Задачи по программированию

Андрей Строганов (savthe@gmail.com)

9 июля 2014 г.

1 Первая программа

Одним из основных понятий в языке программирования является *функция*. Функцией мы будем называть подпрограмму, имеющую имя. Любая программа на языке C++ должна иметь функцию с именем *main*, а результатом работы программы должно быть целое число (`int`).
Пример. Программа, имеющая функцию `main` типа `int`.

```
int main()
{
}
```

В круглых скобках указываются параметры функции, но в данном случае main не принимает параметров. Между фигурными скобками помещается тело функции, то есть набор действий, которые выполняет программа. Текст программы также называется исходным кодом, исходником или сорцами (от английского source code, source, sources).

Часто для большей ясности в исходный код программы добавляются *комментарии*, которые начинаются сразу после двух слешей ("//"), либо заключаются в блок "/* */":

```
int main() //начало моей программы
{
    /*
    Это -- пустая функция,
    она ничего не делает.
    */
}
```

Комментарии игнорируются компилятором и нужны исключительно для сопровождения исходных кодов.

Для вывода текста на экран используется объект `cout`, который объявлен в библиотеке *iostream*, в пространстве имен `std`. Например, выведем на экран строку ”Трям! =)”:

```
#include <iostream> // подключаем библиотеку iostream

int main()
{
    std::cout << "Трям! =)";
}
```

В строке 1 с помощью директивы `#include` мы включили файл *iostream* в программу, чтобы компилятор знал, что существует объект `cout` и как он работает. В строке 4 строка ”Трям =)” записывается (<<) в объект `cout`, определенный в пространстве имен `std`.

Чтобы явно не указывать пространство имен `std`, можно в начало программы добавить строку `using namespace std;`

Пример.

```
#include <iostream>

using namespace std;

int main()
{
    cout << "Трям! =)";
}
```

Не забывайте, что программа начинает выполняться с функции *main*. Все, что написано до нее нужно чтобы компилятор правильно понимал незнакомые слова.

Рассмотрим пример:

```
#include <iostream>

using namespace std;

int main()
{
    cout << "Кто ходит в гости по утрам,";
    cout << "Тот поступает мудро!";
}
```

На экране появится надпись:

```
Кто ходит в гости по утрам,Тот поступает мудро!
```

Чтобы строки не склеивались, необходимо после каждой строки выводить символ-разделитель, который в C++ обозначается `endl`.

```
#include <iostream>

using namespace std;

int main()
{
    cout << "Кто ходит в гости по утрам," << endl;
    cout << "Тот поступает мудро!";
}
```

Обратите внимание на то, что символы << обозначают *запись* строки в объект cout, а не являются литературным ковычками « ».

1.1 Выведите на экран ваше имя и фамилию.

1.2 Выведите на экран звездочки в виде прямоугольного треугольника:

```
*
**
***
****
*****
```

1.3 Выведите на экран звездочки в виде равнобедренного треугольника:

```
    *
   ***
  *****
 *******
*********
```

1.4 Выведите свое имя в рамке из звездочек.

Пример.

```
******************
* Доктор Зойдберг *
******************
```

В C++ поддерживаются стандартные математические операции `+`, `-`, `*`, `/` и скобки `()`.
Пример. Вычисление выражения $\mathbf{7 + \frac{6}{1+2}}$.

```
cout << 7 + 6/(1+2);
```

При вычислении выражений есть разница между целыми и дробными числами. Например, выражение $\mathbf{5/2}$ будет равно $\mathbf{2}$, так как по умолчанию оба числа будут считаться целыми, поэтому результат деления тоже будет считаться целым, при этом дробная часть ответа будет отброшена (а не округлена в большую сторону). Чтобы «подсказать» компилятору, что выражение должно быть дробным, достаточно сделать дробным одно из чисел. Для этого можно было записать $\mathbf{5.0/2}$ или даже проще: $\mathbf{5./2}$. В результате получится число $\mathbf{2.5}$.

1.5 Вычислите выражения, используя операции `+, -, *, /` и скобки
а) $\mathbf{7 + 8}$, б) $\frac{5}{4}$, в) $\mathbf{(2 + 3)^2}$, $\frac{1}{(2+3)^2}$

1.6 Вычислите выражение, используя операции `+, -, *, /` и скобки

$$\mathbf{1 + \frac{1}{1 + \frac{1}{1+\frac{1}{1}}}}$$

2 Тип данных int

Рассмотрим проблему хранения простейших данных. Например, нужно считать с клавиатуры длины двух катетов и вычислить длину гипотенузы.

Переменные. Для хранения данных используются переменные — участки в оперативной памяти, имеющие тип и имя. Тип данных определяет множество допустимых значений данных и возможные операции над ними. Пока используем уже известный нам тип целых чисел `int`.

Чтобы объявить переменную, достаточно указать ее тип и имя:

```
int age; //Объявили целочисленную переменную с именем age
```

Для записи значений в переменные используется оператор присваивания "=":

```
age = 15; //Присвоили значение 15 в переменную age
```

Не путайте оператор присваивания = с математичеким символом, обозначающим равенство. В программировании = обозначает не *приравнять*, а *записать*.

На разных архитектурах компьютеров переменная типа `int` может хранить разные по величине числа. Во многих современных компьютерах `int` позволяет хранить отрицательные и положительные числа, не превышающие нескольких миллиардов (точные значения будут приведены в следующей главе).

