход и несколько альтернативных выходов. По какой ветви будет произведена дальнейшая работа – определяется значением этого условия. Значения могут быть подписаны рядом с соответствующими линиями</p>

Форма блока	Назначение блока
	<p>Блок – подготовка (цикл с параметром) Может содержать установку переключателя, модификацию индекса</p>
<p>Начало цикла</p>  <p>Конец цикла</p> 	<p>Блок – граница цикла Состоит из двух частей, отображающих начало и конец цикла, имеющих один и тот же идентификатор. Условия инициализации, приращения, завершения цикла помещают в начале или конце цикла в зависимости от того, с предусловием цикл или постусловием</p>
	<p>Блок – соединитель Используют при необходимости обрыва линии и продолжении её в другом месте. Соответствующие символы-соединители должны содержать одно и то же уникальное обозначение</p>
	<p>Комментарий Используют для описания и/или пояснения каких-то действий</p>
<p>...</p>	<p>Пропуск Используют для отображения пропущенного символа или группы символов</p>

Структурные схемы основных алгоритмических конструкций

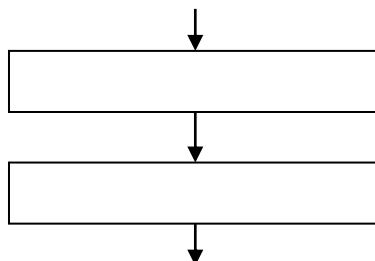
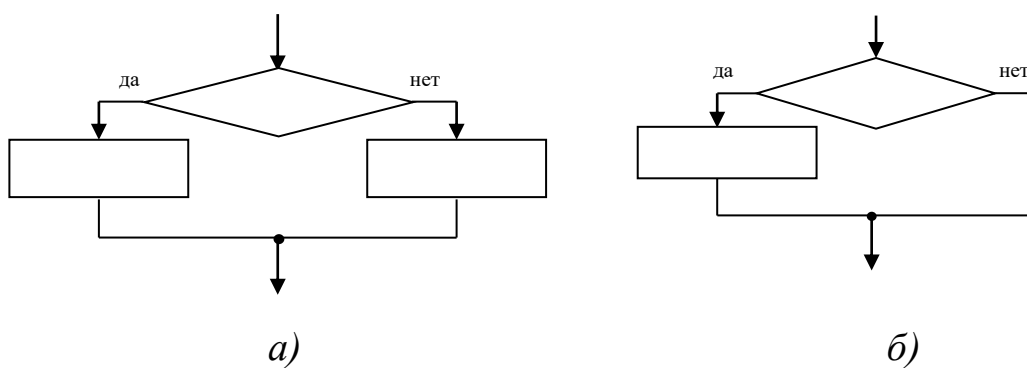


Рис. П4.1. Схематичное представление алгоритмической структуры «следование»



*Рис. П4.2. Алгоритмическая структура «ветвление»:
а – полное ветвление; б – неполное ветвление*



*Рис. П4.3. Алгоритмическая структура «цикл»:
а – цикл с предусловием; б – цикл с постусловием*

Приложение 5

Базовые типы данных C++

В таблице представлены основные типы данных языка C++. В первом столбце указано зарезервированное слово, определяющее тип данных, во втором – количество байт, отводимое под переменную с соответствующим типом данных. В третьем столбце приведён диапазон допустимых значений.

Тип	Количество байт	Диапазон принимаемых значений
Целочисленный (логический) тип данных		
bool	1	0 / 255
Целочисленный (символьный) тип данных		
char	1	0 / 255
Целочисленные типы данных		
short int	2	-32 768 / 32 767
unsigned short int	2	0 / 65 535
int	4	-2 147 483 648 / 2 147 483 647
unsigned int	4	0 / 4 294 967 295
long int	4	-2 147 483 648 / 2 147 483 647
unsigned long int	4	0 / 4 294 967 295
Типы данных с плавающей точкой		
float	4	-2 147 483 648.0 / 2 147 483 647.0
long float	8	-9 223 372 036 854 775 808 .0 / 9 223 372 036 854 775 807.0
double	8	-9 223 372 036 854 775 808 .0 / 9 223 372 036 854 775 807.0

Учебное электронное издание

ШИШКИНА Мария Викторовна

УЧЕБНАЯ ПРАКТИКА

Практикум

Редактор Е. А. Лебедева

Технический редактор Ш. Ш. Амирсейидов

Компьютерная верстка Л. В. Макаровой, А. Н. Герасина

Корректор О. В. Балашова

Выпускающий редактор А. А. Амирсейидова

Системные требования: Intel от 1,3 ГГц; Windows XP/7/8/10; Adobe Reader;
дисковод CD-ROM.

Тираж 9 экз.

Владимирский государственный университет
имени Александра Григорьевича и Николая Григорьевича Столетовых
Изд-во ВлГУ
rio.vlgu@yandex.ru

Институт информационных технологий и электроники
кафедра физики и прикладной математики
familyshishka@mail.ru