Министерство науки и высшего образования Российской Федерации
ФГБОУ ВО «Удмуртский государственный университет»
Институт математики, информационных технологий и физики
Кафедра теоретических основ информатики

А.Е. Анисимов

Практикум
по программированию на C++
Часть 1

Ижевск
2022

УДК 004.424(075.8)
ББК 32.973.22я73-5
А674

Рекомендовано к изданию Учебно-методическим советом УдГУ

Рецензент: канд. физ.-мат. наук, доцент М. А. Клочков

Анисимов А. Е.

А674 Практикум по программированию на C++: ч. 1: учеб. пособие – Ижевск – Удмуртский университет, 2022.– 119 с.

ISBN 978-5-4312- 1018-1

Предлагаемый практикум предназначен для использования в рамках дисциплин начального обучения программированию образовательных программ высшего образования по направлениям подготовки бакалавриата «Прикладная информатика», «Информационные системы», «Фундаментальная информатика и информационные технологии» и других.

Освоение практикума предполагает выполнение обучающимся ряда лабораторных работ для формирования начальных практических навыков программирования на языке C++ и закрепления теоретического материала дисциплин.

УДК 004.424(075.8)
ББК 32.973.22я73-5

ISBN 978-5-4312- 1018-1

Содержание

Введение....4
Лабораторная работа № 1. Типы, выражения, ввод и вывод 5
Лабораторная работа № 2. Ветвление....12
Лабораторная работа № 3. Цикл for 22
Лабораторная работа № 4. Циклы с условием 32
Лабораторная работа № 5. Функции 43
Лабораторная работа № 6. Одномерные массивы 54
Лабораторная работа № 7. Матрицы 70
Лабораторная работа № 8. Строки....84
Лабораторная работа № 9. Рекурсия....95
Приложение: Требования к оформлению программного кода ...107
Список литературы 119

Введение

Данный практикум предназначен для использования в рамках дисциплин образовательных программ высшего образования (бакалавриат) направлений "Прикладная информатика", "Информационные системы", "Фундаментальная информатика и информационные технологии", таких как «Информатика и программирование», «Практикум по программированию», «Технологии программирования» и ряда других.

Результатом прохождения практикума предполагается формирование у обучающихся ряда профессиональных компетенций, в том числе способность применять в профессиональной деятельности языки и технологии программирования, разработку алгоритмических и программных решений профессиональных задач. Практикум предназначен для закрепления студентами начальных сведений и получения первых навыков программирования, освоения базовых конструкций языков программирования C и C++. Так как акцента на различиях между языком C++ и его подъязыком C не делается, то в тексте в большинстве случаев упоминается язык программирования C++.

Практикум состоит из серии лабораторных работ. В начале каждой из них даются примеры решения задач по рассматриваемой теме. Предполагается, что необходимый теоретический материал студент самостоятельно освоил на лекциях или с использованием других источников. Практическое задание состоит в решении поставленной задачи на составление компьютерной программы на языке программирования C++. Каждая лабораторная работа содержит по 32 варианта задач. Студенту необходимо представить преподавателю полученное решение и защитить его. Как правило, при сдаче решения необходимо продемонстрировать успешность прохождения программой ряда тестов, пояснить особенности применяемых программистских решений и ответить на вопросы преподавателя.

Основные требования к оформлению программного кода приведены в Приложении.

Особенностью практикума является в большей степени самостоятельная деятельность студента при наличии необходимого контроля со стороны преподавателя и дифференцированный подход к распределению задач.

Лабораторная работа № 1. Типы, выражения, ввод и вывод

Цель работы

Познакомиться с базовыми (скалярными) типами данных языка программирования C++, построением простых выражений, операторами присваивания, консольным потоковым вводом и выводом.

Необходимый теоретический материал

Для решения задач лабораторной работы № 1 необходимо изучить следующий теоретический материал:

- линейные алгоритмы;
- структура простой программы, функция main();
- базовые типы данных, операции с ними;
- определение и инициализация переменных;
- потоковый ввод (std::cin) и вывод (std::cout);
- присваивание.

Примеры

Пример 1.1.

Написать и отладить программу, решающую следующую задачу: *Даны три положительных вещественных числа a, b и c. Считая, что a, b и c означают длины сторон треугольника, найти и вывести его периметр и площадь.*

```
/*
Пример 1.1.
Даны три положительных вещественных числа a, b и c. Считая,
что a, b и c означают длины сторон треугольника,
найти и вывести его периметр и площадь.
*/

#include <iostream>
#include <iomanip>
#include <math.h>

int main()
{
   // Ввод данных
```

```
    double a, b, c;
    std::cout << "Input a, b, c" << std::endl;
    std::cin >> a >> b >> c;

    // Вычисление
    double p, s;
    p = (a + b + c) / 2;
    s = sqrt(p * (p - a) * (p - b) * (p - c));

    // Вывод ответа
    std::cout << "Perimeter = " << p * 2 << std::endl;
    std::cout << "Area = " << s << std::endl;

    system("pause");
    return 0;
}
```

Изображение консольного окна программы из примера 1.1.

```
C:\Example 1.1.exe
Input a, b, c
3  4.5  5
Perimeter = 12.5
Area = 6.66585
Для продолжения нажмите любую клавишу . . .
```

Таблица тестов для программы из примера 1.1

№ теста	Вход			Выход
	a	b	c	
1.	3	4.5	5	Perimeter = 12.5 Area = 6.66585
2.	10	6	8	Perimeter = 24 Area = 24
3.	1.5	1.5	2.5	Perimeter = 5.5 Area = 1.03645
4.	1	3	1	Perimeter = 5 Area = -nan

Вопрос: почему в последнем тесте площадь не была корректно вычислена?

Пример 1.2.

Написать и отладить программу, решающую следующую задачу: *Даны два натуральных числа n и m (m не превосходит длины числа n). Указать m-ую справа цифру числа n и m-ую слева цифру этого числа.*

```
/*
Пример 1.2.
Даны два натуральных числа n и m (m не превосходит длины числа
n). Указать m-ую слева цифру числа n и m-ую справа цифру этого
числа.
*/

#include <iostream>
#include <math.h>
#include <locale.h>
using namespace std;

int main()
{
   setlocale(LC_ALL, "Russian");

   // Ввод данных
   int n, m;
   cout << "Введите n, m" << endl;
   cin >> n >> m;

   // Вычисление
   int digitLeft = n / int(pow(10, int(log10(n)) - m + 1)) % 10;
   int digitRight = n / int(pow(10, m - 1)) % 10;

   // Вывод ответа
   cout << "В числе " << n << endl;
   cout << m << "-ая слева  цифра равна " << digitLeft << endl;
   cout << m << "-ая справа цифра равна " << digitRight << endl;

   system("pause");
   return 0;
}
```

Изображение консольного окна программы из примера 1.2.

```
C:\Example 1.2.exe
Введите n, m
```

```
123456  4
В числе 123456
4-ая слева цифра равна 4
4-ая справа цифра равна 3
Для продолжения нажмите любую клавишу . . .
```

Таблица тестов для программы из примера 1.2

№ теста	Вход		Выход
	n	m	
1.	123456	4	4-ая слева цифра равна 4 4-ая справа цифра равна 3
2.	10000	1	1-ая слева цифра равна 1 1-ая справа цифра равна 0
3.	5	1	1-ая слева цифра равна 5 1-ая справа цифра равна 5

Варианты задач лабораторной работы № 1

Указания. В программных решениях задач не следует применять ветвления, циклы или рекурсию. Необходимо ограничиться присваиванием, выражениями и функциями.

Входные данные задавать с клавиатуры в консольном окне, результат выдавать также в окно. Для отладки программы заранее составьте систему тестов.

Вариант	Текст задачи
1.1	Даны вещественные положительные числа a и b. Считая эти числа длинами сторон прямоугольника, найти его периметр, площадь и длину диагонали.
1.2	Даны вещественные положительные числа a, b. Найти гипотенузу, площадь и периметр прямоугольного треугольника с катетами a и b.
1.3	Даны вещественные положительные числа a, b и c. Считая эти числа длинами ребер прямоугольного параллелепипеда, найти его объём, площадь поверхности и длины диагоналей граней.
1.4	Дано вещественное положительное число V. Считая это число объёмом куба, найти его ребро и площадь поверхности.

1.5	Даны вещественные числа r1 и r2. Найти площадь кольца, образованного двумя окружностями с радиусами r1 и r2.
1.6	Дано натуральное трехзначное число N. Найти сумму цифр этого числа. Указание: операция целочисленного деления div на 10 дает число без последней цифры, а остаток от деления mod на 10 - последнюю цифру числа.
1.7	Даны вещественные числа a, b, c. Найти площадь треугольника со сторонами a и b и углу между ними с градусов. Указание: для перевода градусов в радианы можно использовать формулу $Радианы = Градусы \cdot \frac{\pi}{180}$
1.8	Даны вещественные числа a, d, n. Найти n-ый член и сумму n первых членов арифметической прогрессии с первым членом равным a и разностью d. Указание: $a_n = a_1 + (n-1)d$, $s_n = \frac{a_1+a_n}{2}n$.
1.9	Даны вещественные числа x1, y1, x2, y2, x3, y3. Найти периметр и площадь треугольника, вершины которого имеют координаты (x1, y1), (x2, y2) и (x3, y3). Указание: расстояние между точками (x1, y1) и (x2, y2) можно найти по формуле $L = \sqrt{(x2-x1)^2 + (y2-y1)^2}$
1.10	Даны целые неотрицательные числа h, m и s. Найти углы поворота в градусах часовой и минутной стрелок часов, когда они показывают время h часов m минут и s секунд
1.11	Дано целое неотрицательное число N. Считая, что N обозначает длину, выраженную в верстах, выразить эту длину в километрах, метрах и сантиметрах. В одной версте 1066 м 80 см. Пример: N = 12 верст = 12 км 801 м 60 см.
1.12	Дано целое неотрицательное число N. Считая, что N - это время в секундах, выразить это время в сутках, часах, минутах и секундах. Пример: N = 100000 секунд = 1 сутки 3 часа 46 минут 40 секунд.
1.13	Дано целое неотрицательное число B. Считая, что B – это размер файла в байтах, выразить этот размер в целых Мбайтах, Кбайтах и байтах. Считаем, что 1 Кбайт = 1024 байт, 1 Мбайт = 1024 Кбайт. Пример: N = 2000000 байт = 1 Мбайт 929 Кбайт 128 байт.
1.14	Даны значения трёх целочисленных переменных a, b, c. Переместить их значения так, чтобы переменная a получила бы исходное значение b, b получила бы значения c , а переменная c – значение a.

1.15	Дано натуральное число N, не превосходящее 9999. Получить число M перестановкой цифр числа N в обратном порядке. Циклы не использовать.
1.16	Даны вещественные координаты вершин треугольника координаты (x1, y1), (x2, y2) и (x3, y3). Найти координаты середин его сторон.
1.17	Даны вещественные координаты вершин треугольника (x1, y1), (x2, y2) и (x3, y3). Найти величины углов этого треугольника в градусах.
1.18	Дано натуральное число N. Найдите количество цифр этого числа. Циклами, ветвлением не пользоваться.
1.19	Даны натуральные числа N, m. Найдите количество цифр в записи этого числа в m-ичной системе счисления. Циклами, ветвлением не пользоваться.
1.20	Робин Бобин Бабарек кушает людей. В первый день он съел одного человека, а в каждый последующий день – вдвое больше, чем в предыдущий. Сколько всего человек съест Р. Б. Бабарек за N дней? Циклами, ветвлением не пользоваться.
1.21	Известные переводные курсы двух валют: один доллар равен S рублей, один евро равен E рублей. Рассчитать котировки доллара в евро и евро в доллар через рубль. Все курсы имеют до двух знаков после запятой.
1.22	Даны два натуральных числа N и D (N < D). Найти десятичную дробь, равную дроби $\frac{N}{D}$ с точностью до двух знаков после запятой. Пример: N = 3, D = 7. Ответ: 0.43.
1.23	Даны целые положительные числа n и m, при этом m не превосходит количества цифр числа n. Округлить значение n до m-ой (считая с конца) цифры. Например, n=123456, m = 3. Ответ n = 123500. Циклами не пользоваться.
1.24	Даны положительное вещественное число n, натуральное число m, при этом m <= 8. Округлить значение n до m-ой цифры после запятой. Например, n=123.4567, m = 3. Ответ n = 123.457. Циклами не пользоваться.
1.25	Родители купили своему ребенку набор свечей для торта, всего в наборе N штук свечей. На первый день рождения была потрачена одна свеча, на второй – две свечи, на третий – три и так далее. На сколько дней рождений хватит свечей из набора? Циклами не пользоваться.
1.26	Ребенок-вундеркинд учит слова тарабарского языка. В первый день обучения он выучил 1 слово, в каждый последующий день он учит

	вдвое больше слов, чем в день предыдущий. Сколько дней понадобится ребенку выучить весь тарабарский язык, если всего в нём N слов? Циклами не пользоваться.
1.27	Дано положительное вещественное число R, большее 1. Найдите количество значащих цифр в дробной части этого числа. Например, для числа R=1.203 количество значащих цифр дробной части равно 3. Строками не пользоваться.
1.28	Даны значения двух вещественных переменных a, b. Переместить их значения так, чтобы переменная a получила бы исходное значение b, при этом b получила бы значение a. Дополнительными переменными не пользоваться.
1.29	Даны значения трёх целочисленных переменных a, b, c. Переместить их значения так, чтобы переменная a получила бы исходное значение b, b получила бы значения c , a переменная c - значение a. Дополнительными переменными не пользоваться.
1.30	Даны значения двух логических переменных A и B. Обменять их значениями, не используя дополнительных переменных и ветвления.
1.31	Даны два вещественных значения a и b. Найти среди них максимальное без использования ветвления и условной операции.
1.32	Даны два вещественных значения a и b. Найти среди них минимальное без использования ветвления и условной операции.

Лабораторная работа № 2. Ветвление

Цель

Ознакомиться с программированием разветвляющихся алгоритмов, операторами условного перехода и выбора.

Необходимый теоретический материал

Для решения задач лабораторной работы № 2 необходимо изучить следующий теоретический материал:

- алгоритмы с ветвлением;
- условный оператор `if-else`;
- оператор выбора `switch-case`.

Примеры

Пример 2.1.

Написать и отладить программу, решающую следующую задачу: *Даны вещественные числа a, b, c, причем a не равно 0. Решить квадратное уравнение ax^2 + bx + c = 0.*

```
/*
Пример 2.1.
Даны вещественные числа a, b, c, причем a не равно 0.
Решить квадратное уравнение ax^2 + bx + c = 0.
*/

#include <iostream>
#include <iomanip>
#include <math.h>
#include <locale.h>
using namespace std;

int main()
{
   setlocale(LC_ALL, "Russian");

   // Ввод данных
   double a, b, c;
   cout << "Введите коэффициенты a, b, c квадратного уравнения"
      << endl;
   cin >> a >> b >> c;
```

```
   if (a == 0)
   {
      cout << "Некорректные данные!" << endl;
      system("pause");
      return 0;
   }

   // Вычисление
   cout << fixed;
   double d = b * b - 4 * a * c;
   double x1, x2;
   if (d > 0)
   {
      x1 = (-b - sqrt(d)) / (2 * a);
      x2 = (-b + sqrt(d)) / (2 * a);
      cout << "Корни уравнения" << endl
         << "x1 = " << setprecision(2) << x1 << endl
         << "x2 = " << setprecision(2) << x2 << endl;
   }
   else
      if (d == 0)
      {
         x1 = -b / (2 * a);
         cout << "Корень уравнения" << endl
            << "x1 = " << setprecision(2) << x1 << endl;
      }
      else
         cout << "Уравнение не имеет корней" << endl;

   system("pause");
   return 0;
}
```

Изображение консольного окна программы из примера 2.1.

```
C:\Example 2.1.exe
Введите коэффициенты a, b, c квадратного уравнения
2  1  -5
Корни уравнения
x1 = -1.85
x2 = 1.35
Для продолжения нажмите любую клавишу . . .
```

Таблица тестов для программы из примера 2.1

№ теста	Вход			Выход
	a	b	c	
1.	2	1	-5	x1 = -1.85 x2 = 1.35
2.	4	4	1	x1 = -0.50
3.	2	3	1.5	Уравнение не имеет корней

Пример 2.2.

Написать и отладить программу, решающую следующую задачу: *Даны два вещественных числа x, y. Определить, какой области координатной плоскости принадлежит точка с координатами (x; y)?*

```
/*
Пример 2.2.
Даны два вещественных числа x, y. Определить, какой области ко-
ординатной плоскости принадлежит точка с координатами (x; y)?
*/

#include <iostream>
#include <locale.h>
using namespace std;

int main()
{
   setlocale(LC_ALL, "Russian");

   // Ввод данных
   double x, y;
   cout << "Введите координаты точки x, y" << endl;
   cin >> x >> y;

   // Определение области плоскости
   cout << "Точка с координатами (" << x << ";" << y << ")"
      << " находится ";
   if (x > 0)
      if (y > 0)
         cout << "в 1 четверти." << endl;
      else
         if (y == 0)
            cout << "на оси OX." << endl;
         else
```

```
            cout << "в 4 четверти." << endl;
   else
      if (x == 0)
         if (y == 0)
            cout << "в начале координат." << endl;
         else
            cout << "на оси OY." << endl;
      else
         if (y > 0)
            cout << "в 2 четверти." << endl;
         else
            if (y == 0)
               cout << "на оси OX." << endl;
            else
               cout << "в 3 четверти." << endl;

   system("pause");
   return 0;
}
```

Изображение консольного окна программы из примера 2.2.

```
C:\Example 2.2.exe
Введите координаты точки x, y
-2.5  -3
Точка с координатами (-2.5; -3) находится в 3 четверти.
Для продолжения нажмите любую клавишу . . .
```

Таблица тестов для программы из примера 2.2

№ теста	Вход		Выход
	x	y	
1.	-2.5	-3	Точка с координатами (-2.5; -3) находится в 3 четверти
2.	4	1	Точка с координатами (4; 1) находится в 1 четверти
3.	-10	0	Точка с координатами (-10; 0) находится на оси OX
4	0.0	0.0	Точка с координатами (0; 0) находится в начале координат

Пример 2.3.

Написать и отладить программу, решающую следующую задачу: *Вводится арифметическое выражение в формате "число операция число", где "число" – любое натуральное число, а "операция" - один из знаков +, -, *, /, %. Элементы выражения могут отделяться друг от друга пробелами. Найти значение выражения. Пример: вход 12345 + 999; выход 13344.*

```
/*
Пример 2.3.
Вводится арифметическое выражение в формате "число операция чис-
ло", где "число" - любое целое число, а "операция" - один из
знаков +, -, *, /, %. Элементы выражения могут отделяться друг
от друга пробелами. Найти значение выражения. Пример: вход 12345
+ 999; выход 13344.
*/

#include <iostream>
#include <locale.h>
using namespace std;

int main()
{
   setlocale(LC_ALL, "Russian");

   // Ввод данных
   int a, b;
   char operation = ' ';

   cout << "Введите выражение в формате"
      <<"\"число операция число\"" << endl;
   cin >> a >> operation >> b;

   // Вычисление
   int result = 0;
   bool success = true;

   switch (operation)
   {
   case '+':
      result = a + b;
      break;
   case '-':
      result = a - b;
```

```
      break;
   case '*':
      result = a * b;
      break;
   case '/':
      result = a / b;
      break;
   case '%':
      result = a % b;
      break;
   default:
      success = false;
   }

   // Вывод результата
   if (success)
      cout << "Результат = " << result << endl;
   else
      cout << "Ошибка ввода" << endl;

   system("pause");
   return 0;
}
```

Изображение консольного окна программы из примера 2.3.

```
C:\Example 2.3.exe
Введите выражение в формате "число операция число"
12345 + 999
Результат = 13344
Для продолжения нажмите любую клавишу . . .
```

Таблица тестов для программы из примера 2.3

№ теста	Вход	Выход
1.	12345 + 999	Результат = 13344
2.	791 % 12	Результат = 11
3.	1000 * -1000	Результат = -1000000
4.	1.5 + 2.5	Ошибка ввода

Варианты задач лабораторной работы № 2

Указания. В программных решениях задач использовать ветвление (условный оператор, оператор выбора), не использовать циклы. Постараться оптимизировать применяемый алгоритм так, чтобы не вычислялись повторно одни и те же логические выражения.

