# Массивы

Часто возникает необходимость хранить не одну переменную, а набор однотипных переменных. Например, координаты вершин многоугольника или коэффициенты многочлена – это набор числовых данных. Для хранения наборов данных используются *структуры данных*.

Массив — это структура однотипных данных. Массив имеет размер — количество элементов в нем. Каждый элемент массива имеет свой номер (также называемый индексом), обращение к элементу массива осуществляется путем указания его индекса.

В языке C++ элементы нумеруются начиная с 0, поэтому последний элемент массива имеет номер на 1 меньше размера массива.

Массив в языке C++ задается следующим образом:

```
тип_элементов идентификатор[размер];
```

где *тип_элементов* — произвольный тип данных языка C++, который будут иметь элементы массива, например, `int`, `double` и т.д.; *идентификатор* — имя массива, *размер* — число элементов в нем.

**ВНИМАНИЕ: По стандарту языка C++ размер массива должен быть константой, то есть можно определить массив в виде `int A[10 + 5]`, но нельзя это сделать в виде `int A[n]`.**

**Однако, компилятор gcc, которым мы пользуемся, допускает объявления второго вида, При этом нет никаких гарантий, что ваша программа будет откомпилирована каким-либо другим компилятором. На компиляторе Microsoft Visual C++ это работать не будет!**

К элементу массива можно обращаться, как *идентификатор*`[`*индекс*`]`. Например, если было сделано объявление

```
double A[5];
```

то таким образом создается 5 элементов массива типа `double`: `A[0]`, `A[1]`, `A[2]`, `A[3]`, `A[4]`.

При этом надо быть очень аккуратными: обращение по индексам, выходящим за пределы размера массива, приводит к ошибке. Например, в вышеприведенном примере обращаться к A[-1] и A[99] чревато либо “падением” программы, либо чтением “мусорных” данных - тут уж как повезет.

Пример программы, которая создает массив типа `int[]`, заданного пользователем размера, считывает с клавиатуры его элементы, затем прибавляет к каждому элементу массива число `1`, затем выводит результат на экран:

```
#include <iostream>
using namespace std;
int main()
{
    int n;          // Размер массива
    int i;          // Счетчик в циклах
    cin >> n; // Считываем размер массива
    int A[n];       // Объявление массива
                    // Считываем массив
    for (i = 0; i < n; ++i)
    {
        cin >> A[i];
    }
                    // Прибавляем по 1 к каждому элементу
    for (i = 0; i < n; ++i)
    {
        A[i] += 1;
    }
                     // Выводим массив на экран
    for (i = 0; i < n; ++i)
    {
```

```
        cout << A[i] << " ";
    }
                    // Переведем курсор на новую строку
    cout << endl;
    return 0;
}
```

В этом примере при помощи `//` обозначается начало комментария, весь текст после начала комментария и до конца строки компилятором игнорируется. Второй способ объявления комментария: в начале комментария поставить знаки `/*`, а в конце – `*/`. Это позволяет делать комментарии, занимающие несколько строк.

В некоторых задачах этого листочка будет написана загадочная фраза ***«требуется отделить ввод массива»*** или ***«требуется отделить вывод массива».*** Это означает, что необходимо написать отдельную функцию, которая бы считывала массив с клавиатуры или, соответственно, выводила бы его. В этих функциях ничего, кроме собственно ввода массива или вывода его, происходить не может. Пример функции, которая вводит массив:

```
void inputArray(int array[], int length) {
   int i;
   for (i = 0; i < length; ++i) {
      cin >> array[i];
   }
}
```

# Упражнения

## B: Четные индексы

Выведите все элементы массива с четными индексами (то есть `A[0]`, `A[2]`, `A[4]`, ...).

Программа должна быть эффективной и не выполнять лишних действий!

Необходимо отделить ввод массива.

| **Ввод** | **Вывод** |
|---|---|
| 5<br>1 2 3 4 5 | 1 3 5 |

## C: Четные элементы

Выведите все четные элементы массива.

Необходимо отделить ввод массива.

| **Ввод** | **Вывод** |
|---|---|
| 7<br>1 2 2 3 3 3 4 | 2 2 4 |

# D: Больше предыдущего

Дан массив. Выведите все элементы массива, которые больше предыдущего элемента.

