

# Функции

Решение любой задачи естественно стараться свести к решению нескольких маленьких подзадач. Если в программе приходится несколько раз выполнять один и тот же алгоритм, то имеет смысл выделить его в подпрограмму и вызывать ее при необходимости. Такая подпрограмма будет называться **функцией**.

Функция имеет следующий вид:

```
тип имя(список аргументов)
{
    тело функции
}
```

Аргументы перечисляются через запятую с указанием типов. Если аргументов нет, ставятся пустые скобки.

# Функции: возвращаемое значение

Функции в языке C++ делятся по смыслу на 2 типа.

- Тип `void`: функция выполняет определённые действия (чтение данных, вывод на экран, и т.д.). Точка выхода из функции – её последняя команда или `return`;
- Любой другой тип (`int`, `bool`, `float`, ...): функция вычисляет некоторое значение и **возвращает** его. Точка выхода из функции – `return <значение>;`

Функции типа `void` вызываются по её имени со списком аргументов:

```
print("hello");
```

После имени функции круглые **скобки ставятся обязательно**, даже если функция не имеет аргументов. Функция должна быть объявлена до того места, где она будет вызвана.

При вызове функций других типов, как правило, сохраняется или используется как-то иначе их возвращаемое значение:

```
int m = max(2, 3);
```

## Функции: пример

```
void print(string text)
{
    cout << text << endl;
    // return; – не обязателен
}
int max(int a, int b) // тип нужно указывать у каждого аргумента
{
    // если a > b, функция завершит работу и вернёт значение a
    if (a > b) return a;
    // здесь можно не писать else, т.к. мы попадём сюда только
    // в том случае, если условие не выполнилось
    return b;
}
int main()
{
    print("hello");
    cout << max(4, 2) << endl; // выводим возвращаемое значение
}
```

# Количество цифр

Дано три символа. Требуется определить, сколько из них являются цифрами. При решении данной задачи реализуйте функцию, которая возвращает 1, если символ – цифра, и 0 – иначе.

## Входные данные

На вход подаются 3 символа без разделителей.

## Результат работы

Напечатайте одно натуральное число – количество цифр.

## Пример

Входные данные	Результат работы
123	3
A5!	1

# Сумма простых чисел

Даны N целых чисел. Определите, какие из чисел являются простыми и вычислите их сумму. Также определите, будет ли их сумма простым числом. При решении данной задачи реализуйте функцию, которая проверяет одно целое число на простоту.

## Входные данные

На вход подаётся натуральное число N, далее ещё N целых неотрицательных чисел.

## Результат работы

На первой строке выведите на экран сумму простых чисел, на второй – слово «Yes», если их сумма – простое число; иначе слово «No».

## Пример

Входные данные	Результат работы
3 3 5 11	19 Yes
5 4 2 0 1 9	2 Yes
2 8 8	0 No

# Суммы цифр

Учительница записала на доске целое число  $N$ . Вовочка подсчитал сумму цифр этого числа и записал ее ниже. С полученным числом он проделал то же самое, и продолжал выписывать числа до тех пор, пока два последних записанных числа не совпали. Ваша задача – найти сумму  $S$  всех выписанных на доску чисел. Реализуйте подсчёт суммы цифр в числе в виде отдельной функции.

## Входные данные

На вход подаётся натуральное число  $N \leq 2 \cdot 10^9$ .

## Результат работы

Напечатайте натуральное число  $S$ .

## Пример

Входные данные	Результат работы
34	48
1234	1246
987654	987711

# Подсчет букв

Дано три символа. Требуется определить, сколько из них являются буквами латинского алфавита. При решении данной задачи реализуйте функцию, которая возвращает 1, если символ – цифра, и 0 – иначе.

## Входные данные

На вход подаются 3 символа без разделителей.

## Результат работы

Напечатайте одно натуральное число – количество букв.

