

C++

Урок 6

Лайфхаки у массивов

Хаки массивов

Задача: Пользователь вводит размер массива $n * n$.
Заполните в двумерном массиве эл-ты главной
диагонали значением 1 и выведите результат.

Хаки массивов

Листинг:

```
for (int i = 0; i < n; i++)  
{  
    for (int j = 0; j < n; j++)  
    {  
        if (i == j) //если на главной диагонали  
        {  
            arr[i][j] = 1; //заполняем 1  
        }  
    }  
}
```

Хаки массивов

Листинг:

```
for (int i = 0; i < n; i++)
{
    for (int j = 0; j < n; j++)
    {
        if (i == j) //если на главной диагонали
        {
            arr[i][j] = 1; //заполняем 1
        }
    }
}
```

Можно обойтись **без if**?

Хаки массивов

Листинг:

```
for (int i = 0; i < n; i++)  
{  
    for (int j = 0; j < n; j++)  
    {  
        arr[i][i] = 1; //условие главной диагонали  
    }  
}
```

Да

Хаки массивов

Листинг:

```
for (int i = 0; i < n; i++)  
{  
    for (int j = 0; j < n; j++)  
    {  
        arr[i][i] = 1; //условие главной диагонали  
    }  
}
```

Нужно ли теперь проходить $n * n$ раз?

Хаки массивов

Листинг:

```
for (int i = 0; i < n; i++)  
{  
    for (int j = 0; j < n; j++)  
    {  
        arr[i][i] = 1; //условие главной диагонали  
    }  
}
```

Нужно ли теперь проходить **$n * n$** раз? **НЕТ**

Хаки массивов

Листинг:

```
for (int i = 0; i < n; i++)  
{  
    arr[i][i] = 1; //условие главной диагонали  
}
```

Хаки массивов

Листинг:

```
for (int i = 0; i < n; i++)  
{  
    arr[i][i] = 1; //условие главной диагонали  
}
```

Уменьшили кол-во итераций (сложность) в n раз.

Хаки массивов

Листинг:

```
for (int i = 0; i < n; i++)  
{  
    arr[i][i] = 1; //условие главной диагонали  
}
```

Для **главной** диагонали

Хаки массивов

Листинг:

```
for (int i = 0; i < n; i++)  
{  
    arr[i][n - i - 1] = 1;  
}
```

Для **побочной** диагонали

A hand-drawn blue oval frame with a double-line border, centered on the page. The text 'Разбор ДЗ' is written inside this frame in a bold, black, sans-serif font.

Разбор ДЗ

Разбор дз

4 задача:

Задача: Пользователь вводит размер массива $n * n$ и вводит значения. Требуется найти сумму эл-тов над главной диагональю.

Разбор дз

4 задача:

```
int sum = 0;
for (int i = 0; i < n; i++)
    for (int j = 0; j < n; j++)
        arr[i][j] = (i > j) ? 0 : 1; // заполняем массив

for (int i = 0; i < n; i++)
{
    for (int j = 0; j < n; j++)
    {
        sum += arr[i][i]; //считаем сумму всего
    }
}
```

Разбор дз

4 задача:

```
int sum = 0;
for (int i = 0; i < n; i++)
    for (int j = 0; j < n; j++)
        arr[i][j] = (i > j) ? 0 : 1; // заполняем массив

for (int i = 0; i < n; i++)
{
    for (int j = 0; j < n; j++)
    {
        sum += arr[i][i]; //считаем сумму всего
    }
}
```

Однако лучше расширять возможное решение

Разбор дз

4 задача:

```
int sum = 0;
for (int i = 0; i < n; i++)
    for (int j = 0; j < n; j++)
        arr[i][j] = -34 + (rand() % static_cast<int>(100 - -34 + 1)); // заполняем массив

for (int i = 0; i < n; i++)
{
    for (int j = 0; j < n; j++)
    {
        if(i < j) sum += arr[i][i]; //считаем сумму над гл диаг
    }
}
```

Заполняем **рандомно** + считаем сумму **над главной** диагональю

Разбор дз

5 задача:

Задача: Пользователь вводит размер массива $n * n$. Массив заполняется случайными числами. Требуется найти произведение элементов побочной диагонали.