Необходимо отделить ввод массива.

| Ввод | Вывод |
|---|---|
| 5<br>1 5 2 4 3 | 5 4 |

# E: Наибольший элемент

Выведите значение наибольшего элемента в массиве

Необходимо отделить ввод массива.

| Ввод | Вывод |
|---|---|
| 5<br>1 2 3 2 1 | 3 |

# F: Количество различных элементов

Дан массив, упорядоченный по неубыванию элементов в нем. Определите, сколько в нем различных элементов.

Необходимо отделить ввод массива.

| Ввод | Вывод |
|---|---|
| 6<br>1 2 2 3 3 3 | 3 |

# G: Вывести в обратном порядке

Выведите элементы данного массива в обратном порядке, не изменяя сам массив.

Необходимо отделить ввод массива.

| Ввод | Вывод |
|---|---|
| 5<br>1 2 3 4 5 | 5 4 3 2 1 |

# H: Переставить в обратном порядке

Переставьте элементы данного массива в обратном порядке, затем выведите элементы данного массива.

Эта задача отличается от предыдущей тем, что вам нужно изменить значения элементов самого массива, поменяв местами `A[0]` с `A[n-1]`, `A[1]` с `A[n-2]`, а затем вывести элементы массива, начиная с `A[0]`.

Необходимо отделить ввод массива. Необходимо отделить вывод массива.

| Ввод | Вывод |
|---|---|
| 5<br>1 2 3 4 5 | 5 4 3 2 1 |

# I: Удалить элемент

Дан массив из N элементов и номер элемента в массиве k. Удалите из массива элемент с индексом k, сдвинув влево все элементы, стоящие правее элемента с индексом k.

Программа получает на вход число N, затем N элементов массива, затем число k.

Программа должна вывести N-1 число – элементы массива после удаления k–го элемента.

Программа должна осуществлять сдвиг непосредственно в массиве, а не делать это при выводе элементов. Также нельзя использовать дополнительный массив.

Необходимо отделить ввод массива. Необходимо отделить вывод массива.

| Ввод | Вывод |
|---|---|
| 7<br>7 6 5 4 3 2 1<br>2 | 7 6 4 3 2<br>1 |

# J: Вставить элемент

Дан массив из N чисел, число k и значение C. Необходимо вставить в массив на позицию с индексом k элемент, равный C, сдвинув все элементы имевшие индекс не менее k вправо.

Поскольку при этом количество элементов в массиве увеличивается, необходимо сразу же создавать массив размером N+1.

Вставку необходимо осуществлять уже в считанном массиве, не оставляя “свободного” места при считывании, не делая этого при выводе и не создавая дополнительного массива.

Необходимо отделить вывод массива.

| Ввод | Вывод |
|---|---|
| 7<br>7 6 5 4 3 2 1<br>2 0 | 7 6 0 5 4 3 2 1 |

# K: Уникальные элементы

Дан массив. Выведите те его элементы, которые встречаются в массиве только один раз. Элементы нужно выводить в том порядке, в котором они встречаются в массиве.

Необходимо отделить ввод массива.

| Ввод | Вывод |
|---|---|
| 6<br>1 2 3 3 2 3 | 1 |

# L: Сжатие массива

Массив заполнен целыми числами. Требуется “сжать” его, переместив все ненулевые элементы в левую часть массива, не меняя их порядок, а все нули - в правую часть. Порядок ненулевых элементов изменять нельзя, дополнительный массив использовать нельзя, задачу нужно выполнить за один проход по массиву. Распечатайте полученный массив.

Необходимо отделить ввод массива. Необходимо отделить вывод массива.

| Ввод | Вывод |
|---|---|

| | |
|---|---|
| 8<br>4 0 5 0 3 0 0 5 | 4 5 3 5 0 0 0 0 |

## M: Ферзи

Известно, что на доске 8×8 можно расставить 8 ферзей так, чтобы они не били друг друга. Вам дана расстановка 8 ферзей на доске, определите, есть ли среди них пара бьющих друг друга.

Программа получает на вход восемь пар чисел, каждое число от 1 до 8 - координаты 8 ферзей. Если ферзи не бьют друг друга, выведите слово NO, иначе выведите YES.

| Ввод | Вывод |
|---|---|
| 1 7<br>2 4<br>3 2<br>4 8<br>5 6<br>6 1<br>7 3<br>8 5 | NO |
| 1 8<br>2 7<br>3 6<br>4 5<br>5 4<br>6 3<br>7 2<br>8 1 | YES |

# Символы и строки

## Символьный тип char

До сих пор основным объектом наших преобразований были числа. Именно числа, как мы уже знаем, хранятся в памяти компьютера в двоичном виде. Но информация, которая нас окружает, может быть представлена и во многих других видах: текстовом, графическом, звуковом. Давайте подумаем, как нам лучше обращаться с текстовой информацией при наличии устройства, работающего только с числами. Вывод очевиден: мы можем занумеровать буквы!

