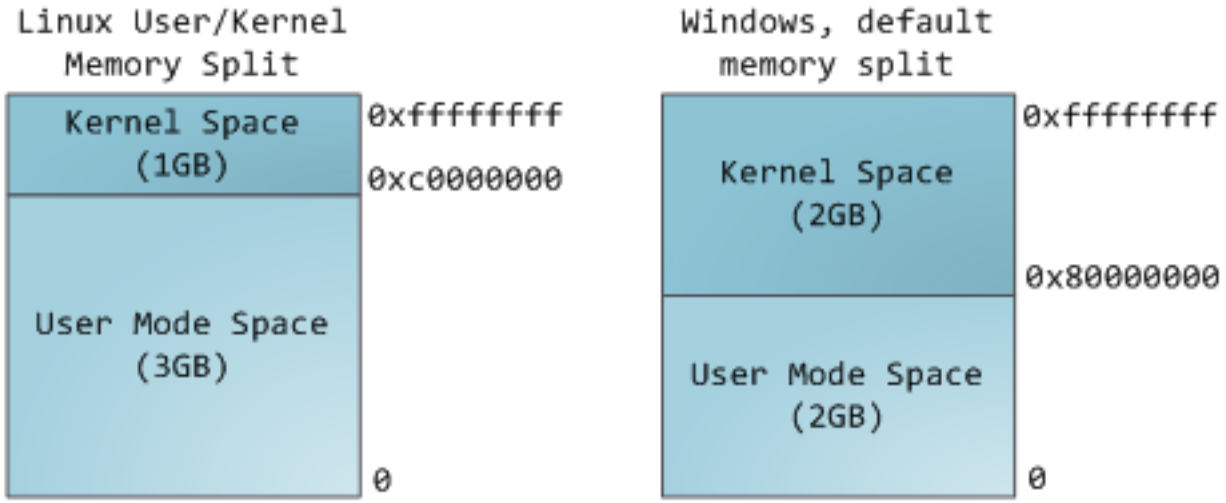
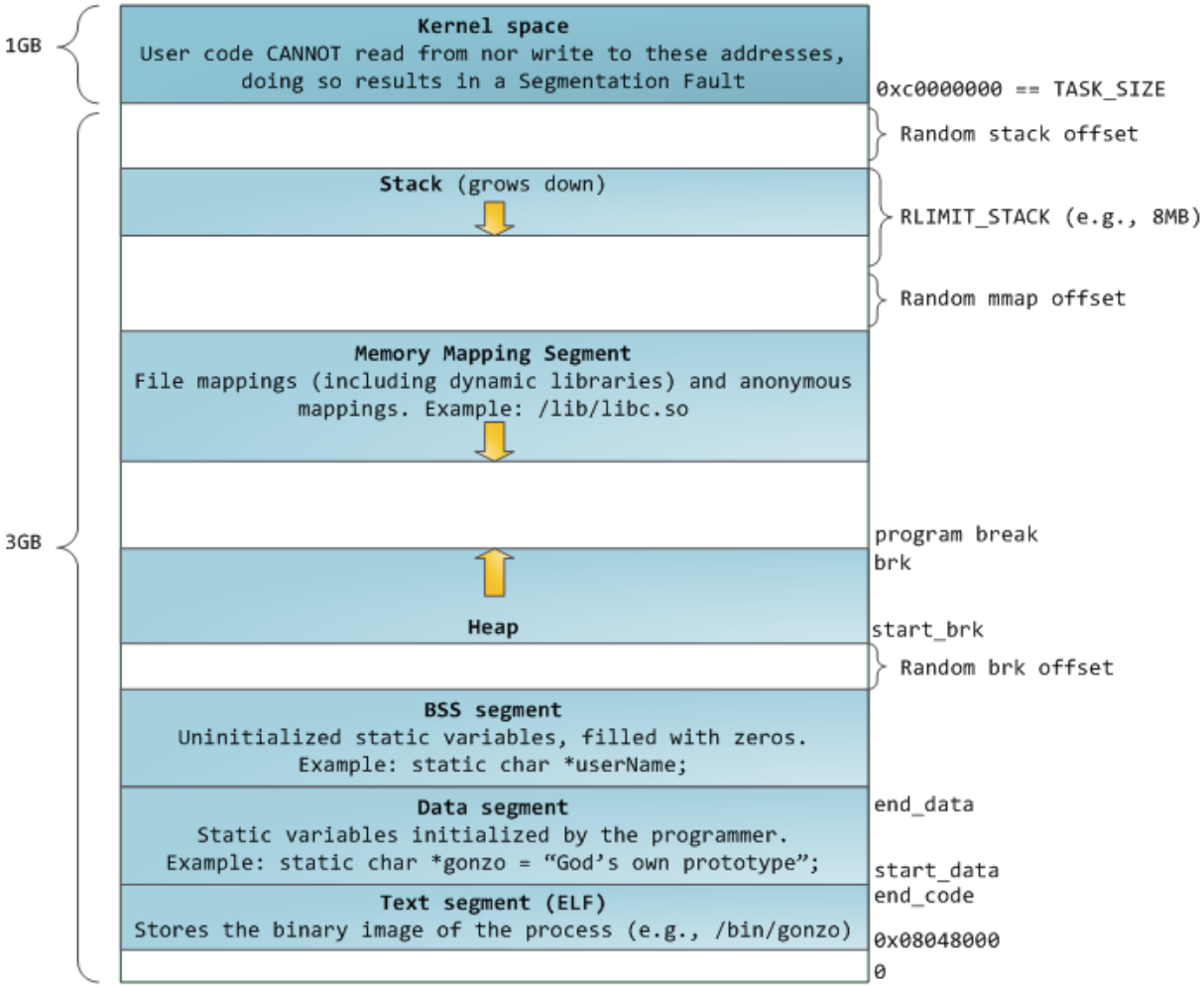
Научно-исследовательская практика

