Функции (часть 1)

В предыдущем листке была задача вычисления числа сочетаний из n элементов по k, для чего необходимо вычисление факториалов трех величин: n, k и n-k. Для этого можно сделать три цикла, что приводит к увеличению размера программы за счет трехкратного повторения похожего кода. Вместо этого лучше сделать одну *функцию*, вычисляющую факториал любого данного числа n и трижды использовать эту функцию в своей программе. Соответствующая функция может выглядеть так:

```
int factorial (int n)
{
    int f=1,i;
    for(i=2;i<=n;++i)
    {
        f=f*i;
    }
    return f;
}
```

Этот текст должен идти до основной программы, то есть до функции `main()`. Первая строчка этого примера является описанием нашей функции.`factorial` – идентификатор, то есть имя нашей функции. После идентификатора в круглых скобках идет список параметров, которые получает наша функция. Список состоит из перечисленных через запятую типов параметров и их идентификаторов. В нашем случае список состоит из одной величины`n`, имеющей тип `int`: наша функция вычисляет факториал целого числа. Функция вычисляет целочисленную величину, поэтому функция будет возвращать значение типа `int`, что указывается до идентификатора функции. Функция может не возвращать никакого значения, тогда в качестве типа возвращаемого значения должно быть указано слово `void`.

Далее идет тело функции в фигурных скобках. Внутри функции вычисляется значение факториала числа `n` и оно сохраняется в переменной `f`. Функция завершается инструкцией `return f`, которая завершает работу функции и возвращает значение переменной `f`. Инструкция `return` может встречаться в произвольном месте функции, ее исполнение завершает работу функции и возвращает указанное значение в место вызова. Если функция не возвращает значения, то инструкция `return` используется без возвращаемого значения, также в функциях, не возвращающих значения, инструкция `return` может отсутствовать.

Функция должна быть описана до начала основной программы. Сама же основная программа, как можно догадаться, также является функцией с именем`main`, не получающая никаких параметров и возвращающее значение типа `int`.

Теперь мы можем использовать нашу функцию `factorial` в основной программе нахождения числа сочетаний:

```
int main ()
{
    int n,k;
    cin>>n>>k;
    cout<<factorial(n)/(factorial(k)*factorial(n-k))<<endl;
    return 0;
}
```

В этом примере мы трижды вызываем функцию `factorial` для вычисления трех факториалов: `factorial(n)`, `factorial(k)`, `factorial(n-k)`.

Мы также можем объявить функцию `binomial`, которая принимает два целочисленных параметра `n` и `k` и вычисляет число сочетаний из `n` по `k`:

```
int binomial (int n, int k)
{
    return factorial(n)/(factorial(k)*factorial(n-k));
}
```

Тогда в нашей основной программе мы можем вызвать функцию `binomial` для нахождения числа сочетаний:

```
    cout<<binomial(n,k)<<endl;
```

Поскольку в этом случае функция `main` вызывает функцию `binomial`, а функция `binomial` вызывает функцию `factorial`, а каждая функция должна быть описана до ее вызова из другой функции, то порядок описания функций в программе должен быть такой:

```
int factorial (int n)

int binomial (int n, int k)

int main ()
```

Вернемся к задаче нахождения наибольшего из двух или трех чисел. Напишем функцию, находящую максимум из двух данных чисел:

```
double max (double a, double b)
{
    if (a>b)
        return a;
    else
        return b;
}
```

Теперь мы можем реализовать функцию `max`, находящую максимум трех чисел:

```
double max (double a, double b, double c)
{
    return max( max(a,b), c);
}
```

В данном примере написаны две различные функции `max`: первая с двумя параметрами, вторая с тремя параметрами. Несмотря на то, что функции имеют одинаковые имена, по количеству передаваемых параметров ясно, какая из них имеется в виду. В нашем случае функция `max (double a, double b, double c)` дважды вызывает функцию `max` для двух чисел: сначала, чтобы найти максимум из `a` и `b`, потом чтобы найти максимум из этой величины и `c`.

