МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
Учреждение образования
«Витебский государственный технологический университет»



# ОСНОВЫ КОМПЬЮТЕРИЗАЦИИ ТЕХНОЛОГИЙ В СИСТЕМАХ АВТОМАТИКИ. ОСНОВЫ ПРОГРАММИРОВАНИЯ НА АЛГОРИТМИЧЕСКОМ ЯЗЫКЕ

Методические указания по выполнению лабораторных работ
для студентов специальности 1-53 01 01-05 «Автоматизация технологических процессов и производств (легкая промышленность)»
дневной и заочной на базе ссуз форм обучения

Витебск
2021

УДК 681.5:004.9

Составители:

А. С. Соколова, В. Е. Казаков

Рекомендовано к изданию редакционно-издательским советом УО «ВГТУ», протокол № 3 от 30.11.2021.



**Основы компьютеризации технологий в системах автоматики. Основы программирования на алгоритмическом языке** : методические указания по выполнению лабораторных работ / сост. А. С. Соколова, В. Е. Казаков. – Витебск : УО «ВГТУ», 2021. – 80 с.

В методических указания изложены теоретические сведения и практические задания по разделу «Основы программирования на языках высокого уровня» дисциплины «Основы компьютеризации технологий в системах автоматики». Издание предназначено для студентов специальности 1-53 01 01-05 «Автоматизация технологических процессов и производств (легкая промышленность)» дневной и заочной на базе ссуз форм обучения.

**УДК 681.5:004.9**

**СОДЕРЖАНИЕ**

1 СОВРЕМЕННЫЕ СРЕДСТВА ПРОГРАММИРОВАНИЯ ..... 5
1.1 Арифметические действия над числами в позиционных системах счисления ..... 5
1.2 Перевод чисел из одной системы счисления в другую ..... 7
1.3 Представление в ЭВМ чисел ..... 9
ЛАБОРАТОРНАЯ РАБОТА 1. СИСТЕМЫ СЧИСЛЕНИЯ ..... 11
ЛАБОРАТОРНАЯ РАБОТА 2. ПРЕДСТАВЛЕНИЕ ЧИСЕЛ В ЭВМ ..... 13
2 ОСНОВНЫЕ ЭЛЕМЕНТЫ ЯЗЫКА С++ ..... 14
2.1 Идентификаторы ..... 14
2.2 Типы данных ..... 14
2.3 Переменные ..... 15
2.4 Константы ..... 15
2.5 Операции, выражения, операторы ..... 15
2.6 Структура программы ..... 18
2.7 Стандартная математическая библиотека ..... 19
2.8 Комментарии ..... 20
2.9 Ввод-вывод данных на консоль ..... 20
ЛАБОРАТОРНАЯ РАБОТА 3. ЗНАКОМСТВО СО СРЕДОЙ CODE::BLOCKS ..... 26
ЛАБОРАТОРНАЯ РАБОТА 4. ВЫЧИСЛЕНИЕ МАТЕМАТИЧЕСКИХ ВЫРАЖЕНИЙ ..... 28
ЛАБОРАТОРНАЯ РАБОТА 5. ЛИНЕЙНЫЕ АЛГОРИТМЫ ..... 30
3 ОПЕРАТОРЫ ЯЗЫКА С++ ..... 31
3.1 Составной оператор ..... 31
3.2 Условный оператор ..... 32
3.3 Оператор выбора ..... 34
3.4 Операторы цикла ..... 36
3.5 Операторы передачи управления ..... 39
3.6 Рекуррентные вычисления сумм и произведений ..... 40
ЛАБОРАТОРНАЯ РАБОТА 6. РАЗВЕТВЛЯЮЩИЕСЯ АЛГОРИТМЫ ..... 43
ЛАБОРАТОРНАЯ РАБОТА 7. ТАБУЛИРОВАННЫЕ ФУНКЦИИ ..... 44
ЛАБОРАТОРНАЯ РАБОТА 8. ЦИКЛИЧЕСКИЕ АЛГОРИТМЫ ..... 45
ЛАБОРАТОРНАЯ РАБОТА 9. ВЫЧИСЛЕНИЯ С ЗАДАННОЙ ТОЧНОСТЬЮ ..... 45
4 МАССИВЫ В ЯЗЫКЕ С++ ..... 46
4.1 Статические массивы ..... 46
4.2 Динамические массивы ..... 50
ЛАБОРАТОРНАЯ РАБОТА 10. ОДНОМЕРНЫЕ МАССИВЫ ..... 52
ЛАБОРАТОРНАЯ РАБОТА 11. ДВУМЕРНЫЕ МАССИВЫ ..... 53
ЛАБОРАТОРНАЯ РАБОТА 12. ДИНАМИЧЕСКИЕ МАССИВЫ ..... 54
5 ФУНКЦИИ В ЯЗЫКЕ С++ ..... 55
5.1 Общие сведения о функциях ..... 55
5.2 Рекурсия ..... 57

5.3 Массивы и функции ..... 58
ЛАБОРАТОРНАЯ РАБОТА 13. ФУНКЦИИ ..... 60
ЛАБОРАТОРНАЯ РАБОТА 14. ФУНКЦИИ И МАССИВЫ ..... 62
6 ОБРАБОТКА СТРОК В ЯЗЫКЕ С++ ..... 64
ЛАБОРАТОРНАЯ РАБОТА 15. КЛАСС STRING ..... 68
7 СТРУКТУРЫ В ЯЗЫКЕ С++ ..... 69
ЛАБОРАТОРНАЯ РАБОТА 16. СТРУКТУРЫ ..... 71
8 ФАЙЛЫ В ЯЗЫКЕ С++ ..... 72
ЛАБОРАТОРНАЯ РАБОТА 17. ТЕКСТОВЫЕ ФАЙЛЫ ..... 76
ЛИТЕРАТУРА ..... 79

# 1 СОВРЕМЕННЫЕ СРЕДСТВА ПРОГРАММИРОВАНИЯ

## 1.1 Арифметические действия над числами в позиционных системах счисления

Арифметические действия над числами в любой позиционной системе счисления производятся по тем же правилам, что и в десятичной системе. При этом нужно только пользоваться теми таблицами сложения и умножения, которые соответствуют данному основанию *P* системы счисления.

Таблица 1.1 – Таблица сложения чисел в двоичной системе счисления

| + | 0 | 1 |
|---|---|---|
| 0 | 0 | 1 |
| 1 | 1 | 10 |

Таблица 1.2 – Таблица умножения чисел в двоичной системе счисления

| × | 0 | 1 |
|---|---|---|
| 0 | 0 | 0 |
| 1 | 0 | 1 |

Таблица 1.3 – Таблица сложения чисел в восьмеричной системе счисления

| + | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 1 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 10 |
| 2 | 2 | 3 | 4 | 5 | 6 | 7 | 10 | 11 |
| 3 | 3 | 4 | 5 | 6 | 7 | 10 | 11 | 12 |
| 4 | 4 | 5 | 6 | 7 | 10 | 11 | 12 | 13 |
| 5 | 5 | 6 | 7 | 10 | 11 | 12 | 13 | 14 |
| 6 | 6 | 7 | 10 | 11 | 12 | 13 | 14 | 15 |
| 7 | 7 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |

Таблица 1.4 – Таблица умножения чисел в восьмеричной системе счисления

| × | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 2 | 0 | 2 | 4 | 6 | 10 | 12 | 14 | 16 |
| 3 | 0 | 3 | 6 | 11 | 14 | 17 | 22 | 25 |
| 4 | 0 | 4 | 10 | 14 | 20 | 24 | 30 | 34 |
| 5 | 0 | 5 | 12 | 17 | 24 | 31 | 36 | 43 |
| 6 | 0 | 6 | 14 | 22 | 30 | 36 | 44 | 52 |
| 7 | 0 | 7 | 16 | 25 | 34 | 43 | 52 | 61 |

Таблица 1.5 – Таблица сложения чисел в шестнадцатеричной системе счисления

| + | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
| 1 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F | 10 |
| 2 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F | 10 | 11 |
| 3 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F | 10 | 11 | 12 |
| 4 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F | 10 | 11 | 12 | 13 |
| 5 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F | 10 | 11 | 12 | 13 | 14 |
| 6 | 6 | 7 | 8 | 9 | A | B | C | D | E | F | 10 | 11 | 12 | 13 | 14 | 15 |
| 7 | 7 | 8 | 9 | A | B | C | D | E | F | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
| 8 | 8 | 9 | A | B | C | D | E | F | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 |
| 9 | 9 | A | B | C | D | E | F | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 |
| A | A | B | C | D | E | F | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 |
| B | B | C | D | E | F | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 1A |
| C | C | D | E | F | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 1A | 1B |
| D | D | E | F | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 1A | 1B | 1C |
| E | E | F | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 1A | 1B | 1C | 1D |
| F | F | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 1A | 1B | 1C | 1D | 1E |

Таблица 1.6 – Таблица умножения чисел в шестнадцатеричной системе счисления

| × | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
| 2 | 0 | 2 | 4 | 6 | 8 | A | C | E | 10 | 12 | 14 | 16 | 18 | 1A | 1C | 1E |
| 3 | 0 | 3 | 6 | 9 | C | F | 12 | 15 | 18 | 1B | 1E | 21 | 24 | 27 | 2A | 2D |
| 4 | 0 | 4 | 8 | C | 10 | 14 | 18 | 1C | 20 | 24 | 28 | 2C | 30 | 34 | 38 | 3C |
| 5 | 0 | 5 | A | F | 14 | 19 | 1E | 23 | 28 | 2D | 32 | 37 | 3C | 41 | 46 | 4B |
| 6 | 0 | 6 | C | 12 | 18 | 1E | 24 | 2A | 30 | 36 | 3C | 42 | 48 | 4E | 54 | 5A |
| 7 | 0 | 7 | E | 15 | 1C | 23 | 2A | 31 | 38 | 3F | 46 | 4D | 54 | 5B | 62 | 69 |
| 8 | 0 | 8 | 10 | 18 | 20 | 28 | 30 | 38 | 40 | 48 | 50 | 58 | 60 | 68 | 70 | 78 |
| 9 | 0 | 9 | 12 | 1B | 24 | 2D | 36 | 3F | 48 | 51 | 5A | 63 | 6C | 75 | 7E | 87 |
| A | 0 | A | 14 | 1E | 28 | 32 | 3C | 46 | 50 | 5A | 64 | 6E | 78 | 82 | 8C | 96 |
| B | 0 | B | 16 | 21 | 2C | 37 | 42 | 4D | 58 | 63 | 6E | 79 | 84 | 8F | 9A | A5 |
| C | 0 | C | 18 | 24 | 30 | 3C | 48 | 54 | 60 | 6C | 78 | 84 | 90 | 9C | A8 | B4 |
| D | 0 | D | 1A | 27 | 34 | 41 | 4E | 5B | 68 | 75 | 82 | 8F | 9C | A9 | B6 | C3 |
| E | 0 | E | 1C | 2A | 38 | 46 | 54 | 62 | 70 | 7E | 8C | 9A | A8 | B6 | C4 | D2 |
| F | 0 | F | 1E | 2D | 3C | 4B | 5A | 69 | 78 | 87 | 96 | A5 | B4 | C3 | D2 | E1 |

ПРИМЕР. Выполним в восьмеричной системе $1170{,}64 \times 46{,}3$, в шестнадцатеричной системе 3B3,6 + E8B,4 и в двоичной системе 1100000011,011 – 101010111,1:

```
    1170,64
  ×   46,3
  ---------
    35523 4
+ 7324 70
  47432 0
  ---------
  57334,134
```

```
  11
   3B3,6
+  E8B,4
  ------
  123E,A
```

```
  ..........
  1100000011,011
- 1010101111,1
  -------------
  1101011,111
```

## 1.2 Перевод чисел из одной системы счисления в другую

При переводе чисел из десятичной системы счисления в систему с основанием $P > 1$ обычно используют следующий алгоритм:

1. Если переводится целая часть числа, то она делится на $P$, после чего запоминается остаток от деления. Полученное частное вновь делится на $P$, остаток запоминается. Процедура продолжается до тех пор, пока частное не станет равным нулю. Остатки от деления на $P$ записываются в порядке, обратном их получению.

2. Если переводится дробная часть числа, то она умножается на $P$, после чего целая часть запоминается и отбрасывается. Вновь полученная дробная часть умножается на $P$ и т.д. Процедура продолжается до тех пор, пока дробная часть не станет равной нулю. Целые части записываются после запятой в порядке их получения. Результатом может быть либо конечная, либо периодическая двоичная дробь. Поэтому, когда дробь является периодической, приходится обрывать умножение на каком-либо шаге и довольствоваться приближенной записью исходного числа в системе с основанием $P$.

ПРИМЕР. Переведем следующие числа из десятичной системы счисления в двоичную: а) 464; б) 380,1875; в) 115,94 (получим пять знаков после запятой в двоичном представлении).

а)

| | |
|---|---|
| 464 | 0 |
| 232 | 0 |
| 116 | 0 |
| 58 | 0 |
| 29 | 1 |
| 14 | 0 |
| 7 | 1 |
| 3 | 1 |
| 1 | 1 |
| 0 | |

б)

| | |
|---|---|
| 380 | 0 |
| 190 | 0 |
| 95 | 1 |
| 47 | 1 |
| 23 | 1 |
| 11 | 1 |
| 5 | 1 |
| 2 | 0 |
| 1 | 1 |
| 0 | |

| | |
|---|---|
| | 1875 |
| 0 | 3750 |
| 0 | 7500 |
| 1 | 5000 |
| 1 | 0000 |

в)

| | |
|---|---|
| 115 | 1 |
| 57 | 1 |
| 28 | 0 |
| 14 | 0 |
| 7 | 1 |
| 3 | 1 |
| 1 | 1 |
| 0 | |

| | |
|---|---|
| | 94 |
| 1 | 88 |
| 1 | 76 |
| 1 | 52 |
| 1 | 04 |
| 0 | 08 |

Получаем: а) $464_{(10)} = 111010000_{(2)}$; б) $380{,}1875 = 101111100{,}0011_{(2)}$; в) $115{,}94_{(10)} \approx 1110011{,}11110_{(2)}$.

ПРИМЕР. Переведем: а) 459,53125 из десятичной системы счисления в восьмеричную; б) 3425,78175 – в шестнадцатеричную.

а)

| | | | | | |
|---|---|---|---|---|---|
| 459 | 3 | ↑ | | 53125 | |
| 57 | 1 | | ↓ 4 | 25000 | |
| 7 | 7 | | 2 | 00000 | |
| 0 | | | | | |

б)

| | | | | | |
|---|---|---|---|---|---|
| 3425 | 1 | | | 78125 | |
| 214 | 6 | | ↓ (12) C | 50000 | |
| 13 | D (13) | ↑ | 8 | 00000 | |
| 0 | | | | | |

Получаем: а) $459,53125_{(10)} = 713,42_{(8)}$; б) $3425,78125 = D61,C8_{(16)}$.

При переводе чисел из системы счисления с основанием $P$ в десятичную систему счисления необходимо пронумеровать разряды целой части справа налево, начиная с 0, и в дробной части слева направо, начиная с -1. Затем вычислить сумму произведений соответствующих значений разрядов на основание системы счисления в степени, равной номеру разряда.

ПРИМЕР. Переведем в десятичную систему счисления следующие числа: а) $1000011111,0101_{(2)}$; б) $137,04_{(8)}$; в) $29A,5_{(16)}$.

$^{9\ 8\ 7\ 6\ 5\ 4\ 3\ 2\ 1\ 0\ -1\ -2\ -3\ -4}$

а) $1000011111,0101_{(2)} = 1\cdot2^9 + 1\cdot2^4 + 1\cdot2^3 + 1\cdot2^2 + 1\cdot2^1 + 1\cdot2^0 + 1\cdot2^{-2} + 1\cdot2^{-4} =$
$= 512 + 16 + 8 + 4 + 2 + 1 + 0,25 + 0,0625 = 543,3125_{(10)}$.

$^{2\ 1\ 0\ -1\ -2}$

б) $137,04_{(8)} = 1\cdot8^2 + 3\cdot8^1 + 7\cdot8^0 + 4\cdot8^{-2} = 64 + 24 + 7 + 0,0625 = 95,0625_{(10)}$.

$^{2\ 1\ 0\ \ -1}$

в) $29A,5_{(16)} = 2\cdot16^2 + 9\cdot16^1 + 10\cdot16^0 + 5\cdot16^{-1} = 512 + 144 + 10 + 0,3125 =$
$= 666,3125_{(10)}$.

Если необходимо перевести число из двоичной системы счисления в систему счисления, основанием которой является степень двойки, достаточно объединить цифры двоичного числа в группы по столько цифр, каков показатель степени. В целой части производят группировку справа налево, в дробной – слева направо. Если в последней группе недостает цифр, дописываются нули: в целой части – слева, в дробной – справа. Затем каждая группа заменяется соответствующей цифрой новой системы. Соответствия для восьмеричной и шестнадцатеричной систем счисления приведены в таблицах 1.7 и 1.8.

Таблица 1.7 – Соответствие групп двоичных цифр восьмеричным цифрам

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| $P$ | 2 | 000 | 001 | 010 | 011 | 100 | 101 | 110 | 111 |
| | 8 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |

Таблица 1.8 – Соответствие групп двоичных цифр шестнадцатеричным цифрам

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| $P$ | 2 | 0000 | 0001 | 0010 | 0011 | 0100 | 0101 | 0110 | 0111 |
| | 16 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| | 2 | 1000 | 1001 | 1010 | 1011 | 1100 | 1101 | 1110 | 1111 |
| | 16 | 8 | 9 | A | B | C | D | E | F |

ПРИМЕР. Переведем в шестнадцатеричную систему счисления двоичное число 1111010101,11:

$$0011\ 1101\ 0101,1100_{(2)} = 3D5,C_{(16)}.$$

ПРИМЕР. Переведем из восьмеричной в двоичную систему счисления двоичное число 1216,04:

$$1216,04_{(8)} = 001\ 010\ 001\ 110,000\ 100 = 1010001110,0001_{(2)}.$$

### 1.3 Представление в ЭВМ чисел

Чтобы получить внутреннее представление знакового целого положительного числа *N*, хранящегося в *K*-разрядной ячейке, необходимо:

1. Перевести число N в двоичную систему счисления.
2. Полученный результат дополнить слева незначащими нулями до *K* разрядов.

ПРИМЕР. Получим внутреннее представление целого числа 1607 в двухбайтовой ячейке памяти:

$$1607_{(10)} = 11001000111_{(2)}.$$

Внутреннее представление этого числа будет: 0000 0110 0100 0111.

Шестнадцатеричная форма внутреннего представления числа: 0647.

Для представления целого отрицательного числа используется дополнительный код. Чтобы получить дополнительный код целого числа *N*, хранящегося в *K*-разрядной ячейке, необходимо:

1. Получить внутреннее представление модуля числа N (прямой код).
2. В прямом коде заменить 0 на 1 или 1 на 0 (обратный код).
3. К обратному коду прибавить 1 (дополнительный код).

ПРИМЕР. Получим внутреннее представление целого числа -1607 в двухбайтовой ячейке памяти.

Внутреннее представление положительного числа: 0000 0110 0100 0111.

Обратный код: 1111 1001 1011 1000;

Дополнительный код: 1111 1001 1011 1001 – внутреннее двоичное представление числа.

Шестнадцатеричная форма внутреннего представления числа: F9B9.

Вещественные числа в ЭВМ хранятся в формате с плавающей точкой.

В формате с плавающей точкой вещественное число *R* представляется в виде произведения *мантиссы m* и основания системы счисления *P* в целой

степени $n$, называемой *порядком*:

$$R = m \cdot P^n .$$

Порядок $n$ указывает, на какое количество позиций и в каком направлении должна сместиться в мантиссе точка (запятая), отделяющая дробную часть от целой.

В ЭВМ используют *нормализованное* представление числа в форме с плавающей точкой. Мантисса в нормализованном представлении должна удовлетворять условию: $0{,}1 \le m < 1$.

В памяти компьютера мантисса представляется как целое число, содержащее только значащие цифры (0 целых и запятая не хранится).

Порядок $n$-разрядного нормализованного числа задается в так называемой смещенной форме: если для задания порядка выделено $k$ разрядов, то к истинному значению порядка прибавляют смещение, равное $2^{k-1}$. Например, порядок, принимающий значения в диапазоне от –128 до +127, представляется смещенным порядком, значения которого меняются от 0 до 255.

Таким образом, для записи внутреннего представления вещественного числа (четырехбайтовая ячейка памяти) необходимо:

1. Перевести модуль данного числа в двоичную систему счисления с 24 значащими цифрами.

2. Нормализовать двоичное число.

3. Найти смещенный порядок в двоичной системе счисления (прибавить к $n$ двоичное число $1000000_{(2)}$).

4. Учитывая знак мантиссы (0 для положительного числа, 1 – для отрицательного), выписать его представление (рис. 1.1).


Рисунок 1.1 – Представление в ЭВМ вещественных чисел

ПРИМЕР. Запишем внутреннее представление числа 250,1875.

Переведенное в двоичную систему счисления с 24 значащими цифрами число: $250{,}1875_{(10)} = 11111010,\ 0011000000000000_{(2)}$.

Нормализованная форма: $0{,}11111010001100000000000000 \cdot 10_{(2)}^{1000}$.

Здесь мантисса, основание системы счисления ($2_{(10)} = 10_{(2)}$) и порядок ($8_{(10)} = 1000_{(2)}$) записаны в двоичной системе.

Машинный порядок в двоичной системе счисления: 1000 + 100 0000 = = 100 1000.

Двоичное представление числа в четырехбайтовой ячейке памяти с учетом знака числа:

0 100 1000  1111 1010  0011 0000  0000 0000.