## Пример

Входные данные	Результат работы
i23	1
A5n	2
282	0

# В одном шаге от счастья

Вова купил билет в трамвае 13-го маршрута и сразу посчитал суммы первых трёх цифр и последних трёх цифр номера билета (номер у билета шестизначный). Оказалось, что суммы отличаются ровно на единицу. «Я в одном шаге от счастья», – подумал Вова, – «или предыдущий или следующий билет точно счастливый». Прав ли он?

## Входные данные

На вход подаётся количество билетов  $N$ , далее  $N$  шестизначных номеров билетов.

## Результат работы

Для каждого номера выведите на экран "Yes", если Вова был прав; иначе "No"

## Пример

Входные данные	Результат работы
3	Yes
715068	No
445219	Yes
012200	



# Секрет

Вам в руки попала секретная записка на английском языке. Текст записки может быть любым, главное - код, заложенный в тексте. Чтобы расшифровать записку нужно посчитать количество букв «b» и «g» в записке (на любом регистре). Если букв «b» больше, чем букв «g», то все плохо. Если букв «b» меньше, чем букв «g», то все хорошо. Ну, а если буквы содержатся в записке в одинаковом количестве, то пока не ясно, как дела пойдут. Напишите программу для расшифровки таких секретных записок.

## Входные данные

На вход подаются строки, состоящие из английских букв, цифр, пробелов и знаков препинания.

## Результат работы

Напечатайте все строки в неизменном виде, после на новой строке выведите слово, определяющее тайный смысл записки:

«GOOD» – если все хорошо;

«BAD» – если все плохо;

«NEUTRAL» – если пока не ясно, как пойдут дела.

# Секрет

## Пример

Входные данные	Результат работы
It is rainy and I have bought umbrella	It is rainy and I have bought umbrella BAD
It is rainy and I have bought tea	It is rainy and I have bought tea NEUTRAL
My aunt Ann is greedy!	My aunt Ann is greedy! GOOD

# Рекурсия

Состояние, в котором функция вызывает сама себя или несколько функций вызывают друг друга, называется **рекурсией**. Использование рекурсии часто позволяет упростить реализацию алгоритма, однако иногда её использование существенно снижает эффективность программы.

Пример рекурсивной функции:

```
unsigned long long fact(int n)
{
    if (n < 2)
        return 1;
    else
        return fact(n-1)*n;
}
```

## Рекурсия: плохой пример

Реализуем рекурсивную функцию, вычисляющую  $n$ -ое число Фибоначчи по определению:  $F_n = F_{n-1} + F_{n-2}$ .

```
unsigned long long fib(int n)
{
    if (n < 3)
        return 1;
    else
        return fib(n-1) + fib(n-2);
}
```

Попробуйте вычислить с её помощью 40-ое число Фибоначчи.  
В чём причина низкой производительности?

? Перепишите функцию без использования рекурсии.

# Разворот

Дано натуральное число  $N$  и последовательность из  $N$  элементов. Требуется вывести эту последовательность в обратном порядке. Для решения задачи напишите рекурсивную функцию.

## Входные данные

На вход подаётся натуральное число  $N$ , за ним ещё  $N$  натуральных чисел.

## Результат работы

Выведите  $N$  чисел в обратном порядке.

## Пример

Входные данные	Результат работы
3 1 2 3	3 2 1

## Рекурсия: задачи

При решении следующих задач используйте рекурсию, пользоваться циклами нельзя. Все числа в задачах натуральные, их количество не превышает 10000.

- ? Напечатайте все числа от 1 до  $n$
- ? Вычислите сумму цифр данного числа
- ? Выведите цифры числа справа налево через пробел
- ? Выведите цифры числа слева направо через пробел
- ? Выведите все нечётные элементы последовательности (0 - признак конца)
- ? Выведите все элементы последовательности с нечётными номерами (0 - признак конца)

# Работа с файлами

Для работы с файловой системой используется библиотека `fstream`. Загрузив файлы на чтение или на запись с ними можно работать, как с потоками (например `cin`, `cout`).

```
#include <fstream> // библиотека для работы с файлами
#include <iostream>
using namespace std;

int main()
{
    ifstream input("input.txt"); // открыть файл на чтение
    ofstream out("out.txt"); // открыть файл на запись

    if (!input || !out) // x^x
    {
        cout << "Ошибка загрузки файла" << endl;
        return 0;
    }

    string s;
    while (getline(input, s))
        out << s << endl;
}
```

# Работа с файлами: задачи

Создайте файл `"matrix.txt"` со следующим содержимым:

```
3 4
1 2 4 8
1 3 9 27
1 5 25 125
```

В первой строке файла заданы размеры матрицы  $m \times n$  (количество строк и столбцов). Далее идут строки матрицы.