Входные данные задавать с клавиатуры в консольном окне, результат выдавать также в окно. Для отладки программы заранее составьте систему тестов.

№	Текст задачи
2.1	Даны три вещественных числа. Найти из них максимальное и минимальное значения.
2.2	Даны три вещественных числа. Найти второе по величине значение (если такое существует). Пример: вход 3.5 2.0 2. выход: 2
2.3	Даны вещественные числа a, b, c. Определить, существует ли треугольник с такими сторонами, и найти его площадь.
2.4	Даны три вещественных числа. Вывести число 3, если они равны друг другу, вывести число 2, если равны друг другу только два числа, и 1 – во всех остальных случаях.
2.5	Даны четыре целых числа. Указать из них наибольшее количество равных друг другу чисел. Пример: вход 3 1 2 1. выход: 2
2.6	Даны целые числа a, b, c, X, Y, Z. В каждый новогодний подарок нужно положить a конфет, b яблок и c печенек . Сколько подарков можно скомплектовать из X конфет, Y яблок и Z печенек?
2.7	Даны вещественные числа a, b, c, d. Решить уравнение $\frac{ax+b}{cx+d} = 0$
2.8	Даны положительные вещественные числа x, y, a, b, c. Определить, пройдёт ли кирпич с ребрами a, b и c в прямоугольное отверстие со сторонами x и y?
2.9	Даны значения трёх вещественных переменных. Обменять местами значения переменных с максимальным и минимальным значениями, вывести новые значения этих переменных.
2.10	Дано натуральное N. Выяснить, является ли високосным год N?
2.11	Дано вещественное положительное значение R, вещественные x, y. На координатной плоскости расположены четыре концентрические окружности с радиусами R, 2R, 3R и 4R, и центрами в начале координат. Окружности образуют мишень. Выстрел произведён в точку с координатами (x, y). Указать количество очков, которые получает

	стрелок за этот выстрел (центральный круг – 4 очка, второй – 3 очка и т.д.).
2.12	Координаты клеток шахматной доски выражаются двумя числами от 1 до 8. Даны натуральные числа a, b, c, d. Проверить, бьёт ли конь, находящийся на клетке (a, b) фигуру, находящуюся на клетке (c, d)?
2.13	Координаты клеток шахматной доски выражаются двумя числами от 1 до 8. Даны натуральные числа a, b, c, d. Проверить, бьёт ли ферзь, находящийся на клетке (a, b) фигуру, находящуюся на клетке (c, d)?
2.14	Даны четыре целых числа, одно из них не равно трём другим, равным между собой. Указать порядковый номер числа, отличного от остальных.
2.15	Даны целочисленные координаты трех вершин прямоугольника, стороны которого параллельны осям координат. Найти координаты его четвертой вершины.
2.16	Студент добирался домой в другой город: в начале пути на автобусе t_1 часов со скоростью v_1 км/ч, затем – на велосипеде t_2 часов со скоростью v_2 км/ч, и уже затем пешком t_3 часов со скоростью v_3 км/ч. На середине пути он сделал остановку в придорожном кафе. На чём он тогда передвигался?
2.17	Даны вещественные положительные числа a, b, c, d, e и f. Считаем, что пары чисел a и b, c и d, e и f обозначают размеры первого, второго и третьего прямоугольников соответственно. Выяснить, можно ли внутри первого прямоугольника разместить два других, не накладывая один на другой?
2.18	Даны натуральные числа m, n, k. В зрительном кинозале находится m рядов по n кресел в каждом. Зрителям продают билеты строго последовательно: вначале первый ряд от первого до последнего места, потом также второй, третий ряды и. т. д. Всего продано k билетов. Определить, на каком по номеру ряду закончатся проданные места и сколько мест в нём осталось непроданных (если такие будут)?
2.19	Дано натуральное число, не превосходящее 999. Вывести на экран это число в виде числительного русского языка. Например: N = 619, выход – «шестьсот девятнадцать».
2.20	Дано вещественное число, содержащее не более трёх знаков после запятой. Вывести на экран дробную часть числа в виде числительного русского языка. Например: N = 1.277, выход – «двести семьдесят семь тысячных».

2.21	Прямоугольник, стороны которого параллельны осям координат, задается координатами двух противолежащих своих вершин. Даны вещественные числа – x_1, y_1, x_2, y_2, x_3, y_3, x_4, y_4. Задано два прямоугольника парами точек (x_1, y_1) и (x_2, y_2) – первый прямоугольник и (x_3, y_3) и (x_4, y_4) –второй прямоугольник. Найти площадь пересечения прямоугольников.
2.22	Даны вещественные числа k, b и R (R > 0). Найти количество точек пересечения прямой, задаваемой уравнением y=kx+b и окружностью с центром в начале координат и радиусом R.
2.23	Даны вещественные числа a, b, c $(a \neq 0)$. Решить уравнение $ax^4 + bx^2 + c = 0$.
2.24	Даны четыре точки на числовой прямой A, B, C и D, причём A ≤ B, C ≤ D. Найти длину пересечения отрезков AB и CD.
2.25	Прямоугольник, стороны которого параллельны осям координат, задается координатами двух противолежащих своих вершин – (x_1, y_1) и (x_2, y_2). Найти площади частей этого прямоугольника, лежащих в каждой из четырёх четвертей координатной плоскости.
2.26	Даны вещественные $a_1, b_1, c_1, a_2, b_2, c_2$. Решить систему уравнений $\begin{cases} a_1 x + b_1 y = c_1 \\ a_2 x + b_2 y = c_2 \end{cases}$.
2.27	Даны целые положительные A, B и C. Какое максимальное число квадратов со стороной C можно разместить в прямоугольнике со сторонами A и B без наложений? Указать площадь занятой квадратами и незанятой частей прямоугольника.
2.28	Даны целые числа a, X, Y, Z. В каждый новогодний подарок нужно положить a конфет; также в подарке должно быть три фрукта: 1 яблоко и 2 груши либо 2 яблока и 1 груша. Сколько подарков можно скомплектовать из X конфет, Y яблок и Z груш?
2.29	Даны натуральные числа N, R, вещественные числа x_1, y_1. Мишень на координатной плоскости представляет собой концентрические круги с центром в начале координат; первый круг имеет радиус R, радиус второго круга равен 2R, третий круг радиуса 3R и так далее. Всего кругов в мишени N штук, за попадание в первый круг назначается N очков, во второй круг - N-1 очков и так далее. За непопадание в мишень даётся ноль очков. Стрелок сделал выстрел в точку с координатами (x_1, y_1). Указать, сколько очков он набрал. Циклами не пользоваться.
2.30	Восточный календарь поделён на двенадцать лет, каждый из которых назван одним из животных: крыса, бык, тигр, кролик, дракон,

	змея, лошадь, коза (овца), обезьяна, петух, собака, свинья. Через каждые двенадцать лет цикл повторяется, но меняется одна из пяти стихий и, соответственно, цвет животных: дерево (зелёный), огонь (красный), земля (жёлтый), металл (белый) и вода (черный). Таким образом, полный цикл восточного календаря составляет 60 лет. По номеру года определить животного и его стихию (цвет), если 1984 был годом зелёной крысы.
2.31	Даны вещественные числа a, b, c. Решить уравнение $ax^4 + bx^2 + c = 0$.
2.32	Из всех натуральных чисел, которые можно представить в виде $2^m 3^n$ (где m, n - натуральные) построили возрастающую последовательность (её начало – 6, 12, 18, 24, 36, ...). Дано число M, принадлежащее этой последовательности. Найти следующее число последовательности.

Лабораторная работа № 3. Цикл for

Цель работы

Изучить назначение и алгоритм цикла с заранее известным числом повторений (цикл со счетчиком). Научиться использовать цикл for.

Необходимый теоретический материал

Для решения задач лабораторной работы № 3 необходимо изучить следующий теоретический материал:

- циклические алгоритмы;
- цикл for.

Примеры

Пример 3.1.

Написать и отладить программу, решающую следующую задачу: *Дано целое положительное число n. Найти сумму чисел*

$$\sum_{i=1}^{n} \frac{1+2+\cdots+i}{2^i}$$

Заметим, что указанную сумму можно расписать по слагаемым:

$$\frac{1}{2^1}+\frac{1+2}{2^2}+\frac{1+2+3}{2^3}+\cdots+\frac{1+2+\cdots+n}{2^n}$$

Также заметим, что вычисление каждого слагаемого (кроме первого) можно выполнить на основе предыдущего слагаемого по рекуррентным формулам.

```
/*
Пример 3.1.
Дано целое положительное число n. Найти сумму чисел
n     1+2+...+i
СУММА ---------
i=1       2^i
*/

#include <iostream>
#include <locale.h>
using namespace std;

int main()
{
```

```
   setlocale(LC_ALL, "Russian");

   // Ввод данных
   int n = 1;
   cout << "Введите n" << endl;
   cin >> n;

   // Вычисление
   int s1 = 0;    // Сумма в числителе 1+2+...+i
   int p1 = 1;    // Степень в знаменателе 2^i
   double s = 0.0;  // Общая сумма
   for (int i = 1; i <= n; i++)
   {
      s1 += i;
      p1 *= 2;
      s += (double) s1 / p1;
   }

   // Вывод ответа
   cout << "Искомая сумма = " << s << endl;

   system("pause");
   return 0;
}
```

Изображение консольного окна программы из примера 3.1.

```
C:\Example 3.1.exe
Введите n
5
Искомая сумма = 3.09375
Для продолжения нажмите любую клавишу . . .
```

Таблица тестов для программы из примера 3.1

№ теста	Вход	Выход
	n	
1.	1	Искомая сумма = 0.5
2.	3	Искомая сумма = 2
3.	5	Искомая сумма = 3.09375
4.	10	Искомая сумма = 3.92285

Пример 3.2.

Написать и отладить программу, решающую следующую задачу:
Последовательность чисел Фибоначчи определяется следующим образом: $a_1 = 1;\ a_2 = 1;\ a_i = a_{i-1} + a_{i-2}$, *для* $i = 3, 4, 5, \ldots$
Дано целое положительное число n. *Найти* a_n.

```
/*
Пример 3.2.
Последовательность чисел Фибоначчи определяется следующим обра-
зом: a(1) = 1; a(2) = 1; a(i) = a(i - 1) + a(i - 2) для i = 3,
4, 5, ... Дано целое положительное число n. Найти a(n).
*/

#include <iostream>
#include <locale.h>
using namespace std;

int main()
{
   setlocale(LC_ALL, "Russian");

   // Ввод данных
   int n = 1;
   cout << "Введите n" << endl;
   cin >> n;

   // Вычисление
   int a = 1;
   int b = 1;
   int c;
   for (int i = 3; i <= n; i++)
   {
      c = a + b;
      a = b;
      b = c;
   }

   // Вывод ответа
   cout << "Число Фибоначчи a(" << n << ") = " << b << endl;

   system("pause");
   return 0;
}
```

Изображение консольного окна программы из примера 3.2.

```
C:\Example 3.2.exe
Введите n
10
Число Фибоначчи a(10) = 55
Для продолжения нажмите любую клавишу . . .
```

Таблица тестов для программы из примера 3.2

№ теста	Вход	Выход
	n	
1.	1	Число Фибоначчи a(1) = 1
2.	6	Число Фибоначчи a(6) = 8
3.	10	Число Фибоначчи a(10) = 55
4.	40	Число Фибоначчи a(40) = 102334155

Пример 3.3.

Написать и отладить программу, решающую следующую задачу: *Даны натуральные числа n, x, y (x и y не превосходят n). На шахматной доске размера n x n клеток расположен ферзь в клетке с координатами (x, y). Вывести изображение доски на консольный экран, при этом в клетке с координатами (x, y) вывести букву Ф, в клетках, которые находятся под боем ферзя, вывести букву x. В остальных клетках – точки.*

```
/*
Пример 3.3.
Даны натуральные числа n, x, y (x и y не превосходят n).
На шахматной доске размера n x n клеток расположен ферзь в клет-
ке с координатами (x, y). Вывести изображение доски на консоль-
ный экран, при этом  в клетке с координатами (x, y) вывести бук-
ву Ф, в клетках, которые находятся под боем ферзя, вывести букву
x. В остальных клетках - точки.
*/

#include <iostream>
#include <iomanip>
#include <locale.h>
using namespace std;
```

```
int main()
{
   setlocale(LC_ALL, "Russian");

   // Ввод данных
   int n, x, y;
   cout << "Введите n, x, y" << endl;
   cin >> n >> x >> y;

   // Вывод шахматной доски
   for (int i = 1; i <= n; i++)
   {
      for (int j = 1; j <= n; j++)
      {
         if (i == x)
            if (j == y)   // Позиция ферзя
               cout << setw(2) << "Ф";
            else        // Горизонталь под боем
               cout << setw(2) << "x";
         else
            if (j == y)   // Вертикаль под боем
               cout << setw(2) << "x";
            else
               if ((x - y) == (i - j) || (x + y) == (i + j))
               // Диагонали под боем
                  cout << setw(2) << "x";
               else     // Клетка не под боем
                  cout << setw(2) << ".";
      }
      cout << endl;
   }

   system("pause");
   return 0;
}
```

Изображение консольного окна программы из примера 3.3.

```
C:\Example 3.3.exe
Введите n, x, y
8 3 5
 . . x . x . x .
 . . . x x x . .
 x x x x Ф x x x
 . . . x x x . .
 . . x . x . x .
 . x . . x . . x
 x . . . x . . .
 . . . . x . . .
Для продолжения нажмите любую клавишу . . .
```

Таблица тестов для программы из примера 3.3

№ теста	Вход			Выход
	n	x	y	
1.	8	3	5	. . x . x . x . . . . x x x . . x x x x Ф x x x . . . x x x . . . . x . x . x . . x . . x . . x x . . . x . . . . x . . .
2.	5	5	5	x . . . x . x . . x . . x . x . . . x x x x x x Ф

Варианты задач лабораторной работы № 3

Указания. В программных решениях задач можно использовать циклы for. Не допускается использовать массивы и строки.

Входные данные задавать с клавиатуры в консольном окне, результат выдавать также в окно. Для отладки программы заранее составьте систему тестов.

№	Текст задачи
3.1	Дано целое положительное число N. Найти сумму $$\sum_{i=1}^{N} \frac{i}{1+2+\cdots+i}$$
3.2	Дано целое положительное число N. Найти сумму $$\sum_{i=1}^{N} \frac{3^i}{i!}$$
3.3	Дано целое положительное число N. Найти сумму $$\sum_{i=1}^{N} \frac{2^i}{\frac{1}{1}+\frac{1}{2}+\cdots+\frac{1}{i}}$$
3.4	Дано целое положительное число N. Найти сумму $$\sum_{i=1}^{N} \frac{i!}{1+2+\cdots+i}$$
3.5	Дано целое положительное число N. Найти произведение $$\prod_{\mathrm{i}=1}^{\mathrm{N}} \frac{1+2+\cdots+\mathrm{i}}{\mathrm{i}!}$$
3.6	Дано целое положительное число N. Найти произведение $$\prod_{\mathrm{i}=1}^{\mathrm{N}} \left(2^{\mathrm{i}}+\frac{1+2+\cdots+\mathrm{i}}{\mathrm{i}!}\right)$$
3.7	Дано целое положительное число N. Найти $\sqrt{1+\sqrt{1+\sqrt{1+\cdots\sqrt{1}}}}$ (всего N корней)
3.8	Дано целое положительное число n. Найти цепную дробь $$\cfrac{1}{2^{\mathrm{n}}+\cfrac{1}{2^{\mathrm{n}-1}+\cfrac{1}{\vdots}}}$$ $$2^2+\frac{1}{2^1}$$

3.9	Дано целое положительное число n. Найти $\sqrt{\mathrm{n}+\sqrt{(\mathrm{n}-1)+\sqrt{(\mathrm{n}-2)+\cdots\sqrt{1}}}}$
3.10	Дано целое положительное число n. Найти $\sqrt{3\mathrm{n}+\sqrt{3(\mathrm{n}-1)+\sqrt{3(\mathrm{n}-2)+\cdots\sqrt{3}}}}$
3.11	Дано целое положительное число n. Найти $\cfrac{1}{\mathrm{n}+\cfrac{1}{(\mathrm{n}-1)+\cfrac{1}{\vdots}}}$ $2+\cfrac{1}{1}$
3.12	Дано целое положительное число n, вещественное число x. Среди чисел $\sin \mathrm{x}, 2\sin \mathrm{x}^2, 3\sin \mathrm{x}^3, \cdots, n\sin \mathrm{x}^{\mathrm{n}}$ найти наибольшее, наименьшее и указать их номера.
3.13	Дано целое положительное число n, вещественное число x. Найти значение многочлена вида $\mathrm{a_n x^n} + \mathrm{a_{n-1} x^{n-1}} + \cdots + \mathrm{a_1 x^1} + \mathrm{a_0}$, если коэффициенты многочлена вводятся последовательно с клавиатуры в следующем порядке: $\mathrm{a_n}, \mathrm{a_{n-1}}, \ldots, \mathrm{a_1}, \mathrm{a_0}$. Указание: массивы не использовать.
3.14	Дано натуральное число n. Среди натуральных чисел от 1 до n найти число с наибольшим количеством делителей.
3.15	Даны вещественные числа a, b $(a \cdot b \neq 0)$, целое неотрицательное число n. Среди чисел вида $a_0 = b\sin a$, $a_i = b\sin(a \cdot a_{i-1})$, для $i = 1, 2, \ldots, n$ найти наименьшее и наибольшее.
3.16	Счастливые билеты. Даны целые положительные M и N $(100000 \leq M \leq N)$. Найти количество целых шестизначных чисел между M и N, у которых сумма первых трёх цифр равна сумме последних трёх цифр.
3.17	Дано вещественное положительное число R. Найти количество точек координатной плоскости с целочисленными координатами, попадающими в круг радиуса R с центром в начале координат.
3.18	Даны вещественные положительные числа R_1 и R_2. Найти количество точек координатной плоскости с целочисленными координа-

	тами, попадающими в кольцо, образованное окружностями с радиусами R_1 и R_2 с центром в начале координат.
3.19	Дано натуральное число n. Напечатать n-таблицу Пифагора (таблица умножения для чисел от 1 до n).
3.20	Даны натуральные числа N и M $(N \le M \le 1000)$. Найти все простые числа p, удовлетворяющие неравенствам $N \le p \le M$.
3.21	Напечатать таблицу истинности бинарных логических операций «и», «или», «исключающее или».
3.22	Дано целое положительное число n, вещественное число x. Найти значение многочлена вида $a_n x^n + a_{n-1}x^{n-1} + \cdots + a_1 x^1 + a_0$, если коэффициенты многочлена вводятся последовательно с клавиатуры в следующем порядке: $a_n, a_{n-1}, \ldots, a_1, a_0$. Указание: массивы не использовать. Рекомендуется использовать схему Горнера.
3.23	Дано натуральное число $n \ge 2$. Среди всех чисел от 1 до n найти такую пару рядом стоящих чисел, сумма цифр в записи которых максимальна.
3.24	Дано натуральное число n. Среди чисел Фибоначчи $a_1, a_2, a_3, \ldots, a_n$ найти все простые числа. Напоминание: простым называется целое число (большее 1), которое делится только на себя и на 1.
3.25	Даны натуральные числа n, m $(n \ge m \ge 0)$. Найти коэффициент бинома Ньютона $C_n^m = \frac{n!}{m!(n-m)!}$ за наименьшее число операций умножения.
3.26	Напечатать треугольник Паскаля, состоящий из n строк.
3.27	Дано натуральное число $N \le 1000$. Найти все тройки натуральных чисел a, b, c $(a \le b \le c)$, удовлетворяющие условию $a^2 + b^2 + c^2 = N$. Указание: использовать только два цикла.
3.28	Тройки Пифагора. Дано натуральное число $N \le 100$. Найти все тройки натуральных чисел a, b, c, не превосходящих N удовлетворяющие условию $a^2 + b^2 = c^2$. Указание: использовать только два цикла.