Упражнения

1. (A) Напишите функцию `int min (int a, int b, int c, int d)`, находящее наименьшее из четырех данных чисел. Функция `main` дожна считывать четыре числа с клавиатуры, вызывать функцию `min`, выводить результат ее работы на экран.
2. (B) Напишите функцию `double power (double a, int n)`, вычисляющую значение a^n . Функция `main` должна считывать числа `a` и `n`, вызывать функцию `power`, выводить результат ее работы на экран.
3. (C) Напишите функцию `bool Xor (bool x, bool y)`, реализующую функцию "Исключающее ИЛИ" двух логических переменных x и y. Функция `Xor` должна возвращать `true`, если ровно один из ее аргументов `x` или `y`, но не оба одновременно равны `true`. Функция `main` в программе должна запрашивать значения переменных `x` и `y`, (два числа, равных 0 или 1), вызывать функцию `Xor(x,y)` и выводить результат на экран.
4. (D) Напишите "функцию голосования" `bool Election(bool x, bool y, bool z)`, которая возвращает то значение (`true` или `false`), которое среди значений ее аргументов `x`, `y`, `z` встречается чаще. Функция `main` должна запрашивать три числа, равных 0 или 1, вызывать функцию `Election` и выводить результат на экран.
5. (E) Напишите функцию `bool IsPrime (int n)`, возвращающую `true`, если натуральное число n>1 простое, и `false`, если составное.

   Функция `main` должна запрашивать число с клавиатуры, вызывать функцию `IsPime`, выводить строку `prime`, если число простое или`composite`, если число составное.

   Указание: число является составным, если оно имеет натуральный делитель, отличный от 1 до n. Программа должна проверить делимость числа n на все числа от 2 до n-1 и вернуть `false` при

нахождении нетривиального делителя. Для того, чтобы проверить, что число n делится на число d нацело, необходимо сравнить остаток от деления n на d с нулем.

Рекурсия

Эпиграф:

```
void ShortStory()
{
    cout<<"У попа была собака, он ее любил."<<endl;
    cout<<"Она съела кусок мяса, он ее убил,"<<endl;
    cout<<"В землю закопал и надпись написал:"<<endl;
    ShortStory();
}
```

Как мы видели выше, функция может вызывать другую функцию. Но функция также может вызывать и саму себя! Рассмотрим это на примере функции вычисления факториала. Хорошо известно, что 0!=1, 1!=1. А как вычислить величину n! для большого n? Если бы мы могли вычислить величину (n-1)!, то тогда мы легко вычислим n!, поскольку n!=n(n-1)!. Но как вычислить (n-1)! ? Если бы мы вычислили (n-2)!, то мы сможем вычисли и (n-1)!=(n-1)(n-2)!. А как вычислить (n-2)! ? Если бы... В конце концов, мы дойдем до величины 0!, которая равна 1. Таким образом, для вычисления факториала мы можем использовать значение факториала для меньшего числа. Это можно сделать и в программе на C++:

```
int factorial (int n)
{
    if (n==0)
        return 1;
    else
        return n*factorial(n-1);
}
```

Подобный прием (вызов функцией самой себя) называется рекурсией, а сама функция называется рекурсивной.

Рекурсивные функции являются мощным механизмом в программировании. К сожалению, они не всегда эффективны (об этом речь пойдет позже). Также часто использование рекурсии приводит к ошибкам, наиболее распространенная из таких ошибок – бесконечная рекурсия, когда цепочка вызовов функций никогда не завершается и продолжается, пока не кончится свободная память в компьютере. Пример бесконечной рекурсии приведен в эпиграфе к этому разделу. Две наиболее распространенные причины для бесконечной рекурсии:

1. Неправильное оформление выхода из рекурсии. Например, если мы в программе вычисления факториала забудем поставить проверку `if (n==0)`, то `factorial(0)` вызовет `factorial(-1)`, тот вызовет `factorial(-2)` и т.д.
2. Рекурсивный вызов с неправильными параметрами. Например, если функция `factorial(n)` будет вызывать `factorial(n)`, то также получиться бесконечная цепочка.

Поэтому при разработке рекурсивной функции необходимо прежде всего оформлять условия завершения рекурсии и думать, почему рекурсия когда-либо завершит работу.