Ввод данных. Для ввода данных с клавиатуры используется объект `cin`, который также как и `cout`, обявлен в файле `iostream`. Чтобы что-то считать с клавиатуры (в общем случае – из потока) нужно с помощью оператора чтения (>>) из `cin` указать переменные, в которые будут записаны считанные значения. Например:

```
int a, b;
cin >> a;
cin >> b;
// можно в одну строку: cin >> a >> b;
```

Пример. Программа, вычисляющая периметр квадрата.

Периметр квадрата вычисляется по формуле $\boldsymbol{p = 4a}$, где $\boldsymbol{a}$ – длина одной стороны квадрата. Видно, что $\boldsymbol{a}$ - это параметр, который программа должна использовать для расчета. Следовательно, именно этот параметр необходимо предстоит считать с клавиатуры.

```
#include <iostream> // необходимо для cout, cin

using namespace std; // чтобы не писать std::cout, std::cin

int main()
{
    cout << "Введите длину стороны квадрата: ";
```

```
    int a; // объявляем переменную a
    cin >> a; // считываем в нее значение с клавиатуры

    cout << "Периметр равен: " << 4*a << endl;
}
```

Рекомендуется объявлять переменные сразу перед использованием, а не за несколько строк кода.

Константы. Для обозначения величин, которые не меняются исользуют константы, которые объявляются также как и переменные, но с модификатором `const`:

```
    cout << "Введите длину стороны квадрата: ";
    int a;
    cin >> a;
    const int p = 4*a;

    cout << "Периметр равен: " << p << endl;
```

Старайтесь все неизменяемые величины делать константными. При случайной попытке изменить значение константы компилятор сразу укажет ошибку.

Пример. Пример программы, вычисляющей скорость точки при равноускоренном движении.

```
#include <iostream>

using namespace std;

int main()
{
    int v0, a, t;
    cout << "Введите начальную скорость, ускорение и время: ";
    cin >> v0 >> a >> t;
    const int v = v0 + a * t;
    cout << "Скорость точки равна: " << v << endl;
}
```

2.1 Вычислите площадь квадрата по длине стороны a.

2.2 Переведите заданное количество километров в метры.

2.3 Вычислите площадь прямоугольника по заданным сторонам a и b.

2.4 Вычислите площадь треугольника по стороне a и высоте h_a.

2.5 Вычислите a^4, использовав не более двух операций умножения.

2.6 Вычислите a^6, использовав не более трех операций умножения.

2.7 Вычислите a^{20}, использовав не более пяти операций умножения.

2.8 *Алгоритм swap*. Даны переменные a и b. Напишите программу, в результате которой эти переменные обменяются своими значениями.

2.9 Решите предыдущую задачу, не используя дополнительных переменных.

3 Типы данных

Целочисленные типы данных. К целочисленным (integer) типам данных относятся `char`, `short` (или `short int`), `int`, `long` (или `long int`). Несколько типов необходимо потому, что переменная каждого типа занимает некоторый объем в памяти компьютера. Так, например, переменная типа `char` занимает 1 байт и может принимать значения от -128 до 127. Переменная типа `int`, обычно, больше и может занимать 2, 4 и более байт, следовательно, и диапазон значений у нее шире. Целочисленные переменные можно объявить беззнаковыми с помощью модификатора `unsigned`, тем самым увеличив максимально возможное значение за счет отрицательной области. Так, переменная типа `unsigned char` будет принимать значения от 0 до 255 включительно.

3.1 Переменная типа `char` всегда имеет размер 1 байт. Объясните, откуда берутся ее максимальное ($\mathbf{127}$) и минимальное ($\mathbf{-128}$) значения? Почему они различаются по модулю?

3.2 Предполагая размеры типов `short int`, `int`, `long int` равными 2, 4, 8 байт соответственно, определите диапазоны их значений и заполните таблицу:

Тип	Min	Max
`char`	-128	127
`unsigned char`	0	255
`short`		
`unsigned short`		
`int`		
`unsigned int`		
`long`		
`unsigned long`		

Типы с плавающей точкой. Типы с плавающей точкой (floating point): `float`, `double`, `long double` используются для хранения дробных чисел и различаются как диапазоном, так и точностью значений (то есть максимальным количеством цифр после запятой). Обычно, переменные типа `float` могут хранить до 6 цифр после запятой, `double` — до 10, `long double` — до 15. Заметим, что к типам с плавающей точкой не применяется модификатор `sizeof`.

Тип	Min	Max
`float`	-3.40282×10^{38}	3.40282×10^{38}
`double`	-1.79769×10^{308}	1.79769×10^{308}
`long double`	$-1.18973 \times 10^{4932}$	1.18973×10^{4932}

Оператор `sizeof`. В зависимости от архитектуры и операционной системы размеры типов могут варьироваться. Чтобы определить размер типа можно воспользоваться оператором `sizeof`, например:

```
cout << "Размер переменной типа int: " << sizeof(int) << endl;
```

Гарантируется, что независимо от архитектуры всегда будет выполняться соотношение:

$$1 \equiv \texttt{sizeof(char)} \leq \texttt{sizeof(short)} \leq \texttt{sizeof(int)} \leq \texttt{sizeof(long)}$$

3.3 Определите, сколько байт занимают переменные целочисленных типов и типов с плавающей точкой на вашем компьютере.

Математические функции из `<cmath>` Для вычисления математических функций, таких как синус, косинус, корень числа и других, используется библиотека `cmath`. Например, квадратный корень вычисляется с помощью функции

```
double sqrt( double )
```

Эта запись означает, что функция `sqrt` принимает действительное число, и результатом ее работы также является действительное число. *Пример.* Пример программы, вычисляющей корень введенного числа.

```
#include <iostream>
#include <cmath> // для функции sqrt

using namespace std;

int main()
{
    cout << "Введите число: ";
    double x;
    cin >> x;

    cout << "Корень из числа " << x << " равен: " << sqrt(x) << endl;
```

Приведем некоторые функции библиотеки `<cmath>`:

```
double fabs( double ) -- модуль числа
double sqrt( double ) -- квадратный корень числа
double sin( double ) -- синус
double cos( double ) -- косинус
double tan( double ) -- тангенс
```

Во всех тригонометрических функциях используется радианная мера углов.