3.29	Дано натуральное число N. Напечатать таблицу перевода сантиметров в дюймы и обратно, состоящую из N строк, с точностью двух знаков после запятой. Строки таблицы должны быть упорядочены по возрастанию. Например, для N=5 таблица перевода выглядит так:
	`см  inch`
	`1.00  0.39`
	`2.00  0.79`

	2.54 1.00 3.00 1.18 4.00 1.57
3.30	Даны натуральные числа n, a, b. В некотором университете занятия начинаются в 8:00, продолжительность учебной пары a минут, перерыва между парами – b минут. Вывести расписание звонков, содержащее n учебных пар. Например, для $n = 5, a = 90, b = 10$ расписание будет выглядеть так: № Пара Перерыв 1 08:00 09:30 2 09:40 11:10 3 11:20 12:50 4 13:00 14:30 5 14:40 16:10
3.31	Дано натуральное n $(1 \leq n \leq 7)$. Вывести весь календарь невисокосного года, 1 января которого является -ым по счёту днём недели. Например, первый месяц такого календаря для $n = 5$ будет выглядеть так: Январь 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
3.32	Дано натуральное число n. Вывести все перестановки первых n натуральных чисел. Например, все перестановки для n = 3: 1 2 3 1 3 2 2 1 3 2 3 1 3 1 2 3 2 1

Лабораторная работа № 4. Циклы с условием

Цель работы

Изучить назначение и алгоритм циклов с заранее неизвестным числом повторений (циклы с условием). Научиться использовать циклы while и do-while.

Необходимый теоретический материал

Для решения задач лабораторной работы № 4 необходимо изучить следующий теоретический материал:

- циклические алгоритмы;
- цикл while;
- цикл do-while.

Примеры

Пример 4.1.

Написать и отладить программу, решающую следующую задачу: *Дано целое число n. Вывести все натуральные степени числа 2 (2, 4, 8, 16 и т.д.), не превосходящие n.*

```
/*
Пример 4.1.
Дано целое число n. Вывести все натуральные степени
числа 2 (2, 4, 8, 16 и т.д.), не превосходящие n.
*/

#include <iostream>
#include <iomanip>
#include <locale.h>
using namespace std;

int main()
{
   setlocale(LC_ALL, "Russian");

   // Ввод данных
   cout << "Введите целое число n" << endl;
   int n;
   cin >> n;

   // Нахождение натуральных степеней числа 2,не превосходящих n
```

```
   if (n >= 2)
   {
      cout << "Натуральные степени числа 2, не превосходящие "
           << n << endl;
      int p = 2;  // Первая степень числа 2
      while (p <= n)
      {
         cout << setw(4) << p;
         p *= 2;  // Получение следующей степени числа 2
      }
      cout << endl;
   }
   else
      cout << "Натуральных степеней числа 2, не превосходящих "
           << n << ", нет" << endl;

   system("pause");
   return 0;
}
```

Изображение консольного окна программы из примера 4.1.

```
C:\Example 4.1.exe
Введите целое n
30
Натуральные степени числа 2, не превосходящие 30
   2   4   8  16
Для продолжения нажмите любую клавишу . . .
```

Таблица тестов для программы из примера 4.1

№ теста	Вход	Выход
	n	
1.	30	2 4 8 16
2.	0	Натуральных степеней числа 2, не превосходящих 0, нет
3.	2	2

Пример 4.2.

Написать и отладить программу, решающую следующую задачу: *Дано вещественное положительное число x. Найти корень квадратный из x с помощью итерационной формулы Герона с точностью 0.000001.*

Итерационная формула Герона определяет последовательность чисел, первое число в которой равно

$a_1 = x,$

а каждое последующее, начиная со второго, находится по формуле

$a_i = \frac{1}{2}\left(a_{i-1} + \frac{x}{a_{i-1}}\right)$, для $i = 2, 3, 4, \ldots$

Последовательность таких чисел быстро сходится к величине $\sqrt{x}$.

Например, для x=2 начало последовательности имеет вид: 2; 1.5; 1.417; 1.414 и т.д. Чем больше номер члена последовательности, тем он ближе к $\sqrt{2}\ \approx 1{,}41421356$

В приведенной программе друг за другом получают числа последовательности (в переменной `a`) до тех пор, пока квадрат очередного числа не будет отличаться от x менее, чем на 0.0000001, то есть пока выполняется условие $|a_i^2 - x| \geq 0.000001$.

```
/*
Пример 4.2.
Дано вещественное положительное число x.
Найти корень квадратный из x с помощью итерационной формулы
Герона с точностью 0.000001.
*/

#include <iostream>
#include <locale.h>
using namespace std;

int main()
{
   setlocale(LC_ALL, "Russian");
   const double EPS = 1E-6;

   // Ввод данных
   double x;
   do
   {
```

```
      cout << "Введите вещественное положительное число x" <<
endl;
      cin >> x;
   } while (x <= 0);

   // Вычисление по формуле Герона
   double a = x;
   while (abs(a * a - x) >= EPS)
      a = (a + x / a) / 2;

   cout << "Квадратный корень из " << x << " равен " << a <<
endl;

   system("pause");
   return 0;
}
```

Изображение консольного окна программы из примера 4.2.

```
C:\Example 4.2.exe
Введите вещественное положительное число x
2
Квадратный корень из 2 равен 1.41421
Для продолжения нажмите любую клавишу . . .
```

Таблица тестов для программы из примера 4.2

№ теста	Вход	Выход
	x	
1.	2	1.41421
2.	16.16	4.01995
3.	100	10

Пример 4.3.

Написать и отладить программу, решающую следующую задачу: *Вводить с клавиатуры последовательность целых числа до 0 (сам 0 не входит в последовательность). Указать количество пар соседних чисел, в которых одно из чисел вдвое больше другого числа.*

```
/*
Пример 4.3.
Вводить с клавиатуры последовательность целых числа до 0 (сам 0
не входит в последовательность). Указать количество пар соседних
чисел, в которых одно из чисел вдвое больше другого числа.
*/

#include <iostream>
#include <locale.h>
using namespace std;

int main()
{
   setlocale(LC_ALL, "Russian");

   // Решение задачи
   int a, b = 0;
   int count = 0;
   cout << "Вводите целые числа до 0" << endl;
   cin >> a;
   while (a)
   {
      if (b)
         if (a == 2 * b || 2 * a == b)
            count++;
      b = a;
      cin >> a;
   }

   cout << "Количество пар соседних чисел, в которых " << endl;
   cout << "одно вдвое больше другого равно " << count << endl;

   system("pause");
   return 0;
}
```

Изображение консольного окна программы из примера 4.3.

```
C:\Example 4.3.exe
Введите целые числа до 0
1 2 3 6 3 6 0
Количество пар соседних чисел, в которых
одно вдвое больше другого равно 4
Для продолжения нажмите любую клавишу . . .
```

Таблица тестов для программы из примера 4.3

№ теста	Вход	Выход
1.	1 2 3 6 3 6 0	4
2.	1 -2 1 0	0
3.	-1 -2 0	1
4.	5 0	0
5.	0	0

Пример 4.4.

Написать и отладить программу, решающую следующую задачу: *Лесенкой назовём такую последовательность (длиной не менее двух) подряд идущих цифр натурального числа, в которой цифры последовательно увеличиваются на 1. Дано натуральное число n. Указать самую длинную лесенку этого числа.*

```
/*
Пример 4.4.
Лесенкой назовём такую последовательность (длиной не менее двух)
подряд идущих цифр натурального числа, в которой цифры последо-
вательно увеличиваются на 1. Дано натуральное число n. Указать
самую длинную лесенку этого числа.
*/

#include <iostream>
#include <locale.h>
using namespace std;

int main()
{
   setlocale(LC_ALL, "Russian");

   // Ввод числа
   long n;
   do
   {
      cout << "Введите натуральное число n" << endl;
      cin >> n;
   } while (n <= 0);

   int digit, prevDigit = 0; // текущая и предыдущая цифры
```

```
int stair = 0, stairLen = 0;    // текущая лесенка и её длина
int   maxStair = 0, maxStairLen = 0;  // максимальная лесенка
                                      //и её длина

// разложение числа n по цифрам с конца
while (n)
{
   digit = n % 10;

   if (prevDigit) // если цифра числа n не первая
   {
      // если соседние цифры образуют лесенку
      if (digit + 1 == prevDigit)
      {
         // присоединение цифры к лесенке
         stair = 10 * stair + digit;
         stairLen++;
         // анализ на максимум длины
         if (stairLen > maxStairLen)
         {
            maxStairLen = stairLen;
            maxStair = stair;
         }
      }
      else   // начало новой лесенки
      {
         stair = digit;
         stairLen = 1;
      }
   }
   else // если цифра числа n первая
   {
      stair = digit;
      stairLen = 1;
   }

   prevDigit = digit;
   n /= 10;
}

// Вывод ответа
if (maxStairLen > 1)
{
   cout << "Самая длинная лесенка ";
   // Вывод цифр лесенки в обратном порядке
   while (maxStair)
```

```
      {
         cout << maxStair % 10;
         maxStair /= 10;
      }
      cout << endl;
   }
   else
      cout << "Лесенок в числе нет" << endl;

   system("pause");
   return 0;
}
```

Изображение консольного окна программы из примера 4.4.

```
C:\Example 4.4.exe
Введите натуральное число n
1256701
Самая длинная лесенка 567
Для продолжения нажмите любую клавишу . . .
```

Таблица тестов для программы из примера 4.4

№ теста	Вход	Выход
	n	
1.	1256701	567
2.	1012355	0123
3.	9876543210	Лесенок в числе нет
4.	341289	89
5.	1	Лесенок в числе нет

Варианты задач лабораторной работы № 4

Указания. В программных решениях задач можно использовать наиболее подходящие виды циклов. Массивами, строками или иными структурными типами данных не пользоваться.

Входные данные задавать с клавиатуры в консольном окне, результат выдавать также в окно. Для отладки программы заранее составьте систему тестов.

№	Текст задачи
4.1	Дано натуральное число a. Найти номер слагаемого в сумме $1 + 2 + 3 + 4 + \cdots$, на котором эта сумма превзойдет a.
4.2	Дано натуральное число a. Найти номер слагаемого в сумме $1! + 2! + 3! + 4! + \cdots$, на котором эта сумма превзойдет a.
4.3	Дано натуральное число a. Найти номер слагаемого в сумме $2^1 + 2^2 + 2^3 + 2^4 + \cdots$, на котором эта сумма превзойдет a.
4.4	Дано вещественное число a. Среди элементов последовательности $\frac{1!}{1}, \frac{2!}{1+2}, \frac{3!}{1+2+3}, \ldots$ указать первый из элементов, больший a.
4.5	Дано натуральное число N. Найти количество единиц в двоичной записи этого числа.
4.6	Дано натуральное число N. Найти наибольшее число, факториал которого не превосходит N.
4.7	Дано натуральное число N. Вывести в порядке возрастания все числа Фибоначчи, не превосходящие N. Указание. Бесконечная последовательность чисел Фибоначчи a_1, a_2, a_3, a_4, … определяется следующим образом: $a_1 = a_2 = 1$: $a_i = a_{i-1} + a_{i-2}$ (для $i > 2$).
4.8	Дано два натуральных числа P и R. P – это пин-код некоторого устройства. R – это код, который ввёл пользователь. Допускается при вводе кода одна ошибочная цифра. Проверить, допустимый ли пин-код ввёл пользователь? Строками не пользоваться.
4.9	Дано вещественное положительное число b. Среди элементов последовательности $\frac{2^1}{1!}, \frac{2^2}{2!}, \frac{2^3}{3!}, \ldots$ указать первый из элементов, меньший b. Рекомендация: при тестировании вводите положительные числа b, близкие к нулю.
4.10	Дано натуральное число N. Представить это число в виде суммы натуральных чисел, являющихся целыми степенями числа 2 так, чтобы каждая степень входила в сумму не более одного раза.
4.11	Дано натуральное число N. Найти сумму цифр этого числа. Указание. Для выделения последней цифры целого числа N можно использовать операцию N % 10, для её отбрасывания из числа – операцию N / 10. Таким образом можно последовательно выделить все цифры исходного числа.
4.12	Дано натуральное число N. Найти наибольшую и наименьшую цифры этого числа. Указание – см. предыдущий вариант.

4.13	Дано натуральное число N. Построить новое число удалением из числа N нечетных цифр. Указание – см. предыдущий вариант.
4.14	Дано натуральное число N. Определить, является ли это число палиндромом? Указание: палиндром - это последовательность символов, одинаково читаемая слева направо и справа налево. Также см. указание к предыдущему варианту.
4.15	Дано натуральное число N. Разложить это число в каноническое произведение степеней с простыми основаниями и натуральными показателями. Пример: 60 = 2^2 × 3^1 × 5^1.
4.16	Вводить с клавиатуры последовательность целых числа до 0 (сам 0 не входит в последовательность). Указать номера наибольшего и наименьшего из введённых чисел.
4.17	Вводить с клавиатуры последовательность целых числа до 0 (сам 0 не входит в последовательность). Найти количество элементов последовательности, имеющих максимальное значение.
4.18	Вводить с клавиатуры последовательность целых числа до 0 (сам 0 не входит в последовательность). Найти номер первого из чисел с максимальным значением, вывести результат на консоль.
4.19	Вводить с клавиатуры последовательность вещественных чисел до 0 (сам 0 не входит в последовательность). Найти *второе* максимальное значение из них (если оно есть). Вывести результат на консоль. Например, для последовательности 1 2 3 3 2 1 3 1 таким числом является 2.
4.20	Вводить с клавиатуры последовательность вещественных чисел до 0 (сам 0 не входит в последовательность). Найти максимальное значение из них и количество чисел, имеющих такое значение. Вывести результат на консоль.
4.21	Вводить с клавиатуры последовательность вещественных чисел до 0 (сам 0 не входит в последовательность). Найти длину самой длинной неубывающей подпоследовательности подряд идущих чисел.
4.22	Вводить с клавиатуры последовательность целых чисел до 0 (сам 0 не входит в последовательность). Найти длину самой длинной подпоследовательности подряд идущих положительных чисел.
4.23	Вводить с клавиатуры последовательность вещественных чисел до 0 (сам 0 не входит в последовательность). Найти *второе* минимальное значение из них (если оно есть). Вывести результат на консоль.

4.24	С клавиатуры вводится последовательность вещественных чисел до 0 (сам 0 не является членом последовательности). Найти количество локальных максимумов последовательности, то есть таких её элементов, которые больше своих соседей. Крайние элементы тоже могут быть локальными максимумами!
4.25	Даны два вещественных положительных числа A и B (A ≤ B). Найти все пары целых взаимно простых чисел, попадающих в отрезок [A; B].
4.26	Дано натуральное число N, не превосходящее 365. Найти число и месяц невисокосного года, на который приходится N-ый по счёту день этого года. Массивами не пользоваться. Усложнение 1: операторами циклов и рекурсией не пользоваться. Усложнение 2: операторами циклов, ветвления, выбора и рекурсией не пользоваться.
4.27	Дано натуральные числа n, m. Проверить, не является ли палиндромом запись цифр этого числа в m-ичной системе счисления?
4.28	Дано натуральное число N. Найти и вывести на экран наиболее длинную неубывающую последовательность подряд идущих цифр этого числа. Например, для N = 121224379 результатом будет число 1224. Если имеется несколько таких последовательностей, то найти первую слева из них.
4.29	Дано натуральное число r, не превышающее 16. Вывести таблицу умножения в системе счисления с основанием r.
4.30	Дано натуральное число N. Представить его в виде суммы одного или нескольких различных чисел Фибоначчи так, чтобы в этом представлении не оказалось двух соседних чисел из последовательности Фибоначчи. Указание: см. теорему Цекендорфа.
4.31	Дано натуральное число N. Указать N-ую цифру последовательности 11235813213455..., в которой выписаны подряд числа Фибоначчи.
4.32	Даны два натуральных числа a, b. Найти десятичную дробь, равную $\frac{a}{b}$, возможно - с периодом. Пример: $\frac{3}{7} = 0,(428571)$

Лабораторная работа № 5. Функции

Цель

Изучить и освоить базовые механизмы процедурного программирования - понятия «функция», «вызов функции», способы передачи параметров, локальные данные. Структурная организация программы

Необходимый теоретический материал

Для решения задач лабораторной работы № 5 необходимо изучить следующий теоретический материал:

- функции;
- вызов функции, возвращаемое значение;
- локальные данные;
- параметры функции, два способа передачи параметров;
- побочный эффект функции;
- нисходящее программирование.

Примеры

Пример 5.1.

Написать и отладить программу, решающую следующую задачу: *Даны три натуральных числа a, b, c. Перераспределить их значения в порядке возрастания их наименьших цифр. Например: вход a = 29, b = 55, c = 188; выход: a = 188, b = 29, c = 55.*

```
/*
Пример 5.1.
Даны три натуральных числа a, b, c. Перераспределить их значения
в порядке возрастания их наименьших цифр. Например: вход a = 29,
b = 55, c = 188; выход: a = 188, b = 29, c = 55.
*/
#include <iostream>
#include <locale.h>

using namespace std;

// Функция обменивает значения переменных x и y
void swap(int& x, int& y)
{
```

```
   int tmp = x;
   x = y;
   y = tmp;
}

// Функция возвращает наименьшую цифру числа x
int minDigit(int x)
{
   int min = 10;
   while (x)
   {
      if (x % 10 < min)
         min = x % 10;
      x /= 10;
   }
   return min;
}

// Функция упорядочивает две переменные x и y по возрастанию их
// наименьших цифр
void sort(int &x, int &y)
{
   if (minDigit(x) > minDigit(y))
      swap(x, y);
}

int main()
{
   setlocale(LC_ALL, "Russian");
   cout << "Введите три натуральных числа" << endl;
   int a, b, c;
   cin >> a >> b >> c;
   sort(a, b);
   sort(a, c);
   sort(b, c);
   cout << "Числа в порядке возрастания их наименьших цифр:"
        << endl;
   cout << a << " " << b << " " << c << endl;
   system("pause");
}
```

Изображение консольного окна программы из примера 5.1.

```
C:\Example 5.1.exe
Введите три натуральных числа
```

```
29 55 188
Числа в порядке возрастания их наименьших цифр
188 29 55
Для продолжения нажмите любую клавишу . . .
```

Таблица тестов для программы из примера 5.1

№ теста	Вход			Выход
	a	b	c	
1.	29	55	188	188 29 55
2.	19	20	21	20 19 21
3.	12	21	221	12 21 221

Пример 5.2.