Шестнадцатеричная форма: 48FA3000.

ПРИМЕР. По шестнадцатеричной форме внутреннего представления C9811000 восстановим число.

Двоичное представление числа: 1100 1001 1000 0001 0001 0000 0000 0000.

Старший бит равен 1, следовательно, число отрицательное.

Порядок числа: 100 1001 – 100 0000 = $1001_{(2)} = 9_{(10)}$.

Нормализованная форма двоичного числа с учетом знака:

$$-0{,}10000001000100000000000 \cdot 10_{(2)}^{1001}.$$

Число в двоичной системе счисления: –100000010,001.

Переводим число в десятичную систему счисления:

$$-100000010{,}001_{(2)} = -(1\cdot 2^{8}+1\cdot 2^{1}+1\cdot 2^{-3}) = -258{,}125_{(10)}.$$

## ЛАБОРАТОРНАЯ РАБОТА 1. СИСТЕМЫ СЧИСЛЕНИЯ

1. Перевести числа в десятичную систему, а затем проверить результаты, выполнив обратные переводы:

Таблица 1.9 – Варианты заданий к задаче 1

| Вариант | Основание системы счисления | | |
|---|---|---|---|
| | 2 | 8 | 16 |
| 1 | 11101001,11 | 145,57 | 67D,12 |
| 2 | 11001,111 | 134,54 | 7AA,23 |
| 3 | 1101011,1 | 125,34 | 6D7,38 |
| 4 | 100001,01 | 130,61 | 700,A7 |
| 5 | 1100001,001 | 145,25 | 7CB,B4 |
| 6 | 10011,0001 | 175,52 | 967,68 |
| 7 | 101101,011 | 133,36 | 83F,55 |
| 8 | 100101,1101 | 163,15 | 6E5,C4 |
| 9 | 1110001,1 | 153,34 | 8D7,29 |
| 10 | 111101,11 | 164,32 | A53,96 |
| 11 | 10101010,101 | 101,37 | 840,D5 |
| 12 | 1010111,011 | 123,27 | AE7,E1 |
| 13 | 1101011,111 | 156,54 | 841,F2 |
| 14 | 10001110,1001 | 136,71 | AC3,74 |
| 15 | 1110101,1111 | 142,25 | A56,54 |

2. Перевести заданные двоичные числа в указанные системы счисления:

Таблица 1.10 – Варианты заданий к задаче 2

| Вариант | Основание системы счисления | |
|---|---|---|
| | 8 | 16 |
| 1 | 1011101001,11 | 10011110111,01011 |
| 2 | 1101101001,111 | 1011110011100,11 |
| 3 | 1101011011,101 | 1010110110101,1 |
| 4 | 1001011001,0111 | 1010101001000,111 |
| 5 | 1100001011,001 | 1100011011110,101 |
| 6 | 1001010111,0001 | 10101010111,1101 |
| 7 | 1011010001,011 | 1010100001001,11 |
| 8 | 1001001101,1101 | 1000010010111,101 |
| 9 | 1110001001,1 | 1001101001111,1001 |
| 10 | 1110101101,11 | 1000101011110,101 |
| 11 | 1010100110,101 | 1010011111010,1001 |
| 12 | 1010101111,011 | 1010010110101,011 |
| 13 | 1101001111,111 | 1001011011110,01 |
| 14 | 1000111110,1001 | 1000011101100,001 |
| 15 | 1011110101,1111 | 10111101010,00011 |

3. Сложить числа, а затем проверить результаты, выполнив соответствующие десятичные сложения:

Таблица 1.11 – Варианты заданий к задаче 3

| Вариант 1 | Вариант 2 | Вариант 3 |
|---|---|---|
| $1011,11_2$ и $111,1_2$ | $A,6_{16}$ и $C,8_{16}$ | $3,6_8$ и $24,2_8$ |
| Вариант 4 | Вариант 5 | Вариант 6 |
| $6,5_8$ и $43,7_8$ | $1010,1_2$ и $11,01_2$ | $8,A_{16}$ и $F,3_{16}$ |
| Вариант 7 | Вариант 8 | Вариант 9 |
| $A,B_{16}$ и $E,F_{16}$ | $7,5_8$ и $14,6_8$ | $1001,01_2$ и $110,1_2$ |
| Вариант 10 | Вариант 11 | Вариант 12 |
| $101,101_2$ и $10,01_2$ | $C,D_{16}$ и $5,2_{16}$ | $3,1_8$ и $26,7_8$ |
| Вариант 13 | Вариант 14 | Вариант 15 |
| $4,5_8$ и $15,3_8$ | $1110,1_2$ и $101,01_2$ | $A,1_{16}$ и $6,F_{16}$ |

4. Умножить числа, а затем проверить результаты, выполнив соответствующие десятичные умножения:

Таблица 1.12 – Варианты заданий к задаче 4

| Вариант 1 | Вариант 2 | Вариант 3 |
|---|---|---|
| $A,1_{16}$ и $2,4_{16}$ | $101_2$ и $1111,001_2$ | $6,25_8$ и $7,12_8$ |
| Вариант 4 | Вариант 5 | Вариант 6 |
| $110_2$ и $1011,101_2$ | $1,23_8$ и $5,14_8$ | $B,8_{16}$ и $7,1_{16}$ |
| Вариант 7 | Вариант 8 | Вариант 9 |
| $4,24_8$ и $6,15_8$ | $9,A_{16}$ и $3,1_{16}$ | $101_2$ и $1001,111_2$ |
| Вариант 10 | Вариант 11 | Вариант 12 |
| $7,C_{16}$ и $1,5_{16}$ | $101_2$ и $1101,011_2$ | $4,27_8$ и $7,41_8$ |
| Вариант 13 | Вариант 14 | Вариант 15 |
| $111_2$ и $1001,101_2$ | $3,62_8$ и $6,12_8$ | $D,2_{16}$ и $1,E_{16}$ |

5. Десятичное число $x$ эквивалентно числу $y$ в некоторой другой системе счисления. Найти основание этой системы.

Таблица 1.13 – Варианты заданий к задаче 5

| Вариант | $x$ | $y$ | Вариант | $x$ | $y$ | Вариант | $x$ | $y$ |
|---|---|---|---|---|---|---|---|---|
| 1 | 59 | 214 | 6 | 116 | 431 | 11 | 62 | 222 |
| 2 | 46 | 232 | 7 | 131 | 245 | 12 | 43 | 111 |
| 3 | 160 | 316 | 8 | 23 | 113 | 13 | 121 | 321 |
| 4 | 183 | 503 | 9 | 133 | 341 | 14 | 145 | 401 |
| 5 | 38 | 123 | 10 | 116 | 164 | 15 | 130 | 202 |

## ЛАБОРАТОРНАЯ РАБОТА 2. ПРЕДСТАВЛЕНИЕ ЧИСЕЛ В ЭВМ

1. Получить шестнадцатеричную форму внутреннего представления целого числа в двухбайтовой ячейке.
2. Получить шестнадцатеричную форму внутреннего представления целого числа в двухбайтовой ячейке.
3. По шестнадцатеричной форме внутреннего представления целого числа в двухбайтовой ячейке восстановить само число.

Таблица 1.14 – Варианты заданий к задачам 1, 2, 3

| Вари-ант | Номер задания | | | Вари-ант | Номер задания | | | Вари-ант | Номер задания | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | | 1 | 2 | 3 | | 1 | 2 | 3 |
| 1 | 1450 | -1450 | F67D | 6 | 1453 | -1453 | F967 | 11 | 2101 | -2101 | F840 |
| 2 | 1341 | -1341 | F7AA | 7 | 1833 | -1833 | F83F | 12 | 2304 | -2304 | FAE7 |
| 3 | 1983 | -1983 | F6D7 | 8 | 2331 | -2331 | F6C5 | 13 | 2345 | -2345 | F841 |
| 4 | 1305 | -1305 | F700 | 9 | 1985 | -1985 | F8D7 | 14 | 2134 | -2134 | FAC3 |
| 5 | 1984 | -1984 | F7CB | 10 | 1689 | -1689 | FA53 | 15 | 2435 | -2435 | FA56 |

4. Получить шестнадцатеричную форму внутреннего представления вещественного числа в четырехбайтовой ячейке.
5. По шестнадцатеричной форме внутреннего представления вещественного числа в четырехбайтовой ячейке восстановить само число.

Таблица 1.15 – Варианты заданий к задачам 4, 5

| Вари-ант | Номер задания | | Вари-ант | Номер задания | | Вари-ант | Номер задания | |
|---|---|---|---|---|---|---|---|---|
| | 4 | 5 | | 4 | 5 | | 4 | 5 |
| 1 | 86,8125 | C5DB0000 | 6 | -51,375 | 488B6000 | 11 | 333,75 | C6870000 |
| 2 | -29,625 | 45D14000 | 7 | 125,125 | C7B7A000 | 12 | -82,75 | 46870000 |
| 3 | 91,8125 | C5ED0000 | 8 | -33,75 | 45DB0000 | 13 | 224,625 | C9A6E000 |
| 4 | -27,375 | 47B7A000 | 9 | 88,3125 | C88B6000 | 14 | -91,875 | 49A6E000 |
| 5 | 139,375 | C5D14000 | 10 | -99,375 | 45ED0000 | 15 | 73,4375 | 48E04000 |

# 2 ОСНОВНЫЕ ЭЛЕМЕНТЫ ЯЗЫКА С++

## 2.1 Идентификаторы

*Идентификатор* – это имя программного объекта (переменной, функции, класса или другого объекта). При выборе идентификатора следует иметь в виду следующее:

– идентификатор может состоять только из букв латинского алфавита, цифр или символов подчёркивания;

– идентификатор должен начинаться с буквы (нижнего или верхнего регистра). Он не может начинаться с цифры;

– не рекомендуется начинать идентификаторы с символа подчеркивания, т.к. в этом случае они могут совпасть с именами системных функций или переменных;

– С++ различает нижний и верхних регистры. **`Min`**, **`MIN`**, **`min`** – три разных идентификатора;

– идентификатор не может быть ключевым словом;

– длина идентификатора по стандарту не ограничена, но некоторые компиляторы и компоновщики налагают на нее ограничения.

Таблица 2.1 – Примеры идентификаторов

| Правильно | Неправильно | Не рекомендуется |
|---|---|---|
| **`B2`**<br>**`i`**<br>**`Max`**<br>**`k_2`**<br>**`kilograms_of_pipe`** | **`8A`**<br>**`while`**<br>**`ид`**<br>**`f(x)`**<br>**`name of variable`** | **`_f`**<br>**`_2m`** |

## 2.2 Типы данных

К *основным* типам данных языка С++ относят:

– **`char`** – символьный;

– **`int`** – целый;

– **`float`** – с плавающей точкой;

– **`double`** – двойной точности;

– **`bool`** – логический;

– **`void`** – пустой.

Типы данных, созданные на базе стандартных типов с использованием спецификаторов, называют *составными*. В С++ определены четыре спецификатора типов данных:

– **`short`** – короткий;

– **`long`** – длинный;

– **`signed`** – знаковый;

– **unsigned** – беззнаковый.

### 2.3 Переменные

*Переменная* – это поименованный участок памяти, в котором хранится значение определенного типа. Переменная имеет тип, имя и значение.

Перед использованием любую переменную надо определить:

```
тип список_переменных;
```

Например:

```
int a;
float g, u, m2;
```

Можно сразу при определении переменной дать ей некоторое начальное значение. Данный прием называется инициализацией. Например:

```
int age = 28;
```

### 2.4 Константы

*Константы* – это именованные ячейки памяти, значения которых фиксируются на начальном этапе выполнения программы и затем в процессе выполнения программы не могут быть изменены.

В С ++ константы определяются следующим образом:

```
const тип имя = значение;
```

Например:

```
const double g = 9.81;
```

### 2.5 Операции, выражения, операторы

*Оператор* – законченное предложение на языке С++. Он указывает компьютеру выполнить некоторые действия. Оператор всегда завершается «**;**».

*Выражение* – конструкция, определяющая состав данных, операции и порядок выполнения операций над данными. Выражение состоит из операндов, знаков операций и круглых скобок.


Рисунок 2.1 – Выражение

*Операнды* – данные, над которыми выполняются действия.

*Операции* выполняют определенные действия над операндами.

В соответствии с количеством операндов операции делятся на *унарные* (один операнд), *бинарные* (два операнда) и, единственную, *тернарную* (в которой три операнда).

Операции имеют приоритет и ассоциативность.

*Приоритет* определяет старшинство операции. Несколько операций могут иметь равный приоритет.

В этом случае включается порядок вычисления – *ассоциативность*. Ассоциативность может быть либо слева-направо, либо справа-налево.

Один и тот же знак может интерпретироваться по-разному в зависимости от контекста.

Приоритет основных операций:

1. Инкремент, декремент.
2. Унарные плюс и минус, логическое и поразрядное НЕ, приведение к типу, взятие адреса.
3. Умножение, деление, остаток от деления.
4. Сложение, вычитание.
5. Больше, меньше, больше или равно, меньше или равно.
6. Равно, не равно.
7. Логическое И.
8. Логическое ИЛИ.
9. Операции присваивания.

Для изменения порядка вычисления используются круглые скобки.

Рассмотрим наиболее часто используемые операции.

*Операции присваивания*

Обычная операция присваивания имеет вид:

```
идентификатор = значение;
```

В C++ существует возможность множественного присваивания, то есть присваивания нескольким переменным одного и того же значения:

```
идентификатор1 = идентификатор2 = ... идентификаторN = значение;
```

Так же в C++ имеются составные операции присваивания:

```
идентификатор операция= значение;
```

Их эквивалентом является следующая запись:

```
идентификатор = идентификатор операция значение;
```

**операция** – одна из операций **+, –, *, /, %, &, |, ^, <<, >>.**

Например, выражение `a += 2` эквивалентно выражению `a = a + 2`.

*Арифметические операции*

Операции **+**, **–**, *****, **/** относят к арифметическим операциям. Их назначение понятно и не требует дополнительных пояснений. Однако стоит отметить, если операция деления применяется к целочисленным операндам, то результатом является целая часть частного (дробная часть отбрасывается). Например:

```
11 / 4; //В результате будет 2
11 / 4.0; //В результате будет 2.75
```

Остаток от деления **%** применяется только к целочисленным аргументам.

Например:

```
11 % 4; //В результате будет 3
```

Операции инкремента **++** и декремента **--** также причисляют к арифметическим, так как они выполняют увеличение и уменьшение на единицу значения переменной. Эти операции имеют две формы записи: префиксную (операция записывается перед операндом) и постфиксную (операция записывается после операнда). Эти формы отличаются при использовании их в выражении. Если знак инкремента (декремента) предшествует операнду, то сначала выполняется увеличение (уменьшение) значения операнда, а затем операнд участвует в выражении.

Например:

```
int x = 12;
int y = ++x; //В результате y = 13
```

Если знак инкремента (декремента) следует после операнда, то сначала операнд участвует в выражении, а затем выполняется увеличение (уменьшение) значения операнда:

```
int x = 12;
int y = x++; //В результате y = 12
```

*Логические операции*

К логическим операциям относятся **&&**, **||**, **!**. Данные операции выполняются над логическими значениями. В таблице 2.2 приведены результаты логических операций.

Таблица 2.2 – Логические операции

| A | B | !A | A&&B | A\|\|B |
|---|---|---|---|---|
| false | false | true | false | false |
| false | true | true | false | true |
| true | false | false | false | true |
| true | true | false | true | true |

*Операции сравнения*

Операции сравнения возвращают в качестве результата логическое значение **true** или **false**. К данной группе операций относят **>**, **>=**, **<**, **<=**, **==**, **!=**.

*Условная операция*

Условная операция имеет вид:

```
условие ? выражение1 : выражение2;
```

Если **условие** истинно (не равно **0**), то результатом будет **выражение1** иначе – **выражение2**. Например:

```
int x = 12;
int y = 4;
int z = (x > y) ? 2 : 5; //В результате z = 2
```

*Операция преобразования типа*

Операция преобразования типа приводит выражение к другому типу данных и имеет вид:

```
(тип) выражение;
```

Например:

```
int x = 5;
int y = x / 2; //В результате y = 2
double z = (double) x / 2; //В результате z = 2.5
int a = 5;
bool b = (bool) a; //В результате b = true
int c = 0;
bool d = (bool) c; //В результате b = false
```

### 2.6 Структура программы

Программа на языке C++ состоит из директив препроцессора, указаний компилятору, объявлений глобальных переменных и/или констант, объявлений и определений функций.

Общая структура программы:

```
директивы препроцессора
описание типов пользователя;
прототипы функций;
описание глобальных переменных;

тип_результата main(параметры)
{
     операторы;
}
тип_результата имя1(параметры)
{
     операторы1;
}
тип_результата имя2(параметры)
{
     операторы2;
}
...
тип_результата имяN(параметры)
{
     операторыN;
}
```

*Препроцессор* – это программа, которая обрабатывает текст программы до компилятора.

Работа препроцессора управляется директивами.

Директива **#include** включает содержимое файла, путь к которому задан,

в компилируемый файл вместо строки с директивой:

**`#include <путь>`** либо **`#include "путь"`**

Если путь заключен в угловые скобки, то поиск файла осуществляется в стандартных директориях. Если путь заключен в кавычки и задан полностью, то поиск файла осуществляется в заданной директории, а если путь полностью не задан – в текущей директории. С помощью этой директивы можно включать в текст программы как стандартные, так и свои файлы.

## 2.7 Стандартная математическая библиотека

В стандартную математическую библиотеку языка C++ входит множество специальных математических функций. Для того, чтобы использовать эти функции, необходимо подключить заголовочный файл, содержащий описания этих функций командой:

**`#include <math.h>`** либо **`#include <cmath>`**

Таблица 2.3 – Основные математические функции

| Функция | Описание |
|---|---|
| 1 | 2 |
| **`abs(x)`** | Модуль целого числа. Например:<br>**`abs(-3); // 3`**<br>**`abs(-3.9); // 3`** |
| **`acos(x)`** | $\arccos(x)$. Возвращается результат из диапазона -π… π |
| **`asin(x)`** | $\arcsin(x)$. Возвращается результат из диапазона -π… π |
| **`atan(x)`** | $arctg(x)$. Возвращается результат из диапазона -π/2… π/2 |
| **`cbrt(x)`** | $\sqrt[3]{x}$ |
| **`ceil(x)`** | Округление до большего целого. Например:<br>**`ceil(2.4); // 3`**<br>**`ceil(-1.5); // -1`** |
| **`cos(x)`** | $\cos(x)$ |
| **`cosh(x)`** | $ch(x)$ |
| **`exp(x)`** | $e^x$ |
| **`fabs(x)`** | Модуль числа. Например:<br>**`fabs(-3); // 3`**<br>**`fabs(-3.9); // 3.9`** |
| **`floor(x)`** | Округление до меньшего целого. Например:<br>**`floor(2.4); // 2`**<br>**`floor(-1.5); // -2`** |
| **`fmod(x, y)`** | Вычисление остатка от деления **`x`** на **`y`**. Например:<br>**`fmod(3,4); // 3`**<br>**`fmod(6.4,3.1); // 0.2`** |
| **`log(x)`** | $\ln(x)$ |

Окончание таблицы 2.3

| 1 | 2 |
|---|---|
| `log2(x)` | $\log_2(x)$ |
| `log10(x)` | $\lg(x)$ |
| `pow(x, y)` | $x^y$ |
| `round(x)` | Округляет число по правилам арифметики. Например:<br>`round(2.4); // 2`<br>`round(-1.5); // -2` |
| `sqrt(x)` | $\sqrt{x}$ |
| `sin(x)` | $\sin(x)$ |
| `sinh(x)` | $sh(x)$ |
| `tan(x)` | $tg(x)$ |
| `tanh(x)` | $th(x)$ |
| `trunc(x)` | Отбрасывание дробной части. Например:<br>`trunc(2.4); // 2`<br>`trunc(-1.5); // -1` |

Стандартная математическая библиотека содержит также определения некоторых математических констант.

Таблица 2.4 – Некоторые математические константы

| Символ | Описание | Значение |
|---|---|---|
| `M_E` | $e$ | `2.71828182845904523536` |
| `M_PI` | $\pi$ | `3.14159265358979323846` |

### 2.8 Комментарии

*Комментарий* – это строка (или несколько строк) текста, которая служит для описания и документирования исходного кода.

*Однострочные комментарии* – это комментарии, которые пишутся после символов **//**. Они размещаются в отдельных строках, и всё, что находится после этих символов до конца строки, игнорируется компилятором.

*Многострочные комментарии* – это комментарии, которые пишутся между символами **/*** и ***/**. Всё, что находится между звёздочками, игнорируется компилятором.

### 2.9 Ввод-вывод данных на консоль

*Ввод-вывод данных* в языке С++ осуществляется либо с помощью *функций ввода-вывода* в стиле С, либо с использованием *библиотеки классов* С++. Преимущество объектов С++ в том, что они легче в использовании,

особенно если ввод-вывод достаточно простой. Функции ввода-вывода, унаследованные от С, более громоздкие, но подходят для задач с форматированным выводом данных.

*Форматированный ввод-вывод*

Функции форматированного ввода и вывода данных расположены в библиотеке **stdio.h**.

Форматированный вывод переменных, указанных в списке, в соответствии со строкой форматирования выполняет функция:

**printf(строка_форматирования, список_выводимых_переменных)**

Ввод переменных, адреса которых указаны в списке, в соответствии со строкой форматирования выполняет функция:

**scanf(строка_форматирования, список_адресов_вводимых_переменных)**

Строка форматирования содержит символы, которые будут выводиться на экран или запрашиваться с клавиатуры, и так называемые спецификации. *Спецификации* – это строки, которые начинаются символом **%** и выполняют управление форматированием:

**%«флаг»«ширина».«точность»«модификатор»«тип»**

Параметры флаг, ширина, точность и модификатор в спецификациях могут отсутствовать.