Любой текст состоит из символов. Символ - это некоторый значок, изображение. Вместо символов хранятся их номера - числовые коды.

Очевидно, что стоит договориться о единообразном способе кодирования символов числовыми кодами, иначе текст, записанный на одном компьютере, невозможно будет прочитать на другом компьютере.

Давайте оценим количество различных символов, требуемых нам. В латинском алфавите 26 букв. Заглавные и строчные буквы стоит представлять разными кодами, чтобы отличить их друг от друга. Итого - 52 буквы. Прибавим к ним все расские буквы в двух вариантах — еще 66 символов. Добавим цифры — 10 штук, знаки препинания и особый символ, кодирующий пробел — еще два десятка. Мы прекрасно обходимся не более чем двумя сотнями символами для передачи любой текстовой информации!

Первоначально договорились под кодирование одного символа отвести один байт, то есть 8 бит информации. Таким образом можно было закодировать 256 различных значений, то есть в записи

текста можно использовать 256 различных символов. Стандарт, указывающий, какие числовые коды соответствуют каким основным символам, называется ASCII. В таблицу ASCII включены символы с кодами от 0 до 127, то есть ASCII - это семибитный код. Русские буквы, а также буквы других алфавитов, которые не укладываются в рамки латиницы, кодируются отдельными стандартами. Возможно, вы встречали в интернете сайты, на которых для того, чтобы прочесть русский тескт, приходилось выбирать в броузере кодировки Win1251, KOI8-R или UTF-8. Это - последствие того, что единого стандарта кодирования текстовой русскоязычной информации не было достаточно долго и все придумывали свои собственные форматы.

В языке C++ для хранения однобайтового символа используется тип данных `char`. Переменную типа `char` можно рассматривать двояко: как целое число, занимающее 1 байт и способное принимать значения от -128 до 127 и как один символ текста.

Как и целые числа, данные типа `char` можно складывать, вычитать, умножать и даже делить. Но если операции умножения и деления, как правило, бессмысленны, то сложение и вычитание вполне осмысленно. Например, если к символу `'A'` прибавить 1, то получится символ `'B'`, а если вычесть 1, то получится символ `'@'`. То есть в следующем фрагменте кода на экран будет выведена буква `B`.

```
char c = 'A';
c = c + 1;
cout << c << endl;
```

В этом примере видно, что переменным типа `char` можно присваивать значения, равные ASCII кодам символов, если эти символы заключать в кавычки. То есть запись `'A'` будет соответствовать символу `A`, или ASCII коду 65.

Аналогично, при считывании переменной типа `char` через поток `cout`, из потока ввода считывается один символ, переменная получает значение, равное его ASCII-коду. Например, если написать программу, содержащую строчку

```
char c;
cin >> c;
```

запустить ее, ввести символ `A` (безо всяких кавычек!), то в переменную `c` будет записано значение 65 - ASCII-код символа `A`.

Эта программа выведет две строки: “`A 65`” и “`~ 126`”, то есть символы с ASCII-кодами 65 (A) и 126 (~) и сами ASCII-коды.

# Строки в языке C++

Текстовая строка - это последовательность символов. Поскольку символы в строке пронумерованы, то естественным представлением для строки был бы массив символов. Так строки и представлялись в языке C - строкой считался массив символов, а для обозначения конца строки использовался символ с ASCII-кодом 0, что позволяло хранить строки переменной длины (то есть в массиве `char[n]` можно было хранить строки любой длины, не превосходящей `n-1`. Такой способ хранения строк порождал ряд неудобств: любая строка была ограничена по длине размером массива, а чтобы вычислить длину строки необходимо было пройти по всей строке до появления нулевого символа, то есть определение длины строки требует количество операций, пропорциональное этой длине.

В языке C++ для представления строк существует более совершенный тип данных `string`, в основе которого лежит такой же массив символов, завершающийся нулевым символом, но содержащий еще ряд дополнительных возможностей.