Написать и отладить программу, решающую следующую задачу: *Даны три действительных числа a, b, c. Если треугольник со сторонами равными a, b и c существует, то найти и вывести его углы в градусах.*

Для нахождения косинуса угла α треугольника со сторонами a, b и c, лежащего напротив стороны a, можно воспользоваться теоремой косинусов:

$$a^2 = b^2 + c^2 - 2bc\cos\alpha$$

откуда

$$\cos\alpha = \frac{b^2 + c^2 - a^2}{2bc}$$

Для перевода значения угла из радианов в градусы можно воспользоваться формулой:

$$\alpha_{\text{градусы}} = \frac{\alpha_{\text{радианы}}}{\pi} \cdot 180$$

Заметим также, что $\pi = \arccos(-1)$

```
/*
Пример 5.2.
Даны три действительных числа a, b, c. Если треугольник со сторонами равными a, b и c существует, то найти и вывести его углы в градусах
*/
```

```
#include <iostream>
#include <locale.h>
#include <cmath>

using namespace std;

// Функция tryangleExist возвращает значение true, если
// существует треугольник со сторонами a, b и c,
// и false - в противном случае.
bool tryangleExist(double a, double b, double c)
{
   return (a < b + c) && (b < a + c) && (c < a + b);
}

// Функция возвращает квадрат числа x
inline double sqr(double x)
{
   return x * x;
}

// Функция angle возвращает значение угла (в градусах),
// лежащего напротив стороны a в треугольнике со
// сторонами  a, b и c.
double angle(double a, double b, double c)
{

   double cosAngle = (sqr(b) + sqr(c) - sqr(a)) / (2 * b * c);
   double angleRad = acos(cosAngle);
   return angleRad * 180 / acos(-1);
}

int main()
{
   setlocale(LC_ALL, "Russian");
   double a(0), b(0), c(0);
   cout << "Введите три действительных числа" << endl;
   cin >> a >> b >> c;

   if (tryangleExist(a, b, c))
   {
      double alpha = angle(a, b, c);
      cout << "Угол напротив стороны " << a << " равен "
           << alpha << " градусов" << endl;

      double beta = angle(b, a, c);
      cout << "Угол напротив стороны " << b << " равен "
```

```
            << beta << " градусов" << endl;

      double gamma = angle(c, a, b);
      cout << "Угол напротив стороны " << c << " равен "
            << gamma << " градусов" << endl;
   }
   else
      cout << "Треугольник не существует" << endl;
   system("pause");
}
```

Изображение консольного окна программы из примера 5.2.

```
C:\Example 5.2.exe
Введите три действительных числа
3 4 5
Угол напротив стороны 3 равен 36.8699 градусов
Угол напротив стороны 4 равен 53.1301 градусов
Угол напротив стороны 5 равен 90 градусов
Для продолжения нажмите любую клавишу . . .
```

Таблица тестов для программы из примера 5.2

№ теста	Вход			Выход
	a	b	c	
1.	3	4	5	36.8699 53.1301 90
2.	3	3	3	60 60 60
3.	1	2	3	Треугольник не существует

Пример 5.3.

Написать и отладить программу, решающую следующую задачу: *Даны два трехмерных вектора a и b, заданные своими координатами. Найти (a + b) x (a - b), где +, - и x означают сумму, разность и скалярное произведение векторов соответственно*.

```
/*
Пример 5.3.
Даны два трехмерных вектора a и b, заданные своими координатами.
Найти
(a + b) x (a - b), где +, - и x означают сумму, разность и ска-
лярное произведение векторов соответственно.
*/
#include <iostream>
#include <locale.h>

using namespace std;
struct Vector
{
   double x, y, z;
};

// Функция задает координаты вектора x
void set(Vector &v, double x1, double y1, double z1)
{
   v.x = x1;
   v.y = y1;
   v.z = z1;
}

// Функция выводит вектор на экран
void print(Vector v)
{
   cout << "( " << v.x << "; " << v.y << "; " << v.z << ")" <<
endl;
}

// Функция возвращает вектор - сумму векторов v и w
Vector sum(Vector v, Vector w)
{
   Vector s;
   set(s, v.x + w.x, v.y + w.y, v.z + w.z);
   return s;
}

// Функция возвращает вектор - разность векторов v и w
Vector diff(Vector v, Vector w)
{
   Vector s;
   set(s, v.x - w.x, v.y - w.y, v.z - w.z);
   return s;
```

```
}


// Функция возвращает число - скалярное произведение векторов v
и w
double prod(Vector v, Vector w)
{
   return v.x * w.x + v.y * w.y + v.z * w.z;
}


int main()
{
   setlocale(LC_ALL, "Russian");
   Vector a, b;
   set(a, 1, 1, 1);
   set(b, 1, 2, 3);

   cout << "Исходные векторы:" << endl;
   cout << " a = ";
   print(a);
   cout << " b = ";
   print(b);


   double result;
   result = prod(sum(a, b), diff(a, b));
   cout << "Результат (a + b) x (a - b) = " << result << endl;

   system("pause");
}
```

Изображение консольного окна программы из примера 5.3.

```
C:\Example 5.3.exe
Исходные векторы
a = ( 1; 1; 1)
b = ( 1; 2; 3)
Результат (a + b) x (a - b) = -11
Для продолжения нажмите любую клавишу . . .
```

Таблица тестов для программы из примера 5.3

№ теста	Вход		Выход
	a	b	
1.	(1; 1; 1)	(1; 2; 3)	-11
2.	(0; -2; 4)	(0; 10; 20)	-480
3.	(1; 1; 1)	(1; 1; 1)	0

Варианты задач лабораторной работы № 5

Указания.

Разработать программу с применением принципа нисходящего программирования. Программа должна состоять из функций, каждая из которых решает конкретную подзадачу. Функции передают и получают данные только через параметры и возвращаемые значения.

№	Текст задачи
5.1	Даны три целых положительных числа a, b и c. Указать из них то число, сумма цифр которого максимальна.
5.2	Даны три целых положительных числа a, b и c. Для каждого числа определить максимальную цифру и выбрать то число из трех, в котором она минимальна.
5.3	Даны три целых положительных числа a, b и c. Выбрать такие два числа из трех, чтобы НОК этой пары было бы наименьшим.
5.4	Даны три целых положительных числа a, b и c. Выбрать такие два числа из трех, чтобы НОД этой пары был бы наибольшим.
5.5	Даны натуральные числа a, b $(a \leq 10, b \leq 10)$. Вывести на экран фразу на русском языке, которая обозначает арифметическое выражение $a + b = c$. Например, для чисел $a = 7, b = 9$ фраза будет выглядеть так: «семь плюс девять равно шестнадцать».
5.6	Даны три целых положительных числа a, b и c. Определить, какие из этих чисел являются полными квадратами.
5.7	Даны три целых положительных числа a, b и c. Определить, какие из этих чисел являются палиндромами. Строками не пользоваться.
5.8	Даны три целых положительных числа a, b и c. Найти в каждом числе произведение минимальной и максимальной цифр.
5.9	Даны три целых положительных числа a, b и c. Указать все такие пары из них, что одно из чисел пары получается чтением цифр другого числа в обратном порядке. Строками не пользоваться.
5.10	Даны три целых положительных числа a, b и c. Указать такое из них, которое получается сцеплением последовательности цифр двух

	других чисел. Например среди числе a = 123, b = 65, c = 65123 таким числом является число c. Строками не пользоваться.
5.11	Даны три целых положительных числа a, b и c. Вывести их в порядке возрастания сумм их цифр. Например, a = 12, b = 9, c = 100; выход 100, 12, 9.
5.12	Даны три трехмерных вектора $\vec{a}$, $\vec{b}$ и $\vec{c}$, заданные своими координатами, вещественное число k. Найти вектор, равный $k\vec{a} + (1-k)\vec{b} + k^2\vec{c}$. Указание: реализовать функции умножения скаляра на вектор, суммы векторов.
5.13	Даны три трехмерных вектора $\vec{a}$, $\vec{b}$ и $\vec{c}$, заданные своими координатами, вещественное число k. Найти вектор, равный $k(\vec{a} + \vec{b}) + (1-k)(\vec{a} - \vec{c})$. Указание: реализовать функции умножения скаляра на вектор, суммы и разности векторов.
5.14	Даны три трехмерных вектора $\vec{a}$, $\vec{b}$ и $\vec{c}$, заданные своими координатами. Найти вектор, равный $(\vec{a} + \vec{b}) + (2\vec{a} + 3\vec{c})$. Указание: реализовать функции умножения скаляра на вектор, суммы векторов.
5.15	Даны три трехмерных вектора $\vec{a}$, $\vec{b}$ и $\vec{c}$, заданные своими координатами. Найти число, равное $\lvert\vec{a} + \vec{b}\rvert + \lvert\vec{b} + \vec{c}\rvert$. Указание: реализовать функции суммы векторов, длины вектора. Длина вектора $\vec{a}$ равна $\lvert\vec{a}\rvert = \sqrt{a_1^2 + a_2^2 + a_3^2}$.
5.16	Даны три трехмерных вектора $\vec{a}$, $\vec{b}$ и $\vec{c}$, заданные своими координатами. Найти вектор с наибольшей длиной. Указание: реализовать функцию, возвращающую длину вектора. Длина вектора $\vec{a}$ равна $\lvert\vec{a}\rvert = \sqrt{a_1^2 + a_2^2 + a_3^2}$.
5.17	Даны три действительных числа a, b и c. Если треугольник со сторонами равными a, b и c существует, то найти и вывести его углы в градусах.
5.18	Даны координаты трёх точек плоскости $(x_1, y_1), (x_2, y_2), (x_3, y_3)$. Найти длины медиан треугольника с вершинами в этих точках.
5.19	Даны координаты четырёх точек плоскости $(x_1, y_1), (x_2, y_2), (x_3, y_3), (x_4, y_4)$. Определить, где лежит ли точка с координатами (x_4, y_4) – внутри или вне треугольника с вершинами $(x_1, y_1), (x_2, y_2), (x_3, y_3)$?
5.20	Даны координаты четырёх точек плоскости $(x_1, y_1), (x_2, y_2), (x_3, y_3), (x_4, y_4)$. Определить, лежат ли какие-либо три из них на одной прямой?

5.21	Даны три вещественных числа p, q, z. Найти все такие перестановки этих чисел, чтобы они, будучи подставленными в качестве коэффициентов квадратного уравнения $ax^2 + bx + c = 0$, дали бы уравнение, имеющее два различных корня.
5.22	Даны вещественные трёхмерные векторы x, y, z. Найти $xy + yz + xz$, где xy означает скалярное произведение векторов.
5.23	Даны вещественные трёхмерные векторы x, y, z. Найти $x + 2y + 3z$.
5.24	Даны вещественные трёхмерные векторы x, y. Найти $\|x\| + \|y\| + \|x + y\|$, где $\|x\|$ означает длину вектора x.
5.25	Даны четыре вещественных положительных числа a, b, c, d. Для каждой тройки этих чисел определить, существует ли треугольник с такими сторонами и среди всех таких треугольников найти тот, площадь которого максимальна.
5.26	Даны четыре вещественных положительных числа a, b, c, d. Найти среди них все тройки Пифагора.
5.27	С клавиатуры вводится последовательность целых положительных чисел до нуля (нуль не является членом последовательности). При этом заранее неизвестно количество членов последовательности. Найти наибольший общий делитель всех членов последовательности. Массивы, векторы и проч. не использовать.
5.28	Дано натуральное число n > 1. Найти несократимую обыкновенную дробь, равную сумме $$\frac{1}{2} + \frac{2}{3} + \frac{3}{4} + \cdots + \frac{n-1}{n}$$
5.29	Даны натуральные числа d, m, y. Написать функцию, возвращающую «истина», если в григорианском календаре существует дата - ое число m-го месяца y-го года, и «ложь» в противном случае.
5.30	С клавиатуры вводится последовательность целых положительных чисел до нуля (нуль не является членом последовательности). При этом заранее неизвестно количество членов последовательности. Найти наименьшее общее кратное всех членов последовательности. Массивы, векторы и проч. не использовать.
5.31	С клавиатуры вводится последовательность целых положительных чисел $a_1, a_2, a_3, \dots, a_n, (n > 1)$ до нуля (нуль не является членом последовательности). При этом заранее неизвестно количество членов последовательности. Найти несократимую обыкновенную дробь, равную сумме

	$$\frac{a_1}{a_2}+\frac{a_2}{a_3}+\frac{a_3}{a_4}+\cdots+\frac{a_n}{a_1}$$ Массивы, векторы и проч. не использовать.
5.32	Даны натуральные n, d, m $(n \le 7, d \le 31, m \le 12)$. Найти номер дня недели, на который выпадает -ое число m-го месяца невисокосного года, если этот год начинается с n-го дня недели. Напишите функцию, возвращающую этот номер.

Лабораторная работа № 6. Одномерные массивы

Цель работы

Изучить способы определения, инициализации и алгоритмы обработки линейных (одномерных) массивов.

Необходимый теоретический материал

Для решения задач лабораторной работы № 6 необходимо изучить следующий теоретический материал:

- фиксированные и динамические одномерные массивы;
- определение и инициализация одномерных массивов;
- типовые алгоритмы обработки одномерных массивов, в том числе ввод и вывод массива, заполнение случайными, поиск экстремума и др.;
- простейшие алгоритмы сортировки массива.

Примеры

Пример 6.1.

Написать и отладить программу, решающую следующую задачу: *Дан одномерный целочисленный массив из N элементов (например, N=10), заполненный случайными числами. Найти среднее арифметическое всех четных элементов массива.*

```
/*
Пример 6.1.
  Дан одномерный целочисленный массив из N элементов (например,
N=10), заполненный случайными числами. Найти среднее арифметиче-
ское всех четных элементов массива.
*/

#include <iostream>
#include <iomanip>
#include <locale.h>
#include <ctime>
using namespace std;

// Заполнение массива случайными числами
```

```
void randomArray(int a[], int N)
{
   srand(time(NULL));
   for (int i = 0; i < N; i++)
      a[i] = rand() % 100 - 50;  //2 * rand() % 100 - 1;
}

// Вывод массива
void printArray(int a[], int N)
{
   for (int i = 0; i < N; i++)
      cout << setw(4) << a[i];
   cout << endl;
}

// Вычисление среднего арифметического evenAvg четных эле-
ментов
// массива a
bool sumEvenNumbers(int a[], int N, double& evenAvg)
{
   int evenSum = 0;// сумма четных
   int evenCount = 0; // количество четных
   for (int i = 0; i < N; i++)
      if (a[i] % 2 == 0)
      {
         evenSum += a[i];
         evenCount++;
      }

   if (evenCount)
   {
      evenAvg = (double)evenSum / evenCount;
      return true;
   }
   else
      return false;
}

int main()
{
   setlocale(LC_ALL, "Russian");

   // Определение массива
```

```
   const int N = 10;
   int a[N];

   randomArray(a, N);
   cout << "Исходный массив:" << endl;
   printArray(a, N);

   // Вычисление среднего арифметического четных элементов
   double evenAvg = 0.0;
   if (sumEvenNumbers(a, N, evenAvg))
      cout << "Среднее арифметическое четных = " << evenAvg
           << endl;
   else
      cout << "Четных элементов в массиве нет" << endl;

   system("pause");
   return 0;
}
```

Изображение консольного окна программы из примера 6.1.

```
C:\Example 6.1.exe
Исходный массив:
  -4  13 -40 -25  13 -25  13   9  31  30
Среднее арифметическое четных = -4.66667
Для продолжения нажмите любую клавишу . . .
```

Таблица тестов для программы из примера 6.1

№ теста	Вход (случайный массив)	Выход
1.	-4 13 -40 -25 13 -25 13 9 31 30	Среднее арифметическое четных = -4.66667
2.	1 93 97 13 11 39 49 89 83 37	Четных элементов в массиве нет

Пример 6.2.

Написать и отладить программу, решающую следующую задачу: *Дано натуральное число n. Два одномерных массива a и b размера n за-*

полнены случайными числами, упорядоченными в порядке не убывания. Переместить значения массивов в массив с размера 2n так, чтобы он также был упорядочен по не убыванию.

```
/*
Пример 6.2.
Дано натуральное число n. Два одномерных массива a и b размера n
заполнены случайными числами, упорядоченными в порядке не убыва-
ния. Переместить значения массивов в массив с размера 2n так,
чтобы он также был упорядочен по не убыванию.
*/

#include <iostream>
#include <iomanip>
#include <locale.h>
#include <ctime>
using namespace std;

// Заполнение массива a неубывающими случайными целыми
// числами
void fillArrayIncNumbers(int* a, int n)
{
   a[0] = rand() % 30;
   for (int i = 1; i < n; i++)
      a[i] = a[i - 1] + rand() % 10;
}

// Вывод массива на экран
void printArray(int* a, int n)
{
   for (int i = 0; i < n; i++)
      cout << setw(4) << a[i];
   cout << endl;
}

// Слияние упорядоченных массивов a и b размера n в массив c
void mergeArray(int* a, int* b, int*& c, int n)
{
   c = new int[2 * n];
   int ia = 0, ib = 0;   // индексы a и b
   int ic = 0;         // индекс c
   // Перенос значений из массивов a и b в массив c
   while (ia < n && ib < n)
```

```
      if (a[ia] < b[ib])
         c[ic++] = a[ia++];
      else
         c[ic++] = b[ib++];
   // Перенос оставшихся значений из a в c
   while (ia < n)
      c[ic++] = a[ia++];
   // Перенос оставшихся значений из b в c
   while (ib < n)
      c[ic++] = b[ib++];
}

int main()
{
   setlocale(LC_ALL, "Russian");
   srand(time(NULL));

   // Ввод n
   int n;
   do
   {
      cout << "Введите размер массива" << endl;
      cin >> n;
   } while (n < 1);

   // Создание динамических массивов a и b
   int* a = new int[n];
   int* b = new int[n];

   // Заполнение массивов a и b неубывающими целыми числами
   fillArrayIncNumbers(a, n);
   fillArrayIncNumbers(b, n);

   // Вывод массивов a и b
   cout << "Массив a:" << endl;
   printArray(a, n);
   cout << "Массив b:" << endl;
   printArray(b, n);

   // Слияние массивов a и b в массив c
   int* c;
   mergeArray(a, b, c, n);
```

```
   // Вывод массива c
   cout << "Массив c, полученный слиянием a и b:" << endl;
   printArray(c, 2 * n);

   system("pause");
   return 0;
}
```

Изображение консольного окна программы из примера 6.2.

```
C:\Example 6.2.exe
Введите размер массива
5
Массив a:
  22  26  31  39  44
Массив b:
  21  24  26  32  38
Массив c, полученный слиянием a и b:
  21  22  24  26  26  31  32  38  39  44
Для продолжения нажмите любую клавишу . . .
```

Пример 6.3.

Написать и отладить программу, решающую следующую задачу: *Дано множество точек координатной плоскости, задаваемых своими координатами (x0; y0), (x1; y1), ... (x(n-1); y(n-1)). Выдать список точек в порядке удалённости их от начала координат.*

```
/*
Пример 6.3.
Дано множество точек координатной плоскости, задаваемых своими
координатами (x0; y0), (x1; y1), ... (x(n-1); y(n-1)).
Выдать список точек в порядке удалённости их от начала координат.
*/

#include <iostream>
#include <iomanip>
#include <locale.h>
#include <ctime>
#include <math.h>
```

```
using namespace std;

// Структура - точка координатной плоскости
struct Point {
   double x, y;     // Координаты точки
   double distance; // Расстояние от точки до начала координат
};

// Функция создает массив точек со случайными координатами
void createRandomPoints(Point * & points, int n)
{
   // Создание динамического массива точек
   points = new Point[n];

   // Заполнение случайными координатами
   srand(time(NULL));
   for (int i = 0; i < n; i++)
   {
      points[i].x = -50.0 + 100.0 * rand() / (double)RAND_MAX;
      points[i].y = -50.0 + 100.0 * rand() / (double)RAND_MAX;
   }
}

// Функция выводит массив точек
void printPoints(Point * points, int n)
{
   // Вывод массива точек
   cout << fixed << setprecision(2);
   for (int i = 0; i < n; i++)
      cout << "(" << setw(6) << points[i].x << "; " << setw(6)
<< points[i].y << ")" << endl;
}

// Расчет расстояния от каждой точки до начала координат
void calculateDistance(Point * points, int n)
{
   for (int i = 0; i < n; i++)
      points[i].distance = sqrt(pow(points[i].x, 2) +
pow(points[i].y, 2));
}

// Вывод расстояний от точек до начала координат
void printDistance(Point * points, int n)
{
   cout << fixed << setprecision(2);
```

```
   for (int i = 0; i < n; i++)
      cout << "D = " << setw(6) << points[i].distance << endl;
}

// Сортировка массива точек по их расстояниям до начала
// координат, метод сортировки - выбором
void sortPoints(Point * points, int n)
{
   for (int i = 0; i < n - 1; i++)
   {
      int imin = i;
      for (int j = i + 1; j < n; j++)
         if (points[j].distance < points[imin].distance)
            imin = j;
      swap(points[i], points[imin]);
   }
}

int main()
{
   setlocale(LC_ALL, "Russian");

   // Ввод количества точек множества
   int n;
   do
   {
      cout << "Введите количество точек плоскости" << endl;
      cin >> n;
   } while (n < 1);

   Point * points = NULL;
   createRandomPoints(points, n);

   cout << "Исходный массив точек:" << endl;
   printPoints(points, n);

   calculateDistance(points, n);

   sortPoints(points, n);

   cout << "Отсортированный массив точек:" << endl;
   printPoints(points, n);

   cout << "Расстояния:" << endl;
   printDistance(points, n);
```

```
   system("pause");
   return 0;
}
```

Изображение консольного окна программы из примера 6.3.

```
C:\Example 6.3.exe
Введите количество точек плоскости
5
Исходный массив точек:
( -3.44; -32.05)
(-44.43; -11.95)
(-47.64; -19.20)
(-38.39; -19.86)
( 13.68; -20.36)
Отсортированный массив точек:
( 13.68; -20.36)
( -3.44; -32.05)
(-38.39; -19.86)
(-44.43; -11.95)
(-47.64; -19.20)
Расстояния:
D = 24.53
D = 32.23
D = 43.22
D = 46.01
D = 51.36
Для продолжения нажмите любую клавишу . . .
```

Пример 6.4.