Таблица 2.5 – Символы управления вводом-выводом

| Символ | Назначение |
|---|---|
| 1 | 2 |
| Флаг | |
| **+** | Перед числом выводится знак «**+**» или «**–**» |
| **пробел** | Перед положительным числом выводится пробел, перед отрицательным – минус |
| **#** | Выводится код системы счисления: **0** – перед восьмеричным числом, **0x** (**0X**) – перед шестнадцатеричным |
| Ширина | |
| **n** | Минимальное число выводимых символов. Незаполненные позиции заполняются пробелам |
| **0n** | Минимальное число выводимых символов. Незаполненные позиции заполняются нулями |
| Точность | |
| **n** | Для вещественных типов выводить **n** знаков после точки |
| Модификатор | |
| **h** | используется в качестве префикса с целыми типами для определения, что аргумент является **short int** |
| **l** | используется в качестве префикса с целыми типами для обозначения, что аргумент является **long int** или **long long int**. Символ **l** используется также как префикс с вещественными типами для определения, что аргумент является скорее **double**, чем **float** |

Окончание таблицы 2.5

| 1 | 2 |
|---|---|
| Тип | |
| **c** | Вывод одного символа |
| **s** | Вывод строки символов |
| **d** | Вывод целого десятичного числа |
| **i** | Вывод целого десятичного, восьмеричного или шестнадцатеричного числа |
| **E, e** | Вывод вещественного числа с плавающей точкой |
| **f** | Вывод вещественного числа с фиксированной точкой |
| **G, g** | Вывод более краткое из **e** и **f** |
| **o** | Вывод целого беззнакового восьмеричного числа |
| **X, x** | Вывод целого беззнакового шестнадцатеричного числа |
| **u** | Вывод целого беззнакового десятичного числа |
| **p** | Вывод значения указателя |
| **n** | Вывод числа прочитанных символов |

Кроме того, строка форматирования может содержать Escape-последовательности.

Таблица 2.6 – Escape-последовательности

| | | | |
|---|---|---|---|
| **\a** | Звуковой сигнал | **** | Обратная косая черта |
| **\b** | Возврат на шаг | **\'** | Одиночная кавычка |
| **\f** | Переход на новую страницу | **\"** | Двойная кавычка |
| **\n** | Переход на новую строку | **\?** | Вопросительный знак |
| **\r** | Возврат в начало строки | **\000** | Символ, заданный восьмеричным кодом ASCII **000** |
| **\t** | Горизонтальная табуляция | **\xdd** | Символ, заданный шестнадцатеричным кодом ASCII **dd** |
| **\v** | Вертикальная табуляция | **\udddd** | Символ, заданный шестнадцатеричным кодом Unicode **dddd** |

ПРИМЕР. Зная длины сторон *a*, *b* и *c*, вычислить площадь *S* и периметр *P* треугольника.

Площадь треугольника можно вычислить по формуле:

$$S = \sqrt{r(r-a)(r-b)(r-c)},$$

где *r* – полупериметр треугольника.

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

using namespace std;

int main()
{
    double a, b, c, S, P, r;
    printf("Input a,b,c:\n");
    scanf("%lf%lf%lf", &a, &b, &c);
    r = (a + b + c) / 2;
    P = 2 * r;
    S = sqrt(r * (r - a) * (r - b) * (r - c));
    printf("S=%5.2lf\tP=%5.2lf", S, P);
    return 0;
}
```

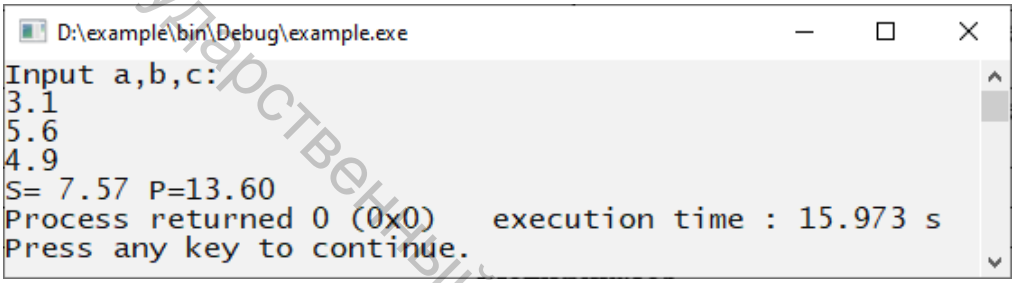



Рисунок 2.2 – Результат выполнения программы

*Потоковый ввод-вывод*

Средства потокового ввода и вывода данных расположены в библиотеке **iostream**. Библиотека iostream определяет три стандартных потока:

- **cin** – стандартный входной поток;
- **cout** – стандартный выходной поток;
- **cerr** – стандартный поток вывода сообщений об ошибках.

Для их использования обязательно необходимо прописать строку:

```
using namespace std;
```

Для выполнения операций ввода-вывода переопределены две операции:

**>>** – получить из входного потока;

**<<** – поместить в выходной поток.

Вывод информации осуществляется командой:

```
cout << значение;
```

Здесь значение преобразуется в последовательность символов и выводится в выходной поток.

Возможно многократное назначение потоков:

```
cout << значение1 << значение2 << ... << значениеN;
```

Ввод информации осуществляется командой:

```
cin >> идентификатор;
```

При этом из входного потока читается последовательность символов до

пробела, затем эта последовательность преобразуется к типу идентификатора, и получаемое значение помещается в идентификатор.

Возможно многократное назначение потоков:

```
cin >> идентификатор1 >> идентификатор2 >> ... >> идентификаторN;
```

При наборе данных на клавиатуре значения для такого оператора должны быть разделены символами пробела, табуляции или окончания строки.

Возможность форматирования строки при использовании потока **cout** обеспечивают специальные функции, флаги и манипуляторы. Различие между функциями, флагами и манипуляторами форматирования состоит в способе их применения.

Таблица 2.7 – Функции вывода

| Функция | Описание |
|---|---|
| **cout.fill(char)** | Заполняет пустые знакоместа символом |
| **cout.width(n)** | Задает ширину поля вывода значения |
| **cout.precision(n)** | Задает количество значащих цифр или количество знаков после точки, если задан манипулятор или флаг **fixed** |
| **cout.setf(flag)** | Устанавливает флаг ввода/вывода |
| **cout.unsetf(flag)** | Отключает флаг вывода |

Таблица 2.8 – Флаги вывода

| Флаг | Описание |
|---|---|
| **ios::boolalpha** | Вывод логических величин в текстовом виде (**true**, **false**) |
| **ios::oct** | Ввод/вывод величин в восьмеричной системе счисления (сначала необходимо снять флаг **dec**) |
| **ios::dec** | Ввод/вывод величин в десятичной системе счисления (флаг установлен по умолчанию) |
| **ios::hex** | Ввод/вывод величин в шестнадцатеричной системе счисления (сначала необходимо снять флаг **dec**) |
| **ios::showpos** | Вывод знака числа |
| **ios::uppercase** | В шестнадцатеричной системе счисления использовать буквы верхнего регистра (по умолчанию установлены буквы нижнего регистра) |
| **ios::showbas** | Вывод индикатора основания системы счисления |
| **ios::scientific** | Вывод чисел в формате с плавающей точкой |
| **ios::fixed** | Вывод чисел в формате с фиксированной точкой |
| **ios::right** | Выравнивание по правой границе (по умолчанию). Сначала необходимо установить ширину поля вывода |
| **ios::left** | Выравнивание по левой границе |

Если при вводе/выводе необходимо установить (снять) несколько флагов, то можно воспользоваться поразрядной логической операцией ИЛИ **|**.

Таблица 2.9 – Манипуляторы вывода

| Манипулятор | Описание |
|---|---|
| **endl** | Переход на новую строку |
| **oct** | Ввод/вывод величин в восьмеричной системе счисления |
| **dec** | Ввод/вывод величин в десятичной системе счисления |
| **hex** | Ввод/вывод величин в шестнадцатеричной системе счисления |
| **right** | Выравнивание по правой границе (по умолчанию) |
| **left** | Выравнивание по левой границе |
| **boolalpha** | Вывод логических величин в текстовом виде (**true**, **false**) |
| **noboolalpha** | Вывод логических величин в числовом виде (по умолчанию) |
| **showpos** | Вывод знака числа |
| **noshowpos** | Отключает вывод знака числа |
| **uppercase** | В шестнадцатеричной системе счисления использовать буквы верхнего регистра |
| **nouppercase** | В шестнадцатеричной системе счисления не использовать буквы верхнего регистра |
| **showbase** | Вывод индикатора основания системы счисления |
| **noshowbase** | Отключает вывод индикатора основания системы счисления |
| **scientific** | Вывод чисел в формате с плавающей точкой |
| **fixed** | Вывод чисел в формате с фиксированной точкой |

ПРИМЕР. Известны плотность $\rho$, высота $h$ и радиус основания $R$ цилиндрического слитка, полученного в металлургической лаборатории. Найти объем $V$, массу $m$ и площадь основания $S$ слитка.

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

using namespace std;

int main()
{
    double ro, h, R, S, V, m;
    cout << "ro=";
    cin >> ro;
    cout << "h=";
    cin >> h;
    cout << "R=";
    cin >> R;
    S = 2 * M_PI * R;
    V = M_PI * R * R * h;
    m = ro * V;
    cout.precision(3);
```

```
    cout << fixed << "S=" << S << endl << "V=" << V << endl << "m=" << m;
    return 0;
}
```

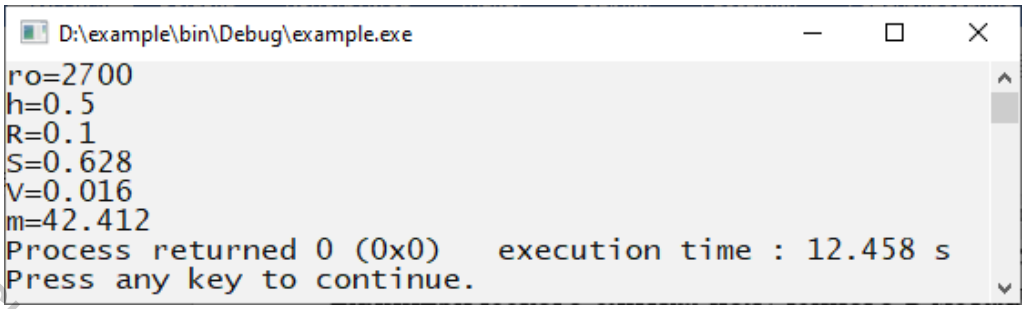



Рисунок 2.3 – Результат выполнения программы

*Кириллица на консоли*

В командной строке Windows кодировка символов соответствует стандарту cp866. Причём поменять кодировку в командной строке Windows нельзя. Во всех русскоязычных Windows кодировка cp1251 является стандартной. И при создании проекта этот стандарт кодирования символов наследуется проектом, то есть программой. В итоге получается, что программа передаёт коды символов сообщения стандарта cp1251. Командная строка принимает эти коды и переводит их в символы, но уже по стандарту cp866, так как другого стандарта не знает. В итоге сообщение передано в консоль, но символы интерпретированы неправильно. Так появляются «иероглифы».

Существует несколько способов решения данной проблемы.

Самый простой – настройка локали.

*Локаль* – это набор параметров: набор символов, язык пользователя, страна, часовой пояс и др. В библиотеке **clocale** есть функция **setlocale()**, которая выполняет настройку локали.

Функция **setlocale()** имеет два параметра, первый параметр – тип категории локали (в данном случае **LC_TYPE** – набор символов), второй параметр – значение локали (в данном случае **"rus"**, **"Russian"** или пустые двойные кавычки).

Функция **setlocale()** работает только для вывода данных. Если же осуществлять ввод, то будут все те же иероглифы.

В библиотеке **windows.h** содержатся функции, позволяющие решить и эту проблему. Функция **SetConsoleCP(1251)** устанавливает нужную кодовую таблицу для ввода, тогда как функция **SetConsoleOutputCP(1251)** устанавливает нужную кодовую таблицу для вывода.

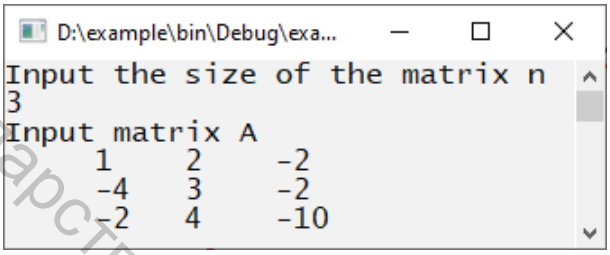
## ЛАБОРАТОРНАЯ РАБОТА 3. ЗНАКОМСТВО СО СРЕДОЙ CODE::BLOCKS

1. Построить проект по следующему исходному коду программы:

```
#include <iostream>
#include <windows.h>

using namespace std;
int main()
{
    COORD c;
    HANDLE hnd = GetStdHandle(STD_OUTPUT_HANDLE);
    float A[10][10], B[10][10];
    int i, j, n;

    // Ввод исходных данных
    cout << "Input the size of the matrix n" << endl;
    cin >> n;
    cout << "Input matrix A" << endl;
    for (i = 0; i < n; i++)
    {
        for (j = 0; j < n; j++)
        {
            c.X = 5 + j * 5;
            c.Y = 3 + i;
            SetConsoleCursorPosition(hnd, c);
            cin >> A[i][j];
        }
    }

    // Обработка данных
    for (i = 0; i < n; i++)
    {
        for (j = 0; j < n; j++)
        {
            if (A[i][j] < 0)
            {
                B[i][j] = 2 * A[i][j];
            }
            else
            {
                B[i][j] = A[i][j];
            }
        }
    }

    // Вывод результата
    for (i = 0; i < n; i++)
    {
        for (j = 0; j < n; j++)
```

```
        {
            c.X = 5 + j * 5;
            c.Y = 3 + i;
            SetConsoleCursorPosition(hnd, c);
            cout << A[i][j];
        }
    }

    return 0;
}
```

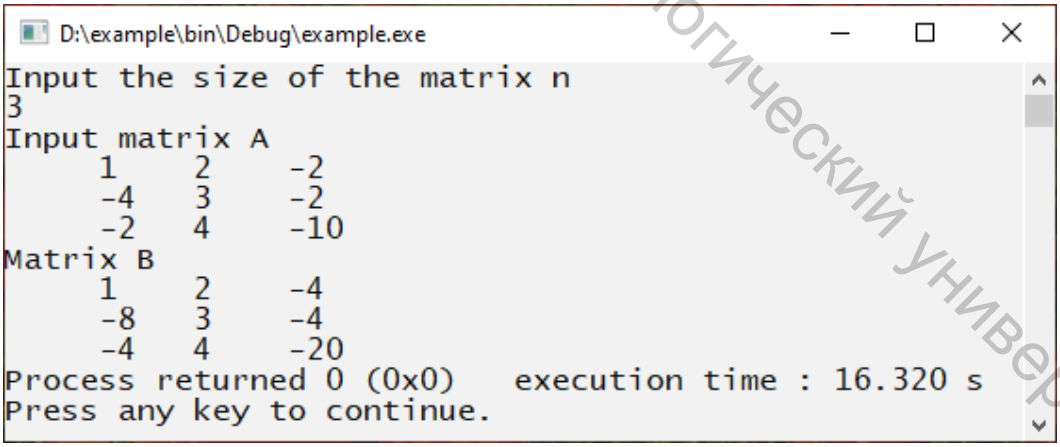
2. Собрать проект и запустить программу. Ввести следующие данные:




После ввода матрицы на консоли ниже должна появиться матрица B, отличающаяся от матрицы A тем, что все отрицательные элементы в ней удвоены.

Правильно ли осуществляется вывод результатов?

3. Исправить логические ошибки в программе так, чтобы результат её запуска выглядел следующим образом:




## ЛАБОРАТОРНАЯ РАБОТА 4. ВЫЧИСЛЕНИЕ МАТЕМАТИЧЕСКИХ ВЫРАЖЕНИЙ

Написать программу вычисления указанной величины. Результат проверить при заданных исходных значениях. Значения переменных *x*, *y*, *z* ввести с консоли. Результат вывести с точностью 6 знаков после точки.

Таблица 2.10 – Варианты заданий

| Вариант | Выражение | Значения переменных | Результат |
|---|---|---|---|
| 1 | 2 | 3 | 4 |
| 1 | $\dfrac{2\cos\left(x-\dfrac{\pi}{6}\right)\left(1+\dfrac{z^2}{\pi-\dfrac{z^2}{5}}\right)}{e+\sin^2(y)}$ | $x = 14{,}26$<br>$y = -1{,}22$<br>$z = 3{,}5\cdot10^{-2}$ | 0,216808 |
| 2 | $\dfrac{\sqrt[3]{2e+\lvert x-y\rvert^2+1}}{x^2+y^2+2}-e^{\lvert x-y\rvert}\left(tg^2(z)+1\right)^x$ | $x = -4{,}5$<br>$y = 0{,}75\cdot10^{-4}$<br>$z = 0{,}845\cdot10^{2}$ | -55,689004 |
| 3 | $\ln\left(y^{\sqrt{\lvert x\rvert}}\right)(x-ey)+\sin^2\left(arctg(z)\right)$ | $x = -15{,}246$<br>$y = 4{,}642\cdot10^{-2}$<br>$z = 20{,}001\cdot10^{2}$ | 185,270319 |
| 4 | $\left\lvert x^{\frac{y}{x}}-\sqrt[3]{\dfrac{y}{x}}\right\rvert+(y-x)\dfrac{\cos(y)-\dfrac{z}{y-x}}{1+(y-x)^2}$ | $x = 1{,}825\cdot10^{2}$<br>$y = 18{,}225$<br>$z = -3{,}298\cdot10^{-4}$ | 1,213078 |
| 5 | $\sqrt[3]{\pi z+e^y}+\dfrac{x}{y\lvert z-y^2\rvert^x\left(x^2+xy+1\right)}$ | $x = -2{,}235\cdot10^{-2}$<br>$y = 2{,}25$<br>$z = 39{,}374$ | 5,095519 |
| 6 | $\sqrt{\pi\left(x^2+x^{y+2}\right)}\cdot\arccos^2\lvert z-x^2+y\rvert^2$ | $x = 16{,}55\cdot10^{-3}$<br>$y = -2{,}75$<br>$z = 3{,}15$ | 16,412261 |
| 7 | $e\cdot arctg\left(x^2\right)-\dfrac{\arccos(x)}{e}\cdot\dfrac{x+7\lvert x^2-y\rvert+x^2}{\lvert x-y\rvert z+x^2}$ | $x = 0{,}1722$<br>$y = 6{,}33$<br>$z = 3{,}23\cdot10^{-4}$ | -719,889361 |
| 8 | $\dfrac{e^{\lvert x-y\rvert}\lvert x-y\rvert^{x+y}}{arctg(x)+arctg(z)}+\sqrt[3]{x^5+\ln^3(y)}$ | $x = -2{,}235\cdot10^{-1}$<br>$y = 2{,}25$<br>$z = 15{,}221$ | 58,655882 |
| 9 | $\left\lvert x^{\frac{y}{x}}-\sqrt[4]{y-x}\right\rvert+(y-x)\dfrac{\cos(y)-\dfrac{z}{y-x}}{e+(y-x)^2}$ | $x = 1{,}825\cdot10^{-2}$<br>$y = 18{,}225$<br>$z = -3{,}298\cdot10^{-2}$ | 2,109949 |
| 10 | $2^{-x}\sqrt{x+\sqrt[3]{\lvert y\rvert}}\sqrt{e^{1-\frac{1}{\sin(z)}}}$ | $x = 3{,}981\cdot10^{-2}$<br>$y = -1{,}625\cdot10^{3}$<br>$z = 0{,}512$ | 1,985231 |
| 11 | $\sqrt[3]{\pi x^6+e^y}+\dfrac{\pi^{\lvert x-y\rvert}\lvert x-y\rvert^{x+y}}{arctg\left(x^2\right)+arctg(z)}$ | $x = 3{,}235\cdot10^{-2}$<br>$y = 1{,}245$<br>$z = 15{,}374$ | 4,916469 |

Окончание таблицы 2.10

| 1 | 2 | 3 | 4 |
|---|---|---|---|
| 12 | $\pi^{yx}+e^{x^y}-\dfrac{y\left(arctg(z)-\dfrac{\pi}{6}\right)}{\lvert x\rvert+\dfrac{1}{y^2+1}}$ | $x=3{,}251$<br>$y=0{,}425$<br>$z=0{,}374\cdot10^{-4}$ | 10,126664 |
| 13 | $y^{\lvert yx\rvert}+\cos^3(y)\dfrac{\lvert x-y\rvert\left(1+\dfrac{\sin^2(z)}{\sqrt{x+y}}\right)}{e^{\lvert x-y\rvert}+\dfrac{x}{\pi}}$ | $x=6{,}251$<br>$y=0{,}748$<br>$z=0{,}847\cdot10^{-3}$ | 0,266035 |
| 14 | $\dfrac{y^{x+1}}{\sqrt[3]{\lvert y-e\rvert}+\pi}+\dfrac{x+\dfrac{y}{e}}{e+\left\lvert\dfrac{x}{\pi}+y\right\rvert}(x+1)^{-\frac{1}{\sin(z)}}$ | $x=4{,}701$<br>$y=1{,}348$<br>$z=1{,}548\cdot10^{-3}$ | 1,290438 |
| 15 | $\sqrt{\lvert x-y\rvert^y}\left(\dfrac{\pi x}{y}+\ln(arctg(z))\right)$ | $x=-11{,}286$<br>$y=9{,}642\cdot10^{-2}$<br>$z=2{,}001\cdot10^{2}$ | -412,965688 |

## ЛАБОРАТОРНАЯ РАБОТА 5. ЛИНЕЙНЫЕ АЛГОРИТМЫ

Написать программу решения задачи согласно варианту:

1. В казино на рулетку бросают шарик. Какую скорость он развивает, если за 30 *с* он совершает *а* кругов? Радиус рулетки равен *r см*.