АДРЕСА, ДИНАМИЧЕСКАЯ ПАМЯТЬ, МАТРИЦЫ

Пару слов про стек и кучу...



И чуть подробнее



Двумерные массивы

Матрица в стековой области памяти

```
#include <iostream>
#define USE_MATH_DEFINES
#include <cmath>

using namespace std;

int main(int argc, char *argv[])
{
	float matrix[3][3];
	matrix[0][0] = cos(M_PI/6);  matrix[0][1] = sin(M_PI/6);
	matrix[0][2] = 0;
	matrix[1][0] = -sin(M_PI/6); matrix[1][1] = cos(M_PI/6);
	matrix[1][2] = 0;
	matrix[2][0] = 0;  matrix[2][1] = 0; matrix[2][2] = 1;

	for (int i=0; i<3; i++)
	{
		for (int j=0; j<3; j++)
		{
			cout << matrix[i][j] << "\t";
		}
		cout << endl;
	}
	return 0;
}
```

Матрицы в динамической памяти

Здесь матрица представляет собой линейный с точки зрения памяти одномерный массив

```
void MakeMatrix(int rows, int columns);
void PrintMatrix(double *array, int rows, int columns);

int main(int argc, char *argv[])
{
    MakeMatrix(3,3);
    return 0;
}

void MakeMatrix(int rows, int columns)
{
          double* pArray = new double [rows*columns];
          for (int i=0; i<rows; i++)
          {
                    for (int j=0; j<columns; j++)
                              pArray[i*rows+j] = 100*i+j;
          }
          PrintMatrix(pArray, rows, columns);
}

void PrintMatrix(double* array, int rows, int columns)
{
          for (int i=0; i<rows; i++)
          {
                    for (int j=0; j<columns; j++)
                              cout << array[i*rows+j] << "\t";
                    cout << endl;
          }
}
```

Матрицы в динамической памяти

```
Starting /Users/amakashov/Documents/Work/test/build-test_proj-Desktop-Debug/test_proj...
0           1           2
100         101         102
200         201         202
/Users/amakashov/Documents/Work/test/build-test_proj-Desktop-Debug/test_proj exited with code 0
```