Упражнения

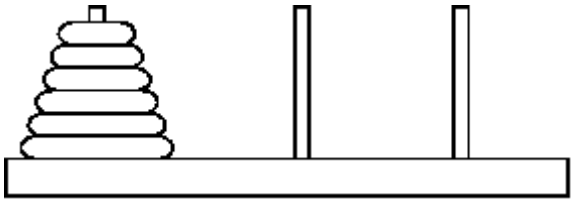
6. (B) Напишите рекурсивную функцию возведения в степень, пользующуюся следующим свойством: $a^n=a*a^{n-1}$.
7. (F) Напишите функцию возведения в степень, которая работала бы как для положительных, так и для отрицательных значений n: $a^{-n}=1/a^n$.
8. (G) Напишите функцию быстрого возведения в степень, которая пользовалась бы следующими свойствами: $a^n=(a^{n/2})^2$ при четном n, $a^n=a*a^{n-1}$ при нечетном n. Подумайте, сколько умножений выполнит эта функция для вычисления a^n?
9. (H) Последовательность Фибоначчи определена следующим образом: $\varphi_0=1$, $\varphi_1=1$, $\varphi_n=\varphi_{n-1}+\varphi_{n-2}$

при n>1. Начало ряда Фибоначчи выглядит следующим образом: 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, ... Напишите функцию `int phi(int n)`, которая по данному натуральному n возвращает φ_n. Функция `n` должна считывать значение n и выводить значение n-го числа Фибоначчи.

10. (I) Для биномиальных коэффициентов (числа сочетаний из n по k) хорошо известна рекуррентная формула: $C_n^k=C_{n-1}^{k-1}+C_{n-1}^k$. Вычислите значение C_n^k, пользуясь этой формулой и учитывая, что $C_n^0=C_n^n=1$.
11. (J) Головоломка "Ханойские башни" состоит из трех колышков, пронумерованных числами 1, 2, 3. На колышек 1 надета пирамидка из n дисков различного диаметра в порядке возрастания диаметра. Диски можно перекладывать с одного колышка на другой по одному, при этом диск нельзя класть на диск меньшего диаметра. Необходимо переложить всю пирамидку с колышка 1 на колышек 2 за минимальное число перекладываний.

    Напишите программу, которая решает головоломку – для данного числа дисков n печатает последовательность перекладываний в формате `"Disk 1 move from 1 to 2"` (диск 1 переложить с колышка 1 на колышек 2), печатая по одной инструкции в строке. Диски пронумерованы числами от 1 до n в порядке возрастания диаметров.

    Программа должна вывести минимальный (по количеству произведенных операций) способ перекладывания пирамидки.

    Указание: подумайте, как переложить пирамидку из одного диска? Из двух дисков? Из трех дисков? Из четырех дисков? Напишите функцию `void move (int n, int x, int y)`, которая перекладывает пирамидку высоты `n` с колышка номер `x` на колышек номер `y`.

Функции (часть 2)

Вся программа в C++ состоит из отдельных блоков — функций. Даже основная программа на самом деле является функцией с именем `main`оговоренного стандартом вида. Синтаксис определения функции в языке C++ такой:

```
возвращаемый_типидентификатор(список_параметров)
{
    Блок инструкций процедуры, включающий инструкцию
    return значение;
}
```

возвращаемый_тип — это тип данных, который возвращает функция. Может быть любым стандартным типом данных C++ (`int`, `double`, `int *` и т.д.) или типом данных, определенным пользователем.

идентификатор — имя функции, как и любой идентификатор в C++ должен состоять из букв, цифр и символа подчеркивания и начинаться не с цифры.

список_параметров — перечисление всех параметров, передаваемых функции. Имеет вид последовательности выражений вида *тип_переменнойидентификатор*, разделенных запятыми. Если функция не принимает ни одного параметра, то список будет пустым.

Наконец, тело функции может содержать любые инструкции, среди которых должна быть хотя бы одна инструкция `return`, которая завершает выполнение процедуры и возвращает указанное значение. Инструкций `return` может быть много, каждая из них немедленно прерывает выполнение функции.

Пример функции, которая по данному действительному числу x и данному натуральному числу n вычисляет x^n:

```
double power(double x, int n) // Функция принимает 2 аргумента: double и int
{                             // и возвращает значение типа double
    double p=1;
    int i;
    for(i=0;i<n;++i)          // Цикл будет пройден n раз
        p*=x;
    return p;                 // Теперь возвращаем значение p
}
```

В основной программе (из функции `main`) или из другой функции мы можем вызвать реализованную функцию, указав ее идентификатор и перечислив в круглых скобках список передаваемых параметров таких же типов, как это указано в объявлении функции:

```
cout<<power(1.5,10)<<endl;   // Вызов функции power с параметрами 1.5 и 10
```

Заметим, что в этом примере мы на занимаемся обработкой ошибок: наша функция будет работать некорректно, если ей передать отрицательное значение в качестве второго параметра. Технология обработки некорректных параметров и сообщения о некорректности входных параметров — отдельная и весьма обширная тема, а лучше всего разрабатывать программы так, чтобы функция получала только корректные параметры.