Написать и отладить программу, решающую следующую задачу: *Имеется набор из n гирек с массами a0, a1, ..., a(n-1) граммов. Продавцу нужно взвесить на чашечных весах s граммов сахара. Найти все сочетания гирек, суммарная масса которых равна s. Все входные данные целые и вводятся с клавиатуры.*

Для поиска нужных комбинаций гирек в приведенной программе используется метод перебора всех возможных подмножеств множества

из n элементов. Суть этого метода заключается в том, что каждая комбинация элементов множества представляется двоичным числом, состоящим из n двоичных цифр (0 или 1). Например, двоичное число 00101 означает подмножество, в которое попадает нулевой и второй элементы (цифры нумеруются с конца числа, первая справа цифра имеет номер 0) набора из пяти элементов. Таким образом, для перебора всех подмножеств потребуется перебрать двоичные представления целых чисел, имеющие n цифр. Количество таких чисел равно 2^n, ровно столько же и всевозможных подмножеств.

Для проверки того, находится ли в j-ой позиции двоичного представления некоторого числа x единица используется операция & – «побитовое И». Первый её операнд – проверяемое число x, а второй – двоичное число, в котором только в j-ой позиции находится 1, остальные нули, то есть это число 2^j. Например, для числа x = 11011 проверим, находится ли в третьей позиции цифра 1? Операция

```
11011 & (1 << 3)
```

даст значение «истина», так как действительно в третьей справа позиции (считая с нулевой) находится единица.

Побитовый сдвиг << используется для получения степени числа 2. То есть 2^j выражается как (`1 << j`).

```
/*
Пример 6.4.
Имеется набор из n гирек с массами a0, a1, ..., a(n-1) граммов.
Продавцу нужно взвесить на чашечных весах s граммов сахара.
Найти все сочетания гирек, суммарная масса которых равна s. Все
входные данные целые и вводятся с клавиатуры.
*/
#include <iostream>
#include <iomanip>
#include <locale.h>
#include <ctime>
#include <math.h>

using namespace std;

int main()
{
   setlocale(LC_ALL, "Russian");

   // Ввод входных данных
   int n;
```

```
   cout << "Введите количество гирек" << endl;
   cin >> n;

   int * a = new int[n];
   cout << "Введите массы " << n << " гирек" << endl;
   for (int i = 0; i < n; i++)
      cin >> a[i];

   int s;
   cout << "Введите массу, которую нужно взвесить" << endl;
   cin >> s;

   // Нахождение всех искомых подмножеств гирек
   int count = 1 << n; // 2^n - количество подмножеств элементов
                       // множества из n элементов
   bool find = false;

   // Перебор всех подмножеств элементов массива a
   cout << "Эту массу можно взвесить следующими наборами гирек:"
        << endl;
   for (int combination = 0; combination < count; combination++)
   {
      // расчет общей массы гирек в подмножестве, задаваемом
      // двоичным  представлением числа combination
      int mass = 0;
      for (int j = 0; j < n; j++)
         // проверка наличия 1 в j-ом разряде двоичного числа
         // combination
         if (combination & (1 << j))
            mass += a[j];
      if (mass == s)
      {
         find = true;
         // вывод найденного набора гирек
         for (int j = 0; j < n; j++)
            if (combination & (1 << j))
               cout << a[j] << " ";
         cout << endl;
      }
   }
   if (!find)
      cout << "Подобрать набор гирек невозможно" << endl;

   system("pause");
   return 0;
}
```

Изображение консольного окна программы из примера 6.4.

```
C:\Example 6.4.exe
Введите количество гирек
5
Введите массы 5 гирек
1 2 3 4 5
Введите массу, которую нужно взвесить
8
Эту массу можно взвесить следующими наборами гирек:
1 3 4
1 2 5
3 5
Для продолжения нажмите любую клавишу . . .
```

Таблица тестов для программы из примера 6.4

№ теста	Вход			Выход
	n	a	s	
1.	5	1 2 3 4 5	8	1 3 4 1 2 5 3 5
2.	3	2 2 2	4	2 2 2 2 2 2
3.	3	1 3 5	7	Подобрать набор гирек невозможно

Варианты задач лабораторной работы № 6

Указания. В решениях предложенных задач необходимо использовать одномерные массивы. Выбор базового типа массива остается за автором решения. Желательно использовать динамические массивы с задаваемым во время исполнения программы размером.

Как правило, массив заполняется случайными числами; в тех задачах, когда необходимо проверить «специальные случаи» исходных данных, можно массиву задавать конкретные значения.

№	Текст задачи
6.1	Даны значения n элементов целочисленного массива. Среди отрицательных элементов массива (если такие есть) найти наибольший.
6.2	Даны значения n элементов целочисленного массива. Среди положительных элементов массива (если такие есть) найти наименьший.
6.3	Даны значения n элементов целочисленного массива. Найти наиболее часто встречающееся число; если таковых будет несколько, то указать их все.
6.4	Даны значения n элементов целочисленного массива. Найти количество различных значений элементов.
6.5	Даны значения n элементов целочисленного массива. Найти количество тех элементов массива, которые меньше своих соседей. Крайние элементы массива тоже имеют соседей!
6.6	Даны значения n элементов вещественного массива. Найти второе максимальное значение (если такое есть).
6.7	Дано число n. Заполнить целочисленный массив из n элементов всеми значениями от 1 до n в случайном порядке.
6.8	Дан целочисленный массив x из n элементов. Заполнить массив y значениями элементов массива x без повторов. Размер массива y будет не больше размера x.
6.9	Коэффициенты многочлена -ой степени $a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0$ можно разместить в вещественном массиве размера n+1: $a_0, a_1 \ldots a_{n-1}, a_n$. Даны два многочлена P(x) и Q(x) степеней m и n соответственно, заданные своими коэффициентами. Найти коэффициенты многочлена, равного сумме многочленов P(x) и Q(x).
6.10	Линейный массив заполняется случайными целыми числами. Осуществить циклический сдвиг элементов массива на такое количество позиций вправо, чтобы максимальный элемент оказался в конце массива.
6.11	Вводится целочисленный массив x из n элементов. Проверить, является ли последовательность всех элементов массива палиндромом? Палиндром – это последовательность, которая читается одинаково с начала и с конца.
6.12	Дан массив из N вещественных чисел. Осуществить циклический сдвиг элементов массива на такое количество позиций влево, чтобы его минимальный элемент оказался в начале массива.
6.13	Дан массив из N вещественных чисел. Разместить все отрицательные элементы массива в его начале, а неотрицательные – в конце

	массива. Сохранить взаимное расположение отрицательных элементов между собой; аналогично – для неотрицательных элементов. Разрешается использовать только один массив.
6.14	Дан массив из N вещественных чисел. Найти элементы с минимальным и максимальным значениями; отсортировать часть массива, состоящую из находящихся между ними элементов. Разрешается использовать только один массив.
6.15	Дан массив из n случайных целых чисел. Переместить в начало массива все нечётные числа, а в конец - все чётные числа. Отсортировать между собой все нечетные, а затем все четные элементы. Разрешается использовать только один массив.
6.16	Протокол соревнований по марафонскому бегу содержит номера финишировавших участников (целые положительные числа, все различные) и их результаты – время в минутах (вещественные числа). Данные хранятся в двух массивах одинакового размера: в массиве numbers номера, а в массиве results - соответствующие результаты. Вывести итоговый протокол соревнований в виде списка номеров и результатов участников, упорядоченный по возрастанию результатов.
6.17	Даны два массива A и B по n целочисленных элементов в каждом. Элементы каждого массива не повторяются. Считая A и B множествами элементов, найти их пересечение.
6.18	Даны два массива A и B по n целочисленных элементов в каждом. Элементы каждого массива не повторяются. Считая A и B множествами элементов, найти их объединение.
6.19	Даны два массива A и B по n целочисленных элементов в каждом. Элементы каждого массива не повторяются. Считая A и B множествами элементов, найти их разность.
6.20	Даны два массива A и B размеров m и n соответственно, заполненные десятичными цифрами. Считая цифры A и B записями чисел, найти сумму этих чисел.
6.21	Даны два массива A и B размеров m и n соответственно, заполненные десятичными цифрами. Считая последовательные цифры A и B записями чисел, найти произведение этих чисел.
6.22	Дан массив из n вещественных чисел. Найти самую длинную неубывающую последовательность подряд идущих элементов массива. Если таких последовательностей будет несколько, то указать первую из них.

6.23	Даны два массива A и B по n целочисленных элементов в каждом. Элементы каждого массива не повторяются. Считая A и B множествами элементов, найти их симметрическую разность.
6.24	Дан массив из n вещественных чисел. Найти самую длинную неубывающую последовательность подряд идущих элементов массива. Если таких последовательностей будет несколько, то указать их все.
6.25	Даны два массива a и b размеров m и n соответственно, заполненные неубывающими последовательностями вещественных значений. Построить новый массив c из m+n значений массивов a и b, также расположенных в нём по не убыванию
6.26	Коэффициенты многочлена $a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0$ размещены в массиве $a_0, a_1 \ldots a_{n-1}, a_n$. Дано вещественное число x. Найти значение многочлена для данного x. Указание: ввод коэффициентов осуществлять с клавиатуры, начиная со старшего коэффициента; операцию возведения в степень не использовать.
6.27	Даны значения двух вещественных массивов x и y размера n. Рассматривая пары значений $(x_1;\ y_1), (x_2;\ y_2), \ldots, (x_n;\ y_n)$ как координаты точек плоскости, найти номера двух наиболее удаленных друг от друга точек и расстояние между ними.
6.28	Даны значения двух вещественных массивов x и y размера n. Рассматривая пары значений $(x_1;\ y_1), (x_2;\ y_2), \ldots, (x_n;\ y_n)$ как координаты точек плоскости, найти номера трёх точек, которые являются вершинами треугольника наибольшей площади и указать площадь этого треугольника.
6.29	Коэффициенты многочлена $a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0$ можно разместить в массиве размера n: $a_0, a_1 \ldots a_{n-1}, a_n$. Даны два многочлена, заданные своими коэффициентами. Найти коэффициенты многочлена, равного произведению двух данных многочленов.
6.30	Дан целочисленный массив x из n элементов. Найти наиболее длинную последовательность подряд идущих элементов, которая является палиндромом. Палиндром – это последовательность, которая читается одинаково с начала и с конца.
6.31	В линейном вещественном массиве найти самую длинную цепочку (длины более 1) подряд идущих элементов, являющихся членами арифметической прогрессии. Если таких цепочек одинаковой длины несколько, указать их все.

6.32	Даны два массива по n элементов: массив строк S и целочисленный массив P, причём массив P заполнен всеми различными числами от 1 до n, расположенными в случайном порядке. Считаем значения P[i] позициями, по которым должны расположиться в своём массива соответствующие строки S[i]. Разместить строки в указанных позициях. Ограничение: дополнительными массивами не пользоваться! Пример. Вход: n=4 Иван Пётр Николай Савва 3 1 4 2 Выход: Пётр Савва Иван Николай.

Лабораторная работа № 7. Матрицы

Цель работы

Изучить способы определения, инициализации и алгоритмы обработки двумерных массивов (матриц).

Необходимый теоретический материал

Для решения задач лабораторной работы № 7 необходимо изучить следующий теоретический материал:

- фиксированные и динамические матрицы;
- типовые алгоритмы обработки матриц.

Примеры

Пример 7.1.

Написать и отладить программу, решающую следующую задачу: *Матрица размера m x n заполнена целыми числами. Найти сумму максимальных элементов строк матрицы.*

```
/*
Пример 7.1.
Матрица размера m x n заполнена целыми числами. Найти сумму максимальных элементов строк матрицы.
*/

#include <iostream>
#include <iomanip>
#include <locale>
#include <ctime>

using namespace std;

void newMatrix(int**& a, int m, int n)
{
   a = new int* [m];
   for (int i = 0; i < m; i++)
      a[i] = new int[n];
}

// Заполнение матрицы случайными числами
void randomMatrix(int** a, const int M, const int N)
{
```

```
   srand(time(NULL));
   for (int i = 0; i < M; i++)
      for (int j = 0; j < N; j++)
         a[i][j] = rand() % 20;
}

// Вывод матрицы на экран
void printMatrix(int** a, const int M, const int N)
{
   for (int i = 0; i < M; i++)
   {
      for (int j = 0; j < N; j++)
         cout << setw(4) << a[i][j];
      cout << endl;
   }
}

// Нахождение суммы максимальных элементов строк матрицы
int sumMaxRows(int** a, const int M, const int N)
{
   int s = 0;
   for (int i = 0; i < M; i++)
   {
      int max = a[i][0];
      for (int j = 1; j < N; j++)
         if (a[i][j] > max)
            max = a[i][j];
      s += max;
   }
   return s;
}

int main()
{
   setlocale(LC_ALL, "Russian");

   // Динамическая матрица
   int m = 7, n = 5;
   int** a;
   newMatrix(a, m, n);
   randomMatrix(a, m, n);
   cout << "Случайная матрица:" << endl;
   printMatrix(a, m, n);
   cout << "Сумма максимальных элементов строк равна "
        << sumMaxRows(a, m, n) << endl;
```

```
   system("pause");
   return 0;
}
```

Изображение консольного окна программы из примера 7.1.

```
C:\Example 7.1.exe
Случайная матрица
   4  30  30   0  43  89  53
  41  95  56  10  24  47  95
  58  49  62  20  61  74  72
  67  30  54  30  79  47  81
  96  70  77  87  20  43  10
Максимум в строке 0 = 89
Максимум в строке 1 = 95
Максимум в строке 2 = 74
Максимум в строке 3 = 81
Максимум в строке 4 = 96
Сумма максимальных элементов строк равна 435
Для продолжения нажмите любую клавишу . . .
```

Пример 7.2.

Написать и отладить программу, решающую следующую задачу: *Даны натуральные числа m и n. Прямоугольную матрицу размера m x n заполнить последовательными целыми числами от 0 до (m * n – 1), двигаясь от левого верхнего угла по спирали, закручивающейся внутрь по часовой стрелке.*

```
/*
Пример 7.2.
Даны натуральные числа m и n. Прямоугольную матрицу размера m x
n заполнить последовательными целыми числами от 0 до (m * n -
1), двигаясь  от левого верхнего угла по спирали, закручивающей-
ся внутрь по часовой стрелке.
*/

#include <iostream>
#include <iomanip>
#include <locale.h>
```

```
using namespace std;

// Создание матрицы a размера m строк n столбцов
// в динамической памяти
void newMatrix(int ** &a, int m, int n)
{
   a = new int*[m];
   for (int i = 0; i < m; i++)
      a[i] = new int[n];
}

// Заполнение матрицы a размера m на n последовательными числами
// от 0 до (m * n - 1) по спирали
void spiralMatrix(int ** a, int m, int n)
{
   int top = 0;        // верхняя граница витка
   int bottom = m - 1; // нижняя граница витка
   int left = 0;       // левая граница витка
   int right = n - 1;  // правая граница витка
   int k = 0;          // заполняющие числа
   do
   {
      // заполнение верхней строны витка
      for (int j = left; j <= right; j++)
         a[top][j] = k++;

      // заполнение правой стороны витка
      for (int i = top + 1; i <= bottom; i++)
         a[i][right] = k++;

      // заполнение нижней стороны витка
      if (top < bottom)
         for (int j = right - 1; j >= left; j--)
            a[bottom][j] = k++;

      // заполнение левой стороны витка
      if (left < right)
         for (int i = bottom - 1; i >= top + 1; i--)
            a[i][left] = k++;

      // сдвиг границ к следующему витку
      top++;
      bottom--;
      left++;
      right--;
   } while (top <= bottom && left <= right);
```

```
}

// Вывод матрицы a на экран
void printMatrix(int ** a, int m, int n)
{
   for (int i = 0; i < m; i++)
   {
      for (int j = 0; j < n; j++)
         cout << setw(4) << a[i][j];
      cout << endl;
   }
}

int main()
{
   setlocale(LC_ALL, "Russian");

   // Ввод размеров матрицы
   int m, n;
   do
   {
      cout << "Введите размеры матрицы m и n" << endl;
      cin >> m >> n;
   } while (m < 1 || n < 1);

   // Создание матрицы в динамической памяти
   int** a = NULL;
   newMatrix(a, m, n);

   // Заполнение матриы по спирали
   spiralMatrix(a, m, n);

   // Вывод полученной матрицы
   printMatrix(a, m, n);

   system("pause");
   return 0;
}
```

Изображение консольного окна программы из примера 7.2.

```
C:\Example 7.2.exe
Введите размеры матрицы
5 7
Матрица, заполненная по спирали:
```

```
  0   1   2   3   4   5   6
 19  20  21  22  23  24   7
 18  31  32  33  34  25   8
 17  30  29  28  27  26   9
 16  15  14  13  12  11  10
Для продолжения нажмите любую клавишу . . .
```

Таблица тестов для программы из примера 7.2

№ теста	Вход		Выход
	m	n	
1.	5	7	0 1 2 3 4 5 6 19 20 21 22 23 24 7 18 31 32 33 34 25 8 17 30 29 28 27 26 9 16 15 14 13 12 11 10
2.	1	10	0 1 2 3 4 5 6 7 8 9
3.	1	1	0

Пример 7.3.