Генератор чисел

Шаги:

```
#include <ctime>
```

```
srand(time(0)); //для обновления результатов  
int num = -34 + (rand() % static_cast<int>(100 - -34 + 1));
```

Разбор дз

5 задача:

```
int pr = 1; // изначально произведение должно быть 1
srand(time(0));
for (int i = 0; i < n; i++) {
    for (int j = 0; j < n; j++)
        arr[i][j] = -34 + (rand() % static_cast<int>(100 - -34 + 1));
}

for (int i = 0; i < n; i++)
    for (int j = 0; j < n; j++)
        if(i + j == n - 1) pr *= arr[i][j]; //условие побочная диагональ
```

A hand-drawn blue oval frame with a slightly irregular, sketchy border, centered on the page. It contains the word 'Повторение' in bold black text.

Повторение

Что такое двумерный массив?



Как хранится?

Функции

Функции

Функции — это блоки кода, выполняющие определенные операции.

на самом деле ф-ии мы уже встречали (`sqrt()`, `cbrt()`, `pow()`)



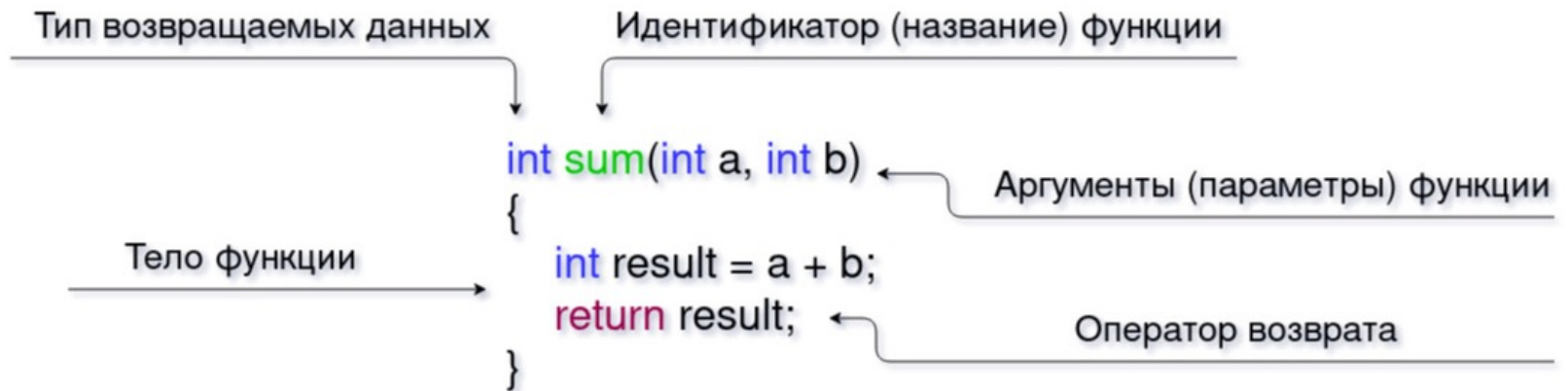
Функции

Виды:

- ❑ **Возвратные** – функции, который имеют определенный тип и что-то возвращают (int, float, double и т.д.).
- ❑ **Невозвратные** – функции неопределенного типа. (void).

Функции

Синтаксис:



Функции

Листинг:

```
#include <iostream>

using namespace std;

void printMess()
{
    //функция неопределенного типа для вывода сообщения
    cout << "Вывод сообщения"; // тело ф-ии
}

int main()
{
    printMess(); //вызов ф-ии
    return 0;
}
```

Описываем ф-ию **перед** основной ф-ией.

Функции

ЛИСТИНГ:

```
#include <iostream>

using namespace std;

float pr(float a, float b) //функция вещественного типа с двумя аргументами
{
    float c = a * b;
    return c; //то, что она будет возвращать
}

int main()
{

    float num1, num2;
    cin >> num1 >> num2;

    cout << pr(num1, num2);
    return 0;
}
```

Функция **вещественного** типа (вар 1)

Функции

ЛИСТИНГ:

```
#include <iostream>

using namespace std;

float pr(float a, float b) //функция вещественного типа с двумя аргументами
{
    return a * b; //то, что она будет возвращать
}

int main()
{

    float num1, num2;
    cin >> num1 >> num2;

    cout << pr(num1, num2);
    return 0;
}
```

Функция **вещественного** типа (вар 2)

Функции

ЛИСТИНГ:

```
#include <iostream>

using namespace std;

float pr(float a, float b) //функция вещественного типа с двумя аргументами
{
    return a * b; //то, что она будет возвращать
}

int main()
{
    float num1, num2;
    cin >> num1 >> num2;
    float num3 = pr(num1, num2); //присваиваем результат ф-ии к переменной
    cout << num3;
    return 0;
}
```

Функция **вещественного** типа (вар 3)

Задача

Задача: Пользователь вводит два вещественных числа (длина и ширина). Напишите ф-ию нахождения периметра и площади.