2. Для выделки *а м*$^2$ кожи расходуется *b кг* дубильного экстракта. Сколько квадратных метров кожи можно обработать, если используется *с кг* экстракта?

3. Сколько заготовок круглой формы можно изготовить из куска материала длиной *а м* и шириной *b м*, если радиус заготовки *R см*? Центры заготовок должны располагаться на одной линии.

4. При производстве пряжи из каждого килограмма сырья получают *а кг* пряжи, отходы составляют *b кг*, потери *с кг* ($a + b + c = 1$). Сколько пряжи, отходов и потерь получается на *т кг* сырья?

5. На изготовление одного изделия затрачивается *а мин*. Подготовительные операции занимают $b_1$ % времени, простые основные операции – $b_2$ %, остальное время – сложные основные операции $b_3$ %. ($b_1 + b_2 + b_3 = 100$). Определить, сколько времени при изготовлении одного изделия затрачивается на выполнение различных операций?

6. При химической обработке кожа в зависимости от вида сырья дает различную усадку: один тип кожи дает усадку $a_1$ %, второй – $a_2$ %, третий – $a_3$ %. Партия кожи площадью *S м*$^2$ состоит из $b_1$ % сырья первого типа, $b_2$ % –

второго, остальное $b_3$ % – третьего ($b_1 + b_2 + b_3 = 100$). Определить предполагаемую площадь готовой продукции.

7. Скорость первого автомобиля $v_1$ *км/ч*, второго – $v_2$ *км/ч*, расстояние между ними – $S$ *км*. Определить расстояние между ними через $t$ *ч*, если автомобили двигаются в одном направлении и второй автомобиль вначале движения находился позади первого.

8. При обслуживании ткацкого станка работница проходит $a$ *м*. Переход от одного станка к другому составляет $b$ *м*. Средняя скорость движения – $S$ *км/ч*. Переход осуществляется только к соседнему станку. Сколько времени потребуется для обслуживания $n$ станков?

9. Отдыхающий жил неделю в гостинице. Причем $n$ дней он прожил в номере «Люкс» (его стоимость составляет $x$ *руб*. в сутки). Но в связи с финансовыми затруднениями он был вынужден оставшиеся дни жить в номере подешевле. Определите стоимость этого номера, если известно, что все проживание в гостинице отдыхающему обошлось в $k$ *руб*.

10. На проходившем в санатории выступлении хора присутствовало $a$ % от общего числа отдыхающих, на юмористический концерт собралось в 1,5 раза больше. Сколько зрителей побывало на этих мероприятиях, если всего в санатории отдыхало $x$ человек?

11. Даны координаты двух противоположных вершин прямоугольника: ($x_1$, $y_1$), ($x_2$, $y_2$). Стороны прямоугольника параллельны осям координат. Найти периметр и площадь данного прямоугольника.

12. Скорость лодки в стоячей воде – $v$ *км/ч*, скорость течения реки – $u$ *км/ч* ($u < v$). Время движения лодки по озеру – $T_1$ *ч*, а по реке (против течения) – $T_2$ *ч*. Определить путь $S$, пройденный лодкой.

13. Найти расстояние между двумя точками с заданными координатами ($x_1$, $y_1$) и ($x_2$, $y_2$) на плоскости.

14. Автомобиль едет из пункта А в пункт Б. Расстояние между ними – $S$ *км*. Первую половину пути автомобиль ехал со скоростью $a$ *км/ч*. С какой скоростью он проехал вторую половину пути, если добрался до пункта Б за $t$ *ч*?

15. Напряжение в цепи $U$ *В*, сопротивление $R_1$ *Ом*. Во сколько раз изменится ток, если последовательно $R_1$ включить сопротивление $R_2$?

# 3 ОПЕРАТОРЫ ЯЗЫКА С++

## 3.1 Составной оператор

*Составной оператор* – это группа операторов, отделённых друг от друга точкой с запятой, начинающихся с открывающей фигурной скобки **{** и заканчивающихся закрывающейся фигурной скобкой **}**:

```
{
    оператор1;
    ...
```

```
    операторN;
}
```

Компилятор воспринимает составной оператор как одно целое.

### 3.2 Условный оператор

Для организации разветвляющихся алгоритмов в C++ предусмотрен условный оператор **if**, который в общем виде записывается следующим образом:

```
if (условие)
    оператор1;
else
    оператор2;
```

Работает условный оператор следующим образом. Сначала вычисляется значение **условие**. Если оно не равно нулю, т.е. имеет значение истина (**true**), выполняется **оператор1**. В противном случае, когда выражение равно нулю, т.е. имеет значение ложь (**false**), выполняется **оператор2**.

**Оператор1** и **оператор2** могут быть составными.

ПРИМЕР. Написать программу решения квадратного уравнения $ax^2 + bx + c = 0$.

```
#include <iostream>
#include <math.h>
#include <windows.h>

using namespace std;

int main()
{
    SetConsoleOutputCP(1251);
    double a, b, c, d, x1, x2;
    cout << "Введите a, b, c:" << endl;
    cin >> a >> b >> c;
    d = b * b - 4 * a * c;
    if (d >= 0)
        {
            x1 = (-b + sqrt(d)) / (2 * a);
            x2 = (-b - sqrt(d)) / (2 * a);
            cout << "x1=" << x1 << endl;
            cout << "x2=" << x2;
        }
    else
        cout << "Вещественных корней нет!";
    return 0;
```

```
}
```

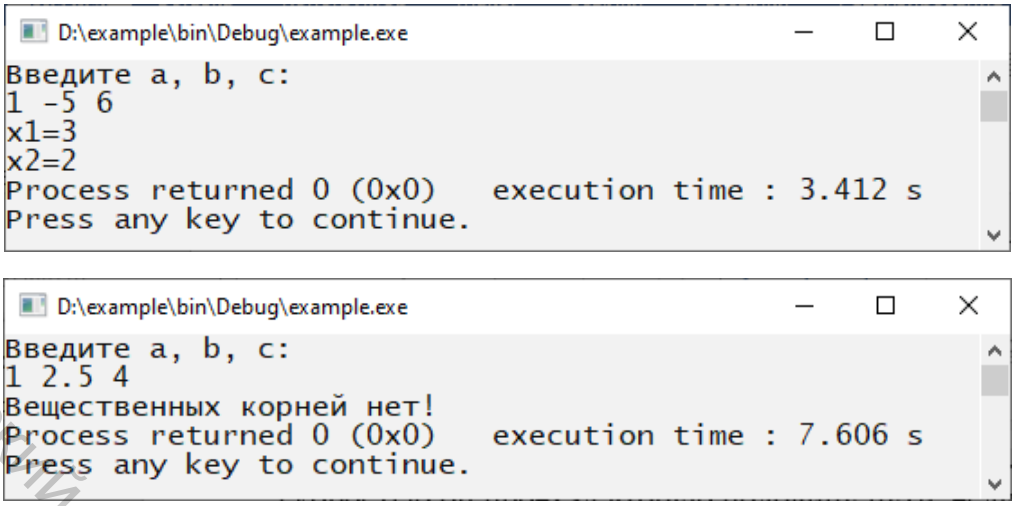



Рисунок 3.1 – Результат выполнения программы

Альтернативная ветвь **else** в условном операторе может отсутствовать, если в ней нет необходимости.

Условные операторы могут быть вложены друг в друга.

ПРИМЕР. Вычислить значение функции:

$$f(x)=\begin{cases}0, x<0\\ x^2, 0\le x\le 1\\ x, x>1\end{cases}$$

```
#include <iostream>
#include <math.h>
#include <windows.h>

using namespace std;

int main()
{
    SetConsoleOutputCP(1251);
    double x, f;
    cout << "Введите x" << endl;
    cin >> x;
    if (x < 0)
        f = 0;
    else if (x <= 1)
        f = pow(x, 2);
    else
        f = x;
    cout << "f(x)=" << f;
    return 0;
}
```

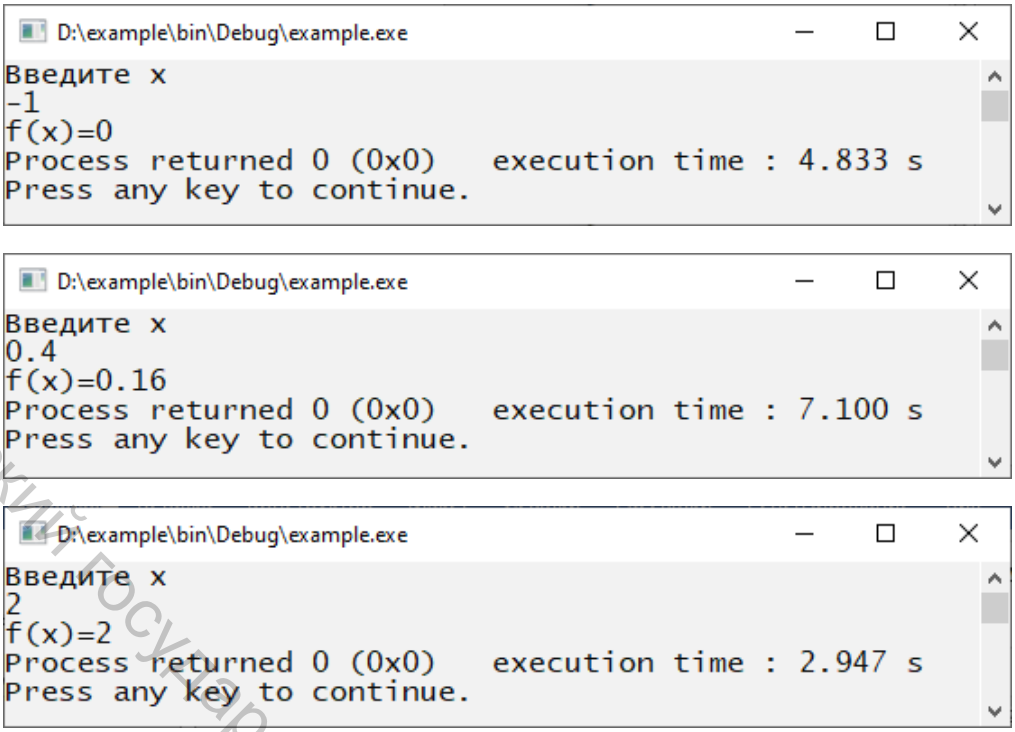



Рисунок 3.2 – Результат выполнения программы

### 3.3 Оператор выбора

Оператор варианта **switch** необходим в тех случаях, когда в зависимости от значений какой-либо переменной надо выполнить те или иные операторы:

```
switch (выражение)
{
    case значение1:
        операторы1;
        break;
    case значение2:
        операторы2;
        break;
    ...
    case значениеN:
        операторыN;
        break;
    default:
        операторы;
}
```

Оператор работает следующим образом. Вычисляется значение **выражения**. Если оно принимает **значение1**, то выполняются **операторы1**. Если выражение принимает **значение2**, то выполняются **операторы2** и так далее. Если **выражение** не принимает ни одно из значений, то выполняются операторы, расположенные после ключевого слова **default**.

Альтернативная ветвь **default** может отсутствовать.

Оператор **break** необходим для того, чтобы осуществить выход из

оператора **switch**. Если оператор **break** не указан, то будут выполняться следующие операторы из списка, несмотря на то, что значение, которым они помечены, не совпадает со значением выражения.

Отсутствие оператора **break** используется, как правило, для того, чтобы выполнить ту или иную ветвь в зависимости не от одного, а от нескольких значений. Выглядеть такая ветвь будет следующим образом:

```
...
case значение1:
case значение2:
case значение3:
    операторы;
    break;
...
```

ПРИМЕР. По заданному номеру дня недели ***n*** вывести его название.

```
#include <iostream>
#include <windows.h>

using namespace std;

int main()
{
    SetConsoleOutputCP(1251);
    int n;
    cout << "Введите номер дня недели: " << endl;
    cin >> n;
    switch (n)
    {
        case 1:
            cout << "Понедельник";
            break;
        case 2:
            cout << "Вторник";
            break;
        case 3:
            cout << "Среда";
            break;
        case 4:
            cout << "Четверг";
            break;
        case 5:
            cout << "Пятница";
            break;
        case 6:
            cout << "Суббота";
```

```
            break;
        case 7:
            cout << "Воскресенье";
            break;
        default:
            cout << "В неделе только 7 дней!";
    }
    return 0;
}
```

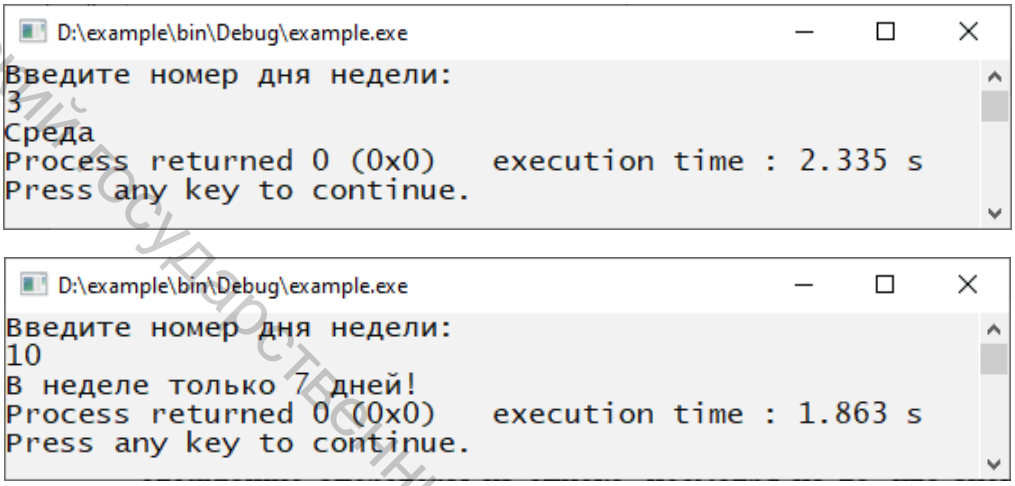



Рисунок 3.3 – Результат выполнения программы

### 3.4 Операторы цикла

*Цикл* – повторение одних и тех же действий. Последовательность действий, которые повторяются в цикле, называют *телом цикла*. Один проход цикла называют *шагом* или *итерацией*. Переменные, которые изменяются внутри цикла и влияют на его окончание, называются *параметрами* цикла.

В C++ для удобства пользователя предусмотрены три оператора, реализующих циклический процесс: **`while`**, **`do...while`** и **`for`**.

*Оператор цикла с предусловием*

Оператор цикла с предусловием имеет вид:

```
while (условие)
    оператор;
```

Работает цикл с предусловием следующим образом. Проверяется **`условие`**. Если оно истинно (не равно нулю), то выполняется **`оператор`**, и **`условие`** проверяется вновь. В противном случае цикл заканчивается, и управление передаётся оператору, следующему за телом цикла.

**`Оператор`** может быть составным.

ПРИМЕР. В последовательности чисел первое число – 1, а каждое следующее больше предыдущего на свой порядковый номер. То есть начало

последовательности имеет вид: 1, 3, 6, 10, 15, 21, … Написать программу, выписывающую все элементы последовательности, меньшие 100.

```
#include <iostream>

using namespace std;

int main()
{
    int number = 1, i = 1;
    while (number < 100)
    {
        cout << number << '\t';
        i++;
        number += i;
    }
    return 0;
}
```

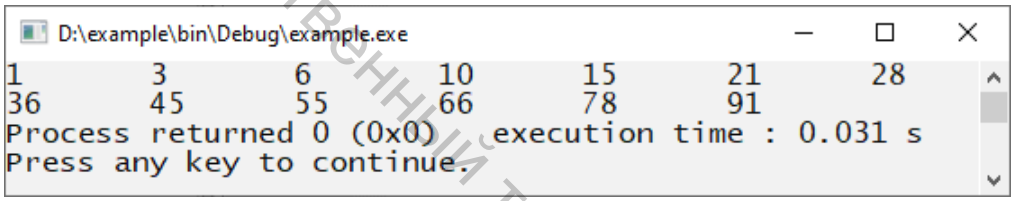



Рисунок 3.4 – Результат выполнения программы

*Оператор цикла с постусловием*

Оператор цикла с постусловием имеет вид:

```
do
    оператор;
while (условие);
```

Работает цикл следующим образом. Вначале выполняется **оператор**. Затем проверяется **условие**. Если оно истинно (не равно нулю), **оператор** выполняется снова. В противном случае цикл завершается, и управление передаётся оператору, следующему за циклом.

**Оператор** может быть составным.

Тело цикла с постусловием будет выполнен хотя бы один раз, в отличие от цикла с предусловием, который может не выполниться ни разу.

ПРИМЕР. Написать программу, реализующую игру «Угадай число». Загадывать число будет компьютер, используя генератор случайных чисел в диапазоне от 1 до 10.

```
#include <iostream>
#include <windows.h>
```

```
using namespace std;
int main()
{
    SetConsoleOutputCP(1251);
    int unknownNumber, enterNumber;
    unknownNumber = 1 + rand() % 10;
    do
    {
        cout << "Введите число: " << endl;
        cin >> enterNumber;
    }
    while (enterNumber != unknownNumber);
    cout << "Победа!!!" << endl;
    return 0;
}
```

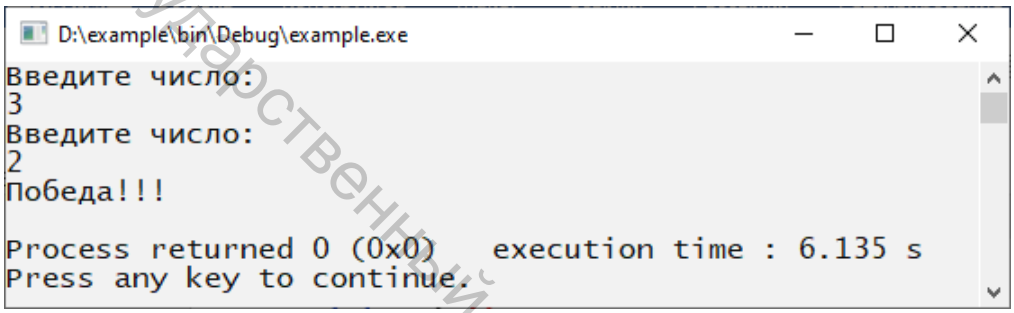



Рисунок 3.5 – Результат выполнения программы

*Оператор цикла со счетчиком*

Оператор цикла со счетчиком имеет вид:

```
for (начальные_присваивания; условие; последействие)
    оператор;
```

Здесь **начальные_присваивания** – оператор или группа операторов, разделённых запятой, которые применяются для присвоения начальных значений величинам, используемым в цикле, в том числе параметру цикла, и выполняются один раз в начале цикла. **Условие** – целое или логическое выражение, которое определяет условие выполнения тела цикла. Если **условие** истинно (не равно нулю), то тело цикла выполняется. **Последействие** – оператор или группа операторов, разделённых запятой, которые выполняются после каждой итерации и служат для изменения параметра цикла. **Оператор** – тело цикла. **Оператор** может быть составным.

**Последействие** или **оператор** должны влиять на **условие**, иначе цикл никогда не закончится. **Начальные_присваивания**, **условие** или **последействие** в записи оператора **for** могут отсутствовать, но при этом точки с запятой должны оставаться на своих местах.

ПРИМЕР. Написать программу, выводящую на экран таблицу значений функции $y = x^2$ на отрезке [0; 5] с шагом 0,5.