3.4 По заданным катетам прямоугольного треугольника определите длину гипотенузы.

3.5 Определите длины катетов прямоугольного треугольника по заданной гипотенузе и углу, образованному ей и одним из катетов.

3.6 Выведите модуль введенного числа.

4 Инструкции ветвления

Инструкция ветвления `if` позволяет выполнять различные действия в зависимости от некоторого логического утверждения (будем называть его условием).

```
if( условие )
{
    действие
}
```

В данном примере действие будет выполнено только в том случае, если условие верно. В тех случаях когда выполняется лишь одна строка, фигурные скобки можно опустить, напрмер:

```
if( x < 0 ) // если верно, что x < 0
    cout << "Вы ввели отрицательное число" << endl;
```

Когда требуется выполнить одно действие при верном условии и другое при неверном, используется блок `else`.

```
if( условие )
{
    действие 1
}
else
{
    действие 2
}
```

Для рассмотреня нескольких вариантов в блок `else` помещают еще один `if`:

```
if( условие1 )
{
    действие 1
}
else if( условие 2 )
{
    действие 2
}
else
{
    действие 3
}
```

В логических выражениях (в условиях) используются операторы `&& (И)`, `|| (ИЛИ)`, `! (НЕ)`.

В случаях, когда требуется сравнить целочисленную переменную со списком констант, удобно использовать инструкцию `switch`:

```
int x;
cin >> x;
switch( x )
{
    case 0:
        cout << "ноль" << endl;
        break;
    case 1:
        cout << "один" << endl;
        break;
    default:
        cout << "неизвестная цифра" << endl;
}
```

Каждое условие в `switch` завершается инструкцией `break`, иначе будут выполнены все остальные блоки `case`. Например, если в данном примере `x = 0`, то при отсутствии `break` в первом `case` на экран будут выведены слова `ноль один`.

Сравнение дробных чисел. Из-за конечной точности представления чисел с плавающей запятой, точное равенство для типов `float`, `double` и `long double` не имеет смысла. То есть, когда нужно сравнить две переменные типа `double` сравнивают модуль их разности с некоторым небольшим значением, играющим роль точности. Например, если модуль разности между `a` и `b` меньше либо равен `0.0001`, то числа считаются примерно равными.

4.1 Не используя функцию `fabs`, вычислите модуль введенного числа.

Решение

Модуль (или абсолютное значение) числа определяется следующим образом:

$$|x| = \begin{cases} x, & \text{если } x \geq 0, \\ -x, & \text{если } x < 0. \end{cases}$$

То есть, если введенное число больше либо равно нулю, то программа его не должна менять, а если оно меньше нуля, то необходимо изменить его знак на противоположенный.

```
#include <iostream>

using namespace std;

int main()
{
    cout << "Введите число: ";
    int x;
    cin >> x;

    int y; // здесь сохраним модуль числа

    if(x >= 0)
        y = x;
    else
        y = -x;

    cout << "Модуль числа равен: " << y << endl;
}
```

4.2 Решите предыдущую задачу без дополнительной переменной.

4.3 Вычислите обратный квадратный корень числа x (то есть $1/\sqrt{x}$).

Решение

Корень можно вычислить, воспользовавшись функцией `sqrt`, которая принимает число с плавающей точкой и возвращает квадратный корень этого числа.

В данной задаче возможно деление на 0, чего следует избегать. Так как x — число с плавающей точкой, точно сравнивать его с нулем нельзя, поэтому потребуем, чтобы x был достаточно отдален от нуля, например, на величину 0.001.

```
#include <iostream>
#include <cmath>

using namespace std;

int main()
{
    cout << "Введите число: ";
    double x;
    cin >> x;

    // убедимся, что x больше 0.001 или меньше -0.001
    if( fabs(x) > 0.001 )
        cout << "Обратный корень равен: " << 1 / sqrt(x) << endl;
    else
        cout << "Ошибка: знаменатель слишком близок к нулю." << endl;
}
```

4.4 Определите, является ли введенное число четным.

4.5 Выведите наименьшее из двух введенных чисел.

4.6 Выведите наименьшее из трех введенных чисел.

4.7 Вводятся два целых числа a и k. Определите, является ли k делителем a.

4.8 Напишите программу вычисления функции

а) $$y = \begin{cases} 3x, & \text{если } x \geq 2, \\ 6, & \text{если } x < 0. \end{cases}$$

б) $$y = \begin{cases} 3, & \text{если } x \geq 5, \\ x + 1, & \text{если } 0 \leq x < 5, \\ x^2 - 2, & \text{если } x < 0. \end{cases}$$

4.9 По двум датам (число, месяц, год) определите, какая из них предшествует другой.

4.10 Определите, принадлежит ли точка с координатой x отрезку $[a, b]$.

4.11 Определите, где находится точка с координатами (x, y) по отношению к прямой $y = 5$ (выше, ниже или лежит на прямой).

4.12 Определите, попадает ли точка с координатами (x, y) в квадрат с координатами цетнра $(x1, y1)$ и длиной стороны a.

4.13 Найдите корни уравнения $ax + b = 0$ по заданным a и b.

4.14 Определите, попадает ли точка с заданными координатами (x, y) в треугольник с вершинами $A(-2, -1)$, $B(0, 5)$ и $C(3, -1)$.