Задача

Задача: Пользователь в пред. задаче посчитал периметр и площадь. Напишите ф-ию, которая выводит значение переданной переменной на экран.

Задача

Задача: оптимизируйте предыдущие задачи и допишите ф-ию ввода значения.

**Можно ли использовать ф-ию
внутри ф-ии?**

**Какую ф-ию нужно описать в
первую очередь?**

Функции

```
#include <iostream>

using namespace std;

void printMess()
{
    //функция неопределенного типа для вывода сообщения
    cout << "Вывод сообщения "; // тело ф-ии
}

float pr(float a, float b) //функция вещественного типа с двумя аргументами
{
    printMess(); //вызываем ф-ию внутри ф-ии
    return a * b; //то, что она будет возвращать
}

int main()
{
    float num1, num2;
    cin >> num1 >> num2;
    float num3 = pr(num1, num2); //присваиваем результат ф-ии к переменной
    cout << num3;
    return 0;
}
```

1

```
#include <iostream>

using namespace std;

float pr(float a, float b) //функция вещественного типа с двумя аргументами
{
    printMess(); //вызываем ф-ию внутри ф-ии
    return a * b; //то, что она будет возвращать
}

void printMess()
{
    //функция неопределенного типа для вывода сообщения
    cout << "Вывод сообщения "; // тело ф-ии
}

int main()
{
    float num1, num2;
    cin >> num1 >> num2;
    float num3 = pr(num1, num2); //присваиваем результат ф-ии к переменной
    cout << num3;
    return 0;
}
```

2

В каком варианте **не будет** ошибки?

Функции

```
#include <iostream>

using namespace std;

void printMess()
{
    //функция неопределенного типа для вывода сообщения
    cout << "Вывод сообщения "; // тело ф-ии
}

float pr(float a, float b) //функция вещественного типа с двумя аргументами
{
    printMess(); //вызываем ф-ию внутри ф-ии
    return a * b; //то, что она будет возвращать
}

int main()
{
    float num1, num2;
    cin >> num1 >> num2;
    float num3 = pr(num1, num2); //присваиваем результат ф-ии к переменной
    cout << num3;
    return 0;
}
```

Компилятор читает программу **последовательно**

Прототипы функций

Функции

Листинг:

```
#include <iostream>

using namespace std;

int main()
{
    printMess(); //вызов ф-ии
    return 0;
}

void printMess()
{
    //функция неопределенного типа для вывода сообщения
    cout << "Вывод сообщения"; // тело ф-ии
}
```

Можно ли сделать так?

Функции

Листинг:

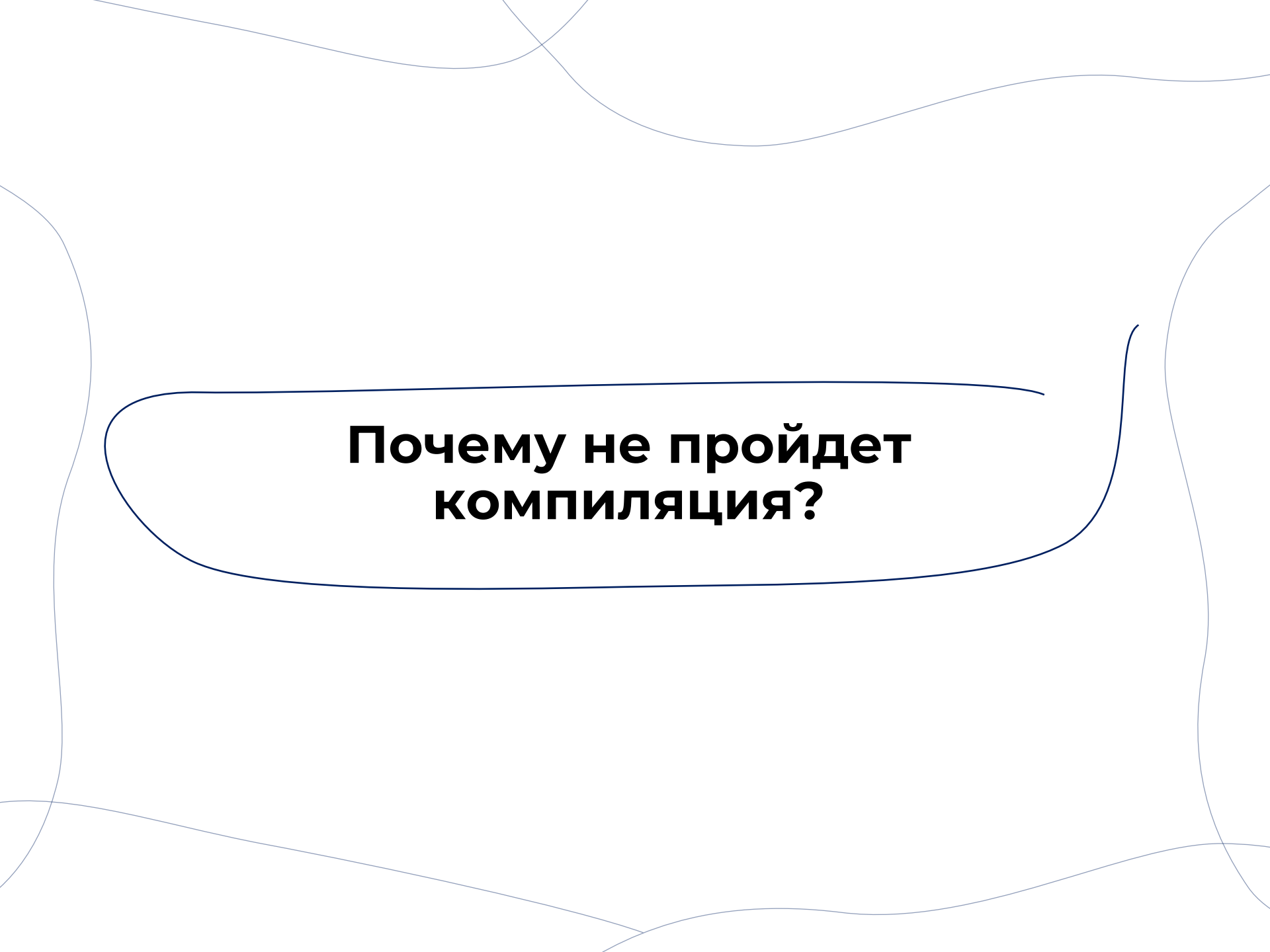
```
#include <iostream>

using namespace std;

int main()
{
    printMess(); //вызов ф-ии
    return 0;
}

void printMess()
{
    //функция неопределенного типа для вывода сообщения
    cout << "Вывод сообщения"; // тело ф-ии
}
```

Хотелось бы, но будет ошибка компиляции(



**Почему не пройдет
компиляция?**

Почему не пройдет компиляция?

```
test1.cpp:7:5: error: use of undeclared identifier 'printMess'  
    printMess(); //вызов ф-ии  
    ^  
1 error generated.
```

Прототипы

Прототип функции — объявление функции, не содержащее тела функции, но указывающее имя функции, типы аргументов и возвращаемый тип данных.

Листинг:

```
#include <iostream>

using namespace std;

void printMess(); //прототип невозвращаемой ф-ии без параметров

int main()
{
    printMess(); //вызов ф-ии
    return 0;
}

void printMess()
{
    //функция неопределенного типа для вывода сообщения
    cout << "Вывод сообщения"; // тело ф-ии
}
```

Прототипы

Листинг:

```
#include <iostream>

using namespace std;

float pr(float a, float b); //прототип функции вещественного типа с двумя аргументами
void printMess(); //прототип ф-ии неопределенного типа

int main()
{
    float num1, num2;
    cin >> num1 >> num2;
    float num3 = pr(num1, num2); //присваиваем результат ф-ии к переменной
    cout << num3;
    return 0;
}

float pr(float a, float b) //функция вещественного типа с двумя аргументами
{
    printMess(); //вызываем ф-ию внутри ф-ии
    return a * b; //то, что она будет возвращать
}

void printMess()
{
    //функция неопределенного типа для вывода сообщения
    cout << "Вывод сообщения "; // тело ф-ии
}
```

Убиваем двух зайцев одновременно

Прототипы

Листинг:

```
#include <iostream>

using namespace std;

float pr(float a, float b); //прототип функции вещественного типа с двумя аргументами
void printMess(); //прототип ф-ии неопределенного типа

int main()
{
    float num1, num2;
    cin >> num1 >> num2;
    float num3 = pr(num1, num2); //присваиваем результат ф-ии к переменной
    cout << num3;
    return 0;
}

float pr(float a, float b) //функция вещественного типа с двумя аргументами
{
    printMess(); //вызываем ф-ию внутри ф-ии
    return a * b; //то, что она будет возвращать
}

void printMess()
{
    //функция неопределенного типа для вывода сообщения
    cout << "Вывод сообщения "; // тело ф-ии
}
```

Пишем **красивый код** и нам **не важна** последовательность объявления.