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

using namespace std;

int main()
{
    double x, y;
    cout << "x\ty" << endl;
    for (x = 0; x <= 5; x += 0.5)
    {
        y = x * x;
        cout << x << "\t" << y << endl;
    }
    return 0;
}
```

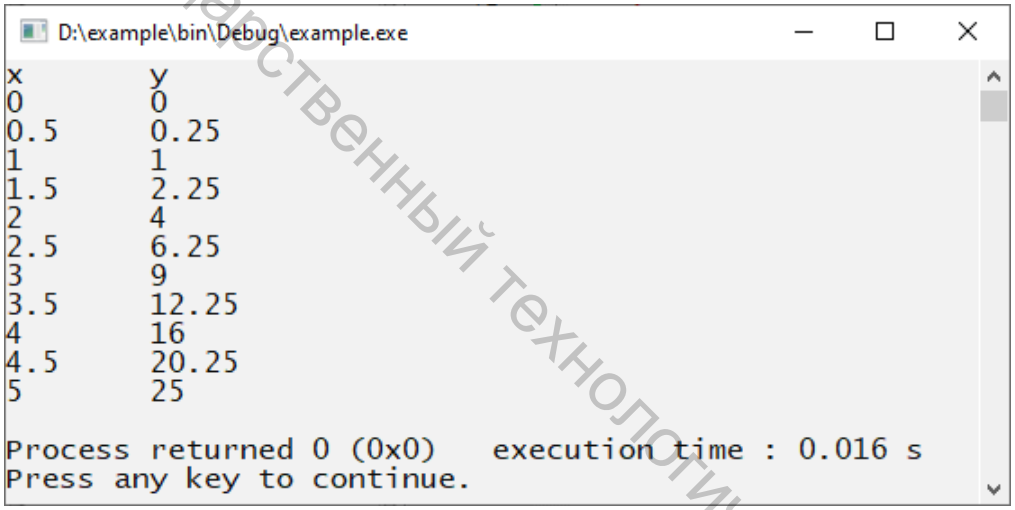



Рисунок 3.6 – Результат выполнения программы

### 3.5 Операторы передачи управления

Оператор **goto** передаёт управление оператору с меткой:

```
goto метка;
...
метка: оператор;
```

Оператор **break** осуществляет немедленный выход из циклов, а также из оператора выбора. Управление передаётся оператору, находящемуся непосредственно за циклом или оператором выбора.

Оператор **continue** начинает новую итерацию цикла, даже если предыдущая не была завершена.

Оператор **return выражение** завершает выполнение функции и передаёт управление в точку её вызова.

### 3.6 Рекуррентные вычисления сумм и произведений

Общая формулировка задач рекуррентного вычисления суммы или произведения элементов выглядит следующим образом:

$$S = \sum_{i=1}^{n} A_i \text{ или } P = \prod_{i=1}^{n} A_i ,$$

где $A_i = F\left(A_{i-1}\right)$. То есть последующее значение элемента $A$ вычисляется на основе предыдущего его значения.

Для вычисления суммы используется прием накопления суммы, состоящий в том, что в цикле каждый раз к промежуточной сумме прибавляем очередное слагаемое. Выполнив описанные вычисления $n$ раз, получим в переменной $S$ требуемое значение суммы. При этом перед циклом, в котором вычисляется сумма, переменной $S$ надо присвоить начальное значение, равное нулю.

Произведение может быть найдено с использованием аналогичного приема накопления произведения, состоящего в том, что в цикле каждый раз произведение предшествующих множителей $P$ умножается на очередной множитель. Начальное значение переменной $P$ перед циклом, в котором оно вычисляется, должно быть задано равным 1.

ПРИМЕР. Написать программу вычисления конечной суммы:

$$S = \sum_{i=2}^{n} (-1)^{i+2} \frac{(x+1)^{i+2}}{(2i-1)!} \sin\left(\frac{\pi}{i}\right).$$

В выражении для вычисления слагаемых выделим следующие части:

$$a = (-1)^{i+2} \; ; \; b = (x+1)^{i+2} ; \; c = (2i-1)! .$$

Таблица 3.1 – Нахождение первых значений $a$, $b$ и $c$

| $i$ | $a$ | $b$ | $c$ |
|---|---|---|---|
| 2 | 1 | $(x+1)^4$ | $3! = 1 \cdot 2 \cdot 3$ |
| 3 | -1 | $(x+1)^5$ | $5! = 1 \cdot 2 \cdot 3 \cdot 4 \cdot 5$ |
| 4 | 1 | $(x+1)^6$ | $7! = 1 \cdot 2 \cdot 3 \cdot 4 \cdot 5 \cdot 6 \cdot 7$ |

Можно заметить, что текущие значения $a$, $b$ и $c$ связаны с предыдущими следующим образом:

$$a_i = a_{i-1} \cdot (-1); \; b_i = b_{i-1} \cdot (x+1); \; c_i = c_{i-1} \cdot (2i-1) \cdot (2i-2).$$

```
#include <iostream>
#include <windows.h>
#include <math.h>

using namespace std;

int main()
{
    int i, n;
    double a, b, c, x, s;
    SetConsoleOutputCP(1251);
    cout << "Введите x, n: ";
    cin >> x >> n;
    a = 1;
    b = pow(x + 1, 4);
    c = 2 * 3;
    s = a * b / c * sin(M_PI / 2);
    for(i = 3; i <= n; i++)
    {
        a = -a;
        b *= x + 1;
        c *= (2 * i - 1) * (2 * i - 2);
        s += a * b / c * sin(M_PI / i);
    }
    cout << "s=" << s;
    return 0;
}
```

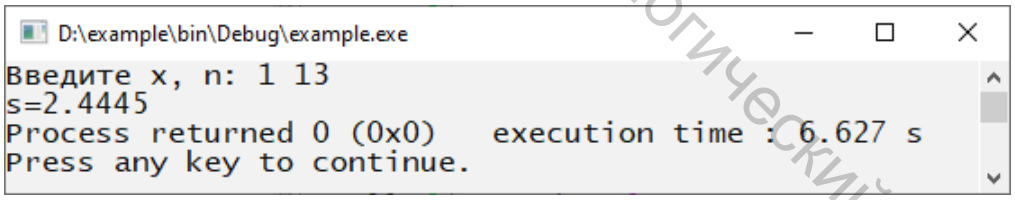



Рисунок 3.7 – Результат выполнения программы

В случае бесконечной суммы вычисление продолжается до тех пор, пока разница между предыдущим и последующим значением суммы не станет меньше, чем заданное пользователем значение точности ε:

$$|S_i - S_{i-1}| < \varepsilon \text{ или } |A_i| < \varepsilon .$$

ПРИМЕР. Написать программу вычисления с заданной точностью суммы бесконечного ряда:

$$S = \sum_{i=1}^{\infty} (-1)^{2i+1} \frac{i \cdot x^{3-2i}}{(i+3)!}.$$

В выражении для вычисления слагаемых выделим следующие части:

$$a = (-1)^{2i+1};\ b = x^{3-2i};\ c = (i+3)!.$$

Таблица 3.2 – Нахождение первых значений *a*, *b* и *c*

| *i* | *a* | *b* | *c* |
|---|---|---|---|
| 1 | -1 | $x$ | $4! = 1 \cdot 2 \cdot 3 \cdot 4$ |
| 2 | -1 | $x^{-1}$ | $5! = 1 \cdot 2 \cdot 3 \cdot 4 \cdot 5$ |
| 3 | -1 | $x^{-3}$ | $6! = 1 \cdot 2 \cdot 3 \cdot 4 \cdot 5 \cdot 6$ |

Можно заметить, что $a_i = -1$, а текущие значения *b* и *c* связаны с предыдущими следующим образом:

$$b_i = \frac{b_{i-1}}{x^2};\ c_i = c_{i-1} \cdot (i+3).$$

```
#include <iostream>
#include <windows.h>
#include <math.h>

using namespace std;

int main()
{
    int i = 1;
    double a, b, c, x, s, e, S = 0;
    SetConsoleOutputCP(1251);
    cout << "Введите x и точность e: ";
    cin >> x >> e;
    a = -1;
    b = x;
    c = 2 * 3 * 4;
    s = a * i * b / c;
    while(fabs(s) > e)
    {
        S += s;
        i++;
        b /= x * x;
        c *= i + 3;
        s = a * i * b / c;
    }
    cout << "s=" << S;
    return 0;
}
```

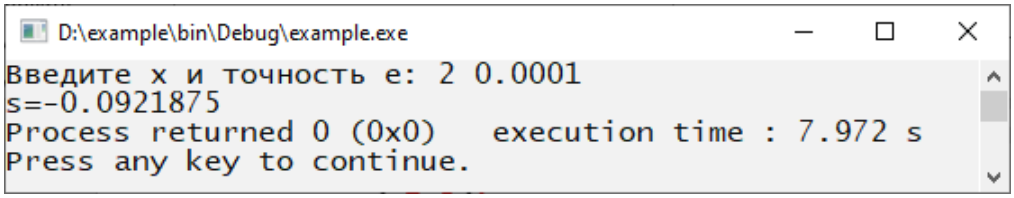



Рисунок 3.8 – Результат выполнения программы

## ЛАБОРАТОРНАЯ РАБОТА 6. РАЗВЕТВЛЯЮЩИЕСЯ АЛГОРИТМЫ

Написать программу решения задачи согласно варианту:

1. Элементы окружности пронумерованы следующим образом: 1 – радиус $R$; 2 – диаметр $D$; 3 – длина $L$; 4 – площадь круга $S$. Дан номер одного из этих элементов и его значение. Вывести значения остальных элементов данной окружности (в том же порядке).

2. В пункте быстрого питания продают гамбургеры по $N$ *руб*. и мороженое по $M$ *руб*. Составить программу, которая сообщает «Возьмите сдачу», «Доплатите ещё» или «Спасибо за покупку» при оплате покупателем за $K$ гамбургеров и $P$ мороженых денег в размере $S$ *руб*.

3. Даны целочисленные координаты точки на плоскости. Если точка совпадает с началом координат, то вывести 0. Если точка не совпадает с началом координат, но лежит на оси $OX$ или $OY$, то вывести соответственно 1 или 2. Если точка не лежит на координатных осях, то вывести 3.

4. Даны три переменные вещественного типа: $A$, $B$, $C$. Вывести их значения в порядке возрастания.

5. Даны координаты точки, не лежащей на координатных осях $OX$ и $OY$. Определить номер координатной четверти, в которой находится данная точка.

6. Дано целое число. Вывести его строку-описание вида «отрицательное четное число», «нулевое число», «положительное нечетное число» и т.д.

7. Арифметические действия над числами пронумерованы следующим образом: 1 – сложение, 2 – вычитание, 3 – умножение, 4 – деление. Дан номер действия $N$ (целое число в диапазоне 1 – 4) и вещественные числа $A$ и $B$ ($B$ не равно 0). Выполнить над числами указанное действие и вывести результат.

8. Элементы равнобедренного прямоугольного треугольника пронумерованы следующим образом: 1 – катет $a$; 2 – гипотенуза $c = \sqrt{2}\,a$; 3 – высота $h$, опущенная на гипотенузу ($h = c / 2$); 4 – площадь $S = c \cdot h / 2$. Дан номер одного из этих элементов и его значение. Вывести значения остальных элементов данного треугольника (в том же порядке).

9. Единицы длины пронумерованы следующим образом: 1 – дециметр, 2 – километр, 3 – метр, 4 – миллиметр, 5 – сантиметр. Дан номер единицы длины (целое число в диапазоне 1 – 5) и длина отрезка в этих единицах (вещественное число). Найти длину отрезка в метрах.

10. Типы соединения резисторов пронумерованы следующим образом: 1 – последовательное; 2 – параллельное. Дан номер соединения и сопротивления двух резисторов $R_1$ и $R_2$. Найти эквивалентное сопротивление этих резисторов.

11. Даны целые числа *a*, *b*, *c*, являющиеся сторонами некоторого треугольника. Вывести на экран заключение о том, каким является этот треугольник (равносторонним, равнобедренным или обычным).

12. Даны целые числа *a*, *b*, *c*. Вывести на экран 1 – если каждое из них отлично от другого, 2 – если только два числа одинаковы и 3 – если одинаковы все три числа.

13. В лавке реализуются два товара. Цена первого $p_1$ *руб*., второго – $p_2$ *руб*. Покупатель купил $n_1$ единиц первого товара и $n_2$ единиц – второго. Написать программу вычисления стоимости покупки с учетом скидки. Скидка в 3 % предоставляется, если сумма покупки больше 100 *руб*., в 5 % – если сумма покупки больше 200 *руб*.

14. Локатор ориентирован на одну из сторон света (1 – север, 2 – запад, 3 – юг, 4 – восток) и может принимать три цифровые команды поворота: 1 – поворот налево, –1 – поворот направо, 2 – поворот на 180°. Дан символ С – исходная ориентация локатора и целые числа $N_1$ и $N_2$ – две посланные команды. Вывести ориентацию локатора после выполнения этих команд.

15. Дан номер года (положительное целое число). Определить количество дней в этом году, учитывая, что обычный год насчитывает 365 дней, а високосный – 366 дней. Високосным считается год, делящийся на 4, за исключением тех годов, которые делятся на 100 и не делятся на 400 (например, годы 300, 1300 и 1900 не являются високосными, а 1200 и 2000 – являются).

## ЛАБОРАТОРНАЯ РАБОТА 7. ТАБУЛИРОВАННЫЕ ФУНКЦИИ

Написать программу вычисления таблицы значений функции *f(x)* на промежутке от *a* до *b* с шагом *h*.

1. $f(x)=\begin{cases}-x, x\le 0\\ x^3, 0<x\le 1\\ x, x>1\end{cases}$

2. $f(x)=\begin{cases}\ln\left(1+x^2\right), x\le e\\ x, e<x\le 9\\ 9e^{8-x}, x>9\end{cases}$

3. $f(x)=\begin{cases}-1, x\le 0\\ 3x^2, 0<x\le 1\\ 1, x>1\end{cases}$

4. $f(x)=\begin{cases}(x+1)^4, x\le -1\\ 1-\cos(\pi x), -1<x\le 1\\ -(x-1)^2, x>1\end{cases}$

5. $f(x)=\begin{cases}|x|, x\le 1\\ \cos\pi x, 1<x\le 2\\ (2-x)^3, x>2\end{cases}$

6. $f(x)=\begin{cases}-x, x\le -1\\ x^3, -1<x\le 1\\ e^{1-x}, x>1\end{cases}$

7. $f(x)=\begin{cases}|x|^x, x\le 0\\ \ln(x+1), 0<x\le 1\\ x, x>1\end{cases}$

8. $f(x)=\begin{cases}e^x, x\le 0\\ \sin \pi x, 0<x\le 1\\ x-1, x>1\end{cases}$

9. $f(x)=\begin{cases}\sqrt{x+1}, x\le 1\\ \frac{x^x}{9}, 1<x\le 3\\ (x-4)^2, x>3\end{cases}$

10. $f(x)=\begin{cases}x^2, x\le 1\\ \frac{\pi}{2}-\mathrm{x}, 1<x\le \frac{\pi}{2}\\ \sin(x), x>\frac{\pi}{2}\end{cases}$

11. $f(x)=\begin{cases}2^{x-1}, x\le 0\\ \sin\frac{\pi x}{2}, 0<x\le 1\\ x+1, x>1\end{cases}$

12. $f(x)=\begin{cases}x\sin(x), x\le 0\\ \sqrt{x}, 0<x\le 1\\ \cos(\pi x), x>1\end{cases}$

13. $f(x)=\begin{cases}\sqrt{|x|}, x\le 1\\ 1, 1<x\le 3\\ (x-4)^2, x>3\end{cases}$

14. $f(x)=\begin{cases}x, x\le 2\\ \ln(x), 2<x\le 4\\ \sin\left(\frac{\pi x}{2}\right), x>4\end{cases}$

15. $f(x)=\begin{cases}x^2, x\le 0\\ \frac{x^3}{3}, 0<x\le 1\\ x, x>1\end{cases}$

## ЛАБОРАТОРНАЯ РАБОТА 8. ЦИКЛИЧЕСКИЕ АЛГОРИТМЫ

Написать программу вычисления значения суммы либо произведения согласно варианту. В программе использовать цикл **`for`**.

1. $S=\sum_{i=1}^{10}\frac{x^2}{4i^2-1}$

2. $P=\prod_{i=1}^{10}\frac{x^i}{i(i+1)}$

3. $S=\sum_{i=1}^{10}\frac{x^i}{i}$

4. $P=\prod_{i=2}^{15}\frac{x^{i-1}}{i^3-1}$

5. $S=\sum_{i=0}^{10}\frac{x^i}{4^i}$

6. $P=\prod_{i=1}^{10}\frac{x+i}{i}$

7. $S=\sum_{i=3}^{8}\frac{\sin^2 x^2}{i^4}$

8. $P=\prod_{i=7}^{14}\frac{i+x}{x}$

9. $S=\sum_{i=11}^{20}\frac{x-i}{i^2}$

10. $P=\prod_{i=5}^{15}\frac{1}{\cos^2(ix)}$

11. $S=\sum_{i=1}^{10}\frac{i^2}{x+i}$

12. $P=\prod_{i=1}^{10}\frac{ix}{i+x}$

13. $S=\sum_{i=1}^{5}\frac{i+x}{\sin i}$

14. $P=\prod_{i=1}^{6}\frac{\sin^2(ix)+x}{\cos i^2}$

15. $S=\sum_{i=-2}^{2}\frac{5i-x^2}{i^2+1}$

## ЛАБОРАТОРНАЯ РАБОТА 9. ВЫЧИСЛЕНИЯ С ЗАДАННОЙ ТОЧНОСТЬЮ

Написать программу вычисления суммы элементов бесконечного ряда с заданной точностью *e*. Использовать цикл **`while`** или **`do while`**.

1. $S=\sum_{n=1}^{\infty}(-1)^{n-1}\frac{x^{4n-2}}{(2n-1)!}$ 2. $S=\sum_{n=1}^{\infty}\frac{(x+3)^{n+1}\sin^n(x)}{(2n)!}$ 3. $S=\sum_{n=0}^{\infty}\frac{2^n x^{6n}}{n!}$

4. $S=\sum_{n=1}^{\infty}\frac{\sin^{n+1}(x+2)}{(2n)!}$ 5. $S=\sum_{n=1}^{\infty}\frac{x^{n+1}}{(n+1)!}$ 6. $S=\sum_{n=1}^{\infty}\frac{x^{n+1}}{(2n-1)!}$

7. $S=\sum_{n=1}^{\infty}(-1)^n\frac{x^n}{(n+1)!}$ 8. $S=\sum_{n=0}^{\infty}(-1)^n\frac{(5x)^{2n}}{(2n)!}$ 9. $S=\sum_{n=1}^{\infty}\frac{x^{2n-1}}{(2n-1)!}$

10. $S=\sum_{n=0}^{\infty}(-1)^n\frac{x^{2n}\ln^n(2)}{n!}$ 11. $S=\sum_{n=1}^{\infty}(-1)^n\frac{x^{n+1}}{(2n)!}$ 12. $S=\sum_{n=0}^{\infty}\frac{3^n x^n}{n!}$

13. $S=\sum_{n=1}^{\infty}(-1)^{n-1}\frac{3^{2n-1}x^{2n-2}}{(2n-1)!}$ 14. $S=\sum_{n=0}^{\infty}\frac{(n^2+1)\left(\frac{x}{2}\right)^n}{n!}$ 15. $S=\sum_{n=1}^{\infty}(-1)^n\frac{x^n}{2^n n!}$

# 4 МАССИВЫ В ЯЗЫКЕ С++

*Массив* – это структурированный тип данных, состоящий из фиксированного числа элементов одного типа, объединенных единым именем.

## 4.1 Статические массивы

Одномерный статический массив описывается следующим образом:

```
тип имя[n];
```

**n** – количество элементов в массиве, которое может быть задано числом либо массивом.

Например:

```
int A[6];
const int n = 15;
double B[n];
```

Доступ к каждому элементу массива осуществляется с помощью индекса – порядкового номера элемента. Для обращения к элементу массива указывают его имя, а затем в квадратных скобках индекс.

Индексация элементов в массивах начинается с 0!

Можно инициализировать массив при объявлении. Например:

```
int a[6] = {1, 4, -8, 3, 0, 6};
```

или

```
int a[] = {1, 4, -8, 3, 0, 6};
```

Если при инициализации указать меньше значений, чем размер массива, последние элементы заполнятся нулями.

ПРИМЕР. Написать программу консольного ввода и вывода массива из 5 элементов.

```
#include <iostream>
#include <windows.h>

using namespace std;

int main()
{
    SetConsoleOutputCP(1251);
    int i, a[5];

    // ввод массива
    for(i = 0; i < 5; i++)
    {
        cout << "a[" << i << "]=";
        cin >> a[i];
    }

    // вывод массива
    cout << "Массив A {";
    for(i = 0; i < 5; i++)
        cout << a[i] << ", ";
    cout << "\b\b}" << endl;
    return 0;
}
```

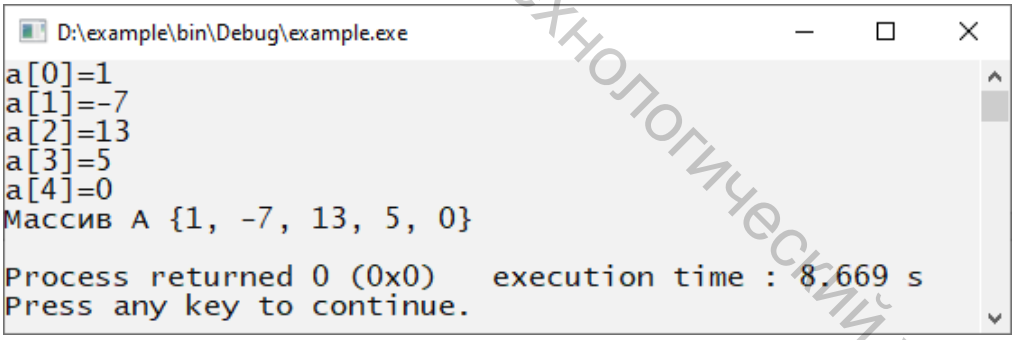



Рисунок 4.1 – Результат выполнения программы

ПРИМЕР. Задан массив из 10 случайных чисел в диапазоне от 0 до 20. Написать программу поиска индекса и значения максимального элемента.

```
#include <iostream>
#include <windows.h>

using namespace std;

int main()
{
    SetConsoleOutputCP(1251);
```

```
    int i, a[10], i_max = 0;

    for (i = 0; i < 10; i++)
        a[i] = rand() % 20;

    cout << "Массив A {";
    for (i = 0; i < 10; i++)
        cout << a[i] << ", ";
    cout << "\b\b}" << endl;

    for (i = 1; i < 10; i++)
        if (a[i] > a[i_max])
            i_max = i;

    cout << "Максимальный элемент a[" << i_max << "]=" << a[i_max];
    return 0;
}
```

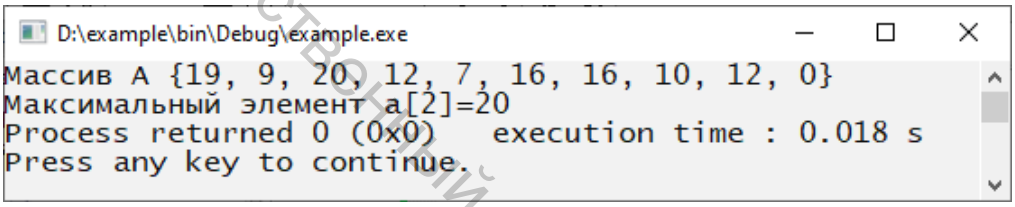



Рисунок 4.2 – Результат выполнения программы

Двумерный статический массив описывается следующим образом:

```
тип имя[m][n];
```

`m` – количество строк в массиве;

`n` – количество столбцов в массиве.

Например:

```
int a[2][3];
```

Пример инициализации двумерного массива:

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

ПРИМЕР. Написать программу вычисления в двумерном массиве размерности 3x4 суммы элементов, кратных 5.