Возвращаемое функцией значение можно вывести на экран, присвоить другой переменной, использовать в арифметическом выражении, а можно и ничего с ним не делать: вызов функции в виде

```
power(1.5,10);
```

является синтаксически правильным, но бессмысленным (как, например, инструкция `2+2;`, вычисляющая значение выражения, но не использующая результат).

Функция может и не возвращать значения, тогда в ее объявлении в качестве типа возвращаемого значения следует указать слово `void`. Функция, не возвращающая значения, должна завершаться инструкцией `return;` без указания значения (а может и вообще не содержать инструкции `return;`). Такие функции (они являются аналогами процедур в Паскале и Бейсике) нельзя использовать в арифметических выражениях. Пример функции, печатающей на экране строку `Hello, world!' и не возвращающей значения:

```
void hello() // Функция не возвращает значения
{            // и не принимает ни одного параметра
    cout<<"Hello, world!"<<endl;
    return;
}
```

и ее вызова (круглые скобки обязательны, даже если передается пустой список параметров)

```
hello();
```

Наконец, заметим, что могут существовать функции, имеющие одинаковые имена, но отличающиеся типами передаваемых параметров. Например, можно сделать две функции для возведения в степень чисел типа `int` и типа `double`:

```
int power(int x, int n);
double power(double x, double n);
```

При этом первая функция будет работать быстрее, поскольку операции с целочисленными данными выполняются быстрее, чем операции над числами с плавающей точкой, поэтому для целых чисел пользоваться первой функцией предпочтительнее.

Упражнения

Во всех этих задачах необходимо написать программу, содержащую реализацию указанной в задании функции, а также демонстрирующую работу этой функции.

1. Напишите функцию `bool IsUpper(char)`, которая определяет, является ли входной символ заглавной буквой латинского алфавита.
2. Напишите функцию `bool IsDigit(char)`, которая определяет, является ли входной символ цифрой.
3. Напишите функцию `char ToUpper(char)`, которая переводит строчный символ латинского алфавита в аналогичный заглавный.
4. Напишите функции `string ToUpper(string)` и `string ToLower(string)`, которая переводит строку из нижнего регистра в верхний и наоборот. Символы, не являющиеся латинскими буквами не меняются.
5. В шифре Цезаря каждая буква заменяется на третью по счету букву латинского алфавита с циклическим сдвигом (то есть A меняется на D, B на E, ..., Z на C). Напишите функции `string CaesarCrypt(string)` и `string CaesarDecrypt(string)`, осуществляющие шифрование и дешифрование строки при помощи шифра Цезаря.
6. Напишите функцию, шифрующую строку по следующему принципу: строка разбивается на пары символов, два символа в паре переставляются местами (то есть строка `"abcdef"` будет зашифрована, как `"badcfe"`).

Передача параметров функции

Локальные и глобальные переменные

Внутри функции могут быть объявлены переменные, как, например, переменные `p` и `i` в функции `power`. Такие переменные называются *локальными* и они определены только внутри этой функции. Переменные, определенные вне тела любой функции (ранее мы такие переменные вообще не рассматривали) называются *глобальными* и они определены всюду начиная с места определения и до конца файла. Глобальными переменными можно пользоваться во всех функциях, но с современной точки зрения использовать глобальные переменные рекомендуется как можно реже.

Локальные переменные создаются каждый раз при входе в функцию и уничтожаются при выходе из нее. Таким образом, значения, сохраненные в локальных переменных, пропадут после завершения работы функции.