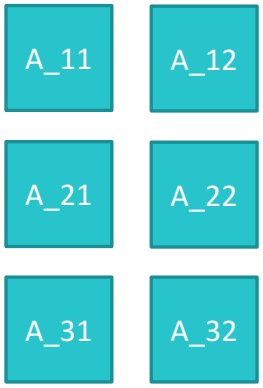
Матрицы в динамической памяти

Двумерный массив через двойной указатель

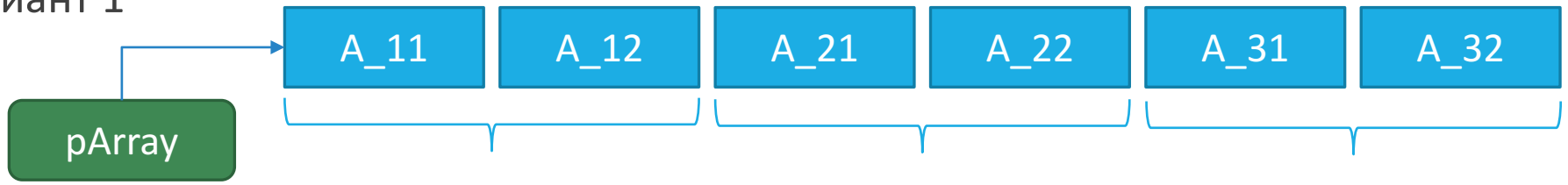
```
int main(int argc, char *argv[])
{
        MakeMatrix(3,3);
        return 0;
}

void MakeMatrix(int rows, int columns)
{
        double** pArray = new double* [rows];
        for (int i=0; i<rows; i++)
        {
                pArray[i] = new double[columns];
                for (int j=0; j<columns; j++)
                        pArray[i][j] = 100*i+j;
        }
        PrintMatrix(pArray, rows, columns);
}

void PrintMatrix(double** array, int rows, int columns)
{
        for (int i=0; i<rows; i++)
        {
                for (int j=0; j<columns; j++)
                        cout << array[i][j] << "\t";
                cout << endl;
        }
}
```