- ? Считайте содержимое файла и выведите матрицу на экран
- ? Запишите матрицу, умноженную на 2, в файл `"out.txt"`
- ? Вычислите сумму каждой строки матрицы
- ? Вычислите сумму каждого столбца матрицы



# Графы: матрица смежности

Рассмотрим следующую матрицу:

i/j	1	2	3	4	5
1	0	1	0	0	1
2	1	0	0	1	0
3	1	0	0	1	1
4	0	0	0	0	0
5	0	0	1	0	1

Единица в позиции  $(i, j)$  означает, что между **вершинами**  $i$  и  $j$  существует **ребро**. Квадратная матрица, составленная из нулей и единиц, называется **матрицей смежности**, которая однозначно задаёт наборы вершин  $V$  и рёбер между ними  $E$ :

$$V = \{v_1, v_2, v_3, v_4, v_5\},$$

$$E = \{(v_1, v_2), (v_1, v_5), (v_2, v_1) \dots, (v_5, v_5)\}.$$

Множество, объединяющее эти 2 набора  $G = \{V, E\}$ , называется **графом**. Графы часто используются в программировании: с их помощью удобно задавать карты дорог, связи между компьютерами в сети и т.д.

# Матрица смежности: задачи

Создайте файл `"matrix.txt"` со следующим содержимым:

```
5 5
0 1 0 0 1
1 0 0 1 0
1 0 0 1 1
0 0 0 0 0
0 0 1 0 1
```

В первой строке файла заданы размеры матрицы  $m \times n$  (количество строк и столбцов). Далее идут строки матрицы смежности. Будем считать, что матрицей задана карта дорог между городами.

? Подсчитайте общее количество дорог между городами

? Определите, между какими городами есть дороги

? Определите, есть ли прямая дорога из города  $i$  в город  $j$

? Определите, сколько в графе **концевых вершин**, т.е. городов, из которых нет ни одной дороги

? Определите, сколько в графе **изолированных вершин**, т.е. городов из которых нет и в которые нет ни одной дороги

? Определите, сколько в графе **петель**, т.е. дорог, ведущих из города  $i$  в город  $i$

# Матрица смежности: ещё задачи

Создайте файл `"matrix.txt"` со следующим содержимым:

```
5 5
0 1 0 0 1
1 0 0 1 0
1 0 0 1 1
0 0 0 0 0
0 0 1 0 1
```

В первой строке файла заданы размеры матрицы  $m \times n$  (количество строк и столбцов). Далее идут строки матрицы смежности. Будем считать, что матрицей задана карта дорог между городами.

? Найдите **степени** всех вершин графа, т.е. количество дорог, входящих и выходящих из каждого города

? Граф называется **регулярным**, если степени всех его вершин равны. Проверьте, является ли заданный граф регулярным

? Граф называется **ориентированным**, если его рёбрам задано направление. Проверьте, является ли заданный граф ориентированным

# Надёжный пароль

Будем считать пароль надёжным, если он включает в себя хотя бы одну строчную английскую букву, хотя бы одну заглавную английскую букву и хотя бы одну цифру, при этом его длина должна быть не менее 12 символов. Требуется по данному паролю определить, надёжный ли он.

## Входные данные

На вход программе подаётся одна строка.

## Результат работы

Выведите на экран "Safe" если пароль надёжный, и "Unsafe" в противном случае.