4.15 Определите, попадает ли точка с заданными координатами (x, y) в круг с центром в точке (a, b) и радиусом r.

4.16 Нарисуйте на плоскости область, в которой и только в которой истинно указанное выражение:

```
а) (y >= x) && (y + x >= 0) && (y <= 1)
б) (sqrt(x) + sqrt(y) < 1) || fabs(x) < 1
в) (fabs(x) <= 1) && (fabs(y) >= 1)
г) (sqrt(x) + sqrt(y) <= 4) && (y <= x)
```

4.17 Найдите корни уравнения $ax^2 + bx + c = 0$ по заданным a, b, c.

4.18 Найдите корни уравнения $ax^4 + bx^2 + c = 0$ по заданным a, b, c.

4.19 У исполнителя две команды: 1) прибавь A, 2) прибавь B. Первая из них увеличивает число на экране на A, вторая — на B. Програма для этого исполнителя — это последовательность команд. Сколько различных чисел можно получить из числа 1 с помощью программы, которая содержит ровно n команд? Входные данные: A, B, n.

4.20 Треугольник задан координатами своих вершин. Определите, является ли он прямоугольным.

4.21 Найдите площадь пересечения двух прямоугольников, заданных координатами своих вершин (не обязательно целыми).

5 Инструкции циклов

В C++ существует три базовые инструкции циклов. Инструкция `while` позволяет выполнять действия пока верно некоторое логическое условие:

```
while( условие )
{
    тело цикла
}
```

Один проход по телу цикла называется итерацией.

Часто для цикла требуются дополнительные данные, например – переменная, которая будет играть роль счетчика цикла чтобы выполнить необходимое количество итераций. В таких случаях удобно использовать инструкцию `for`.

```
for( инициализация; условие; действие )
{
    тело цикла
}
```

При инициализации можно объявлять локальные (для цикла) переменные и задавать их начальные значения, цикл будет продолжаться, пока верно логическое условие, в качестве действия подразумевается изменение параметров, обеспечивающих цикл (например, увеличение счетчика). Любой из параметров может быть опущен. Инструкции `while` и `for` называются инструкциями цикла с предусловием. То есть логическое условие проверяется перед выполнением очередной итерации.

Для организации цикла с постусловием, то есть где условие проверяется после выполнения итерации используется инструкция `do-while`.

```
do
{
    тело цикла
}
while( условие );
```

Такая конструкция удобна когда требуется независимо ни от чего выполнить первую итерацию.

5.1 Выведите на экран n звездочек.

Решение

Пример использования инструкции `while`.

```
#include <iostream>

using namespace std;

int main()
{
    cout << "Введите количество звездочек: ";
    int n;
    cin >> n;

    // объявим переменную, которая будет считать
    // количество уже выведенных звездочек
    int i = 0;

    // пока выведено меньше n звездочек, будем
    // выводить по одной и увеличивать i на 1.
    while( i < n )
    {
        cout << "*";
        ++i;
    }
}
```

В данном примере для обеспечения цикла используется переменная `i`, которую необходимо объявить до цикла и не забывать изменять на каждой итерации. Такой механизм можно реализовать более лаконично с помощью инструкции `for`.

```
#include <iostream>

using namespace std;

int main()
{
    cout << "Введите количество звездочек: ";
    int n;
    cin >> n;

    for(int i = 0; i < n; ++i)
        cout << "*";
}
```

5.2 Решите предыдущую задачу, не вводя дополнительную переменную `i`.

5.3 Выведите на экран квадрат, нарисованный из $n \times n$ звездочек.

5.4 Выведите на экран все целые числа от 0 до n.

5.5 Выведите на экран квадраты целых чисел от 0 до n.

5.6 Выведите на экран четные числа от 0 до n.

5.7 Проверьте, является ли введенное натуральное число степенью числа 3.

5.8 Выведите на экран все делители целого числа n.

5.9 Вычислите наибольший общий делитель двух целых чисел.

5.10 Выведите на экран таблицу умножения всех натуральных чисел, не превышающих n.

6 Суммы, произведения, последовательности, неравенства

6.1 Вычислите сумму:
а) $s = 1 + 2 + \ldots + n$
б) $s = 1^2 + 2^2 + \ldots + n^2$
в) $s = 1 + \sqrt{1.1} + \sqrt{1.2} + \ldots + \sqrt{2}$
г) $s = 3^2/2 + 6^2/3 + 9^2/4 + \ldots + (3n)^2/(n+1)$
д) $s = \sqrt{x} + \sqrt{x^2} + \ldots + \sqrt{x^n}$
е) $s = 1! + 2! + \ldots + n!$
ё) $s = 1 \cdot 3 + 2 \cdot 3^2 + 3 \cdot 3^3 + \ldots + n \cdot 3^n$
ж) $s = 1! + (1! + 2!) + \ldots + (1! + 2! + \ldots + n!)$
з) $s = 1 - 2 + 4 - 8 + \ldots \pm 2^n$

6.2 Вычислите произведение
а) $P = (1 - 1/2^2)(1 - 1/3^2)\ldots(1 - 1/n^2)$
б) $P = (1/1 + 2^2)(1/2 + 3^2)\ldots(1/n + (n+1)^2)$
в) $P = (2/1 - x)(2/2 - x^2)(2/3 - x^3)\ldots(2/n - x^n)$
г) $P = (1/2 - 5)(1/3 - 5^2)(1/4 - 5^3)\ldots(1/(n+1) - 5^n)$
д) $P = (1/9 + 2)(1/9^2 + 4)(1/9^3 + 6)\ldots(1/9^n + 2n)$
е) $P = (1^2 + 2^2)(1^2 + 2^2 + 3^2)\cdots(1^2 + 2^2 + \ldots + n^2)$

6.3 Вычислите k-й член последовательности $\{a_n\}$, если
а) $a_1 = 3$, $a_{n+1} = na_n + 1/n$
б) *Последовательность Фибоначчи.* $a_1 = 1$, $a_2 = 1$, $a_{n+1} = a_n + a_{n-1}$
в) $a_0 = 1$, $a_1 = 1$, $a_{k+1} = 2a_k - a_{k-1} + 1$

6.4 Найдите наименьшее натуральное число, удовлетворяющее неравенству
а) $x^3 + a > 3x$, б) $x^2 + ax + a(a+1) < 2^x$, в) $3^x + a < (a^2+3)^x$.