В различных функциях можно определять переменные с одним и тем же именем. Это будут различные переменные. Также можно объявлять в функциях переменные, имена которых совпадают с именами глобальных переменных. Тогда будет создана новая локальная переменная, а изменить значение глобальной переменной с таким именем будет невозможно. Пример:

```
#include<iostream>
using namespace std;
int i;                  // i - глобальная переменная
void f1() {
    int i;              // Определена локальная переменная i
```

```
    i=5;              // Изменяется значение локальной переменной
    cout<<i<<endl;    // Будет напечатано 5
}
 void f2() {
    i=3;              // Изменяется значение глобальной переменной
}
int main(){
    i=1;              // Изменяется значение глобальной переменной
    f1();             // f1() не меняет значение глобальной переменной
    cout<<i<<endl;    // Будет напечатано 1
    f2();             // f2() меняет значение глобальной переменной
    cout<<i<<endl;    // Будет напечатано 3
    return 0;      }
```

Вопрос: как будет работать программа, если добавить в функцию `main` определение переменной `i`, как локальной? А если при этом убрать объявление глобальной переменной i?

Передача параметров по значению и по ссылке

Переменные, в которых сохраняются параметры, передаваемые функции, также являются локальными для этой функции. Эти переменные создаются при вызове функции и в них копируются значения, передаваемые функции в качестве параметров. Эти переменные можно изменять, но все изменения этих переменных будут "забыты" после выхода из функции. Рассмотрим это на примере следующей функции, "меняющей" значения двух переданных ей переменных:

```
#include<iostream>
using namespace std;
void swap(int a, int b)
{
    int t;
    t=b;
    b=a;
    a=t;
}
int main()
{
    int p=3,q=5;
    swap(p,q);
    cout<<p<<" "<<q<<endl;
    return 0;
}
```

При вызове функции `swap` создаются новые переменные `a` и `b`, им присваиваются значения 3 и 5. Эти переменные никак не связаны с переменными `p` и`q` и их изменение не изменяет значения `p` и `q`. Такой способ передачи параметров называется *передачей параметров по значению*.

Чтобы функция могла изменять значения переменных, объявленных в других функциях, необходимо указать, что передаваемый параметр является не просто константной величиной, а переменной, необходимо передавать значения *по ссылке*. Для этого функцию `swap` следовало бы объявить следующим образом:

```
void swap(int & a, int & b)
```

Амперсанды перед именем переменной означают, что эта переменная является не локальной переменной, а ссылкой на переменную, указанную в качестве параметра при вызове функции. Теперь при вызове `swap(p,q)` переменные `a` и `b` являются синонимами для переменных `p` и `q`, и изменение их значений влечет изменение значений `p` и `q`. А вот вызывать функцию в виде `swap(3,5)` уже нельзя, поскольку `3` и `5` — это константы, и сделать переменные синонимами констант нельзя.

Однако в языке C (не C++) вообще не было такого понятия, как передача параметров по ссылке. Для того, чтобы реализовать функцию, аналогичную`swap` в рассмотренном примере, необходимо было передавать адреса переменных `p` и `q`, а сама функция при этом должна быть объявлена, как

```
void swap(int * a, int * b)
```

Поскольку в этом случае функция `swap` знает физические адреса в оперативной памяти переменных `p` и `q`, то разыменовав эти адреса функция `swap`сможет изменить значения самих переменных `p` и `q`.

Теперь вспомним, что в C++ массивы и указатели — это почти одно и то же. А поскольку передача массива по значению, то есть создание копии всего массива является трудоемкой операцией (особенно для больших массивов), то вместо массива всегда передается указатель на его начало. То есть два объявления функции `void f(int A[10])` и `void f(int * A)` абсолютно идентичны. В любом случае функция `f` получит указатель на начало массива, а, значит, будет способна изменять значения его элементов.

Упражнения

Функции — удобный способ структурирования программы. Существует мнение, что если фрагмент программы реализует некоторую законченную алгоритмическую идею и состоит более чем из 10 строк, то его обязательно надо оформить в виде отдельной функции.

Упражнения к этому листку объединены темой генерирования комбинаторных объектов (перестановок, подмножеств и т.д.) Такие объекты следует хранить в массивах, например, перестановка чисел от 1 до n — это массив типа `int[n]`, заполненный числами от 1 до `n`.

В программе должны быть следующие функции (`p` — массив для хранения комбинаторного объекта, `n` — число элементов в нем):

```
int * Init (int n);          // Создание и инициализация массива
bool  Next (int * p, int n); // Построение следующего комбинаторного объекта
void  Print(int * p, int n); // Печать комбинаторного объекта
void  Done (int * p);        // Освобождение памяти
```

а функция `main` должна иметь следующую структуру:

```
cin>>n;
p=Init(n);
Print(p,n);
while( Next(p,n) )
    Print(p,n);
Done(p);
```

Кроме того, программа должны работать за оптимальное время.