Различия

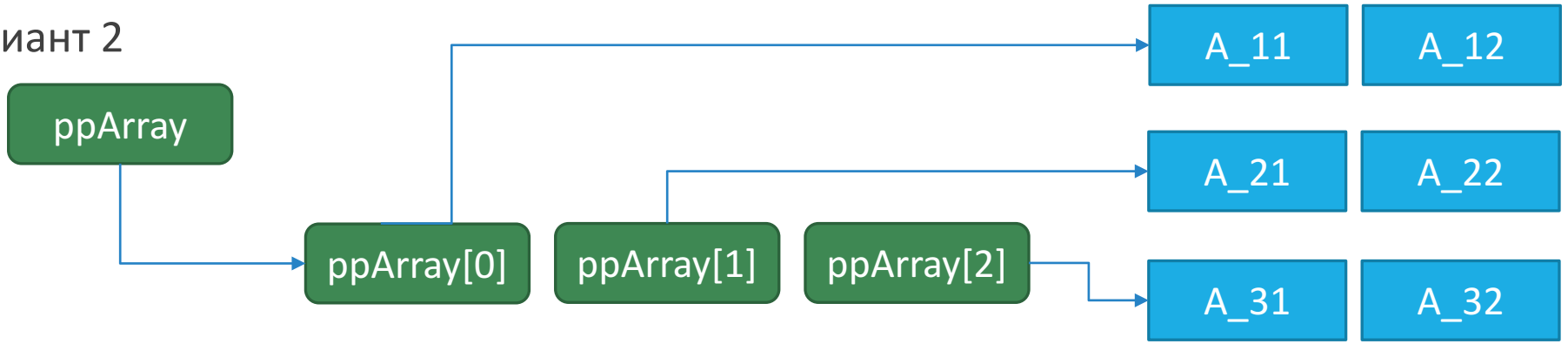
Матрица 3x2



Вариант 1



Вариант 2



Можно сделать даже тройной указатель

```
const int size = 3;

int main(int argc, char *argv[])
{
    int*** testPointer;
    testPointer = new int**[size];
    for (int i=0; i<size; i++)
    {
        testPointer[i] = new int*[size];
        for (int j=0; j<size; j++)
        {
            testPointer[i][j] = new int[size];
            for(int k=0; k<size; k++)
                testPointer[i][j][k]=100*i+10*j+k;
        }
    }
    for (int i=0; i<size; i++)
    {
        for (int j=0; j<size; j++)
        {
            for(int k=0; k<size; k++)
                cout << testPointer[i][j][k] << "\t";
            cout << endl;
        }
        cout << "\n*********************************\n";
    }
    return 0;
}
```

Матрицы в динамической памяти

```
int main(int argc, char *argv[])
{
        MakeMatrix(3,3);
        return 0;
}

void MakeMatrix(int rows, int columns)
{
        double** pArray = new double* [rows];
        for (int i=0; i<rows; i++)
        {
                pArray[i] = new double[columns];
                for (int j=0; j<columns; j++)
                        pArray[i][j] = 100*i+j;
        }
        PrintMatrix(pArray, rows, columns);
```

Утечки памяти

```
int main(int argc, char *argv[])
{
        MakeMatrix(3,3);
        return 0;
}

void MakeMatrix(int rows, int columns)
{
        double** pArray = new double* [rows]; // Выделили память
        for (int i=0; i<rows; i++)
        {
                pArray[i] = new double[columns]; // И ещё раз
                for (int j=0; j<columns; j++)
                        pArray[i][j] = 100*i+j;
        }
        PrintMatrix(pArray, rows, columns);
        // А теперь мы потеряли указатель. Что стало с памятью?
}

void PrintMatrix(double** array, int rows, int columns)
{
        for (int i=0; i<rows; i++)
        {
                for (int j=0; j<columns; j++)
                        cout << array[i][j] << "\t";
                cout << endl;
        }
}
```

Утечки памяти

```
int main(int argc, char *argv[])
{
        MakeMatrix(3,3);
        return 0;
}

void MakeMatrix(int rows, int columns)
{
        double** pArray = new double* [rows]; // Выделили память
        for (int i=0; i<rows; i++)
        {
                pArray[i] = new double[columns]; // И ещё раз
                for (int j=0; j<columns; j++)
                        pArray[i][j] = 100*i+j;
        }
        PrintMatrix(pArray, rows, columns);
        for (int i=0; i<rows; i++)
                delete[] pArray[i]; // Удалили массивы строк
        delete[] pArray; // А теперь вобще всё
}

void PrintMatrix(double** array, int rows, int columns)
{
        for (int i=0; i<rows; i++)
        {
                for (int j=0; j<columns; j++)
                        cout << array[i][j] << "\t";
                cout << endl;
        }
}
```

Краткое резюме по динамической памяти

int* ptr = new int – создание одного элемента в динамической памяти

int* ptr = new int[size] – создание массива элементов в динамической памяти

delete ptr – удаление указателя

delete[] ptr – удаление указателя на массив

В C – использовались malloc() и free()

Строки в стиле С

Такая строка – просто указатель на массив байтов

```
#include <iostream>
#include <cstring> // <string.h>

using namespace std;

int main(int argc, char *argv[])
{
        char cString[] = "My sample string";
        cout << "C-string: " << cString << endl;
        cout << "Length of string: " << strlen(cString) << endl;
        cout << "Size of string: " << sizeof(cString) << endl;

        cString[9] = 0; // '\0';
        cout << "\nAfter modification:\n";
        cout << "C-string: " << cString << endl;
        cout << "Length of string: " << strlen(cString) << endl;
        cout << "Size of string: " << sizeof(cString) << endl;

        char secondString[] = " text";
        char* resultingString = strcat(cString, secondString);
        cout << resultingString << endl;
        cout << "Length of string: " << strlen(cString) << endl;
        cout << "Size of string: " << sizeof(cString) << endl;
}
```