6.5 Найдите наибольшее целое отрицательное число, удовлетворяющее неравенству
а) $x^3 + a(a+1) < 3x$, б) $4^x > x - a$ в) $5^{-x} > (a^2+3)^x$.

6.6 Вычислите с точностью 0.001:
а) $y = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \ldots$
б) $y = x + x^3/3! + x^5/5! + \ldots$;
в) $y = x - x^3/3 + x^5/5 - x^7/7 + \ldots$.

6.7 Найдите минимальное из n введенных чисел.

6.8 В ящике находятся мячики n различных цветов. Количество мячиков каждого цвета известно. Определите наименьшее количество мячиков, которое нужно достать чтобы среди них оказалось p мячиков одного цвета. Данные вводятся в следующем виде: n, p, а далее вводится n чисел, определяющих количество мячиков каждого цвета.

6.9 Выведите на экран квадраты целых чисел от 0 до n, не используя операцию умножения и вложенные циклы.

6.10 Целое число n, большее 7, может быть представлено в виде $3m + 5n$, где m и n — целые неотрицательные числа. Напишите программу, которая находит любую из возможных пар m и n для введенного числа.

6.11 *Гипотеза Коллатца (гипотеза* $3n + 1$*).* Рассмотрим некоторое натуральное число n. Если оно четно, то поделим его на 2, а если нечетно — то умножим на 3 и прибавим 1. Применяя это правило к n, и, продолжая дальше, получим последовательность чисел.

В 1937 году Лотар Коллатц предположил, что, начав с любого натурального n в конце концов мы неминуемо получим число 1. Пусть, например, $n = 14$. Запишем последовательность:

$$14, 7, 22, 11, 34, 17, 52, 26, 13, 40, 20, 10, 5, 16, 8, 4, 2, 1.$$

Напишите программу, которая для введенного n выводит на экран последовательность Коллатца.

7 Циклы. Цифры и остаток от деления

Для решения некоторых задач из этого раздела понадобится оператор `%`, определяющий остаток от деления двух целых чисел. Например, `8%3 = 2`.

7.1 Для введенной целочисленной переменной выведите на экран значения `x*10`, `x/10`, `x%10`? Проанализируйте результаты и объясните связь указанных операций с числом 10.

7.2 Посчитайте количество цифр в десятичной записи данного натурального числа.

7.3 Вычислите сумму цифр заданного натурального числа.

7.4 Напишите программу, переставляющую цифры в натуральном числе в обратном порядке.

7.5 Проверьте, является ли десятичная запись введенного целого неотрицательного числа палиндромом (то есть читается одинаково слева направо и справа налево, например: 18981, 5, 6226).

7.6 Определите количество трехзначных натуральных чисел, сумма цифр которых равна n.

7.7 Определите, может ли введенное число n быть записью некоторого числа в системе счисления p, где $1 < p \leq 10$

7.8 Напишите программу, переводящую введенное десятичное число в двоичную систему счисления.

7.9 Напишите программу, переводящую введенное двоичное число в десятичную систему счисления.

7.10 Напишите программу, определяющую количество переносов разрядов, которые необходимо сделать при сложении в столбик натуральных чисел m и n.

7.11 Проверьте, могут ли цифры в данном натуральном числе n быть переставлены так, что получится палиндром.

7.12 Рассмотрим натуральное число n построим число m, переставив цифры n в обратном порядке (например, при $n = 438$, $m = 834$). Сложив эти числа проверим, получился ли палиндром. Если нет, то повторим действия. Найдется ли для каждого начального значения палиндром — неизвестно, хотя, для большого количества чисел палиндромы найдены.

Напишите программу, которая для введенного натурального n ищет палиндром следуя приведенному алгоритму и определяет количество итераций, за которое палиндром найден. Максимальное количество итераций – 10000.

7.13 Для натурального числа n найдите минимальное число p такое, что первые цифры числа 2^p совпадают с числом n.

7.14 Даны натуральные числа n и k, $n > 1$. Напишите программу, которая выводит на экран k десятичных цифр числа $1/n$. При наличии двух десятичных разложений выбирается то из них, которое не содержит девятки в периоде.

7.15 Дано натуральное число $n > 1$. Определить длину периода десятичной записи дроби $1/n$.

7.16 Вводятся целые числа $n \neq 0$ и $p \neq 0$. Затем вводится n целых чисел. Определите, делится ли их сумма на p. Все введенные числа по модулю не превышают 10^9.

7.17 Натуральное число называется *двояким*, если в его десятичной записи встречается не более двух различных цифр. Например: 3, 23, 33, 100, 12121 — двоякие, а числа 123 и 9980 — нет. Для заданного натурального n найдите ближайшее к нему двоякое число. Если таких числа два, то любое из них.

7.18 Для любого заданного целого числа $0 \leq n \leq 10000$, не кратного 2 и 5, существует число, кратное n, такое, что в десятичной записи оно является последовательностью единиц.