```
#include <iostream>
#include <windows.h>

using namespace std;

int main()
{
    SetConsoleOutputCP(1251);
```

```
    int i, j, a[3][4] , s = 0;

    // ввод массива
    for (i = 0; i < 3; i++)
        for (j = 0; j < 4; j++)
        {
            cout << "a[" << i << "][" << j << "]=";
            cin >> a[i][j];
        }

    // вывод массива
    cout << "Матрица A:" << endl;
    for (i = 0; i < 3; i++)
    {
        for (j = 0; j < 4; j++)
            cout << a[i][j] << "\t";
        cout << endl;
    }

    for (i = 0; i < 3; i++)
        for (j = 0; j < 4; j++)
            if (a[i][j] % 5 == 0)
                s += a[i][j];
    cout << "Сумма элементов кратных 5 равна " << s;

    return 0;
}
```

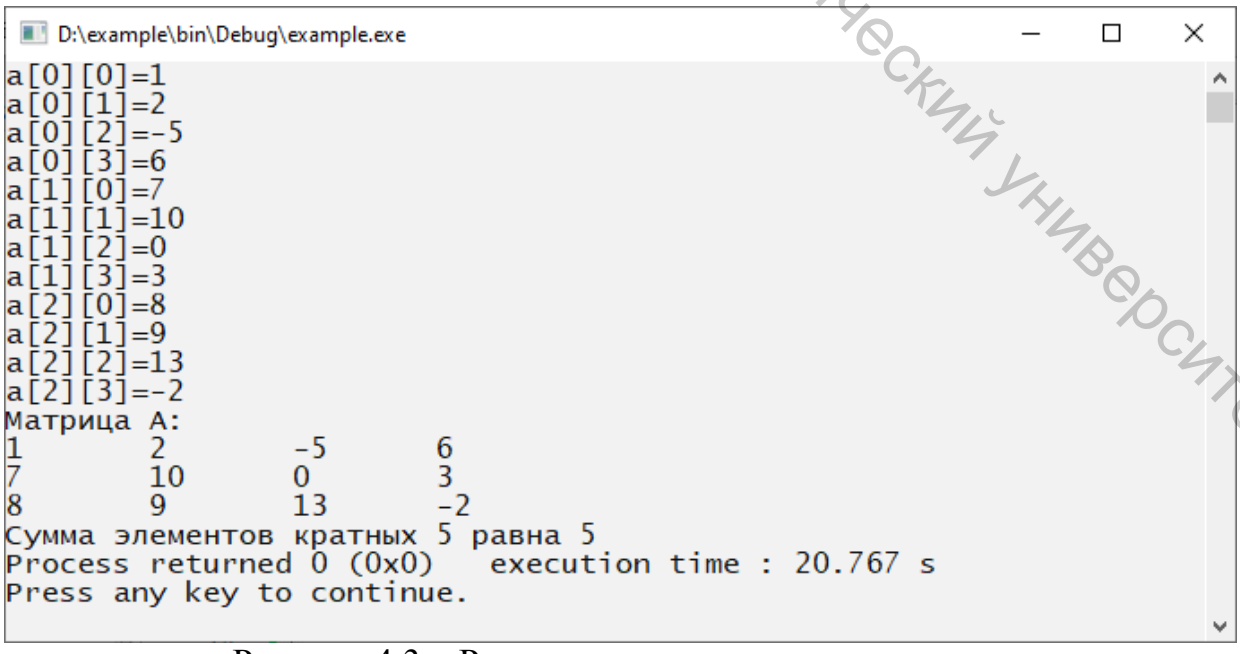



Рисунок 4.3 – Результат выполнения программы

### 4.2 Динамические массивы

Для создания динамического массива необходимо:

– описать указатель (**тип *имя_указателя;**);

– определить размер массива;

– выделить участок памяти для хранения массива и присвоить указателю адрес этого участка памяти.

Наиболее предпочтительным подходом к динамическому распределению памяти является использование операций языка C++ **new** и **delete**.

Операция **new** выделяет память из области свободной памяти, а операция **delete** высвобождает выделенную память.

Минимальный набор действий, необходимых для динамического размещения одномерного массива действительных чисел размером $n$:

```
int n;
double *a;
     . . .
a = new double[n]; // Захват памяти для n элементов
     . . .
delete []a;    // Освобождение памяти
```

Минимальный набор действий, необходимых для динамического размещения двухмерного массива действительных чисел размером $m \times n$:

```
int i, n, m;   // m, n – размеры массива
double **a;
     . . .
a = new double *[m];     // Захват памяти под указатели
for (i = 0; i < m; i++)
     a[i] = new double[n]; // и под элементы
     . . .
for (i = 0; i < m; i++)
    delete []a[i]; // Освобождение памяти
delete []a;
```

ПРИМЕР. Задана целочисленная матрица $A$ размером $m$x$n$. Получить вектор $B$, присвоив его $j$-му элементу значение 1, если все элементы $j$-го столбца матрицы положительные, –1 – если все элементы $j$-го столбца матрицы отрицательные, и 0 – в остальных случаях.

```
#include <iostream>
#include <windows.h>

using namespace std;

int main()
{
    SetConsoleOutputCP(1251);
```

```
int i, j, m, n, **A, *B;

cout << "Введите m и n: ";
cin >> m >> n;

A = new int *[m];
for (i = 0; i < m; i++)
     A[i] = new int[n];

cout << "Матрица A:" << endl;
for (i = 0; i < m; i++)
{
    for (j = 0; j < n; j++)
    {
        A[i][j] = rand() % 10 - 3;
        cout << A[i][j] << "\t";
    }
    cout << endl;
}

B = new int[n];

cout << "Вектор B:" << endl;
for (j = 0; j < n; j++)
{
    B[j] = 1;
    for (i = 0; i < m; i++)
        if (A[i][j] <= 0)
        {
            B[j] = 0;
            break;
        }
    if (B[j] == 1)
    {
        cout << B[j] << "\t";
        continue;
    }
    B[j] = -1;
    for (i = 0; i < m; i++)
        if (A[i][j] >= 0)
        {
            B[j] = 0;
            break;
        }
    cout << B[j] << "\t";
}
```

```
    for (i = 0; i < m; i++)
        delete []A[i];
    delete []A;
    delete []B;

    return 0;
}
```

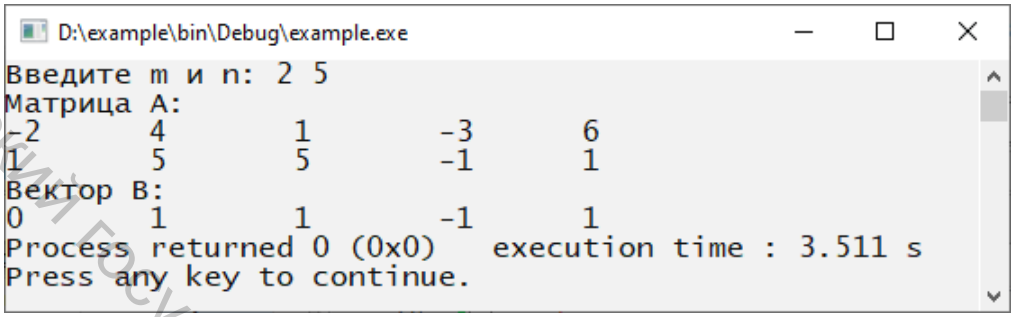



Рисунок 4.4 – Результат выполнения программы

## ЛАБОРАТОРНАЯ РАБОТА 10. ОДНОМЕРНЫЕ МАССИВЫ

Написать программу вычисления в одномерном целочисленном массиве:

1. Произведения элементов массива, расположенных между максимальным и минимальным элементами.
2. Суммы элементов массива, расположенных между первым и последним нулевыми элементами.
3. Суммы элементов массива, расположенных до последнего положительного элемента.
4. Суммы элементов массива, расположенных между первым и последним положительными элементами.
5. Произведения элементов массива, расположенных между первым и вторым нулевыми элементами.
6. Суммы элементов массива, расположенных между первым и вторым отрицательными элементами.
7. Суммы элементов массива, расположенных до минимального элемента.
8. Суммы элементов массива, расположенных после последнего отрицательного элемента.
9. Суммы элементов массива, расположенных после последнего элемента, равного нулю.
10. Номера элемента с наименьшим отклонением от среднего арифметического всех элементов.
11. Суммы элементов, не превосходящих среднее арифметическое всех элементов.
12. Количества четных элементов между минимальным и

максимальным элементами.

13. Количества положительных элементов между первым и последним нулями.

14. Произведения элементов после последнего нулевого элемента.

15. Количества нечетных элементов между первым и последним положительным элементом.

## ЛАБОРАТОРНАЯ РАБОТА 11. ДВУМЕРНЫЕ МАССИВЫ

Написать программу решения задачи согласно варианту:

1. Дана целочисленная прямоугольная матрица. Определить количество строк, не содержащих ни одного нулевого элемента.

2. Дана целочисленная прямоугольная матрица. Определить количество столбцов, содержащих, по крайней мере, один нулевой элемент.

3. Дана целочисленная прямоугольная матрица. Определить произведение элементов в тех строках, которые не содержат отрицательных элементов.

4. Дана целочисленная прямоугольная матрица. Определить сумму элементов в тех строках, которые содержат отрицательные элементы.

5. Дана целочисленная прямоугольная матрица. Вывести на экран номер первого столбца, содержащего нулевой элемент.

6. Дана целочисленная прямоугольная матрица. Найти номер первого столбца, не содержащего ни одного отрицательного элемента.

7. Дана целочисленная прямоугольная матрица. Найти количество столбцов, среднее арифметическое элементов которых меньше заданной величины.

8. Дана целочисленная прямоугольная матрица. Определить сумму элементов строк, не содержащих ни одного нулевого элемента.

9. Дана целочисленная прямоугольная матрица. Определить сумму элементов столбцов, содержащих, по крайней мере, один нулевой элемент.

10. Дана целочисленная прямоугольная матрица. Определить сумму положительных элементов, лежащих выше главной диагонали.

11. Дана целочисленная прямоугольная матрица. Определить сумму отрицательных элементов, лежащих ниже главной диагонали.

12. Дана целочисленная прямоугольная матрица. Найти номер последнего из ее столбцов, содержащих только положительные элементы.

13. Дана целочисленная прямоугольная матрица. Найти номер последней из ее строк, содержащих нулевой элемент.

14. Дана целочисленная прямоугольная матрица. Найти количество ее столбцов, которые состоят только из чётных элементов.

15. Дана целочисленная квадратная матрица. Поменять местами главную диагональ и первый столбец.

## ЛАБОРАТОРНАЯ РАБОТА 12. ДИНАМИЧЕСКИЕ МАССИВЫ

Написать программу решения задачи согласно варианту. Использовать динамическое размещение данных:

1. Дана матрица $A$, размером $m \times n$. Заполнить одномерный массив, найдя сумму положительных элементов в каждом столбце матрицы.
2. Дана матрица $A$, размером $m \times n$. Заполнить одномерный массив, найдя произведение положительных элементов в каждом столбце матрицы.
3. Дана матрица $A$, размером $m \times n$. Найти минимальный элемент в каждой строке матрицы. Затем каждую строку матрицы разделить на минимальный элемент строки.
4. Дана матрица $A$, размером $m \times n$. Заполнить одномерный массив, найдя минимальный среди положительных элементов в каждой строке матрицы.
5. Дана матрица $A$, размером $m \times n$. Заполнить одномерный массив, найдя количество положительных элементов в каждом столбце матрицы.
6. Дана матрица $A$, размером $m \times n$. Заполнить одномерный массив, найдя минимальный среди отрицательных элементов в каждой строке матрицы.
7. Дана матрица $A$, размером $m \times n$. Заполнить одномерный массив, найдя среднее арифметическое положительных элементов в каждом столбце матрицы.
8. Дана матрица $A$, размером $m \times n$. Найти минимальный элемент в каждой строке матрицы. Затем каждую строку матрицы умножить на минимальный элемент строки.
9. Дана матрица $A$, размером $m \times n$. Заполнить одномерный массив, найдя среднее геометрическое положительных элементов в каждом столбце матрицы.
10. Дана матрица $A$, размером $m \times n$. Найти минимальный элемент в каждой строке матрицы. Затем к каждому элементу каждой строки прибавить минимальный элемент строки.
11. Дана матрица $A$, размером $m \times n$. Найти минимальный элемент и его номер в каждой строке матрицы. Затем из каждого элемента каждой строки вычесть номер минимального элемента строки.
12. Дана матрица $A$, размером $m \times n$. Заполнить одномерный массив, найдя сумму отрицательных элементов в каждом столбце матрицы.
13. Дана матрица $A$, размером $m \times n$. Найти номер минимального элемента в каждой строке матрицы. Затем к каждому элементу каждой строки прибавить номер минимального элемента строки.
14. Дана матрица $A$, размером $m \times n$. Заполнить одномерный массив, найдя произведение отрицательных элементов в каждом столбце матрицы.
15. Дана матрица $A$, размером $m \times n$. Заполнить одномерный массив, найдя количество отрицательных элементов в каждом столбце матрицы.

# 5 ФУНКЦИИ В ЯЗЫКЕ С++

## 5.1 Общие сведения о функциях

*Подпрограмма* – именованная, логически законченная группа операторов языка, которую можно вызвать для выполнения любое количество раз из различных мест программы. В языке С++ подпрограммы реализованы в виде функций.

Как известно, программа на С++ состоит из одной или нескольких функций. Функция должна быть описана перед своим использованием.

Описание функции состоит из заголовка и тела функции.

```
тип имя_функции([список_параметров]) // заголовок функции
{
    операторы; // тело функции
}
```

**Тип** функции показывает, какое значение возвращает функция: целое, вещественное, строковое и так далее.

**список_параметров** содержит перечень типов и имен величин, передаваемых в функцию и разделенных запятыми. Если функция не имеет параметров, то скобки все равно необходимы, хотя в них ничего не указывается.

В случае, если вызываемые функции определены после функции **main** *или даже в другом файле***,** необходимо до первого вызова функции (перед функцией **main**) сообщить программе тип возвращаемого функцией значения, а также количество и типы ее параметров. Подобное сообщение называется *прототипом функции*.

Прототип может полностью совпадать с заголовком функции, хотя при объявлении функции компилятору необходимо знать только имя функции, количество аргументов, их типы и тип возвращаемого функцией значения. Поэтому имена параметров необязательно указывать в прототипах функций.

Если возврата значения функцией не требуется, то тип функции – **void**.

*Возврат значений* функцией выполняется с помощью оператора **return**:

```
return значение;
```

Если тип функции – **void**, то оператор возврата просто – **return;** или его можно даже опустить в теле функции.

**Return** вызывает немедленный выход из функции и возврат в вызвавшую ее подпрограмму.

Функция возвращает единственное скалярное значение. Это может быть число, адрес, структура.

Для вызова функции в программе пишется имя функции, за которым в скобках следует список аргументов, которые являются переменными и константами, определенными в программе. При этом происходит выполнение тела функции, в котором параметры заменяются значениями аргументов. Тип аргумента должен соответствовать типу соответствующего параметра или

допускать приведение к типу параметра.

ПРИМЕР. Написать программу для вычисления расстояния между двумя точками по их координатам.

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

using namespace std;

double R(double X1, double Y1, double X2, double Y2)
{
    return sqrt(pow(X2 - X1, 2) + pow(Y2 - Y1, 2));
}

int main()
{
    double x1, x2, y1, y2, r;
    cout << "Input x1, y1, x2, y2:" << endl;
    cin >> x1 >> y1 >> x2 >> y2;
    r = R(x1, y1, x2, y2);
    cout << "r=" << r << endl;
    return 0;
}
```

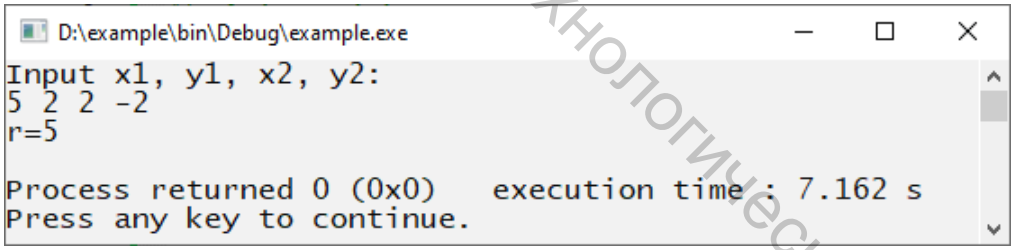



Рисунок 5.1 – Результат выполнения программы

ПРИМЕР. Написать программу для вывода таблицы умножения на заданное число.

```
#include <iostream>

using namespace std;

void multab(int);

int main()
{
    int n;
    cout << "Input n:" << endl;
```

```
    cin >> n;
    multab(n);
    return 0;
}

void multab(int n)
{
    for (int i = 1; i <= 10; i++)
        cout << n << "*" << i << "=" << n * i << endl;
}
```


Рисунок 5.2 – Результат выполнения программы

### 5.2 Рекурсия

*Рекурсия* – вызов функции из неё же самой, непосредственно (простая рекурсия) или через другие функции (сложная или косвенная рекурсия).

ПРИМЕР. Написать программу, использующую рекурсивную функцию, вычисляющую факториал числа.

```
#include <iostream>

using namespace std;

long long int fact(int n)
{
    if (n == 1)
        return n;
    else
        return n * fact(n - 1);
}
```

```
int main()
{
    int n;
    cout << "n=";
    cin >> n;
    cout << n << "!=" << fact(n);
    return 0;
}
```

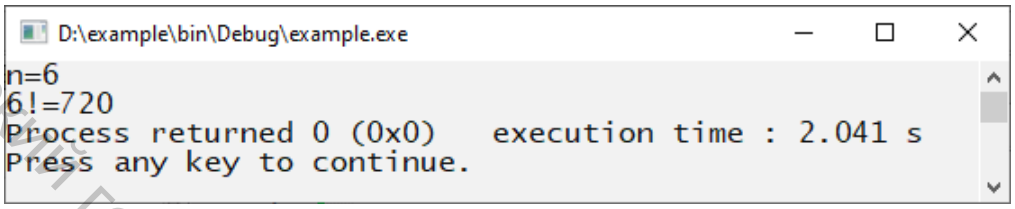



Рисунок 5.3 – Результат выполнения программы

### 5.3 Массивы и функции

Массив передаётся в функцию как указатель. Причём неважно, какой это массив: статический или динамический. Также обычно в функцию необходимо передать размер массива (количество элементов), поскольку в общем случае, имея только указатель, невозможно определить размер массива.

ПРИМЕР. Написать программу, которая будет вызывать функцию для вывода на печать элементов одномерного массива.

```
#include <iostream>

using namespace std;

void print(int x[], int n)
{
    cout << "Massiv:" << endl;
    for (int i = 0; i < n; i++)
        cout << x[i] << '\t';
}

int main()
{
    const int n = 5;
    int x[n] = {3, 5, 1, 7, 4};
    print(x, n);
    return 0;
}
```

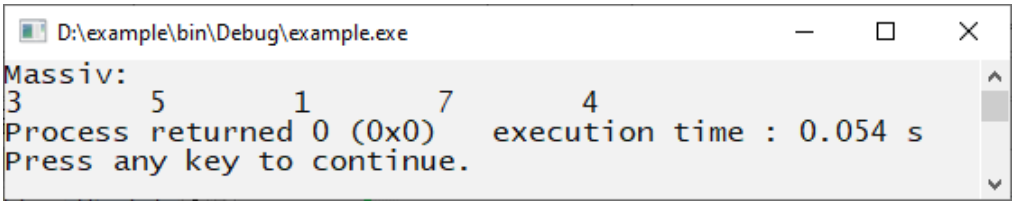



Рисунок 5.4 – Результат выполнения программы

Заголовок может выглядеть и так:

**void print(int *x, int n)**

Передача в подпрограмму матрицы в качестве параметра несколько сложнее, чем передача одномерного массива. Когда мы передаем многомерные массивы, необходимо указывать все размерности, за исключением первой, то есть в списке параметров допустима запись:

**тип_данных имя[][константа]**

Это приемлемо только для небольших программ с матрицами одинакового размера.

Решением является использование динамического массива. В этом случае в подпрограмму будет передаваться указатель на указатель.