Написать и отладить программу, решающую следующую задачу: *Схема лабиринта задается прямоугольной матрицей: если в ячейке матрицы записано число 0, то это свободная ячейка (проход), по ней можно пройти; если число 1 – то это стенка, через неё пройти нельзя. В лабиринте можно перемещаться из одной свободной ячейки в другую свободную ячейку, если они стыкуются по общей стороне. Дана прямоугольная матрица размера m x n (размер до 10 x 10), задающая лабиринт; даны координаты одной из ячеек матрицы. Указать все свободные ячейки лабиринта, в которые можно попасть из данной ячейки.*

```
/*
Пример 7.3.
Схема лабиринта задается прямоугольной матрицей: если в ячейке
матрицы записано число 0, то это свободная ячейка (проход), по
ней можно пройти; если число 1 - то это стенка, через неё пройти
нельзя. В лабиринте можно перемещаться из одной свободной ячейки
в другую свободную ячейку, если они стыкуются по общей стороне.
Дана прямоугольная матрица размера m x n (размер до 10 x 10),
```

```
задающая лабиринт; даны координаты одной из ячеек матрицы. Указать все свободные ячейки лабиринта, в которые можно попасть из данной ячейки. */
#include <iostream>
#include <iomanip>
#include <locale.h>

using namespace std;

// Создание матрицы a размера m строк n столбцов
// в динамической памяти
void newMatrix(int ** &a, int m, int n)
{
   a = new int*[m];
   for (int i = 0; i < m; i++)
      a[i] = new int[n];
}

// Заполнение матрицы случайными числами 0 (свободная ячейка)
// и 1 (стенка)
void createLabirint(int** a, int m, int n)
{
   srand(time(NULL));
   for (int i = 0; i < m; i++)
      for (int j = 0; j < n; j++)
         a[i][j] = rand() % 2;
}

// Вывод лабиринта на экран
void printLabirint(int** a, int m, int n)
{
   cout << "  ";
   for (int j = 0; j < n; j++)
      cout << j;
   cout << endl;
   for (int i = 0; i < m; i++)
   {
      cout << i << ")";
      for (int j = 0; j < n; j++)
         switch (a[i][j])
         {
         case 0:             // свободная ячейка
            cout << " "; break;
         case 1:             // стенка
            cout << "#"; break;
         case 2:             // свободная достижимая ячейка
```

```
            cout << "+"; break;
         }
      cout << endl;
   }
}

// Функция проверяет, существует ли в матрице размером m x n
// элемент с координатами (x; y)
bool valid(int x, int y, int m, int n)
{
   return (0 <= x && x < m && 0 <= y && y < n);
}

// Нахождение всех доступных ячеек лабиринта a
// из ячейки a[xStart][yStart]
void findAccessible(int ** a, int xStart, int yStart, int m, int
n)
{
   struct Coordinates {
      int x, y;
   };
   Coordinates* stack = new Coordinates[m * n];
   int stackIndex = -1;
   int x1, y1;

   if (a[xStart][yStart] == 0)
   {
      stack[++stackIndex].x = xStart;
      stack[stackIndex].y = yStart;
      while (stackIndex >= 0)
      {
         x1 = stack[stackIndex].x;
         y1 = stack[stackIndex].y;
         stackIndex--;
         a[x1][y1] = 2;   // пометка, что ячейка достижима
         if (valid(x1 - 1, y1, m, n) && a[x1 - 1][y1] == 0)
         {
            stack[++stackIndex].x = x1 - 1;
            stack[stackIndex].y = y1;
         }
         if (valid(x1, y1 - 1, m, n) && a[x1][y1 - 1] == 0)
         {
            stack[++stackIndex].x = x1;
            stack[stackIndex].y = y1 - 1;
         }
         if (valid(x1, y1 + 1, m, n) && a[x1][y1 + 1] == 0)
```

```
         {
            stack[++stackIndex].x = x1;
            stack[stackIndex].y = y1 + 1;
         }
         if (valid(x1 + 1, y1, m, n) && a[x1 + 1][y1] == 0)
         {
            stack[++stackIndex].x = x1 + 1;
            stack[stackIndex].y = y1;
         }
      }
   }
}

int main()
{
   setlocale(LC_ALL, "Russian");

   // Создание матрицы-лабиринта в динамической памяти
   int** a = NULL;
   int m = 7, n = 7;
   newMatrix(a, m, n);

   // Создание случайного лабиринта
   createLabirint(a, m, n);

   // Вывод исходного лабиринта
   cout << "Исходный лабиринт:" << endl;
   printLabirint(a, m, n);

   // Ввод начальной ячейки путей в лабиринте
   int xStart, yStart;
   do
   {
      cout << "Введите координаты ячейки матрицы, "
           << "от которой ищем все пути" << endl;
      cin >> xStart >> yStart;
   } while (!valid(xStart, yStart, m, n));

   // Нахождение всех доступных ячеек из (xStart; yStart)
   findAccessible(a, xStart, yStart, m, n);

   // Вывод лабиринта с доступными из (xStart; yStart) ячейками
   cout << "Достижимые ячейки обозначены знаком +" << endl;
   printLabirint(a, m, n);

   system("pause");
```

```
    return 0;
}
```

Изображение консольного окна программы из примера 7.3.

```
C:\Example 7.3.exe
Исходный лабиринт
  0123456
0)  # #
1)#  # ##
2)#     #
3)   ###
4)# # #
5)   ## #
6)#### ##
Введите координаты ячейки лабиринта, от которой ищем все пути
3 0
Достижимые ячейки обозначены знаком +
  0123456
0)++# #
1)#++#+##
2)#++++#
3)+++###
4)#+# #
5)+++## #
6)#### ##
Для продолжения нажмите любую клавишу . . .
```

Варианты задач лабораторной работы № 7

Указания. В решениях предложенных задач необходимо использовать двумерные массивы – матрицы. Желательно использовать динамические матрицы с задаваемым во время исполнения программы размером.

Как правило, матрица заполняется случайными числами; в тех задачах, когда необходимо проверить «специальные случаи» исходных данных, можно матрице задавать конкретные значения.

№	Текст задачи
7.1	Дана квадратная целочисленная матрица. Проверить, является ли она симметричной относительно главной диагонали и относительно побочной диагонали.
7.2	Дана квадратная целочисленная матрица. Найти максимальный элемент среди элементов выше главной диагонали, и минимальный элемент среди элементов ниже главной диагонали.
7.3	Дана квадратная целочисленная матрица. Найти сумму элементов, расположенных выше побочной диагонали, и максимальное значение среди элементов, расположенных ниже побочной диагонали.
7.4	Дана прямоугольная целочисленная матрица размера m строк и n столбцов. Среди сумм элементов строк матрицы найти наибольшую.
7.5	Дана прямоугольная целочисленная матрица размера m строк и n столбцов. Найти в каждом столбце матрицы минимальный элемент и среди таких элементов найти максимальный, указать его координаты.
7.6	Дана прямоугольная целочисленная матрица размера m строк и n столбцов. Найти в каждой строке матрицы максимальный элемент и среди таких элементов найти минимальный, указать его координаты.
7.7	Дана прямоугольная целочисленная матрица размера m строк и n столбцов. Найти среди строк матрицы палиндромы, то есть строки, элементы которых читаются одинаково с начала и с конца.
7.8	Дана прямоугольная целочисленная матрица размера m строк и n столбцов. Найти среди столбцов матрицы палиндромы, то есть такие столбцы, элементы которых читаются одинаково сверху вниз и снизу вверх.
7.9	Дана прямоугольная целочисленная матрица размера m строк и n столбцов. Произвести циклический сдвиг элементов внешнего слоя матрицы по часовой стрелке на одну позицию.
7.10	Дана прямоугольная вещественная матрица размера m строк и n столбцов. Каждую строку матрицы сдвинуть циклически влево так, чтобы её минимальный элемент оказался первым в строке.
7.11	Даны две вещественные квадратные матрицы A и B размера $n \times n$. Найти произведение этих матриц $C = A \times B$, в котором каждый элемент c_{ij} матрицы C равен $c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}$.

7.12	Дана прямоугольная вещественная матрица размера m строк и n столбцов. Найти все пары одинаковых строк матрицы.
7.13	Дана прямоугольная вещественная матрица размера m строк и n столбцов. Каждую строку матрицы сдвинуть циклически вправо так, чтобы её максимальный элемент оказался первым в строке.
7.14	Дана квадратная целочисленная матрица размера не менее 3 строки на 3 столбца. Произвести циклический сдвиг элементов внешнего слоя матрицы по часовой стрелке, а элементов следующего за ним слоя – против часовой стрелки.
7.15	Дана квадратная целочисленная матрица размера $m \times n$. Произвести разворот матрицы на 90 градусов по часовой стрелке.
7.16	Дана прямоугольная вещественная матрица размера m строк и n столбцов. Каждый столбец матрицы сдвинуть циклически вверх так, чтобы его минимальный элемент оказался верхним в столбце.
7.17	Соседом элемента матрицы a_{ij} называется такой другой её элемент, индексы которого отличаются от каждого из индексов i и j не более чем на 1. Дана прямоугольная вещественная матрица A размера m строк и n столбцов. Построить матрицу B такого же размера, чтобы каждый её элемент b_{ij} равнялся наименьшему из соседей элемента a_{ij}.
7.18	Дана прямоугольная вещественная матрица A размера m строк и n столбцов. Построить матрицу B такого же размера, чтобы каждый её элемент b_{ij} равнялся наименьшему из элементов матрицы A, за исключением элементов, находящихся в том же столбце и в той же строке.
7.19	Дана целочисленная квадратная матрица размера $n \times n$. Проверить, является ли эта матрица магическим квадратом, то есть такой, в которой суммы элементов во всех строках и во всех столбцах совпадают.
7.20	Дана целочисленная квадратная матрица размера $n \times n$. Проверить, является ли эта матрица латинским квадратом, то есть такой, в которой каждая строка и каждый столбец содержат числа от 1 до n.
7.21	Элемент матрицы называется седловой точкой, если он является наименьшим в строке и одновременно наибольшим в столбце, либо, наоборот, является наибольшим в строке и одновременно наименьшим в столбце. Для данной вещественной матрицы размером $m \times n$ указать индексы всех седловых точек.

7.22	Дано натуральные числа n и k, два целочисленных линейных массива x и y по k элементов в каждом, заполненных числами из диапазона от 1 до n. Рассматриваем пары значений $(x_1;\ y_1), (x_2;\ y_2), \dots, (x_k;\ y_k)$ как координаты клеток шахматной доски размера $n \times n$, на которых стоят шахматные ферзи. Указать координаты тех клеток этой шахматной доски, которые не находятся под боем ни одного ферзя, вывести шахматную доску на экран с указанием нахождения ферзей и клеток «под боем».
7.23	Поле игры «Сапёр». Дано натуральное число n (n >=2). В квадратной целочисленной матрице размера n x n случайным образом расставлены числа -1, которые называются «мины», количество мин также случайное число из диапазона [n . . 2n]. В каждую из оставшихся клеток матрицы записать число, равное количеству мин в соседних клетках. Клетка называется соседней, если она граничит с данной по одной из сторон или углов. Усложнение: реализовать игровую программу «Сапёр».
7.24	Дана целочисленная матрица A размера $m \times n$. Обозначим $A'(i,j)$ - верхний левый угол матрицы A до -й строки и j-го столбца (подматрица). Каждому элементу исходной матрицы a_{ij} присвоить наименьшее значение среди всех элементов его подматрицы $A'(i,j)$.
7.25	Дана целочисленная матрица A размера $m \times n$. Обозначим $A'(i,j)$ - верхний левый угол матрицы A до -й строки и j-го столбца (подматрица). Каждому элементу исходной матрицы a_{ij} присвоить значение суммы всех элементов его подматрицы $A'(i,j)$.
7.26	Дана прямоугольная целочисленная матрица размера m строк и n столбцов. Последовательность элементов матрицы, расположенных по чётным строкам слева направо, по нечётным строкам справа налево, (строки рассматриваем последовательно сверху вниз) назовем «змейкой». Сдвинуть циклически элементы змейки на одну позицию к концу.
7.27	Дана прямоугольная целочисленная матрица размера m строк и n столбцов (m и n не меньше 3). *Пятном* в матрице называем подматрицу размером 3×3, заполненную нулями. Поставить в исходную матрицу такое пятно, чтобы сумма оставшихся в матрице ненулевых элементов была бы минимальна.
7.28	Дана прямоугольная вещественная матрица размера m строк и n столбцов. Упорядочить столбцы матрицы по не возрастанию их нижних элементов

7.29	Дана прямоугольная вещественная матрица размера m строк и n столбцов. Упорядочить строки матрицы по не убыванию их первых слева элементов.
7.30	Дана целочисленная квадратная матрица размера $n \times n$, заполненная числами 0 или 1; также даны два целых неотрицательных числа x и y, не превосходящие n. Рассматриваем матрицу как шахматную доску, в клетке которой с координатами $(x; y)$ находится белый шахматный ферзь. Считаем, что если клетка матрицы равна 1, то в ней находится какая-то чёрная фигура, если 0 – то клетка пуста. Определить, сколько чёрных фигур находится под боем белого ферзя.
7.31	Дана целочисленная квадратная матрица размера $n \times n$, заполненная случайными числами, также даны два целых неотрицательных числа x и y, не превосходящие n. Рассматриваем матрицу как шахматную доску, в клетке которой с координатами $(x; y)$ находится шахматный конь. Конь может сделать путь *в два хода* из своей исходной клетки, пройдя, таким образом, по двум клеткам доски (исходную клетку не считаем). Найти такой путь коня, чтобы сумма чисел в этих двух клетках была максимальна.
7.32	Игра «Судоку». Поле игры «Судоку» представляет собой квадратную матрицу размера 9 x 9, в каждой клетке которой находится одно из чисел от 1 до 9, при этом в каждой строке, в каждом столбце и каждом малом квадрате 3 x 3 цифры не повторяются. Игроку видны в начале игры только некоторые из чисел. Цель игры - открыть всё поле. За каждый ход игрока он открывает по одной клетке, в случае ошибки игра считается проигранной. Уровень 1: написать программу, которая генерирует случайные исходные значения игрового поля. Уровень 2: написать программу, которая реализует игру «Судоку» для игры с человеком. Уровень 3: написать программу, алгоритм которой самостоятельно ищет решения последовательного открытия клеток игрового поля.

Лабораторная работа № 8. Строки

Цель работы

Изучить свойства строковых типов данных в языках C и C++, их особенности и различия. Рассмотреть применение алгоритмов с использованием строкового типа данных.

Необходимый теоретический материал

Для решения задач лабораторной работы № 8 необходимо изучить следующий теоретический материал:

- строковый тип данных языка C (библиотека string.h или cstring), стандартные операции со строками;
- строковый тип данных языка C++ (класс string, библиотека string), стандартные операции со строками;
- символьный тип данных char;
- типовые алгоритмы со строками.

Примеры

Пример 8.1.

Написать и отладить программу, решающую следующую задачу: *Дана строка символов. Проверить, сбалансирована ли эта строка по круглым скобкам? Строка считается сбалансированной, если для любой открывающей круглой скобки далее в строке имеется закрывающая скобка, при этом фрагмент строки между ними также сбалансирован по скобкам. При отсутствии скобок строка также считается сбалансированной.*

Предлагаемое решение выполнено на языке C с использованием строк C-style.

```
/*
Пример 8.1.
Дана строка символов. Проверить, сбалансирована ли эта строка по
круглым скобкам? Строка считается сбалансированной, если для
любой открывающей круглой скобки далее в строке имеется закрыва-
ющая скобка, при этом фрагмент строки между ними также сбаланси-
рован по скобкам. При отсутствии скобок строка также считается
сбалансированной.
*/
```

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

// Функция проверяет, сбалансирована ли по скобкам строка s
bool strBalance(char s[])
{
   int count = 0;  // счетчик открытых скобок
   int i = 0;
   while (s[i] && (count >= 0))
   {
      if (s[i] == '(')
         count++;
      if (s[i] == ')')
         count--;
      i++;
   }
   return count == 0;
}

int main()
{
   setlocale(LC_ALL, "Russian");

   printf("Введите символьную строку\n");
   char s[256];
   gets_s(s);

   if (strBalance(s))
      printf("Строка сбалансирована по скобкам\n");
   else
      printf("Строка не сбалансирована по скобкам\n");

   system("pause");
   return 0;
}
```

Изображение консольного окна программы из примера 8.1.

```
C:\Example 8.1.exe
Введите символьную строку
(sqrt(pow(a, 2) + pow(b, 2))) * (a + b) / exp(2 * log(c))
Строка сбалансирована по скобками
```

```
Для продолжения нажмите любую клавишу . . .
```

Таблица тестов для программы из примера 8.1

№ теста	Вход	Выход
1.	(sqrt(pow(a, 2) + pow(b, 2))) * (a + b) / exp(2 * log(c))	Строка сбалансирована по скобкам
2.	;)(-%	Строка не сбалансирована по скобкам
3.	Uno, dos, tres, cuatro	Строка сбалансирована по скобкам

Пример 8.2.

Написать и отладить программу, решающую следующую задачу: *В строке словом называется последовательность латинских букв, не прерываемая другими символами. Дана строка символов. Найти в строке слово, имеющее наибольшую длину.*

```
/*
Пример 8.2.
В строке словом называется последовательность латинских букв, не
прерываемая другими символами. Дана строка символов. Найти в
строке слово, имеющее наибольшую длину.
*/
#include <iostream>
#include <locale.h>
#include <string>

using namespace std;

// Функция возвращает самое длинное слово из строки s
string maxWordStr(string s)
{
   int i = 0;
   string maxWord = "";
   while (i < s.length())
   {
      // Пропуск не-букв
      while (i < s.length() && !isalpha(s.at(i)))
         i++;
      // Выделение слова
```

```
      string word = "";
      while (i < s.length() && isalpha(s.at(i)))
         word += s.at(i++);
      // Анализ слова
      if (word.length() > maxWord.length())
         maxWord = word;
   }
   return maxWord;
}

int main()
{
   setlocale(LC_ALL, "Russian");

   string s("");
   cout << "Введите строку символов" << endl;
   getline(cin, s);

   string maxWord = maxWordStr(s);

   if (maxWord.length() > 0)
      cout << "Слово, имеющее наибольшую длину = " << maxWord
<< endl;
   else
      cout << "В строке нет слов..." << endl;

   system("pause");
   return 0;
}
```

Изображение консольного окна программы из примера 8.2.

```
C:\Example 8.2.exe
Введите строку символов
 London is a capital of Great+Britain...!
Слово, имеющее наибольшую длину = capital
Для продолжения нажмите любую клавишу . . .
```

Таблица тестов для программы из примера 8.2

№ теста	Вход	Выход
1.	a10b11c12	Слово, имеющее наибольшую длину = a
2.	SELECT AVG(MARK) AS AVERAGE FROM STUDENTS;	Слово, имеющее наибольшую длину = STUDENTS
3.	2 + 2 = 4	В строке нет слов...

Пример 8.3.

Написать и отладить программу, решающую следующую задачу: *Расстоянием между двумя словами одинаковой длины назовём количество несовпадающих у них символов на одинаковых позициях. Например, между словами "LIKE" и "LOVE" расстояние равно 2. В первой строке входного файла расположено контрольное слово, а во второй строке – текст, содержащий слова (в словах используется только латиница). Найти в тексте слово, имеющее самое малое расстояние до контрольного слова и указать это расстояние.*

```
/*
Пример 8.3.
Расстоянием между двумя словами одинаковой длины назовём количество несовпадающих у них символов на одинаковых позициях. Например, между словами "LIKE" и "LOVE" расстояние равно 2.
В первой строке входного файла расположено контрольное слово, а во второй строке - текст, содержащий слова (в словах используется только латиница). Найти в тексте слово, имеющее самое малое расстояние до контрольного слова и указать это расстояние.
*/
#include <iostream>
#include <locale.h>
#include <string>
#include <fstream>
using namespace std;

void readFile(string fileName, string &controlWord, string
&textString)
{
   // Открытие файла для чтения
   ifstream inputFile(fileName, ios::in);
   if (!inputFile)
   {
      cerr << "Файл невозможно открыть" << endl;
      exit(1);
```

```
   }

   // Чтение первой строки с контрольным словом
   inputFile >> controlWord;
   getline(inputFile, textString);

   // Чтение второй строки с текстом
   getline(inputFile, textString);

   inputFile.close();
}

void findNearestWord(string controlWord, string textString,
string &nearestWord, int &minDistance)
{
   minDistance = INT_MAX;
   int i = 0;
   while (i < textString.length())
   {
      // Пропуск не-букв
      while (i <textString.length()&&!isalpha(textString.at(i)))
         i++;

      // Выделение очередного слова строки
      string word = "";
      while (i < textString.length()&&isalpha(textString.at(i)))
         word += textString.at(i++);

      // Анализ слова: расстояние между word и controlWord
      if (word.length() == controlWord.length())
      {
         // Расчет расстояния
         int distance = 0;
         for (int j = 0; j < word.length(); j++)
            if (word.at(j) != controlWord.at(j))
               distance++;

         // Поиск наименьшего расстояния
         if (distance < minDistance)
         {
            nearestWord = word;
            minDistance = distance;
         }
      }
   }
}

int main()
```

```
{
   setlocale(LC_ALL, "Russian");

   string controlWord = "", textString = "";
   readFile("InputText.txt", controlWord, textString);
   cout << "Контрольное слово - " << controlWord << endl;
   cout << "Текст - " << textString << endl;

   // Обработка строки - поиск самого близкого слова
   string nearestWord; // самое близкое слово в textString
   int minDistance; // расстояние от controlWord до nearestWord
   findNearestWord(controlWord, textString, nearestWord,
      minDistance);

   if (nearestWord != "")
   {
      cout << "К слову " << controlWord
         << " наиболее близко слово " << nearestWord << endl;
      cout << "Расстояние равно " << minDistance << endl;
   }
   else
      cout << "Нет слов для сравнения" << endl;

   system("pause");
   return 0;
}
```

Изображение консольного окна программы из примера 8.3.