1. По данным числам n и k выведите на экран все строки длины n, состоящие из чисел 1, ..., k в лексикографическом (алфавитном) порядке.
2. По данному n выведите на экран все подмножества множества {1, ...,n}.
3. По данным n и k выведите на экран все двоичные строки длины n, содержащие ровно k единиц в лексикографическом порядке.
4. По данному n напечатайте все перестановки чисел от 1 до n в лексикографическом порядке.
5. По данному n напечатайте все способы размещения из n элементов k элементов (количество способов выбрать из чисел от 1 до n ровно k чисел с учетом порядка выбора).
6. По данному числу n напечатайте всевозможные его представления в виде суммы натуральных слагаемых. Представления, различающиеся порядком слагаемых, считать за одно.
7. По данному слову (или последовательности цифр) напечатайте всевозможные перестановки его букв. Учтите, что буквы исходного слова могут совпадать.

Задача №306. Минимум 4 чисел

Данные вводятся с клавиатуры или из файла input.txt, выводятся на экран или в файл output.txt. Первые тесты не всегда совпадают с примерами из условия.

:: Результаты :: Вопросы :: Посылки :: Разбор :: Темы :: Лучшие решения :: Источники

		Язык	Free Pascal	GNU C	GNU C++	Delphi	Java	Python 2.7	Mono C#	Ruby	Python 3.1
Ограничение по времени, *сек*	1	**Min время, *сек***	0.001	0.001	0.001	0.002	0.098	0.006	0.044	0.01	0.046
Ограничение по памяти, *мегабайт*	64	**Среднее время, *сек***	0.007	0.008	0.009	0.01	0.243	0.111	0.063	0.027	0.067
		Верных решений	1281	103	1353	487	97	13	66	2	32

Напишите функцию `int min (int a, int b, int c, int d)` (C/C++), `static int min (int a, int b, int c, int d)` (Java)`function min (a,b,c,d: integer):integer` (Pascal), находящую наименьшее из четырех данных чисел.

Входные данные

Вводится четыре числа.

Выходные данные

Необходимо вывести наименьшее из 4-х данных чисел.

Примеры

входные данные

```
4 5 6 7
```

выходные данные

```
4
```

Задача №307. Степень

Данные вводятся с клавиатуры или из файла input.txt, выводятся на экран или в файл output.txt. Первые тесты не всегда совпадают с примерами из условия.

:: Результаты :: Вопросы :: Посылки :: Разбор :: Темы :: Лучшие решения :: Источники

		Язык	Free Pascal	GNU C	GNU C++	Delphi	Java	PHP	Python 2.7	Mono C#	Python 3.1
Ограничение по времени, *сек*	1	**Min время, *сек***	0.001	0.001	0.002	0.002	0.098	0.034	0.007	0.048	0.047
Ограничение по памяти, *мегабайт*	64	**Среднее время, *сек***	0.008	0.013	0.014	0.01	0.257	0.082	0.093	0.083	0.056
		Верных решений	881	66	1090	423	63	8	10	63	15

Напишите функцию `double power (double a, int n)` (C/C++), `function power (a:real; n:longint): real` (Pascal), вычисляющую значение a^n.

Входные данные

Вводится 2 числа - *a* (вещественное) и *n* (целое неотрицательное).

Выходные данные

Необходимо вывести значение a^n.

Примеры

входные данные

```
2 2
```

выходные данные

```
4
```

Задача №308. Исключающее ИЛИ

Данные вводятся с клавиатуры или из файла input.txt, выводятся на экран или в файл output.txt. Первые тесты не всегда совпадают с примерами из условия.