Для работы со строками языка C++ необходимо в начале программы подключить описание типа `string`, которое находится в одноименном файле:

```
#include <string>
```

Переменная для хранения строковых данных объявляется так:

```
string S;
```

Присвоить строковой переменной некоторое константное значение можно так:

```
S = "Hello, world!";
```

С записью строк в тексте программы в кавычках мы уже встречались, когда выводили текст в поток `cout`. Обратите внимание - константы типа `char` записываются в одинарных кавычках, а строки - в двойных кавычках. В частности, `'A'` - это символ, а `"A"` - это строка, состоящая из одного символа. Поэтому переменной типа `char` нельзя присвоить значение `"A"`, поскольку они имеют несовместимые типы данных.

Итак, строка является массивом символов и с каждым символом этой строки можно работать по-отдельности, обращаясь к ним по индексу, как к элементам массива. Например:

char S[12] = “Hello, World”;

```
cout << S[0]; // Будет выведен символ H
S[0] = 'h'; // Изменили первый символ в строке
cout << S; // Будет выведено hello, world!
```

Для определения длины строки есть метод `size()`, применяемый к строке. Он возвращает целое число - количество символов в строке. Его можно использовать так:

```
string S;
cin >> S;
cout << "Вы ввели слово " << S << " в котором " << S.size() << " символов" << endl;
```

Основная операция над строками - сложение: например, при сложении строк `"Hello, "` и `"world!"` получится строка `"Hello, world!"`. Такая операция над строками называется *конкатенацией*.

Вот пример использования конкатенации строк:

```
string S, S1, S2;            // Объявление трех строк
cout << "Who are you? ";
cin >> S1;                   // Считали строку S1
S2 = "Hello, "               // Присвоили строке значение
S = S2 + S1;                 // Использование конкатенации
cout << S << endl;           // Вывод строки на экран
cout << S.size();          // Длина строки S
```

При считывании строк из входного потока считываются все символы, кроме символов–разделителей (пробелов, табуляций и новых строк), которые являются границами между строками. Например, если при выполнении следующей программы

```
string S1, S2, S3; // объявили 3 строки
cin>>S1>>S2>>S3;
```

ввести текст ‘`Мама мыла раму`’ (с произвольным количеством пробелов между словами), то в массив `S1` будет записана строка `"Мама"`, в `S2` — `"мыла"`, в `S3` — `"раму"`.

Таким образом, организовать считывание всего файла по словам, можно следующим образом:

```
string s;
while (cin >> s) // Цикл пока считывание успешно
{             // Делаем необходимые действия
}
```

Если нужно считать строку со всеми пробелами, то необходимо использовать функцию `getline` следующим образом:

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

В данном случае если запустить эту программу и ввести строку `"Мама мыла раму"`, то именно это значение и будет присвоено строке `S`. Считать же весь входной поток по строкам можно при помощи следующего кода:

```
string s;
while (getline(cin, S)) // Цикл пока считывание успешно
{                        // Делаем необходимые действия
}
```

# N: ASCII-код символа

Считайте со стандартного ввода символ и выведите его ASCII-код.

Программа получает на вход один символ с ASCII кодом от 33 до 126.

| Ввод | Вывод |
|---|---|
| A | 65 |

# O: IsDigit

Для данного символа, считанного со стандартного ввода, проверьте, является ли он цифрой. Программа должна вывести слово YES, если символ является цифрой, или слово NO.

Решение оформите в виде функции `bool IsDigit(char c)`. В решении нельзя использовать циклы. В решении нельзя использовать константы с неочевидным значением типа 48 или 57.

| Ввод | Вывод |
|---|---|
| 0 | YES |
| A | NO |

# P: ToUpper

Напишите функцию `char ToUpper(char c)`, которая переводит символ в верхний регистр, то есть для строчной буквы латинского алфавита возвращает соответствующую заглавную букву латинского алфавита, а для остальных символов возвращает тот же символ.

Считайте один символ со стандартного ввода и переведите его в верхний регистр. В решении нельзя использовать циклы. В решении нельзя использовать константы с неочевидным значением.

| Ввод | Вывод |
|---|---|
| f | F |
| F | F |
| 4 | 4 |

# Q: Сменить регистр символа

Напишите функцию `char CaseChange (char c)`, меняющую регистр символа, то есть переводящую заглавные буквы в строчные, а строчные - в заглавные, остальные символы не меняющие.

Считайте один символ со стандартного ввода, выведите результат работы данной функции. В решении нельзя использовать циклы. В решении нельзя использовать константы с неочевидным значением.

| Ввод | Вывод |
|---|---|

| | |
|---|---|
| f | F |
| F | f |
| 4 | 4 |

# R: Проверить строки на равенство

Даны две строки (возможно, с пробелами). Проверьте, равны ли они. Если строки равны, выведите слово YES, если строки не равны, выведите слово NO.