```
C:\Example 8.3.exe
Контрольное слово - nifnif
Текст - nufnuf nafnaf nafnif nifnuf whatis finite
К слову nifnif наиболее близко слово nafnif
Расстояние равно 1
Для продолжения нажмите любую клавишу . . .
```

При этом текстовый файл `InputText.txt` имеет следующее содержимое:

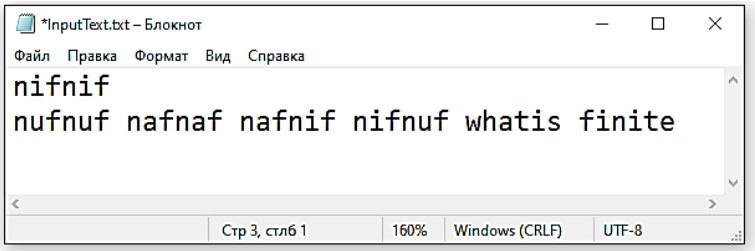

Таблица тестов для программы из примера 8.3 (в колонке «Выход» указаны только искомое слово и расстояние)

№ теста	Вход (текстовый файл)	Выход
1.	nifnif nufnuf nafnaf nafnif nifnuf whatis finite	nafnif 1
2.	BE TO BE OR NOT TO BE	BE 0
3.	IZHEVSK SARAPUL IRKUTSK BARNAUL USTINOV	IRKUTSK 4
4.	C++ JAVA PYTHON PHP GO R	PHP 3
5.	UDSU 1 2 3 4 5	Нет слов для сравнения

Варианты задач лабораторной работы № 8

Указания. В решениях предложенных задач для представления текста необходимо пользоваться строковым типом данных. Рекомендуется использовать строки в стиле языка C++ (класс string), однако допускается использование строк из C (string.h, cstring).

В некоторых задачах входные данные задаются в текстовом файле, в этом случае необходимо для тестирования создать несколько входных файлов. В задачах под словом понимают последовательность букв, не прерываемую другими символами.

№	Текст задачи
8.1	Дана символьная строка. Определить количество слов в этой строке. Словом называется русское слово (только из букв кириллицы) или английское слово (только из букв латиницы).
8.2	Назовем пароль криптостойким, если выполнены 5 критериев: пароль содержит строчные латинские буквы; пароль содержит заглавные латинские буквы; пароль содержит цифры; пароль содержит символы какие-либо из символов ! » # $ % & ‘ ( ) * +; длина пароля не менее 8 символов. Требуется по данному паролю определить, сколько критериев из пяти выполнено.

8.3	Анаграмма. Даны две буквенные строки s1 и s2. Определить, является ли строка s1 анаграммой строки s2? Анаграмма - это строка, полученная из другой строки перестановкой её символов.
8.4	Дана строка символов. Определить, является ли она палиндромом? Палиндром - это строка, которая читается одинаково с начала и с конца.
8.5	Даны две строки символов s1 и s2. Найти количество вхождений строки s2 в строку s1.
8.6	Даны две строки символов s1 и s2. Удалить все вхождения строки s2 в строку s1.
8.7	В двух данных строках заданы два довольно длинных натуральных числа (таких, которые не уместятся в стандартные целые типы данных). Найти и вывести в файл запись нахождения их суммы, выполненной столбиком.
8.8	Дана символьная строка. Найти количество вхождений в неё последнего слова строки.
8.9	Предложением-палиндромом называется такая строка, в которой первое слово совпадает с последним, второе – с предпоследним и так далее. Проверить, является ли входная строка предложением-палиндромом.
8.10	Дана символьная строка. Найти в ней наибольшую подстроку длины более двух, которая является палиндромом. Палиндром – это такая строка, которая одинаково читается как с начала, так и с конца.
8.11	Дана символьная строка. Определить наиболее частое слово этой строки.
8.12	Дана символьная строка. Определить количество различных слов в этой строке и вывести их.
8.13	Дана символьная строка. Удалить из неё все слова длины, равной длине самого короткого слова.
8.14	Во входном текстовом файле располагаются текстовые строки. Найти количество различных слов, встречающихся в тексте.
8.15	Изображение каждой цифры кодируется матрицей 5*5 из единиц и точек. Например, цифре 7 соответствует код 11111 ...1. ..1.. .1... 1....

	Дано натуральное число N. Вывести изображение цифр этого числа последовательно на экран. Изображения цифр числа должны быть в одной «строке» и отделяться друг от друга пробелами!
8.16	Во входном текстовом файле дана строка символов, содержащая слова. Записать в выходной файл слова данной строки, чтобы первая буква каждого слова была в верхнем регистре, все остальные буквы слова в нижнем.
8.17	В каждой строке входного текстового файла располагается логин и пароль пользователя (через пробел); количество пользователей не менее 10. На вход подаются две строки - логин и пароль. Проверить, совпадают ли они с данными из файла.
8.18	Во входном файле расположен текст, содержащий слова и пробелы. Вывести в выходной файл сжатый текст за счет оставления между словами по одному пробелу и удалением пробелов перед первым словом и после последнего слова.
8.19	Во входном текстовом файле даны две строки символов, содержащих слова. Записать в выходной файл слова, содержащиеся в каждой из двух строк.
8.20	В двух данных строках заданы два довольно длинных натуральных числа (таких, которые не уместятся в стандартные целые типы данных). Найти и вывести в файл запись нахождения их произведения, выполненного столбиком.
8.21	Дана строка, в которой записано арифметическое выражение из целых чисел и знаков суммы и разности. Найти значение этого выражения. Пример: вход 123+33-69+0001, выход 88.
8.22	Во входном текстовом файле находится информация о студентах: фамилия, имя и оценка - целое число от 2 до 5 (в отдельных строках). Вывести список студентов в выходной текстовый файл в порядке убывания оценок студентов; информация о каждом студенте - в одной строке.
8.23	Во входном текстовом файле располагаются текстовые строки. Найти сколько раз встречается в тексте каждое слово (частоты слов) и вывести слова и их частоты в выходной файл.
8.24	Во входном текстовом файле даны два фрагмента текста из слов. Фрагменты разделены пустой строкой. Вывести в выходной текстовый файл список слов, имеющихся как в первом, так и во втором фрагментах.

8.25	Во входном файле имеется текст, состоящий из слов, пробелов и точек. Считаем, что точка заканчивает каждое предложение текста. Вывести в выходной файл этот текст, чтобы первое слово предложения начиналось с заглавной буквы, а все остальные буквы были бы строчными.
8.26	Даны две строки символов s1 и s2, а также натуральное число L. Выяснить, если у этих строк общая подстрока длиной L.
8.27	Во входном файле дан текст, в котором есть слова. Вывести в выходной файл такие пары слов, чтобы одно из них можно было получить перестановкой букв другого слова. Например, слова «слово» и «волос» образуют такую пару. Каждая пара – в отдельной строке выходного файла.
8.28	Дана символьная строка. В строке могут встречаться скобки трёх типов: круглые ( ), квадратные [ ] и фигурные { }. Проверить, сбалансирована ли строка по таким скобкам?
8.29	Во входном файле расположен список логинов и паролей пользователей: в каждой строке один логин и один пароль. Строк не менее 10 и не более 20. Программа предлагает либо войти в систему (ввести логин и пароль и проверить их корректность), либо пройти регистрацию (ввести новые логин и пароль, и если такого логина не было, сохранить эту пару в файле).
8.30	Дана строка, в которой записано арифметическое выражение из целых чисел и знаков суммы и произведения. Найти значение этого выражения. Пример: вход 123+12*22*01+5*10, выход 437.
8.31	В строке особую роль играет символ # – он отменяет предыдущий символ. Построить из данной строки s новую строку с учётом роли символа #. Пример: вход ПП#РЮЮ##ИВЕД#Т, выход ПРИВЕТ.
8.32	Во входном файле расположены текстовые строки, в которых есть слова. Вывести в выходной файл этот текст с теми же строками, но выровненными по одной ширине за счет равномерной вставки дополнительных пробелов между словами. Ширину строк взять равной длине самой длинной из входных строк.

Лабораторная работа № 9. Рекурсия

Цель работы

Освоить рекурсивный подход к программированию алгоритмов. Научиться сравнивать и оценивать рекурсивную и итеративную реализации алгоритма.

Необходимый теоретический материал

Для решения задач лабораторной работы № 9 необходимо изучить следующий теоретический материал:

- рекурсивный и итеративный алгоритмы;
- механизм рекурсивного вызова;
- типовые рекурсивные алгоритмы.

Примеры

Пример 9.1.

Написать и отладить программу, решающую следующую задачу: *Дано натуральное число n. Реализовать рекурсивную функцию, возвращающую сумму цифр числа n.*

```
/*
Пример 9.1.
Дано натуральное число n. Реализовать рекурсивную функцию, воз-
вращающую сумму цифр числа n.
*/
#include <iostream>
#include <locale.h>
using namespace std;

// Функция возвращает сумму цифр числа n
int sumDigits(int n)
{
   return n > 0 ? n % 10 + sumDigits(n / 10) : 0;
}

int main()
{
   setlocale(LC_ALL, "Russian");

   int n(0);
```

```
    cout << "Введите натуральное число n = ";
    cin >> n;
    cout << "Сумма цифр числа " << n << " равна "
         << sumDigits(n) << endl;

    system("pause");
}
```

Изображение консольного окна программы из примера 9.1.

```
C:\Example 9.1.exe
Введите натуральное число n = 1234
Сумма цифр числа 1234 равна 10
Для продолжения нажмите любую клавишу . . .
```

Таблица тестов для программы из примера 9.1

№ теста	Вход	Выход
1.	1234	10
2.	1000000000	1
3.	999	27
4.	1	1

Пример 9.2.

Написать и отладить программу, решающую следующую задачу: *Ханойская башня. Древняя легенда гласит, что в храме города Варанаси Буддой были установлены три алмазных стержня - левый, средний и правый. На левом стержне находятся n золотых дисков (по легенде n = 64) разного размера, нанизанные "пирамидкой". Монахи могут перемещать диски с одного стержня на другой, но при этом за один раз можно переносить лишь один диск и ставить его только на диск большего диаметра. В тот момент, когда все диски будут перемещены на правый стержень, под раскаты грома этот мир погибнет. Необходимо указать монахам последовательность переноса дисков, чтобы достичь цели, указанной Буддой.*

```
/*
Пример 9.2.
Ханойская башня.
```

```
Имеется три стержня - левый, средний и правый. На левом
стержне находятся n дисков разного диаметра, размещенные
«пирамидкой». Можно перемещать диски с одного стержня на
другой, но при этом за один раз можно переносить лишь один
диск и ставить его только на диск большего диаметра. Необхо-
димо указать последовательность переноса дисков, чтобы пере-
нести все диски с левого стержня на правый.
*/
#include <iostream>
#include <iomanip>
#include <locale.h>
using namespace std;

// Ханойская башня: перенос n дисков со стержня from
// на стержень to, если стержень buf является вспомогатель-
ным
void Hanoi(int n, char from, char buf, char to)
{
   if (n == 1)
      cout << "Перенеси диск с " << from << " на " << to
           << endl;
   else
   {
      Hanoi(n - 1, from, to, buf);
      cout << "Перенеси диск с " << from << " на " << to
           << endl;
      Hanoi(n - 1, buf, from, to);
   }
}

int main()
{
   setlocale(LC_ALL, "Russian");
   int n;
   cout << "Монах, введи количество дисков:";
   cin >> n;
   cout << "А теперь выполни следующие действия:" << endl;
   Hanoi(n, 'Л', 'С', 'П');
   system("pause");
}
```

Изображение консольного окна программы из примера 9.2.

```
C:\Example 9.2.exe
Монах, введи количество дисков: 3
А теперь выполни следующие действия:
Перенеси диск с Л на П
Перенеси диск с Л на С
Перенеси диск с П на С
Перенеси диск с Л на П
Перенеси диск с С на Л
Перенеси диск с С на П
Перенеси диск с Л на П
Для продолжения нажмите любую клавишу . . .
```

Пример 9.3.

Написать и отладить программу, решающую следующую задачу: *Вводить с клавиатуры последовательность целых числа до 0 (сам 0 не входит в последовательность). Реализовать рекурсивную функцию, возвращающую наибольшее значение из введённых.*

```
/*
Пример 9.3.
Вводить с клавиатуры последовательность целых числа до 0 (сам 0
не входит в последовательность). Реализовать рекурсивную функ-
цию, возвращающую наибольшее значение из введённых.
*/
#include <iostream>
#include <locale.h>
using namespace std;

int findMax()
{
   int x;
   cin >> x;
   if (x)
   {
      int y = findMax();
      return x > y ? x : y;
   }
   else
      return INT_MIN;
}
```

```
int main()
{
   setlocale(LC_ALL, "Russian");
   cout << "Вводите числа до 0" << endl;
   int max = findMax();
   cout << "Максимум = " << max << endl;
   system("pause");
}
```

Изображение консольного окна программы из примера 9.3.

```
C:\Example 9.3.exe
Вводите числа до 0
12 -55 16 49 1 17 0
Максимум = 49
Для продолжения нажмите любую клавишу...
```

Таблица тестов для программы из примера 9.3

№ теста	Вход	Выход
1.	12 -55 16 49 1 17 0	49
2.	-1 -2 -3 -4 0	-1
3.	100 0	100
4.	1 1 1 0	1

Пример 9.4.

Написать и отладить программу, решающую следующую задачу: *Даны натуральные числа n, m (2 <= m <= 16) . Реализовать рекурсивную функцию, возвращающую строковое представление числа n в системе счисления с основанием m.*

```
/*
Пример 9.4.
Даны натуральные числа n, m (2 <= m <=16) . Реализовать рекурсивную функцию, возвращающую строковое представление числа n в системе счисления с основанием m.
*/
#include <iostream>
#include <iomanip>
```

```
#include <locale.h>
#include <string>
using namespace std;

string toM(int n, int m)
{
   if (n)
   {
      int digit = n % m;
      char cdigit = digit < 10 ? char('0' + digit) :
                                 char('A' + digit - 10);
      return toM(n / m, m) + cdigit;
   }
   else
      return "";
}

int main()
{
   int n = 127, m = 2;
   cout << n << "=" << toM(n, m) << "(" << m << ")" << endl;

   system("pause");
}
```

Изображение консольного окна программы из примера 9.4.

```
C:\Example 9.4.exe
127=1111111(2)
Для продолжения нажмите любую клавишу...
```

Таблица тестов для программы из примера 9.4

№ теста	Вход		Выход
	n	m	
1.	127	2	1111111(2)
2.	160	16	A0(16)
3.	65536	8	200000(8)
4.	92	3	10102(3)

Пример 9.5.

Написать и отладить программу, решающую следующую задачу: *Даны координаты клетки шахматной доски (x; y) размера n x n. Обойти все клетки доски ходом шахматного коня, начиная с клетки (x; y) и побывав в каждой клетке ровно один раз.*

Приведенная программа находит первое возможное решение и останавливается. Можно найти все решения, если исключить принудительный останов программы exit(1).

```
/*
Пример 9.5.
Даны координаты клетки шахматной доски (x; y) размера n x n.
Обойти все клетки доски ходом шахматного коня, начиная с клетки
(x; y) и побывав в каждой клетке ровно один раз.
*/
#include <iostream>
#include <iomanip>
using namespace std;

typedef int** Board;
Board board;     // шахматная доска (матрица)
const int n = 8; // размер шахматной доски

// Вывод шахматной доски board на консоль
void PrintBoard()
{
   for(int i = 0; i < n; i++)
   {
      for(int j = 0; j < n; j++)
         cout << setw(4) << board[i][j];
      cout << endl;
   }
   cout << "-------------------------" << endl;
}

// Создание шахматной доски и инициализация нулями
Board NewBoard(int n)
{
   Board b = new int* [n];
   for(int i = 0; i < n; i++)
   {
      b[i] =  new int[n];
      for (int j = 0; j < n; j++)
         b[i][j] = 0;
   }
```

```
   return b;
}

// Рекурсивная функция: попытка поставить в клетку board[x][y]
// коня на шаге номер step и перебор возможных следующих
// ходов из этой клетки
void Knight(int x, int y, int step)
{
   if((0 <= x) && (x < n) && (0 <= y) && (y < n))
   {
      if(board[x][y] == 0)
      {
         board[x][y] = step;
         if (step < n * n)
         {
            Knight(x - 2, y - 1, step + 1);
            Knight(x - 2, y + 1, step + 1);
            Knight(x - 1, y - 2, step + 1);
            Knight(x - 1, y + 2, step + 1);
            Knight(x + 1, y - 2, step + 1);
            Knight(x + 1, y + 2, step + 1);
            Knight(x + 2, y - 1, step + 1);
            Knight(x + 2, y + 1, step + 1);
         }
         else
         {
            PrintBoard();
            exit(1); // Здесь можно system("pause");
         }
         board[x][y] = 0;
      }
   }
}

void main()
{
   setlocale(LC_ALL, "Russian");
   int x0, y0;
   cout << "Введите координаты начальной клетки" << endl;
   cin >> x0 >> y0;
   board = NewBoard(n);
   Knight(x0, y0, 1);
   system("pause");
}
```

Изображение консольного окна программы из примера 9.5.