Напишите программу, которая по введенному n вычисляет количество единиц в минимальном из чисел вида $111\ldots1$ кратных n.

8 Алгоритм Евклида

8.1 *Алгоритм Евклида*. Пусть a и b - целые числа, не равные одновременно нулю. Требуется найти НОД(a, b). Докажите, что

$$\text{НОД}(a, b) = \text{НОД}(a - b, b) = \text{НОД}(a, b - a).$$

Заметим, что НОД$(x, 0) = x$. Реализуйте следующий алгоритм: пока a и b не равны нулю, выбираем большее из них и вычитаем из него меньшее. Когда a или b обратится в ноль, ненулевая переменная будет равна искомому наибольшему общему делителю. Обоснуйте, почему.

8.2 *Модифицированный алгоритм Евклида*. Вместо вычитания меньшего из большего, докажите и используйте следующее свойство:

$$\text{НОД}(a, b) = \text{НОД}(a \bmod b, b) = \text{НОД}(a, b \bmod a).$$

Требуется вычислить последовательность остатков от деления. Последний ненулевой остаток от деления будет равен НОД(a, b).

8.3 Напишите модификацию алгоритма Евклида, доказав и воспользовавшись следующими соотношениями:

$$\text{НОД}(2a, 2b) = 2\text{НОД}(a, b), \quad \text{НОД}(2a, b) = \text{НОД}(a, b) \text{ при нечетном b.}$$

8.4 Исследуйте связь наибольшего общего делителя и наименьшего общего кратного. Используя алгоритм Евклида, вычислите наименьшее общее кратное двух чисел.

8.5 Вводится n целых чисел. Найдите их наибольший общий делитель.

8.6 *Соотношение Безу*. Пусть a и b – не равные одновременно нулю целые числа. Тогда найдутся такие x и y, что выполняется соотношение: НОД$(a, b) = ax + by$. Напишите программу, которая по заданным a и b вычисляет x и y, удовлетворяющие соотношению Безу.

9 Делимость и простые числа

9.1 *Основная теорема арифметики.* Любое натуральное число $n > 1$ можно представить единственным образом (с точностью до порядка сомножителей) в виде

$$n = p_1 \cdot p_2 \cdot \ldots \cdot p_k,$$

где p_i — простые числа (заметим, что некоторые из них могут совпадать).

Напишите программу, выводящую разложение данного натурального числа на простые множители.

9.2 *Каноническое разложение на простые множители..* Любое натуральное число $n > 1$ можно представить единственным образом (с точностью до порядка сомножителей) в виде:

$$n = p_1^{q_1} \cdot p_2^{q_2} \cdot \ldots \cdot p_k^{q_k}.$$

где p_i — различные простые числа.

Напишите программу, выводящую каноническое разложение данного натурального n.

9.3 Напечатайте все простые числа, меньшие n.

9.4 Дано натуральное число n. Требуется представить его в виде суммы двух натуральных чисел a и b таких, что НОД(a, b) максимален.

9.5 Дана дробь $\frac{a}{b}$. Требуется ее сократить, то есть найти такие c и d, что $\frac{c}{d} = \frac{a}{b}$, где c – целое, а d – наименьшее возможное натуральное число.

9.6 Для введенного натурального числа a найдите наименьшее натуральное n такое, что n^n делится на a.

9.7 *Теорема о рациональных корнях многочлена.* Пусть многочлен с целыми коэффициентами

$$P_n(x) = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 p^1 + a_0$$

имеет рациональный корень $x = \frac{p}{q}$. Тогда p является делителем коэффициента a_0, а число q является делителем a_n.

Напишите программу, которая будет искать все целые x, удовлетворяющие уравнению:

$$ax^3 + bx^2 + cx + d = 0$$

при заданных целых a, b, c, d, не превосходящих по модулю $2 \cdot 10^9$.

Сравните скорость решения с полным перебором всех корней.

9.8 Пусть $a_1 = 2$, $a_2 = 3$, $a_n = a_1 \cdot a_2 \cdot \ldots \cdot a_{n-1} - 1$. Назовем числа a_i псевдопростыми. Для заданного натурального числа x нужно ответить на вопрос: можно ли x однозначно представить в виде произведения псевдопростых чисел (представления, отличающиеся только порядком множителей, считаются одинаковыми), и, если можно, вывести разложение.

10 Одномерные массивы

Массивом называется последовательность элементов одного типа, имеющих общее имя. Обращение к элементу массива осуществляется через его номер (индекс). В общем виде объявление массива можно записать так:

```
T m[N];
```

Здесь **T** – некоторый тип, *m* – имя массива, *N* – количество элементов. Важно помнить, что *N* должно быть целочисленной константой. То есть, это может быть некоторое число или предварительно объявленная константа, но не переменная. Рассмотрим несколько примеров.

```
char m[10]; //правильно
```

```
int N = 5;
float m[N]; //неправильно, N -- переменная

const int N = 5;
float m[N]; //правильно
```

Если в массиве N элементов, тогда их номера будут иметь значения $\{0, 1, 2, ..., N-1\}$. При обращении к элементу массива, которого нет (например к N или к $N + 10$), произойдет выход за границы массива. В лучшем случае это приведет к некорректной работе программы, в худшем – к аварийному завершению работы программы.

10.1 Напишите программу, вычисляющую сумму введенных чисел.