ПРИМЕР. Дана прямоугольная матрица. Написать программу, выводящую на консоль сумму элементов данной матрицы.

```
#include <iostream>
#include <windows.h>

using namespace std;

int sumArr(int **a, int m, int n)
{
    int s = 0;
    for (int i = 0; i < m; i++)
        for (int j = 0; j < n; j++)
            s += a[i][j];
    return s;
}

int main()
{
    SetConsoleOutputCP(1251);
    int **a, i, j, m, n;
    cout << "Введите размеры матрицы: ";
    cin >> m >> n;
    a = new int *[n];
        for (i = 0; i < n; i++)
            a[i] = new int[m];
```

```
    for (i = 0; i < m; i++)
        for (j = 0; j < n; j++)
            a[i][j] = rand() % 10;

    for (i = 0; i < m; i++)
    {
        for (j = 0; j < n; j++)
            cout << a[i][j] << '\t';
        cout << endl;
    }

    cout << "Сумма элементов матрицы равна " << sumArr(a, m, n);

    for (i = 0; i < n; i++)
        delete []a[i];
    delete []a;

    return 0;
}
```

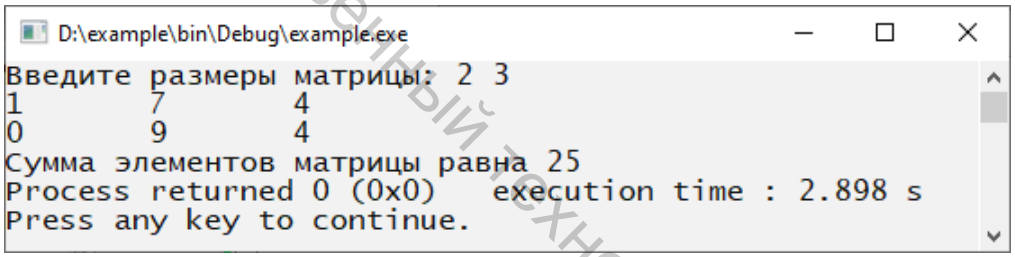



Рисунок 5.5 – Результат выполнения программы

## ЛАБОРАТОРНАЯ РАБОТА 13. ФУНКЦИИ

Написать программу решения задачи согласно варианту:

1. Описать функцию fibonacci(n), вычисляющую n-е число Фибоначчи. Числа Фибоначчи определяются следующим образом: $f_1 = 0, f_2 = 1, f_3 = 1, \ldots, f_n = f_{n-1} + f_{n-2}$. С помощью этой функции заполнить массив первыми 15 числами Фибоначчи.

2. Описать функцию invertDigits(K), меняющую порядок следования цифр целого положительного числа $K$ на обратный. С помощью этой функции поменять порядок следования цифр на обратный для каждого из пяти заданных целых чисел.

3. Описать функцию exp1(x, e), находящую приближенное значение функции $y = e^x$ по формуле $y = \sum_{i=0}^{\infty} \frac{x^i}{i!}$. В сумме учитывать все слагаемые, большие $e$ ($0 < e < 1$). С помощью этой функции найти приближенное значение экспоненты для заданного $x$ при шести различных $e$.

4. Описать функцию convertTime(T), возвращающую по времени $T$ (в секундах) строку в виде HH:MM:SS, где HH – количество часов, MM – количество минут и SS – количество секунд во временном отрезке. Используя эту функцию, получить результат для пяти заданных отрезков времени.

5. Описать функцию root(a, n), вычисляющую значение $x = \sqrt{a}$ по формуле $x_n = \frac{1}{2}\left( x_{n-1} + \frac{a}{x_{n-1}} \right)$. В качестве начального приближения использовать значение $x_1 = \frac{1}{2}(1 + a)$. С помощью этой функции вычислить корень для пяти различных $a$ при заданном $n$.

6. Описать функцию isPalindrom(K), возвращающую *true*, если целый параметр $K$ (>0) является палиндромом (последовательность составляющих его чисел читается одинаково слева направо и справа налево), и *false* – в противном случае. С помощью этой функции найти количество палиндромов в наборе из 5 целых положительных чисел.

7. Описать функцию element(x, i, j), возвращающую по индексам элемента в матрице его значение. Матрица заполняется следующим образом:

$$\begin{bmatrix} x^n & x^{n-1} & x^{n-2} & \cdots & 1 \\ x^{n-1} & 0 & 0 & \cdots & x \\ x^{n-2} & 0 & 0 & \cdots & x^2 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x & x^2 & \cdots & x^n \end{bmatrix}$$

С использованием этой функции заполнить квадратную матрицу заданного размера.

8. Описать функцию even(K), возвращающую новое число, которое состоит только из четных цифр исходного числа $K$. Получить таким образом пять чисел.

9. Описать функцию convertDigits(K, p), переводящую целое десятичного числа $K$ в систему счисления с основанием $p$ (>1).

10. Описать функцию y(n), вычисляющую значение выражения по формуле:

$$y(n) = \cfrac{1}{n + \cfrac{1}{(n-1) + \cfrac{1}{(n-2) + \cfrac{1}{\ldots}}}}$$

$$\ldots + \cfrac{1}{1 + \cfrac{1}{2}}$$

Используя эту функцию, получить результат для пяти различных $n$.

11. Описать функцию shortFract(n, d) сокращения простой дроби, которая задаётся двумя целочисленными значениями: числителем *n* и знаменателем *d*. Функция должна выводить ответ на экран в виде дроби (например: 1/4, 3/17). С помощью этой функции сократить три заданные дроби.

12. Описать функцию pow1(x, e), находящую приближенное значение функции:

$$y = (1+x)^n = 1 + nx + \frac{n(n-1)}{2!}x^2 + \ldots + \frac{n(n-1)(n-2)\cdot\ldots\cdot(n-i+1)}{i!}x^i + \ldots.$$

В сумме учитывать все слагаемые, большие *e* ($0 < e < 1$). С помощью этой функции найти приближенное значение *y* для трех заданных *x* ($-1 < x < 1$) при *e*, равном 0,1 и 0,001.

13. Описать функцию sumDigits(K, p), вычисляющую сумму цифр, составляющих заданное целое число. С использованием этой функции получить сумму цифр для пяти заданных положительных чисел.

14. Описать функцию prod(n), вычисляющую произведение *n* ($n > 2$, четное) первых множителей следующего выражения:

$$y = \frac{2}{1}\cdot\frac{2}{3}\cdot\frac{4}{3}\cdot\frac{4}{5}\cdot\frac{6}{5}\cdot\frac{6}{7}\cdot\ldots.$$

С использованием этой функции получить значения *y* при пяти различных *n*.

15. Описать функцию isPow2(N), возвращающую *true*, если целый параметр *N* является точным значением двойки в какой-либо целой степени и *false* – в противном случае. С помощью этой функции проверить пять чисел.

## ЛАБОРАТОРНАЯ РАБОТА 14. ФУНКЦИИ И МАССИВЫ

Написать программу решения задачи согласно варианту:

1. Координаты первой группы точек заданы массивом X(n, 2), координаты второй группы точек – массивом Y(m, 2). Для каждой группы точек с использованием функции определить, сколько точек из группы находится внутри окружности радиусом R и с центром в начале координат.

2. Даны два массива X(n) и Y(m). Вычислить

$$Z = \frac{s_1 + s_2}{k_1 \cdot k_2},$$

где $s_1$ и $k_1$ – сумма и количество четных элементов массива X(n);
$s_2$ и $k_2$ – сумма и количество четных элементов массива Y(m).

Сумму четных элементов массива вычислять с помощью одой функции, количество – с помощью другой.

3. Даны две матрицы A(n, m) и B(k, l). Для каждой матрицы с использованием функции вывести на консоль количество отрицательных элементов в каждом столбце.

4. Даны две матрицы A(n, m) и B(k, l). Для каждой матрицы с

использованием функции подсчитать среднее арифметическое элементов, принадлежащих отрезку [0, 1].

5. Даны две матрицы A(n, m) и B(k, l). Для каждой матрицы с использованием функции вывести на консоль все элементы, кратные 3.

6. Даны два массива X(n) и Y(m). Вычислить

$$Z = \frac{\sum \sin(X_i) - \sum \sin(Y_i)}{2}.$$

Сумму синусов элементов вычислять с использованием функции.

7. Даны две матрицы A(m, m) и B(n, n). Для каждой матрицы с использованием функции вычислить сумму отрицательных элементов на главной диагонали.

8. Даны две матрицы A(m, m) и B(n, n). Для каждой матрицы с использованием функции вычислить количество четных элементов под главной диагональю.

9. Даны два массива X(n) и Y(m). С использованием функции вычислить среднеквадратичное отклонение элементов в каждом массиве. Среднеквадратичное отклонение элементов массива определять по формуле:

$$S = \sqrt{\frac{\sum_{i=1}^{n} \left( array_i - \overline{array} \right)^2}{n}}.$$

где $\overline{array}$ – среднее арифметическое значение элементов в массиве.

10. Даны два массива X(n) и Y(m). Вычислить

$$Z = \frac{X_{\max} - Y_{\min}}{2}.$$

$X_{\max}$ и $Y_{\min}$ вычислять с использованием функций.

11. Даны два массива X(n) и Y(m). С использованием функции вычислить среднее арифметическое положительных элементов в каждом массиве.

12. Даны две целочисленные матрицы A(n, m) и B(k, l). Для каждой матрицы с использованием функции вывести на консоль количество четных элементов в каждой строке.

13. Даны два массива X(n) и Y(m). Для каждого массива с использованием функций вычислить

$$Z = \frac{k_p + k_n}{2},$$

где $k_p$ – количество положительных элементов массива;
$k_n$ – количество отрицательных элементов массива.

14. Даны два массива X(n) и Y(m). Для каждого массива с использованием функций вычислить среднее значение и среднее отклонение элементов.

15. Даны две целочисленные матрицы A(n, m) и B(k, l) и число a. Для каждой матрицы с использованием функции подсчитать количество нечетных элементов, принадлежащих отрезку [a, 2a].

# 6 ОБРАБОТКА СТРОК В ЯЗЫКЕ С++

В языке С++ для удобной работы со строками есть класс **string**, для использования которого необходимо подключить заголовочный файл **string**.

Строки можно объявлять и одновременно присваивать им значения:

```
string S1, S2 = "Hello";
```

Строка **S1** будет пустой, строка **S2** будет состоять из 5 символов.

К отдельным символам строки можно обращаться по индексу. Например, **S[0]** – это первый символ строки.

Строки можно создавать с использованием следующих конструкторов:

**string()** – создает пустую строку;

**string(S)** – копия строки **S**;

**string(n, c)** – повторение символа **c** заданное число **n** раз;

**string(c)** – строка из одного символа **c**;

**string(S, pos, n)** – строка, содержащая **n** символов строки **S**, начиная с символа с индексом **pos**.

Строка выводится точно так же, как и числовые значения:

```
cout << S;
```

Для считывания строки с консоли можно использовать операцию **>>** для объекта **cin**:

```
cin >> S;
```

В этом случае считывается строка до первого пробела или символа конца строки. Это удобно для того, чтобы разбивать текст на слова или чтобы читать данные до конца файла при помощи **while (cin >> S)**.

Можно считывать строки целиком вместе с пробелами, используя функцию **getline**:

```
getline(cin, S);
```

Со строками можно выполнять следующие арифметические операции:

**=** – присваивание значения.

**+=** – добавление в конец строки другой строки или символа.

**+** – конкатенация двух строк, конкатенация строки и символа.

**==**, **!=** – посимвольное сравнение.

**<**, **>**, **<=**, **>=** – лексикографическое сравнение.

У строк есть разные методы, многие из них можно использовать несколькими разными способами (с разным набором параметров).

Таблица 6.1 – Функции и методы класса **string**

| Название | Описание |
|---|---|
| 1 | 2 |
| **S.size()**<br>**S.length()** | Возвращает длину строки **S** |
| **S.resize(n)**<br>**S.resize(n, c)** | Изменяет длину строки **S** на **n**. Если задан параметр **c**, то при увеличении длины строки **S** добавляемые символы будут равны **c** |

Продолжение таблицы 6.1

| 1 | 2 |
|---|---|
| **S.at(n)** | Обращение к символу строки **S** с индексом **n**. Данный метод аналогичен обращению **S[n]** |
| **S.begin()** | Возвращает итератор, указывающий на первый символ строки **S** |
| **S.end()** | Возвращает итератор, указывающий на последний символ строки **S** |
| **S.front()** | Возвращает первый символ строки **S** |
| **S.back()** | Возвращает последний символ строки **S** |
| **S.assign(n, c)**<br>**S.assign(str)**<br><br>**S.assign(str, pos, n)** | Заменяет содержимое строки **S n** символами **c**.<br>Заменяет содержимое строки **S** копией строки **str**.<br>Заменяет содержимое строки **S n** символами строки **str**, начиная с символа с индексом **pos** |
| **S.append(n, c)**<br><br>**S.append(str)**<br><br>**S.append(str, pos, n)** | Добавляет в конец строки **n** одинаковых символов, равных **c**.<br>Добавляет в конец строки **S** содержимое строки **str**.<br>Добавляет в конец строки **S n** символов строки **str**, начиная с символа с индексом **pos** |
| **S.insert(i, n, c)**<br><br>**S.insert(i, str)**<br>**S.insert(i, str, pos, n)** | Вставляет в строку **S n** одинаковых символов, равных **c**.<br>Вставляет содержимое строки **str.**<br>Вставляет **n** символов строки **str**, начиная с символа с индексом **pos**.<br>Первый вставленный символ будет иметь индекс **i**, а все символы, которые ранее имели индекс **i** и более, сдвигаются вправо |
| **S.replace(pos, n, n2, c)**<br><br>**S.replace(pos, n, str)**<br><br>**S.replace(pos, n, str, pos2, n2)** | Заменяет **n** символов строки **S**, начиная с символа с индексом **pos**, на **n2** символов **c**.<br>Заменяет **n** символов строки **S**, начиная с символа с индексом **pos**, на содержимое строки **str**.<br>Заменяет **n** символов строки **S**, начиная с символа с индексом **pos**, на **n2** символов строки **str**, начиная с символа с индексом **pos2** |
| **S.swap(str)** | Обменивает содержимым строки **S** и **str** |
| **S.push_back(c)** | Добавляет в конец строки символ **c** |
| **S.pop_back()** | Удаляет последний символ строки |
| **S.clear()**<br>**S.erase()** | Очищает строку, строка становится пустой |

Окончание таблицы 6.1

| 1 | 2 |
|---|---|
| **S.erase(pos)**<br><br>**S.erase(pos, n)** | Удаляет символы из строки **S**, начиная с символа с индексом **pos** и до конца строки.<br>Удаляет из строки **S n** символов, начиная с символа с индексом **pos** |
| **S.empty()** | Возвращает **true**, если строка пуста, **false** – в противном случае |
| **S.substr(pos)**<br><br>**S.substr(pos, n)** | Возвращает подстроку данной строки, начиная с символа с индексом **pos** и до конца строки.<br>Возвращает подстроку данной строки, начиная с символа с индексом **pos** количеством **n** символов |
| **S.find(str)**<br>**S.find(str, pos)** | Ищет первое вхождение строки **str** в строку **S**.<br>Ищет первое вхождение строки **str**, начиная с позиции **pos**.<br>Если строка **str** не найдена, то возвращается **-1** |
| **S.rfind(str)**<br>**S.rfind(str, pos)** | Ищет последнее вхождение строки **str** в строку **S**. Способы вызова аналогичны способам вызова метода **find**. |
| **S.find_first_of(str)**<br>**S.find_first_of(str, pos)** | Ищет в строке **S** первое появление любого из символов данной строки **str**. Если задано значение **pos**, то поиск начинается с позиции **pos** |
| **S.find_last_of(str)**<br>**S.find_last_of(str, pos)** | Ищет в строке **S** последнее появление любого из символов данной строки **str.** Если задано значение **pos**, то поиск начинается с позиции **pos** |
| **S.find_first_not_of(str)**<br>**S.find_first_not_of(str, pos)** | Ищет в данной строке первое появление символа, отличного от символов строки **str**. Если задано значение **pos**, то поиск начинается с позиции **pos** |
| **S.find_last_not_of(str**<br>**S.find_last_not_of(str, pos)** | Ищет в данной строке последнее появление символа, отличного от символов строки **str**.<br>Если задано значение **pos**, то поиск начинается с позиции **pos** |
| **S.compare(str)** | Сравнивает две строки. Возвращает **0**, если **S=str**, **1**, если **S>str**, и **–1**, если **S<str** |
| **int stoi(str)** | Преобразует строку **str** в целочисленное значение |
| **float stof(str)**<br>**double stod(str)** | Преобразует строку **str** в вещественное значение |
| **to_string(value)** | Преобразует числовое значение в строку **string** |
| **S.c_str()** | Возвращает массив символов строки **S** |

ПРИМЕР. Рассмотрим применение некоторых из описанных выше методов.

```
#include <iostream>
#include <string>

using namespace std;

int main()
{
    int k, m, n;
    string s, p, t, digits = "0123456789";

    s = string("Pupkin Vasiliy Vasilyevich");
    cout << s << endl;
    n = s.length();
    cout << "n=" << n << endl;
    k = s.find("Microsoft");
    cout << "index of Microsoft: " << k << endl;
    k = s.find("Pupkin");
    cout << "index of Pupkin: " << k << endl;
    k = s.find("Vasil");
    cout << "index of Vasil: " << k << endl;
    k = s.find("Vasil", 10);
    cout << "index of Vasil from 10: " << k << endl;
    p = s.substr(7);
    cout << "substr(7): " << p << endl;
    s.erase(7, 8);
    cout << "erase(7, 8): " << s << endl;
    s.erase(8);
    cout << "erase(8): " << s << endl;
    s.pop_back();
    cout << "pop_back: " << s << endl;
    s[2] = 's';
    cout << "s[2] = 's': " << s << endl;
    s.insert(3, "h");
    cout << "insert(3,\"h\"): " << s << endl;
    s += "Aleksandr Sergeevich";
    cout << "+=Aleksandr Sergeevich: " << s << endl;
    cout << "front: " << s.front() << endl;
    cout << "back: " << s.back() << endl;

    p.assign("My phone number is 291234567. Remember!");
    cout << endl << p << endl;
    k = p.find_first_of(digits);
    cout << "first of digits " << k << endl;
    m = p.find_last_of(digits);
    cout << "last of digits " << m << endl;
    p = p.substr(k, m - k + 1);
    cout << "substr(" << k << ", " << m - k + 1 << " ): " << p << endl;
```

```
    t = "45";
    m = stoi(t);
    cout << endl << "stoi(\"t\"): " << m << endl;

    return 0;
}
```

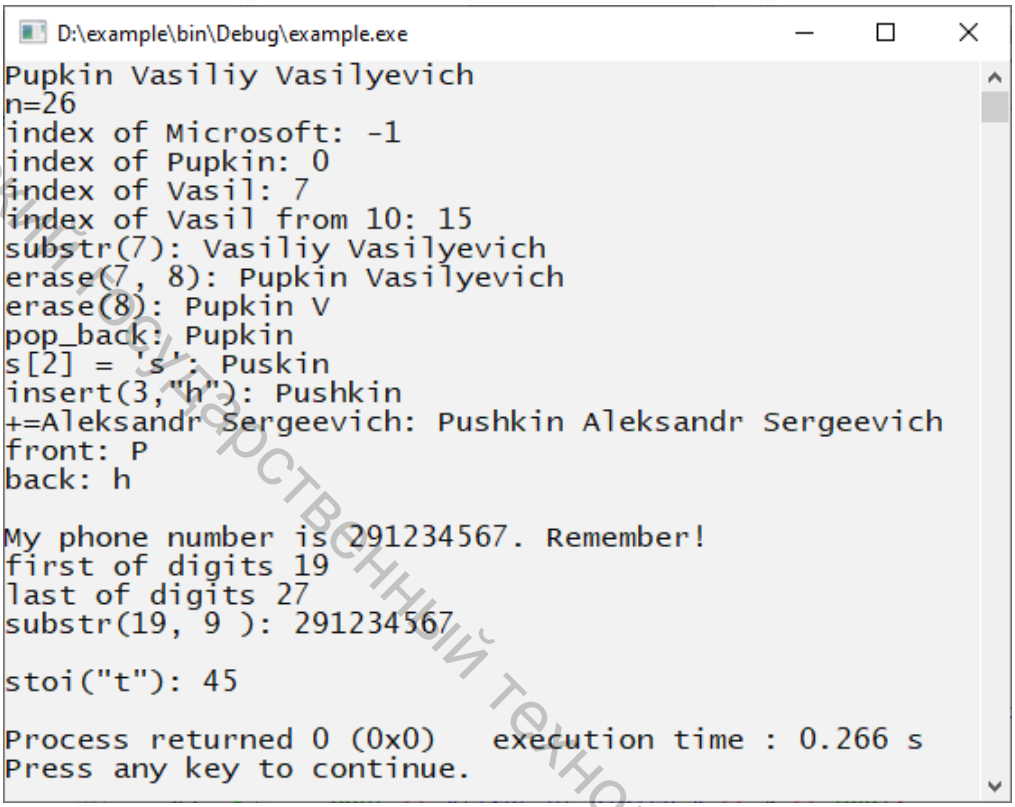



Рисунок 6.1 – Результат выполнения программы

## ЛАБОРАТОРНАЯ РАБОТА 15. КЛАСС STRING

Написать программу обработки строки согласно варианту. Ввод строки осуществлять с консоли.

1. Вывести все слова наименьшей длины.
2. Вывести те слова, в которых нет повторения букв.
3. Вывести те слова, в которых буква «а» повторяется дважды.
4. Вывести те слова, которые начинаются и оканчиваются одинаковой буквой.
5. Вывести те слова, которые не содержат цифр.
6. Вывести те слова, у которых есть хотя бы одна буква «а», стоящая рядом с «м».
7. Вывести все слова в обратном порядке.
8. Вывести все слова, поменяв местами первую и последнюю буквы.
9. Удалить во всех словах букву «а», позиции справа заполнить «*».
10. Заменить во всех словах каждое вхождение буквы «х» на «ks».

11. Удалить все слова, заканчивающиеся на гласную букву.

12. Определить, есть ли в заданном предложении цифры, если есть – найти их сумму.

13. Символы самого длинного слова заменить символами 'm'.

14. В словах с нечетным количеством символов удалить среднюю букву.

15. В каждом слове, где есть буква «а», добавить после нее «да».