Решение оформите в виде функции `bool IsEqual( string S1, string  S2)`.

После того, как вы решите эту задачу, вам разрешается использовать оператор `==` для сравнения строк.

| **Ввод** | **Вывод** |
|---|---|
| Hi<br>HI | NO |
| Bye<br>Bye | YES |

# S: Палиндром

Дано слово, состоящее только из заглавных и строчных латинских букв. Проверьте, верно ли что это слово читается одинаково как справа налево, так и слева направо (то есть является палиндромом), если считать заглавные и строчные буквы не различающимися. Выведите слово YES, если слово является палиндромом и словов NO, если не является.

Решение оформите в виде функции `bool IsPalindrome (string  S)`. При решении этой задачи нельзя пользоваться вспомогательными массивами или строками.

| **Ввод** | **Вывод** |
|---|---|
| Radar | YES |
| YES | NO |

# T: Извлечь цифры

Дана строка, возможно, содержащая пробелы. Извлеките из этой строки все символы, являющиеся цифрами и составьте из них новую строку. Решение оформите в виде функции `string ExtractDigits ( string  S)`, получающей на вход исходную строку `S` и возвращающую новую строку, содержащую только цифры данной строки.

Указание. Заведите строку `Answer`, пройдите по всем символам данной строки, при обнаружении цифры добавляйте ее в конец строки `Answer`, увеличивая ее размер на 1. По завершении цикла верните значение `Answer`.

| **Ввод** | **Вывод** |
|---|---|
| 2+2=4 | 224 |

# U: Значение выражения - 1

Дана строка, состоящая из n цифр, между которыми стоит n-1 знак операции, каждый из которых может быть либо +, либо -. Вычислите значение данного выражения.

Решение оформите в виде функции `int Evaluate( string S)`.

| Ввод | Вывод |
|---|---|
| 1+2-3 | 0 |

# V: StrToInt

Дана строка, содержащее запись в виде символов целого числа от 0 до $10^9$-1. Определите значение этого числа в виде переменной int. Решение задачи оформите в виде функции `int StrToInt ( string S)`.

Функция `main` должна быть такой:

```
int main()
{
    string S;
    cin >> S;
    cout << StrToInt(S) << endl;
    return 0;
}
```

| Ввод | Вывод |
|---|---|
| 661 | 661 |

# W: IntToStr

Дана целое число от -$10^9$+1 до $10^9$-1. Запишите это число в строку, то есть выполните преобразование, обратное предыдущей задаче (но только допускаются отрицательные числа).

Решение задачи оформите в виде функции `string IntToStr (int n)`.

Функция `main` должна быть такой:

```
int main()
{
    int n;
    cin >> n;
    cout << IntToStr(n) << endl;
    return 0;
}
```

| Ввод | Вывод |
|---|---|
| -661 | -661 |

# X: Количество слов

Дана строка, возможно, содержащая пробелы. Определите количество слов в этой строке. Слово - это несколько подряд идущих букв (как заглавных, так и строчных).

Решение оформите в виде функции `int CountWords ( string S)`. При решении этой задачи нельзя пользоваться дополнительными строками и массивами.

| Ввод | Вывод |
|---|---|

| | |
|---|---|
| `Yesterday, all my troubles seemed so far away` | `8` |

# Y: Слова с прописной буквы

Дана строка. Измените регистр символов в этой строке так, чтобы первая буква каждого слова была заглавной, а остальные буквы - строчными.

Решение оформите в виде функции `void Capitalization (string S)`.

| **Ввод** | **Вывод** |
|---|---|
| `In a hole in the ground there lived a hobbit.` | `In A Hole In The Ground There Lived A Hobbit.` |

# Z: Шифр Цезаря

В шифре Цезаря каждый символ заменяется на другой символ, третий по счету в алфавите после данного, с цикличность. То есть символ A заменяется на D, символ B - на E, символ C - на F, ..., символ Z на C. Аналогично строчные буквы заменяются на строчные буквы. Все остальные символы не меняются.

Дана строка, зашифруйте ее при помощи шифра Цезаря. Решение оформите в виде функции `void CaesarCipher (string S)`.

Указание: сделайте функцию `char CaesarCipher (char c)`, шифрующую один данный символ.

| **Ввод** | **Вывод** |
|---|---|
| `In a hole in the ground there lived a hobbit.` | `Lq d kroh lq wkh jurxqg wkhuh olyhg d kreelw.` |