:: Результаты :: Вопросы :: Посылки :: Разбор :: Темы :: Лучшие решения :: Источники

		Язык	Free Pascal	GNU C	GNU C++	Delphi	Java	PHP	Python 2.7	Mono C#	Ruby	Python 3.1
Ограничение по времени, *сек*	1	**Min время, *сек***	0.001	0.001	0.001	0.002	0.098	0.019	0.008	0.036	0.01	0.048
Ограничение по памяти,*мегабайт*	64	**Среднее время,*сек***	0.005	0.011	0.008	0.006	0.272	0.02	0.028	0.067	0.01	0.056
		Верных решений	727	61	971	357	57	4	12	53	1	29

Напишите функцию

`bool Xor (bool x, bool y)` (C/C++),

`function _Xor (x, y:boolean): boolean` (Pascal),

`def xor(x, y):`(Python)

реализующую функцию "Исключающее ИЛИ" двух логических переменных `x` и `y`.

Функция `Xor` должна возвращать `true`, если ровно один из ее аргументов `x` или `y`, но не оба одновременно равны `true`.

Входные данные

Вводится 2 числа - *x* и *y* (*x* и *y* равны `0` или `1`, `0` соответствует значению `false`, `1` соответствует значению `true`).

Выходные данные

Необходимо вывести `0` или `1` - значение функции от *x* и *y*.

Примеры

входные данные

```
0 1
```

выходные данные

```
1
```

Задача №309. Голосование

Данные вводятся с клавиатуры или из файла input.txt, выводятся на экран или в файл output.txt. Первые тесты не всегда совпадают с примерами из условия.

		Язык	Free Pascal	GNU C	GNU C++	Delphi	Java	PHP	Python 2.7	Mono C#	Ruby	Python 3.1
Ограничение по времени, *сек*	1	**Min время, *сек***	0.001	0.001	0.001	0.002	0.099	0.02	0.006	0.036	0.02	0.047
Ограничение по памяти,*мегабайт*	64	**Среднее время,*сек***	0.006	0.011	0.008	0.008	0.2	0.037	0.056	0.074	0.02	0.053
		Верных решений	658	59	914	325	75	7	10	50	1	15

Напишите "функцию голосования" `bool Election(bool x, bool y, bool z)` (C/C++), `function Election (x, y, z:boolean): boolean` (Pascal), возвращающую то значение (`true` или `false`), которое среди значений ее аргументов `x`, `y`, `z` встречается чаще.

Входные данные

Вводится 3 числа - *x*, *y* и *z* (*x*, *y* и *z* равны `0` или `1`, `0` соответствует значению `false`, `1` соответствует значению `true`).

Выходные данные

Необходимо вывести значение функции от *x*, *y* и *z*.

Примеры

входные данные

```
0 0 1
```

выходные данные

```
0
```

Задача №310. Проверка на простоту

Данные вводятся с клавиатуры или из файла input.txt, выводятся на экран или в файл output.txt. Первые тесты не всегда совпадают с примерами из условия.

		Язык	Free Pascal	GNU C	GNU C++	Delphi	Java	PHP	Python 2.7	Perl	Mono C#	Python 3.1	Haskell
Ограничение по времени, *сек*	1	**Min время, *сек***	0.002	0.002	0.002	0.002	0.118	0.03	0.013	0.033	0.044	0.059	0.004
Ограничение по памяти,*мегабайт*	64	**Среднее время,*сек***	0.015	0.009	0.015	0.025	0.288	0.046	0.05	0.033	0.069	0.107	0.004
		Верных решений	963	70	1105	437	82	4	11	1	56	105	1

Проверьте, является ли число простым.

Входные данные

Вводится одно число *n*.

Выходные данные

Необходимо вывести строку `prime`, если число простое, или `composite`, если число составное.

Примеры

входные данные

```
5
```

выходные данные

```
prime
```

Задача №252. Степень для отрицательного показателя

Данные вводятся с клавиатуры или из файла input.txt, выводятся на экран или в файл output.txt. Первые тесты не всегда совпадают с примерами из условия.

:: Результаты :: Вопросы :: Посылки :: Разбор :: Темы :: Лучшие решения :: Источники

		Язык	Free Pascal	GNU C	GNU C++	Delphi	Java	Python 2.7	Mono C#	Python 3.1
Ограничение по времени, *сек*	1	**Min время, *сек***	0.001	0.002	0.002	0.002	0.118	0.028	0.05	0.048
Ограничение по памяти, *мегабайт*	64	**Среднее время, *сек***	0.014	0.044	0.013	0.011	0.289	0.042	0.081	0.093
		Верных решений	374	35	635	223	38	6	45	8

Напишите *рекурсивную функцию*, возводящую число *a* в степень *n*. Гарантируется, что все числа "помещаются" в стандартные вещественные (a и ответ) и целые (n) типы.