Решение

```
#include <iostream>

using namespace std;

int main()
{
    const int n = 10;
    int v[n]; //объявляем массив

    for(int i = 0; i < n; ++i)
    {
        cout << "Введите элемент массива " << i << ": ";
        cin >> v[i]; //считываем в цикле числа
    }

    int s = 0; // переменная для накопления суммы

    //перебираем массив и добавляем каждый элемент в s
    for(int i = 0; i < n; ++i)
        s += v[i];

    cout << "Сумма элементов массива равна: " << s << endl;
}
```

10.2 Заполните массив случайными числами от -25 до 25.

Решение

В задачах на массивы вместо того чтобы каждый раз вводить числа с клавиатуры, часто удобно заполнить массив автоматически случайными числами. При этом разумно после заполнения вывести полученный массив на экран.

```
#include <iostream>
#include <ctime>
#include <cstdlib>

using namespace std;

int main()
{
    srand(time(0));

    const int n = 10;
    int v[n]; //объявляем массив

    /*
       Так как функция rand() возвращает некторое неотрицательное
       целое случайное число, вычтем возьмем остаток от деления на 51,
       получив случайное число от 0 до 50, а затем вычтем из него 25,
       получив случайное число от -25 до 25.
    */
    for(int i = 0; i < n; ++i)
        v[i] = rand() % 51 - 25;

    // выводим полученный массив
    for(int i = 0; i < n; ++i)
        cout << v[i] << " ";
    cout << endl;
}
```

10.3 Замените в массиве все отрицательные эелементы нулями

Решение

Заполним массив случайными числами от -25 до 25, а затем заменим отрицательные элементы нулями. В конце программы выведем на экран полученный после преобразования массив.

```
#include <iostream>
#include <ctime>
#include <cstdlib>

using namespace std;

int main()
{
    srand(time(0));

    const int n = 10;
    int v[n]; //объявляем массив

    for(int i = 0; i < n; ++i)
        v[i] = rand() % 51 - 25;

    // выводим массив
    for(int i = 0; i < n; ++i)
        cout << v[i] << " ";
    cout << endl;

    // заменяем отрицательные элементы нулями
    for(int i = 0; i < n; ++i)
        if(v[i] < 0) v[i] = 0;

    // выводим полученный массив
    for(int i = 0; i < n; ++i)
        cout << v[i] << " ";
    cout << endl;

}
```

10.4 Вычислите сумму элементов массива.

10.5 Вычислите сумму квадратов положительных элементов массива

10.6 Вычислите среднее арифметическое отрицательных элементов одномерного массива.

10.7 Подсчитате количество четных элементов одномерного массива.

10.8 В одномерном массиве вычислите сумму элементов, стоящих на четных местах.

10.9 Вычислите среднее арифметическое элементов массива, значения которых попадают в интервал $[-8.1, 1.3]$.

10.10 Вычислите произведение элементов массива, значения которых попадают в интервал $[a, b]$.

10.11 Подсчитайте количество положительных элементов массива, имеющих нечетные индексы.

10.12 Замените в массиве все нечетные элементы значениями их индексов.

10.13 В одномерном массиве переставьте 1-й и 2-й элементы, 3-й и 4-й, 5-й и 6-й и т. д.

10.14 Вычислите отношение количества отрицательных элементов к количеству положительных в массиве.

10.15 Определите количество перемен знака в массиве.

10.16 Определите, все ли элементы данного массива положительны.

10.17 Определите, чередуются ли положительные и отрицательные элементы данного массива.

10.18 Определите, является ли последовательность элементов массива до первого отрицательного числа возрастающей.

10.19 Определите, являются ли элементы массива арифметической прогрессией.

10.20 Подсчитайте количество локальных максимумов среди элементов массива.

11 Минимум и максимум

11.1 Определите положение максимального элемента в массиве. Если в массиве более одного максимального — определить положение последнего из них.

11.2 В заданном массиве переставьте минимальный и максимальный элементы.

11.3 Найдите разность между минимальным и максимальным элементами массива.

11.4 Замените все элементы массива, большие чем a, значением максимального элемента этого массива.

11.5 Подсчитайте количество элементов массива, равных по модулю максимальному элементу.

11.6 Определите положение минимального элемента из находящихся до первого отрицательного в массиве.

11.7 Найдите максимальный элемент массива из элементов, стоящих на нечетных местах.

11.8 Найдите максимальный элемент в первой половине массива и минимальный во второй.

11.9 Определите количество элементов одномерного массива, являющихся делителями максимального элемента в этом массиве.

11.10 Замените нулями все элементы, не являющиеся делителями максимального элемента.

11.11 Найдите два последовательных наибольших элемента в одномерном массиве.

12 Сортировка

12.1 *Сортировка пузырьком.* Запустим цикл, в котором счетчик `i` принимает значения от $n-1$ до 1 включительно. На каждой итерации этого цикла будем запускать вложенный цикл, в котором переменная `k` принимает значения от 0 до $i-1$. Таким образом получается, что `k < i`. На каждой вложенного цикла сравним элементы с индексами `k` и `k+1`. Если элемент с индексом `k` больше элемента с индексом `k+1`, то поменяем их местами.

12.2 Найдите способ сокращения количества операций в сортировке пузырьком, если известно, что исходный массив либо уже отсортирован, либо почти отсортирован.

12.3 *Сортировка выбором.* Найдем в массиве минимальный элемент и поменяем его местами с нулевым элементом. Затем повторим эту же операцию на отрезке `[1, n]`, затем на отрезке `[2, n]` и т.д. Так, помещая каждый раз в начало отрезка минимальный элемент, получим массив отсортированный по позрастанию.