Задача

Задача: напишите ф-ию решения квадратного уравнения и функцию вывода.

На вход: три переменные (коэффициента) a , b , c .

Прототипы

Приятный момент

При объявлении прототипа **необязательно** указывать название переменных (типы **обязательны**), но **нужно** при объявлении самой ф-ии.

Прототипы

Листинг:

```
#include <iostream>

using namespace std;

float pr(float, float); //прототип функции без указания названий аргументов НО ТИПЫ НУЖНЫ

int main()
{
    float num1, num2;
    cin >> num1 >> num2;
    float num3 = pr(num1, num2); //присваиваем результат ф-ии к переменной
    cout << num3;
    return 0;
}

float pr(float a, float b) //функция вещественного типа с двумя аргументами
{
    return a * b; //то, что она будет возвращать
}
```

Задача

Задача: напишите ф-ию вывода одномерного массива. (void)

Задача

Задача: напишите ф-ию вывода двумерного массива. (void)

Перегрузка функций



Что такое перегрузка?

Задача

Задача: напишите ф-ию суммы для двух целых и вещественных чисел.

Перегрузка

Перегрузка — это возможность использовать одну и ту же функцию для разных типов данных.

Полиморфизм - способность функции обрабатывать данные разных типов.

Перегрузка

Перегрузка — это возможность использовать одну и ту же функцию для разных типов данных.

Полиморфизм - способность функции обрабатывать данные разных типов.

перегруженная функция = полиморфная функция



Почему это удобно?

Перегрузка

```
#include <iostream>

using namespace std;

int sum(int, int); // для целых чисел
float sum(float, float); // для вещественных чисел
double sum(double, double); // для вещественных чисел

int main()
{
    int a = 1, b = 4;
    float a1 = 1.12, b1 = 4.56;
    double a2 = 2.25, b2 = 8.45;

    cout << sum(a, b) << endl; // передаются целые числа
    cout << sum(a1, b1) << endl; // передаются вещественные числа
    cout << sum(a2, b2) << endl; // передаются вещественные числа
}

int sum(int a, int b)
{
    return a + b;
}

float sum(float a, float b)
{
    return a + b;
}

double sum(double a, double b)
{
    return a + b;
}
```

Не зависим от **типов** поступающих переменных.

Задача

Задача: Пользователь вводит три числа (три стороны). Напишите ф-ию нахождения площади. (учтите разные типы данных).

Рекурсия



Что такое рекурсия?



Рекурсия

Функция является **рекурсивной**, если оператор в теле функции вызывает функцию, содержащую данный оператор.

Вызов самой функции внутри нее.

Задача

Задача: По приезду в лилипутию жителей земли встречает конвертер валют. 1 рубль = 1.33 лилипутским дублонам.

Переведите рубли в дублоны и выведите эту сумму, если пользователь вводит отрицательно число, попросите ввести снова.

Рекурсия

Листинг:

```
#include <iostream>

using namespace std;

void run(int); //объявление прототипа

int main()
{
    int value;
    cout << "Введите изначальное кол-во денег: ";
    cin >> value;
    run(value);
}

void run(int num)
{
    if (num < 0)
    {
        cout << "Введите положительное число для подсчета: " << endl;
        int n;
        cin >> n;
        run(n); //вызов самой ф-ии
    }
    else
    {
        cout << "В лилипутии будет: " << num * 1.33 << " дублонов" << endl;
    }
}
```


Рекурсия

Задача:

Выведите факториал до введенного числа.

Рекурсия

Листинг:

```
#include <iostream>
using namespace std;

unsigned int factorial(unsigned int); // прототип рекурсивной функции

int main(int argc, char *argv[])
{
    int n; // локальная переменная для передачи введенного числа с клавиатуры
    cout << "Enter n!: ";
    cin >> n;
    for (int i = 1; i <= n; i++)
    {
        cout << i << "!" << "=" << factorial(i) << endl; // вызов рекурсивной функции
    }
    return 0;
}

unsigned int factorial(unsigned int f) // рекурсивная функция для нахождения n!
{
    unsigned long int result;
    if (f == 1 || f == 0) // 1!=1 и 0!=1
        return 1;
    result = f * factorial(f - 1); // вызов самой себя
    return result;
}
```