Входные данные

Вводится 2 числа - *a* и *n* (число *n* может быть отрицательным).

Выходные данные

Необходимо вывести значение a^n

Примеры

входные данные

```
2 -1
```

выходные данные

```
0.5
```

Задача №311. Быстрое возведение в степень

Данные вводятся с клавиатуры или из файла input.txt, выводятся на экран или в файл output.txt. Первые тесты не всегда совпадают с примерами из условия.

:: Результаты :: Вопросы :: Посылки :: Разбор :: Темы :: Лучшие решения :: Источники

		Язык	Free Pascal	GNU C	GNU C++	Delphi	Java	PHP	Python 2.7	Mono C#	Python 3.1
Ограничение по времени, *сек*	1										

Ограничение по памяти, *мегабайт*	64

	Min время, *сек*	0.001	0.002	0.001	0.002	0.098	0.036	0.006	0.047	0.048
	Среднее время, *сек*	0.011	0.04	0.059	0.014	0.25	0.036	0.081	0.071	0.06
	Верных решений	228	27	587	230	27	1	8	45	25

Напишите функцию быстрого возведения в степень. Количество действий должно быть пропорционально двоичному логарифму *n*.

Входные данные

Вводится 2 числа - *a* (вещественное) и *n* (целое неотрицательное).

Выходные данные

Необходимо вывести значение a^n.

Примеры

входные данные

```
2 2
```

выходные данные

```
4
```

Задача №312. Числа Фибоначчи

Данные вводятся с клавиатуры или из файла input.txt, выводятся на экран или в файл output.txt. Первые тесты не всегда совпадают с примерами из условия.

:: Результаты :: Вопросы :: Посылки :: Разбор :: Темы :: Лучшие решения :: Источники

Ограничение по времени, *сек*	1
Ограничение по памяти,*мегабайт*	64

Язык	Free Pascal	GNU C	GNU C++	Delphi	Java	Python 2.7	Mono C#	Ruby	Python 3.1	Haskell
Min время, *сек*	0.001	0.001	0.001	0.002	0.12	0.01	0.044	0.184	0.047	0.01
Среднее время,*сек*	0.008	0.016	0.01	0.012	0.282	0.056	0.076	0.184	0.066	0.012
Верных решений	386	48	687	259	33	9	44	1	13	2

Последовательность Фибоначчи определена следующим образом: $\varphi_0=1$, $\varphi_1=1$, $\varphi_n=\varphi_{n-1}+\varphi_{n-2}$ при n>1. Начало ряда Фибоначчи выглядит следующим образом: 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, ... Напишите функцию `int phi(int n)` (C/C++),`function phi (n:integer): integer`, (Pascal), которая по данному натуральному *n* возвращает φ_n.

Входные данные

Вводится одно число *n*.

Выходные данные

Необходимо вывести значение φ_n.

Примеры

входные данные

```
3
```

выходные данные

```
3
```

Задача №313. Биномиальные коэффициенты

Данные вводятся с клавиатуры или из файла input.txt, выводятся на экран или в файл output.txt. Первые тесты не всегда совпадают с примерами из условия.

:: Результаты :: Вопросы :: Посылки :: Темы :: Лучшие решения :: Источники

Ограничение по времени, *сек*	1
Ограничение по памяти, *мегабайт*	64

Язык	**Free Pascal**	**GNU C**	**GNU C++**	**Delphi**	**Java**	**Python 2.7**	**Mono C#**	**Ruby**	**Python 3.1**
Min время, *сек*	0.001	0.003	0.001	0.002	0.118	0.1	0.044	0.304	0.05
Среднее время, *сек*	0.011	0.012	0.015	0.017	0.213	1.041	0.075	0.304	0.128
Верных решений	137	27	413	152	16	5	28	1	13

Для биномиальных коэффициентов (числа сочетаний из n по k) хорошо известна рекуррентная формула: $C_n^k=C_{n-1}^{k-1}+C_{n-1}^k$, $C_n^0=C_n^n=1$.

Входные данные

Вводится 2 числа - n и k.

Выходные данные

Необходимо вывести значение C_n^k.

Примеры

входные данные

```
4 2
```

выходные данные

```
6
```