# 7 СТРУКТУРЫ В ЯЗЫКЕ С++

*Структура* – это составной тип данных, в котором под одним именем объединены данные различных типов. Отдельные данные структуры называются полями.

Объявляется структура следующим образом:

```
struct ИмяСтруктуры
{
    тип1 имяПоля1;
    тип2 имяПоля2;
    ...
    типN имяПоляN;
};
```

К полям структуры можно обращаться следующим образом:

```
имяСтруктуры.имяПоля
```

или

```
указательНаСтруктуру->имяПоля
```

Полями структур могут быть произвольные типы данных. Можно, например, создать структуру, полями которой будут массивы или другие структуры.

Структуры можно передавать в функции и возвращать из них.

Пример. Написать программу вывода на консоль среднего балла студентов. О студенте хранится следующая информация: фамилия, имя и оценки по трем предметам (математика, физика, химия).

```
#include <iostream>
#include <string>
#include <windows.h>

using namespace std;

struct student
{
    string lastName;
    string firstName;
    int math;
```

```
    int physics;
    int chemistry;
};

void printAverage(student st)
{
    cout << "Студент: " << st.lastName << " " << st.firstName << endl;
    cout << "Средний балл: " << (double) (st.math + st.physics +
st.chemistry) / 3 << endl;
}

int main()
{
    SetConsoleOutputCP(1251);

    student st1 = {"Иванов", "Михаил", 7, 9, 8};

    student st2 = {lastName: "Сорокин", firstName: "Сергей", math:
9, physics: 10, chemistry: 8};

    student st3 =  {.lastName = "Михайлова", .firstName = "Анна",
.math = 7, .physics = 4, .chemistry = 5};

    student st4, *st5;

    st4.lastName = "Петрова";
    st4.firstName = "Мария";
    st4.math = 6;
    st4.physics = 6;
    st4.chemistry = 6;

    st5 = new student;
    st5->lastName = "Березкин";
    st5->firstName = "Василий";
    st5->math = 9;
    st5->physics = 10;
    st5->chemistry = 9;

    printAverage(st1);
    printAverage(st2);
    printAverage(st3);
    printAverage(st4);
    printAverage(*st5);

    return 0;
}
```

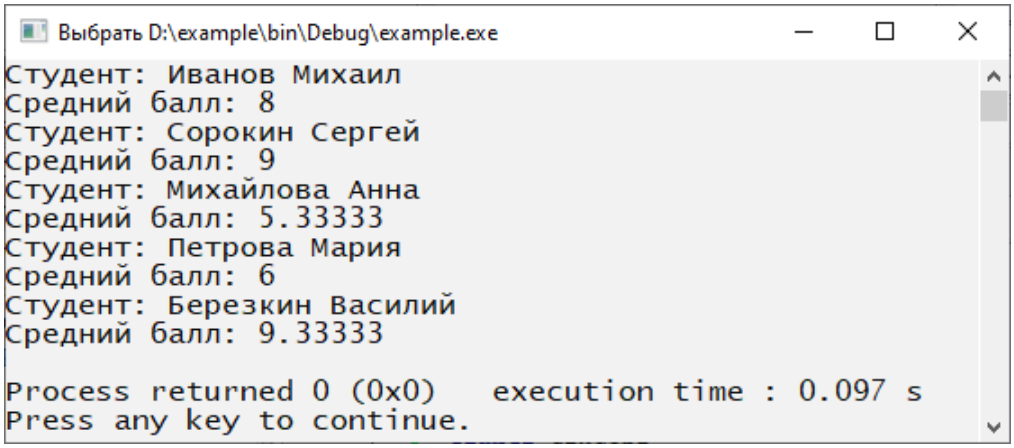



Рисунок 7.1 – Результат выполнения программы

## ЛАБОРАТОРНАЯ РАБОТА 16. СТРУКТУРЫ

Написать программу на языке C++ решения задачи согласно своему варианту. В программе при необходимости использовать функции, реализующие операции со структурами, а также массив переменных созданной структуры.

1. Создать структуру fraction, описывающую простую дробь. Поля: *n* – числитель, *d* – знаменатель. Для заданных двух дробей выполнить операции умножения и деления.

2. Создать структуру date, описывающую дату. Поля: *day*, *month*, *year*. Среди *N* человек с заданной датой рождения найти самого младшего и самого старшего человека.

3. Создать структуру parallelepiped, описывающую прямоугольный параллелепипед. Поля: *a* и *b* – длины сторон прямоугольника-основания, *h* – высота. Найти, у какого из *N* заданных параллелепипедов объем наименьший.

4. Создать структуру time, описывающую время. Поля: *hours*, *minutes*. С использованием данной структуры решить следующую задачу: у прибора зафиксировано время начала работы и время завершения работы в текущий день. Определить общее время работы прибора за неделю.

5. Создать структуру triangle, описывающую треугольник. Поля: *a, b, c* – длины трех сторон. Найти, у какого из трех заданных треугольников площадь наибольшая.

6. Создать структуру straightLine, описывающую прямую. Поля: *x1*, *y1*, *x2*, *y2* – координаты двух точек, через которые проходит прямая. Для *K* заданных прямых вывести их уравнение в виде $y = a\,x + b$.

7. Создать структуру conus, описывающую конус. Поля: *r* – радиус круга-основания, *h* – высота. Определить для *N* заданных конусов объем и площадь поверхности.

8. Создать структуру point, описывающую точку на плоскости. Поля: *x*, *y* – координаты точки. С помощью этой структуры для каждой из *N* точек вывести сообщение, в какой координатной четверти она расположена.

9. Создать структуру parabola, описывающую параболу. Поля: *a*, *b*, *c* – коэффициенты уравнения $y = a\,x^2 + b\,x + c$. Для *N* заданных парабол определить точки пересечения с осью *OX*.

10. Создать структуру parallelogram, описывающую параллелограмм. Поля: *a*, *b* – длины сторон и *phi* – угол между сторонами. Для *K* заданных фигур определить $k_1$ – количество квадратов и $k_2$ – количество прямоугольников.

11. Создать структуру vector3, описывающую вектор в трехмерной системе координат. Поля: *x*, *y*, *z* – координаты вектора. Для *M* заданных векторов определить длину каждого и найти номер самого длинного вектора.

12. Создать структуру circle, описывающую окружность. Поля: *x0*, *y0* – координаты центра, *R* – радиус. Для каждой из *N* заданных окружностей определить, пресекается ли она с осями *OX* и *OY*.

13. Создать структуру rhomb, описывающую ромб. Поля: *d1*, *d2* – диагонали ромба. Для *K* заданных ромбов определить *k1* – количество квадратов и найти ромб с минимальной площадью.

14. Создать структуру complex, описывающую комплексное число. Поля: *re* – действительная и *im* – мнимая часть. Для *N* заданных комплексных чисел найти числа, у которых модуль наибольший и наименьший.

15. Создать структуру point3, точку в пространстве. Поля: *x*, *y*, *z* – координаты точки. Найти расстояние между двумя заданными точками.

# 8 ФАЙЛЫ В ЯЗЫКЕ C++

В языке C++ для работы с файлами имеется три потока: **ifstream** для чтения данных из файла, **ofstream** – для записи в файл и **fstream** – для работы как в режиме чтения, так и записи. Для работы с данными потоками подключают заголовочный файл **fstream**.

Для записи данных в текстовый файл необходимо:

1. Описать файловый поток:

```
ofstream f;
```

2. Открыть файл с помощью метода **open**.
3. Записать информацию в файл с помощью операции помещения в поток **<<**.
4. Закрыть файл методом **close**.

Для чтения данных из текстового файла необходимо:

1. Описать файловый поток:

```
ifstream f;
```

2. Открыть файл с помощью метода **open**.
3. Записать информацию в текстовый файл с помощью операции извлечения из потока **>>** или функции **getline**. При чтении каждой порции данных необходимо проверять, что чтение данных возможно.
5. Закрыть файл методом **close**.

Открывается файл следующим образом:

```
f.open("name", mode);
```

Здесь **f** – имя файлового потока; **name** – полное имя файла; **mode** – режим работы с открытым файлом.

Таблица 8.1 – Возможные значения **mode**

| | |
|---|---|
| **ios::in** | Открытие файла в режиме чтения (режим по умолчанию для **ifstream**) |
| **ios::out** | Открытие файла в режиме записи (режим по умолчанию для **ofstream**). Если файл не существует, он создается |
| **ios::app** | Открытие файла в режиме дозаписи (запись в конец файла) |
| **ios::ate** | Передвинуться в конец уже открытого файла |
| **ios::trunc** | Удалить содержимое файла, если он существует |
| **ios::binary** | Открыть двоичный файл |

После удачного открытия файла в **f** будет храниться **true**, в противном случае – **false**.

После открытия файла для записи в него можно писать точно так, как и выводить на консоль, только вместо стандартного потока вывода **cout** необходимо указывать имя файлового потока:

```
f << a;
```

Чтение данных из файла осуществляется так, как и с консоли, только вместо потока ввода **cin** используют файловый поток:

```
f >> a;
```

Для проверки достигнут ли конец файла, служит метод **f.eof()**.

ПРИМЕР. Написать программу записи в текстовый файл следующих данных о людях: ФИО, город проживания, телефон.

```
#include <iostream>
#include <fstream>
#include <string>
#include <windows.h>

using namespace std;

int main()
{
    SetConsoleOutputCP(1251);
    SetConsoleCP(1251);
    int i, n;
    ofstream fo;
    string temp;

    cout << "n=";
    cin >> n;
    cin.ignore();
```

```
    fo.open("D:\\abc.txt");
    if (fo)
    {
        for (i = 1; i <= n; i++)
        {
            cout << "ФИО: ";
            getline(cin, temp);
            fo << temp << '\t';

            cout << "город: ";
            getline(cin, temp);
            fo << temp << '\t';

            cout << "телефон: ";
            getline(cin, temp);
            fo << temp;

            if (i != n)
                fo << endl;
        }
        fo.close();
    }
    else
        cout << "Ошибка при открытии файла";

    return 0;
}
```

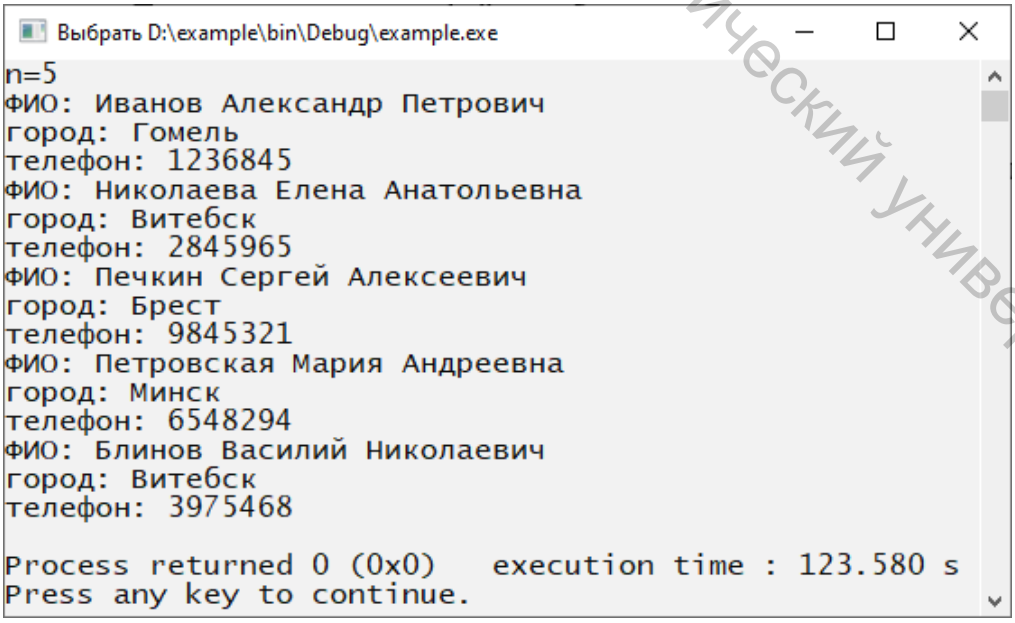



Рисунок 8.1 – Результат выполнения программы (консоль)

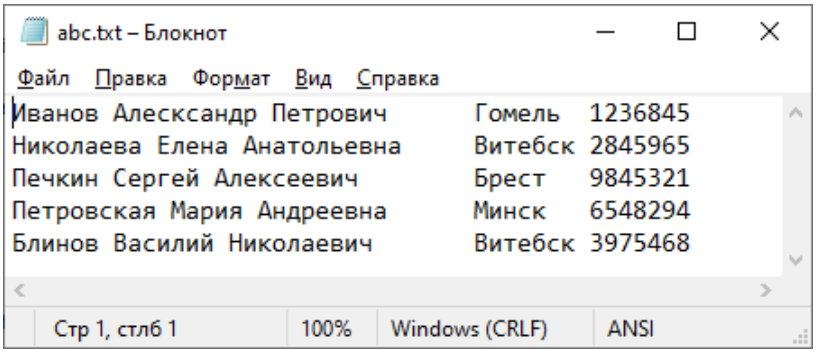



Рисунок 8.2 – Результат выполнения программы (записанный файл)

ПРИМЕЧАНИЕ. После ввода значения **n** в потоке остается символ **'\n'**. Если следующим вводом должно быть извлечение символьных или строковых данных, то пользователю не будет предложено их ввести. Вместо них будет записан символ конца строки. Чтобы удалить данный символ из потока, использован метод **ignore.**

ПРИМЕР. Прочитать данные о людях из файла, записанного в предыдущем. Вывести на консоль сведения о людях, проживающих в Витебске.

```
#include <iostream>
#include <fstream>
#include <string>
#include <windows.h>

using namespace std;

struct person
{
    string name;
    string city;
    long long int phone;
};

int main()
{
    SetConsoleOutputCP(1251);
    ifstream fi;
    string temp;
    int prev, next;
    person a;

    fi.open("D:\\abc.txt");

    if (fi)
    {
        cout << "В Витебске проживают: " << endl;
```

```
        while (!fi.eof())
        {
            getline(fi, temp);
            prev = 0;
            next = temp.find('\t', prev + 1);
            a.name = temp.substr(prev, next);
            prev = next + 1;
            next = temp.find('\t', prev);
            a.city = temp.substr(prev, next - prev);
            a.phone = stoi(temp.substr(next + 1, temp.length() - next));

            if (a.city == "Витебск")
                cout << a.name << '\t' << a.phone << endl;
        }
        fi.close();
    }
    else
        cout << " Ошибка при открытии файла" << endl;

    return 0;
}
```

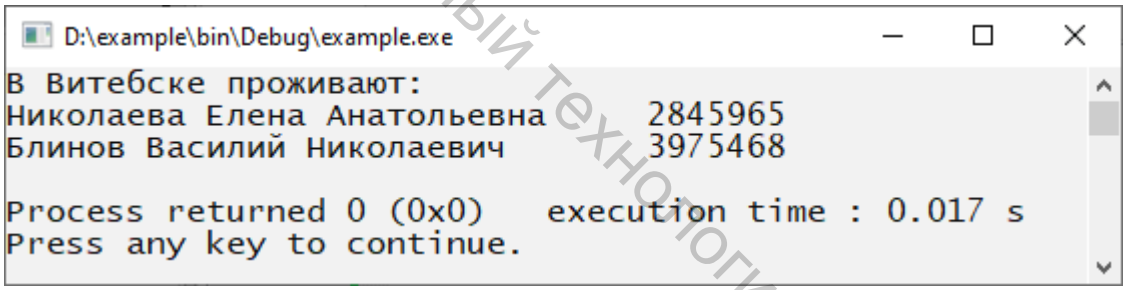



Рисунок 8.3 – Результат выполнения программы

## ЛАБОРАТОРНАЯ РАБОТА 17. ТЕКСТОВЫЕ ФАЙЛЫ

Написать две программы: первую – для записи данных в текстовый файл, вторую – для чтения и обработки данных согласно варианту:

1. В файле находится список студентов группы. О каждом студенте хранятся следующие сведения: Ф.И.О., год рождения, номер зачетной книжки, пол, результаты последней сессии по четырем экзаменам (физика, математика, история, химия). Вывести на консоль информацию о студентах, получивших отличные оценки по математике.
2. В файле находятся результаты метеорологических наблюдений по месяцам. Элемент списка хранит следующие данные: период наблюдения (месяц и год), количество дней месяца, количество выпавших осадков, количество облачных дней, количество дней с переменной облачностью.

Определить и вывести на консоль сведения о месяце в заданном году, в течение которого выпало наибольшее количество осадков.

3. В файле находится список книг, хранящихся в библиотеке. О каждой книге хранятся следующие сведения: инвентарный номер, шифр УДК, название книги, Ф.И.О. автора, место издания, год издания. Вывести на консоль сведения о книгах данного автора, выпущенных не раньше указанного года.

4. В файле находится список читателей библиотеки. О каждом читателе хранятся следующие сведения: Ф.И.О., номер читательского билета, адрес, место работы или учебы, количество взятых книг, срок возврата книг (день, месяц, год). Вывести на консоль сведения о читателях, для которых истек срок возврата книг.

5. В файле находится список студентов факультета. О каждом студенте хранятся следующие сведения: Ф.И.О., группа, адрес (город, улица, дом). Вывести на консоль сведения о студентах, заданной группы, проживающих в указанном городе.

6. В файле находится список товаров в магазине. О каждом товаре хранятся следующие сведения: наименование, отдел магазина, цена, дата поступления, срок годности. Вывести на консоль сведения о товарах заданного отдела, для которых срок годности истекает в указанный день.

7. В файле находится список дисциплин кафедры. О каждой дисциплине хранятся следующие сведения: название, курс, группа, количество часов лекций, лабораторных, практических, сдается зачет или экзамен. Вывести на консоль сведения о дисциплинах с экзаменом, у которых более 32 часов лекций.

8. В файле находится список занятий в аудитории. О каждом занятии хранятся следующие сведения: группа, название дисциплины, день недели, номер пары, количество человек. Вывести на консоль сведения о занятиях в самый загруженный день недели.

9. В файле находится список оргтехники. О каждом устройстве хранятся следующие сведения: вид устройства, модель, год приобретения, цена, подразделение. Вывести на консоль сведения об устройствах указанного подразделения, срок службы которых более 5 лет.

10. В файле находится расписание полетов самолетов. О каждом рейсе хранятся следующие сведения: пункт назначения, день недели, время отправления, время полета, стоимость билета. Вывести на консоль сведения о самом дешевом рейсе в указанный город.

11. В файле находятся сведения о сотрудниках организации. О каждом сотруднике хранятся следующие сведения: Ф.И.О., возраст, должность, подразделение, стаж, стаж работы в организации. Вывести на консоль сведения о сотрудниках, работавших только в данной организации.

12. В файле находится список CD-дисков. О каждом CD-диски хранятся следующие сведения: название альбома, исполнитель, стиль, год выпуска, длительность, стоимость. Для заданного стиля вывести на консоль имена исполнителей, диски которых выпущены не ранее 2015 года.

13. В файле находится список отгружаемой со склада продукции. О каждой отгрузке хранятся следующие сведения: число, месяц, год, наименование продукции, количество, куда отгружен. Для заданного года вывести на консоль сведения о продукции, отгруженной в указанную организацию.

14. В файле находится список сотрудников подразделения. О каждом сотруднике хранятся следующие сведения: Ф.И.О., должность, год рождения, образование. Вывести на консоль сведения о сотрудниках с высшим образованием не старше 1980 года рождения.

15. В файле находится список автотранспорта предприятия. О каждом автомобиле хранятся следующие сведения: тип транспортного средства (легковой, автобус, грузовой), номер, год выпуска, пробег. Вывести на консоль сведения об автобусах, не старше 7 лет.

## ЛИТЕРАТУРА

1. Дэвис, Стефан Р. C++ для «чайников» / Стефан Р. Дэвис; пер. с англ. – 4-е изд. – М. : Вильямс, 2003. – 336 с.

2. Стивен, Прата. Язык программирования C++ (C++11). Лекции и упражнения / Прата Стивен; пер. с англ. – 6-е изд. – М. : Вильямс, 2012. – 1248 с.

3. Страуструп, Бьярне. Программирование: принципы и практика с использованием C++ / Бьярне Страуструп; пер. с англ. – 2-е изд. – М. : Вильямс, 2016. – 1328 с.

4. Шилдт, Герберт. C++. Базовый курс / Герберт Шилдт; пер. с англ. – 3-е изд. – М. : Вильямс, 2019. – 624 с.

Учебное издание

# ОСНОВЫ КОМПЬЮТЕРИЗАЦИИ ТЕХНОЛОГИЙ В СИСТЕМАХ АВТОМАТИКИ. ОСНОВЫ ПРОГРАММИРОВАНИЯ НА АЛГОРИТМИЧЕСКОМ ЯЗЫКЕ

Методические указания по выполнению лабораторных работ

Составители:
Соколова Анна Сергеевна
Казаков Вадим Евгеньевич

Редактор *Т.А. Осипова*
Корректор *Т.А. Осипова*
Компьютерная верстка *А.С. Соколова*

---

Подписано к печати 13.12.2021. Формат 60x90 1/16. Усл. печ. листов 5,0.
Уч.-изд. листов 6,3. Тираж 30 экз. Заказ № 318.

Учреждение образования «Витебский государственный технологический университет»
210038, г. Витебск, Московский пр., 72.
Отпечатано на ризографе учреждения образования
«Витебский государственный технологический университет».
Свидетельство о государственной регистрации издателя, изготовителя,
распространителя печатных изданий № 1/172 от 12 февраля 2014 г.
Свидетельство о государственной регистрации издателя, изготовителя,
распространителя печатных изданий № 3/1497 от 30 мая 2017 г.