## Пример

Входные данные	Результат работы
VasyaPupkin007	Safe
xXx1337xXx	Unsafe
PupkinVasiliyVasilyevich	Unsafe

# IP-адрес

Для того чтобы выходить в Интернет, каждому компьютеру присваивается так называемый IP-адрес. Он состоит из четырех целых чисел в диапазоне от 0 до 255, разделенных точками. По заданной строке требуется определить, является ли она правильным IP-адресом.

## Входные данные

На вход программе подаётся произвольная строка не длиннее 1000 символов.

## Результат работы

Выведите на экран "Good" если строка является правильным IP-адресом, и "Bad" в противном случае.

## Пример

Входные данные	Результат работы
192.168.1.1	Good
1.2.3.4	Good
123.456.321.1	Bad
1..2.3	Bad
1.2.3.	Bad
10.010.0010.12	Good

# Напёрстки

Шулер показывает следующий трюк. Он имеет три одинаковых наперстка. Под первый (левый) он кладет маленький шарик. Затем он очень быстро выполняет ряд перемещений наперстков, каждое из которых – это одно из трех перемещений - А, В, С:

А - обменять местами левый и центральный наперстки,

В - обменять местами правый и центральный наперстки,

С - обменять местами левый и правый наперстки.

Необходимо определить, под каким из наперстков окажется шарик после всех перемещений.

## Входные данные

На вход программе подаётся строка, состоящая из букв А, В, С.

## Результат работы

Выведите на экран номер наперстка, под которым окажется шарик: 1, 2 или 3.

## Пример

Входные данные	Результат работы
СВАВСАССС	1
С	3

# Цветной дождь

В Банановой республике очень много холмов, соединенных мостами. На химическом заводе произошла авария, в результате чего испарилось экспериментальное удобрение. На следующий день выпал цветной дождь, причем он прошел только над холмами. В некоторых местах падали красные капли, в некоторых - синие, а в остальных - зеленые. Президенту Банановой республики захотелось покрасить мосты между вершинами в цвет холмов, которые они соединяют. Если холмы разного цвета, то покрасить мост таким образом не удастся. Посчитайте количество таких "плохих" мостов.

## Входные данные

В файле `m1.txt` в первой строке записано число холмов  $n$ , а в следующих  $n$  строках записана матрица смежности, описывающая наличие мостов между холмами. В последней строке записаны  $n$  чисел, обозначающие цвета холмов: 1 - красный, 2 - синий, 3 - зеленый.

## Результат работы

Выведите на экран количество "плохих" мостов.

# Цветной дождь

## Пример

Входные данные	Результат работы
7 0 1 0 0 0 1 1 1 0 1 0 0 0 0 0 1 0 0 1 1 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 1 0 1 0 1 0 0 1 0 0 0 0 0 0	4



# Числа-палиндромы

Палиндромом называется строка, которая одинаково читается как слева направо, так и справа налево. Рассмотрим все натуральные числа, запись которых в десятичной системе счисления является палиндромом (при этом запись не начинается с нуля). Например, числа 121 и 1331 являются палиндромами, а число 123 — нет. По данному числу  $N$  найдите  $N$ -е в порядке возрастания число-палиндром.

## Входные данные

Программа получает на вход одно натуральное число  $N$ , не превосходящее 100000.

## Результат работы

Программа должна вывести одно натуральное число —  $N$ -е в порядке возрастания число-палиндром.