```
C:\Example 9.5.exe
Введите координаты начальной клетки
2 2
  9   2  11  16   7   4  21  18
 12  15   8   3  20  17  24   5
 27  10   1  14  25   6  19  22
 50  13  26  29  60  23  36  31
 45  28  49  40  35  30  59  64
 54  51  44  61  48  39  32  37
 43  46  53  56  41  34  63  58
 52  55  42  47  62  57  38  33
------------------------
Для продолжения нажмите любую клавишу...
```

Таблица тестов для программы из примера 9.5

№ теста	Вход			Выход
	n	x	y	
1.	5	0	0	1 18 5 10 3 6 11 2 19 14 17 22 13 4 9 12 7 24 15 20 23 16 21 8 25
2.	4	1	1	*нет решений*

Варианты задач лабораторной работы № 9

Указания. Требуется реализовать рекурсивную функцию (функции), возвращающую требуемый результат. В указанной функции использовать циклы не допускается.

№	Текст задачи
9.1	Последовательность чисел Фибоначчи $a_1, a_2, \dots, a_i, \dots$ определяется так: $a_i = \begin{cases} 1, \text{если } i = 1 \text{ или } i = 2 \\ a_{i-1} + a_{i-2}, \text{если } i > 2 \end{cases}$ Реализовать а) рекурсивную функцию, возвращающую a_i б) итеративную функцию, возвращающую a_i. в) функцию, возвращающую a_i, основанную на формуле Бине.

9.2	Дано натуральное число n. Реализовать рекурсивную функцию, возвращающую а) сумму нечетных цифр числа n; б) произведение цифр числа n, находящихся на нечетных позициях, считая справа.
9.3	Дано натуральные числа n, m (2<=m<=10). Реализовать рекурсивную функцию, возвращающую а) максимальную цифру числа n в записи этого числа в m-ичной системе счисления; б) минимальную цифру числа n в в m-ичной системе счисления.
9.4	Дано натуральные числа n, m (2≤m≤16). Реализовать рекурсивную функцию, возвращающую а) сумму цифр числа n в m-ичной системе счисления; б) произведение цифр цифр числа n в m-ичной системе счисления.
9.5	Дано натуральное число n. Реализовать булевскую рекурсивную функцию, возвращающую истину, если запись этого числа является палиндромом, и ложь в противном случае.
9.6	Дан массив из n целых случайных чисел. Реализовать рекурсивную функцию, осуществляющую а) вывод значений массива на экран в прямом порядке; б) вывод значений массива на экран в обратном порядке.
9.7	Дан массив из n целых чисел. Реализовать булевскую рекурсивную функцию, возвращающую истину, если последовательность элементов массива является палиндромом, и ложь в противном случае.
9.8	Дан массив из n целых случайных чисел. Реализовать рекурсивную функцию, возвращающую количество элементов массива, имеющих максимальное значение.
9.9	Дан массив из n целых случайных чисел. Реализовать рекурсивную функцию, вычисляющую индекс и значение минимального элемента массива.
9.10	Дан массив из n действительных случайных чисел. Реализовать рекурсивную функцию, возвращающую среднее арифметическое элементов массива.
9.11	Дан целочисленный массив из n элементов, заполненный нулями и единицами. Реализовать рекурсивную функцию, возвращающую длину самой длинной последовательности подряд идущих единиц.
9.12	Дано натуральное число n. Реализовать рекурсивную функцию, выводящую разложение это число в произведение простых множителей.
9.13	Дано натуральное число n. Реализовать рекурсивную функцию, возвращающую число, получаемое из n выбрасыванием всех нечетных цифр.

9.14	Дано натуральное число n. Реализовать булевскую рекурсивную функцию, возвращающую истину, если десятичная запись цифр этого числа является палиндромом, и ложь – в противном случае.
9.15	На вход подается строка, состоящая только из десятичных цифр. Реализовать рекурсивную функцию, возвращающую целое число, получаемое из его строкового представления.
9.16	Даны два натуральных числа n и m $(m \leq 16)$. Реализовать рекурсивную функцию от двух аргументов n и m, возвращающую строковое представление числа n в -ичной системе счисления. Стандартными функциями перевода не пользоваться.
9.17	На вход подаются две строки s1 и s2. Реализовать рекурсивную функцию, возвращающую индекс первого вхождения подстроки s2 в строку s1, либо значение -1, если таких вхождений нет. Стандартными функциями поиска подстроки не пользоваться.
9.18	Вводить с клавиатуры последовательность целых числа до 0 (сам 0 не входит в последовательность). Реализовать рекурсивную функцию, возвращающую сумму положительных чисел введенной последовательности. Массивами и другими структурами не пользоваться.
9.19	Вводить с клавиатуры последовательность целых числа до 0 (сам 0 не входит в последовательность). Реализовать рекурсивную функцию, возвращающую через параметры значения наибольшего и наименьшего из введённых чисел. Массивами и другими структурами не пользоваться.
9.20	Вводить с клавиатуры последовательность целых числа до 0 (сам 0 не входит в последовательность). Реализовать рекурсивную функцию, возвращающую номер первого из чисел с максимальным значением, вывести результат на консоль. Массивами и другими структурами не пользоваться.
9.21	Вводить с клавиатуры последовательность целых чисел до 0 (сам 0 не входит в последовательность). Реализовать рекурсивную функцию, выводящую введённые числа на экран (в любом порядке). Массивами и другими структурами не пользоваться.
9.22	Вводить с клавиатуры последовательность целых чисел до 0 (сам 0 не входит в последовательность). Реализовать рекурсивную функцию, выводящую положительные числа на консоль (в любом порядке). Массивами и другими структурами не пользоваться.
9.23	Дана случайная целочисленная матрица. Реализовать рекурсивную функцию, выводящую её на консоль (без использования циклов!).

9.24	Дана символьная строка. Реализовать булевскую рекурсивную функцию, возвращающую истину, если строка является палиндромом, и ложь – в противном случае.
9.25	Дана символьная строка. Реализовать рекурсивную функцию, возвращающую из неё новую строку, удалив все пробелы.
9.26	Дана символьная строка, в которой могут быть круглые скобки. Реализовать булевскую рекурсивную функцию, возвращающую истина, если в строке соблюден баланс круглых скобок, и ложь – в противном случае.
9.27	Дано натуральное число n. Реализовать рекурсивную функцию, выводящую все простые числа, не превосходящие n.
9.28	Дано натуральное число r, не превышающее 16. Реализовать рекурсивную функцию, выводящую таблицу умножения в системе счисления с основанием r.
9.29	Дано натуральное число n. Реализовать рекурсивную функцию, выводящую треугольник Паскаля, состоящий из n строк.
9.30	Дана символьная строка. Реализовать рекурсивную функцию, возвращающую самое длинное слово строки.
9.31	Шашки. На стандартной 64-клеточной доске расположена шашечная позиция, состоящая из одной белой шашки и некоторого количества чёрных шашек. Определить, какое максимальное количество чёрных шашек сможет срубить белая шашка? Переход в «дамки» не осуществлять.
9.32	Лабиринт. Дана матрица-лабиринт, заполненная нулями и единицами. Единица называется «стеной», нулик - «комнатой». В лабиринте можно переходить из комнаты в соседнюю комнату, если они соприкасаются сторонами. Определить, можно ли в лабиринте добраться от начальной до конечной клетки. Лабиринт, координаты начальной и конечной клеток задаются.

Приложение: Требования к оформлению программного кода

В этом разделе приведены основные требования и рекомендации по оформлению исходного кода программ при выполнении и сдаче лабораторных задач. Изучить полный перечень требований к оформлению программного кода, в том числе – профессионального, можно будет в последующих дисциплинах.

Требования носят обязательный характер и им необходимо следовать. *Рекомендации* носят характер необязательный, но им следовать настоятельно рекомендуется.

В этом разделе используются следующие условные обозначения.

Примеры правильного оформления программного кода будут помещёны в рамку голубого цвета (при монохромной печати граница отрисована сплошной линией):

```
int sum = 0; // это правильный код
```

Примеры неверного оформления программ или нерекомендуемые для применения конструкции приводятся в рамке красного цвета (при монохромной печати граница пунктирная):

```
i = j = k = 1; // так не рекомендуется!
```

А. Требования к именованию

В программах используется именование различных программных объектов (сущностей) – переменных, констант, функций, классов и т.д. Кроме этого, именуются файлы с исходными кодами программ.

Правильно подобранные имена делают код более читабельным и понятным. Также имена могут дать информацию о том, чем является сущность: константой, переменной, функцией, классом и т.д.

Основные требования к именованию программных объектов:

А1. Имена должны быть осмысленными.

Имя должно не только идентифицировать объекты данных или функции в программе, но и содержать в себе краткое описание смысла и назначения сущности. Имена составляют из одного, двух, реже – трех слов.

Как правило, имена объектов данных (переменных, констант, полей, классов) являются существительными, имена блоков кода (функций, методов) – глаголами.

Допускаются однобуквенные имена, если область видимости таких переменных невелика (несколько строк кода). К переменным с короткой областью видимости относятся индексы массива, указатели, итераторы.

A2. Имена должны быть основаны на английском языке.

Английский язык де-факто стал языком международного общения в IT-сообществе. Кроме этого, синтаксис самых распространённых ЯП основан на английском.

A3. Составные имена должны набираться в стиле CamelCase.

CamelCase («верблюжий» стиль) – это стиль написания фразы из нескольких слов без пробелов, когда каждое слово начинается с заглавной буквы.

lowerCamelCase – это стиль CamelCase, когда первое слово фразы в отличие от остальных начинается со строчной буквы. Так записываются имена *переменных, объектов, полей, методов, функций*.

UpperCaseName – это стиль CamelCase, когда все слова фразы, в том числе первое слово, начинаются с заглавной буквы. Так записываются имена *классов, структур и других типов*.

Именованные константы записываются в верхнем регистре с нижним подчёркиванием в качестве разделителя.

Примеры правильно подобранных имён:

```
int i;                                  //переменная-индекс
int countPositive;                      //переменная
double averageSalaryEmploees;           //переменная
Point xCenter(0), yCenter(0);           //объекты
int getDistance(Point x, Point y); //метод
const int MAX_NUMBER = 1000;            // константа
class ListProducts{};                   // класс
```

Примеры неверно подобранных имён:

```
int abcdef;                    // имя ни о чём не говорит
short int i1, i2, i3, p2, j2, j3, j23; // запутанно!
string imyaFaila;              // лучше fileName
void vivodmatrici();           // лучше printMatrix()
```

```
string input_filename;        // нарушение CamelCase
int A, B, C;                  // нарушение low-
erCamelCase
class listBooks{};            // нарушение Upper-
CamelCase
```

В. Требования к форматированию

Следующий набор правил определяет, как набирать и форматировать программные конструкции, их взаимное расположение. Цель правил – максимальная удобочитаемость текста и, как следствие, быстрое восприятие читающим его человеком смысла и логики программы.

В1. Вложенные конструкции оформляются с помощью отступов (принцип «лесенки»).

Для быстрого понимания этой структуры используется принцип «лесенки», когда вложенная конструкция набирается с отступом относительно конструкции, её содержащей. Лучше всего использовать отступ в два, три или четыре пробела, но размер отступа должен быть единообразным во всей программе.

В2. Каждая конструкция должна находиться (начинаться) в отдельной строке.

Нельзя располагать разные конструкции в одной и той же строке.

К таким конструкциям относятся:

- описания переменных, констант;
- все заголовки (функций, методов, классов);
- операторы;
- операции присваивания;
- вызовы функций и методов;
- директивы препроцессора.

В3. Написание операторов.

Условный оператор:

```
if (условие) {
  операторы;
```

```
}
else{
  операторы;
}
```

Операторы циклов:

```
for (инициализация; условие; модификация) {
  операторы;
}
while (условие) {
  операторы;
}
do {
  операторы;
} while (условие);
```

Оператор выбора:

```
switch (условие) {
  case значение1 :
    операторы;
    break;

  case значение2 :
  case значение3 :
    операторы;
    break;

  default :
    операторы;
}
```

B4. Определение функций, методов.

Определение функций и методов оформляется следующим образом:

```
ТипЗначения nameFunction(параметры)
{
  операторы;
}
```

B5. Пробелы и пустые строки

Следует отбивать пробелами с двух сторон бинарные операции, двоеточие. Пробел ставится после запятой, точки с запятой, зарезервированных слов.

Фрагменты кода, решающие отдельные подзадачи, рекомендуется разделять пустой строкой. В этом случае программа будет выглядеть не сплошным текстом, а как разделённый на логически завершённые фрагменты программный код. Так он лучше читается.

Примеры правильно форматированного кода:

```
if (d > 0)
{
  x1 = (-b - sqrt(d)) / (2 * a);
  x2 = (-b + sqrt(d)) / (2 * a);
}
else
  if (d == 0)
    x1 = -b / (2 * a);
  else
    cout << “no roots” << endl;

// определения переменных с инициализацией
int x = 0;
int y = 0;

// отдельные операторы
while (ptr != nullptr){
  ptr++;
  count++;
}

// определение функции
// функция возвращает максимальное из двух целых значений
int max(int a, int b)
{
```

```
  return a > b ? a : b ;
}

//...
x = (- b + sqrt(d)) / (2 * a); // бинарные операции
```

Примеры неверно форматированного кода:

```
if (a > b) max = a;      // в одной строке два оператора

if (a > b)
if (a > c) max = a;
else max = c;            // не выдержаны отступы

if (a > b)
{ i++; j++;}             // два оператора в строке

x=(-b+sqrt(d))/(2*a);    // нет пробелов в бин.операциях
```

C. Требования к документированию

Исходный текст должен читаться так, чтобы у читающего его человека, как правило – специалиста, не должно остаться неразрешённых вопросов. Этому способствует комментирование программного кода.

Комментарии используются для описания отдельных строк, блоков кода или целого алгоритма. Комментарии должны содержать лишь ту информацию, которая необходима для чтения и понимания программы. Обсуждение нетривиальных вещей или неочевидных решений необходимо, но не нужно описывать то, что и так ясно из кода. Такие избыточные комментарии перегружают текст.

Документирующие комментарии – это особый вид комментариев, которые используются для описания *спецификации кода*, то есть требований к внешнему поведению, которые не зависят от реализации. Обычно документируют классы, функции, методы, интерфейсы и подобные элементы программы. Такие комментарии делаются для разработчиков, которые будут использовать чужие программы, не имея исходного кода.

С1. Оформление комментариев.

Для поясняющего комментария используют две наклонные черты //, после которых обязательно должен следовать пробел.

Обозначения /* и */ обычно применяются для временного отключения фрагментов кода при его отладке; можно использовать для заглавного комментария.

Комментарии блоков кода, спецификации классов, методов и функций размещают перед самим кодом, то есть в строках выше него.

Для отдельного оператора допускается размещение комментария справа от него, если только такой комментарий не слишком велик. Желательно последовательность таких комментариев выравнивать относительно друг друга.

```
// Функция производит поиск в массиве array элемента //
со значением desiredValue и возвращает его индекс.
// В случае отсутствия значения в массиве функция
// возвращает -1.
int findValue(Value* array, Value desiredValue)
{ //...
}
// ...
int countPositiv = 0; // счетчик положительных чисел
list<Book> books;        // список книг библиотеки
moveChess(knight, x, y);      // ход конём в клетку x, y
```

С2. Заглавный комментарий файла должен исчерпывающе описывать его назначение.

В заглавном комментарии (иначе – «шапке») файла должно быть описано, какую часть задачи решает код этого файла. В таком описании можно указать:

- имя файла;
- назначение содержимого файла: объявление или реализация класса, реализация функций, в том числе main(). и т.п.;
- постановка исходной задачи;
- фамилия автора;

- краткая техническая информация о состоянии кода, версии, программных интерфейсах и функциональности приложения;
- и другое.

Например, для учебного приложения, состоящего из одного исходного файла, «шапка» может выглядеть так:

```
// labwork03.cpp
// Лабораторная работа № 3 по предмету
// Практикум по программированию, 1 курс ПИ, 2 семестр
// ---- Задача: Дан случайный целочисленный массив из
// N элементов. Найти в массиве самую длинную цепочку
// подряд идущих положительных элементов -----
// Автор: студент группы ОАБ020302-11 Иванов С.А.
// Дата готовности : 15.10.2022 года
```

С3. Каждую функцию, метод необходимо документировать.

В тексте спецификации функции, метода необходимо (но не избыточно) описать:

- поведение и/или цель функции, метода;
- семантику входных параметров;
- семантику выходных параметров;
- возвращаемое значение;
- исключения, если функция их порождает.

Спецификация функции, то есть точное описание её поведения, помогает разработчику сосредоточиться на соответствии её реализации внешним требованиям. Поэтому спецификацию необходимо оформлять *до начала* написания её кода.

```
// Функция sum возвращает сумму своих вещественных
// параметров. Количество параметров - три, два или
один.
// Вход:
//   a, b, c - вещественные
```

```
// Возвращаемое значение:
//   сумма a + b + c

double sum(double a, double b = 0.0, double c = 0.0)
{
  return a + b + c;
}
```

D. Другие рекомендации

Правила этого раздела будет разумнее отнести не к требованиям по оформлению программного кода, а к рекомендациям по созданию более эффективных и ошибкоустойчивых программ.

D1. Переменные следует объявлять в как можно меньшей области видимости (принцип локализации).

Переменной предоставляется только та область программы, где она нужна и используется. Переменные, не используемые в части программы, но видимые в ней, образуют избыточный балласт, который может привести к ошибкам или неожиданным эффектам.

Из этого правила следует, что необходимо полностью избегать глобальных переменных. Переменные должны быть локализованы в своих функциях, классах или даже блоках. Неявное изменение функцией не принадлежащих её данных является побочным эффектом.

В следующем примере продемонстрирован побочный эффект, возникающий при использовании глобальной переменной.

```
int x;        // глобальная переменная

void inc()
{
  x++;        // побочный эффект
}

void main()
{
  x = 1;      // значение 1
```

```
  inc();
  cout << x << endl; // значение 2 - непонятно изменение
}
```

Для обмена данными между функциями нужно использовать параметры, а не глобальные переменные.

```
void inc(int& x)          // явная передача переменной
                          // через параметр по ссылке
{
  x++;
}

void main()
{
  int x = 1;              // значение 1
  inc(x);                 // явное изменение переменной x
  cout << x << endl;      // значение 2
}
```

D2. Следует инициализировать переменные перед их использованием и не полагаться на значения по умолчанию.

Использование неинициализированных переменных может привести к ошибкам! Правильным будет всегда инициализировать переменные перед использованием.

В языках C и C++ переменные *могут получать* значения по умолчанию, но полагаться на них в алгоритме является плохим стилем. Тем более, от версии к версии языка его механизм встроенной инициализации меняется.

```
int n;
cin >> n;
int sum;       // переменная не инициализирована!
for (int i = 1; i <= n; i++)
  sum += i;
cout << sum << endl;
```

Правильным будет инициализировать переменные перед использованием.

```
int n;
cin >> n;
int sum = 0;        // переменная инициализирована
for (int i = 1; i <= n; i++)
  sum += i;
cout << sum << endl;
```

D3. Программа должна быть разбита на функции, каждая из которых выполняет свою часть работы.

Подпрограммы (функции, процедуры, методы) были созданы в своё время в том числе для того, чтобы избежать дублирования идентичного кода.

Следуя принципам процедурного программирования, необходимо структуру программу создать из совокупности подпрограмм (функций), каждой из которой поручить выполнение своей части работы. Каждая из функций должна иметь понятный интерфейс, то есть чёткое определение того, какую работу функция производит, какие данные у неё на входе и какие данные она выводит (возвращает). Все эти свойства необходимо максимально точно описать в *спецификации функции* – комментария, предшествующего её определению.

D4. Следует избегать «магических» чисел.

«Магические» числа – это явно используемые в программе числовые значения. Когда такое число встречается в программе, то неочевидно, какой смысл оно несёт в её логике. Кроме этого, если от значения числа зависит поведение программы (а так чаще всего и бывает), то для изменения этого поведения придётся искать все вхождения «магического» числа в код и их синхронно править.

```
int a[16];          // 16 - магическое число!
...                 // заполнение массива
int min = 100;      // еще одно магическое число - 100
for (int i = 0; i < 16; i++) // опять 16
  if (a[i] < min)   // поиск min будет работать неверно,
    min = a[i];     //если в массиве все числа больше 100
```

Правильным будет определить и использовать именованные константы:

```
const int n = 16;
int a[n];          // размер массива задан константой
...                // заполнение массива
int min = INT_MAX; // стандартная константа
for (int i = 0; i < n; i++)
  if (a[i] < min)       // поиск min будет работать
       min = a[i];      // правильно
```

Список литературы

1. Страуструп, Б. Язык программирования C++. Специальное издание. Пер. с англ. — М.: Издательство Бином, 2011 г. — 1136 с.
2. Павловская, Т.А. Процедурное и объектно-ориентированное программирование. - СПб.: Питер, 2018. – 496 с.
3. Прата, С. Язык программирования C++ (C++11). Лекции и упражнения, 6-е издание — М.: Диалектика-Вильямс, 2018. — 1248 с.
4. Васильев, А. Н. Программирование на C++ в примерах и задачах — М.: ЭКСМО, 2020. — 368 с. – ISBN 978-5-699-87445-3.
5. Уроки по C++ для начинающих [Электронный ресурс].– Режим доступа: https://ravesli.com/uroki-cpp/.
6. Анисимов, А. Е. Требования и рекомендации по оформлению программного кода на языках C и C++ / А. Е. Анисимов. – Ижевск : Удмуртский государственный университет, 2020. – 47 с. – ISBN 978-5-4312-0787-7.
7. Анисимов, А. Е. Практикум по основам программирования : учебно-методическое пособие / А. Е. Анисимов. – Ижевск : Издательский дом "Удмуртский университет", 2014. – 94 с. – ISBN 978-5-4312-0275-9.
8. Анисимов, А. Е. Сборник заданий по основаниям программирования : учеб. пособие / А. Е. Анисимов, В. В. Пупышев ; А. Е. Анисимов, В. В. Пупышев. – Москва : Интернет-ун-т информ. технологий :, 2006. – 348 с. – (Основы информационных технологий). – ISBN 5-9556-0059-0.
9. Online C++ Compiler [Электронный ресурс]. – Режим доступа: https://www.tutorialspoint.com/compile_cpp_online.php.

Учебное издание

Анисимов Андрей Евгеньевич

ПРАКТИКУМ
ПО ПРОГРАММИРОВАНИЮ НА С++
ЧАСТЬ 1

Учебное пособие

Подписано в печать 15.09.2022. Формат 60x84 $^{1}/_{16}$.
Усл. печ. л. 7,0. Уч.-изд. л. 3,5.
Тираж 53 экз. Заказ № 1547.

Издательство «Удмуртский университет»
426034, Ижевск, ул. Ломоносова, 4Б, каб. 021
Тел./факс: + 7 (3412) 916-364 E-mail: editorial@udsu.ru

Типография Издательского центра
«Удмуртский университет»
426034, Ижевск, ул. Университетская, 1, корп. 2.
Тел. 68-57-18