12.4 *Сортировка вставками.* Пусть последовательность элементов с индексами `0 ... i` уже отсортирована. Требуется вставить элемент `i + 1` в эту последовательность. Сохраним его в некоторую переменную `t` и, начиная с `i`-го элемента, будем сдвигать все элементы с индексами `i`, `i-1`, `...` вправо на одну позицию до тех пор, пока очередной элемент не окажется меньше `t` или мы не достигнем начала массива. Так мы найдем позицию для вставки изначально `i+1`-го элемента в отсортированную часть массива.

12.5 *Сортировка слиянием.* Основная идея алгоритма заключается в том, что две отсортированные последовательности можно объединисть в одну отсортированную последовательность за линейное время. Будем считать, что каждый элемент массива является отсортированным массивом, состоящим из одного элемента. Тогда можно эффективно объединить элементы с индексами 0 и 1, 2 и 3 и т.д. в пары отсортированных элементов. Затем объединяем 0,1 и 2,3; 4,5 и 6,7 и т.д. Так исходный массив представляется в виде нескольких отсортированных последовательностей, которые попарно объединяются.

12.6 *Пирамидальная сортировка.* Одномерный массив можно рассматривать в виде бинарного дерева, где родительскому элементу `i` будут соответствовать потомки с индексами `2i+1` и `2i+2`. Если при этом `v[i]` не меньше элементов `v[2*i+1]` и `v[2*i+2]`, то такое дерево называют *кучей* (*heap*). Сортировка реализуется в два этапа. На первом этапе исходный массив преобразуется так, чтобы выполнялось свойство кучи. При этом на вершину попадает максимальный элемент. Этот элемент меняется местами с последним элементом массива. Такая операция нарушает свойство кучи, т.к. на вершину попадает произвольный элемент. Поэтому, рассматривая дерево уже без последнего элемента в массиве, элемент на вершине проталкивается вниз так чтобы свойство кучи снова выполнялось. Очевидно, что на вершине снова окажется максимальный элемент (уже без последнего) и процесс повторяется вновь.

12.7 *Сортировка Хоара.* Рассмотрим отрезок массива `[l, r]`. Пусть `m = v[(l+r)/2]` – значение серединного элемента данного отрезка. Переставим элементы `[l, r]` таким образом чтобы сначала шли все элементы не большие `m`, затем не меньшие `m`. Применим этот алгоритм к двум полученным отрезкам.

12.8 Известно, что элементы массива – натуральные числа, не превосходящие 1000. Отсортируйте этот массив за линейное время.

12.9 В заданном массиве подсчитайте количество различных элементов.

12.10 Найдите в массиве пару наименее отличающихся друг от друга элементов.

12.11 Задан массив натуральных чисел. Найдите минимальное натуральное число, не представимое в виде суммы элементов этого массива. Каждый элемент должен входить в сумму только один раз.

13 Разные задачи на циклы

13.1 Вычислите значение многочлена

$$P_n(x) = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x + a_0$$

по заданной переменной x и массиву коэффициентов a.

13.2 Выведите все пары элементов массива, которые в сумме дают число k.

13.3 Заполните массив случайными числами, причем так, чтобы числа шли в порядке возрастания.

13.4 Переставьте элементы одномерного массива так, чтобы сначала шли все нечетные элементы, а потом четные.

13.5 Определите, являются ли элементы массива арифметической прогрессией.

13.6 Выведите на экран элементы, являющиеся степенями 2.

13.7 Выведите на экран элементы, являющиеся факториалами чисел.

13.8 Запишите в массив B подряд все отрицательные элементы из массива A. Остаток B заполнить нулями.

13.9 Переставьте элементы одномерного массива в обратном порядке.

13.10 Переставьте элементы массива таким образом, чтобы сначала шли все положительные, а затем все отрицательные элементы.

13.11 Перенесите в массиве все нулевые элементы в конец массива (порядок всех остальных элементов должен остаться прежним).

13.12 Определите максимальную длину последовательности из подряд идущих нулей в массиве.

13.13 Сдвиньте элементы массива циклически на k позиций влево.

13.14 В массивах $a1$ и $a2$ хранятся результаты двух забегов. Запишите в массив b результаты десяти лучших участников.

13.15 Координаты n точек на плоскости заданы в виде двух массивов X и Y. Найдите пару самых близких из них.

13.16 Задан массив M длины n. Вычислите следующую сумму:
а) $M[0] - M[1] + M[2] - ... \pm M[n-1]$;
б) $M[0] \cdot M[n-1] + M[1] \cdot M[n-2] + \ldots + M[n-1] \cdot M[0]$.

13.17 По данным коэффициентам многочленов A и B вычислите коэффициенты многочлена C, являющегося их произведением.

13.18 В заданном целочисленном массиве найдите элементы, сумма которых равна данному числу (если известно, что такие существуют).

13.19 Напечатайте все последовательности длины k из чисел $1 \ldots n$.

13.20 Напечатайте все перестановки из чисел $1 \ldots n$.

13.21 Напечатайте все подмножества множества $1 \ldots n$.

13.22 В отсортированном массиве найдите заданный элемент за $O(logn)$ операций.

13.23 Определите, может ли число n быть представлено в виде суммы каких-либо элементов массива, взятых по одному разу.

Литература

1. Яворский Р. Сборник задач по программированию.
2. Шень А. Программирование: теоремы и задачи. — 4-е изд., стереотипн. — М.:МЦНМО, 2011. — 296 с.
3. Московские олимпиады по информатике. — 2-е изд., доп. / Под ред. Е.В. Андреевой, В.М. Гуровица и В.А. Матюхина — М.:МЦНМО, 2009. — 415 с.
4. Стивен Скиена, Мигель Ревилла. Олимпиадные задачи по программированию. Руководство по подготовке к соревнованиям. М.: КУДИЦ-ОБРАЗ, 2005. — 416 с.