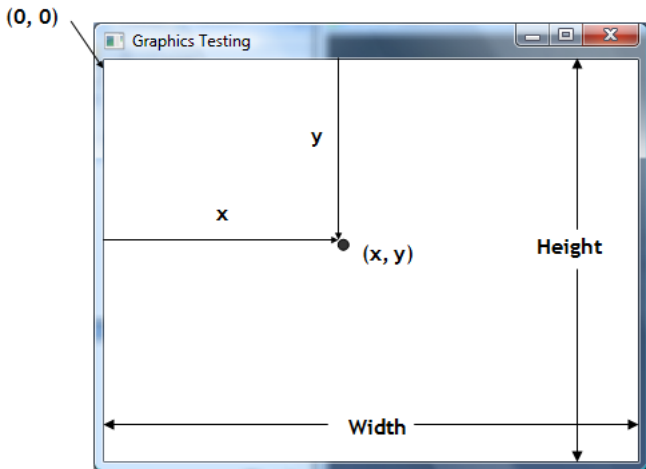
## Пример

Входные данные	Результат работы
20	111

## Библиотека glcanvas: создание окна

Начало координат в окне – левый верхний угол; ось X направлена влево, ось Y – вниз. Функция принимает размеры окна в пикселях; размер по умолчанию:  $640 \times 480$  пикселей.

```
void window(int w = 640, int h = 480)
```



## Функция прорисовки

Все операции, связанные с прорисовкой графики, нужно помещать внутри одной функции. В main эту функцию необходимо специально "зарегистрировать" как glutDisplayFunc. После этого функция будет вызываться автоматически, когда необходимо перерисовать содержимое окна. Для работы с функциями glcanvas нужно подключить саму библиотеку, также удобно использовать пространство имён cnv.

```
// минимальный пример
#include <glcanvas.hpp>
using namespace cnv;

void draw()
{
    glutSwapBuffers();
}

int main()
{
    window();
    glutDisplayFunc(draw);
    glutMainLoop();
}
```

# Основные функции

```
void draw()
{
    // очистка экрана (цвет фона)
    clear(255, 255, 255);
    // окружность и круг (координаты центра и радиус)
    circle(100, 100, 20);
    circle_fill(240, 175, 25);
    // прямоугольники (координаты двух углов)
    rect(120, 150, 200, 450);
    rect_fill(500, 300, 520, 320);
    // отрезки (координаты концов)
    line(40, 40, 80, 80);
    line(80, 80, 20, 80);
    line(20, 80, 40, 40);
    // вывод текста
    position(300, 300);
    text_out << "Hello, world!";
    // вывод содержимого буфера в окно
    glutSwapBuffers();
}
```

# Цвет

Цвет в формате RGB задаётся тремя компонентами (красный, зелёный, синий). Он может быть записан двумя способами: как тройка десятичных чисел (r, g, b) от 0 до 255 или как одно шестнадцатеричное число 0xrrggbb, в котором каждый байт содержит соответствующую компоненту.

#ff0000	#ffff00	#404040	#123456
rgb(255, 0, 0)	rgb(255, 255, 0)	rgb(64, 64, 64)	rgb(18, 52, 86)

В `glcanvas` цвет по умолчанию – чёрный. Изменить его можно функциями `color(hex)` и `color4(r, g, b)`. Кроме того, в функции рисования фигур (`circle`, `rect` и т.д.) можно передать цвет в формате HEX последним аргументом:

```
circle(100, 200, 40, 0xff0000);
```

Для перевода цвета из RGB в HEX есть функция `hexcolor(r, g, b)`:

```
circle(100, 200, 40, hexcolor(255, 0, 0));
```

# Мышь и клавиатура, анимация

Анимация графики осуществляется путём её перерисовки после некоторого события: например, движение мышки, нажатие клавиши на клавиатуре или срабатывание таймера.

```
// Реагирует на движение мыши с нажатой кнопкой (перетаскивание)
void glutMotionFunc(void (*func)(int x, int y));
// Реагирует на любое движение мыши
void glutPassiveMotionFunc(void (*func)(int x, int y));
// Реагирует на клики
// button: GLUT_LEFT_BUTTON, GLUT_MIDDLE_BUTTON, GLUT_RIGHT_BUTTON
// state: GLUT_UP, GLUT_DOWN
void glutMouseFunc(void (*func)(int button, int state, int x, int y));
// Реагирует на нажатие клавиши
void glutKeyboardFunc(void (*func)(unsigned char key, int x, int y));
// В функциях выше x, y - координаты курсора
// Срабатывает по таймеру каждые msecс миллисекунд
void glutTimerFunc(unsigned int msecс, void (*func)(int value), value);
```