Результат

```
Starting /Users/amakashov/projects/build-test_2seminar-Desktop-Debug/test_2seminar...
C-string: My sample string
Length of string: 16
Size of string: 17

After modification:
C-string: My sample
Length of string: 9
Size of string: 17
My sample text
Length of string: 14
Size of string: 17
/Users/amakashov/projects/build-test_2seminar-Desktop-Debug/test_2seminar exited with code 0
```

Когда выделяется память?

```
#include <iostream>
#include <cstring> // <string.h>

using namespace std;

int main(int argc, char *argv[])
{
        char cString[] = "My sample string";
        cout << "C-string: " << cString << endl;
        cout << "Length of string: " << strlen(cString) << endl;
        cout << "Size of string: " << sizeof(cString) << endl;

        cString[9] = 0; // '\0';
        cout << "\nAfter modification:\n";
        cout << "C-string: " << cString << endl;
        cout << "Length of string: " << strlen(cString) << endl;
        cout << "Size of string: " << sizeof(cString) << endl;

        char secondString[] = " long-long-long text";
        char* resultingString = strcat(cString, secondString);
        cout << resultingString << endl;
        cout << "Length of string: " << strlen(cString) << endl;
        cout << "Size of string: " << sizeof(cString) << endl;
}
```

Получаем ошибку

```
Starting /Users/amakashov/projects/build-test_2seminar-Desktop-Debug/test_2seminar...
C-string: My sample string
Length of string: 16
Size of string: 17

After modification:
C-string: My sample
Length of string: 9
Size of string: 17
My sample long-long-long text
Length of string: 29
Size of string: 17
The program has unexpectedly finished.
/Users/amakashov/projects/build-test_2seminar-Desktop-Debug/test_2seminar crashed.
```

В динамической памяти

```
#include <iostream>
#include <cstring> // <string.h>

using namespace std;

int main(int argc, char *argv[])
{
          char* cString = new char[80];
          memcpy (cString, "My sample string", strlen("My sample
string"));
          cout << "C-string: " << cString << endl;
          cout << "Length of string: " << strlen(cString) << endl;
          cout << "Size of string: " << sizeof(cString) << endl;

          cString[3] = '\0';
          cout << "\nAfter modification:\n";
          cout << "C-string: " << cString << endl;
          cout << "Length of string: " << strlen(cString) << endl;
          cout << "Size of string: " << sizeof(cString) << endl;

          char secondString[] = " long-long-long text";
          char* resultingString = strcat(cString, secondString);
          cout << resultingString << endl;
          cout << "Length of string: " << strlen(cString) << endl;
          cout << "Size of string: " << sizeof(cString) << endl;
          delete[] cString;
}
```

С динамическим выделением памяти…

```
Starting /Users/amakashov/projects/build-test_2seminar-Desktop-Debug/test_2seminar...
C-string: My sample string
Length of string: 17
Size of string: 8

After modification:
C-string: My
Length of string: 3
Size of string: 8
My  long-long-long text
Length of string: 23
Size of string: 8
/Users/amakashov/projects/build-test_2seminar-Desktop-Debug/test_2seminar exited with code 0
```

К слову о синтаксисе main…

```
#include <iostream>
#include <cstring> // <string.h>

using namespace std;

int main(int argc, char *argv[])
{                    VS2017_COMMUNITY_RTW.3_ENU
        cout << "Number of arguments " << argc << endl;
        for (int i=0; i<argc; i++)
                cout << i+1 << " argument " <<argv[i] << endl;
}
```

```
Starting /Users/amakashov/projects/build-test_2seminar-Desktop-Debug/test_2seminar...
Number of arguments 1
1 argument /Users/amakashov/projects/build-test_2seminar-Desktop-Debug/test_2seminar
/Users/amakashov/projects/build-test_2seminar-Desktop-Debug/test_2seminar exited with code 0
```

Ссылки

```cpp
#include <iostream>

using namespace std;

void substract(double& value)
{
	value -= 1;
}

int main(int argc, char *argv[])
{
	double a = 3.14;
	double& b = a; // Ссылка на переменную a
	// double& c; // Ошибка - ссылка ВСЕГДА должна быть инициализирована
	cout << "a=" << a << " and b = " << b << endl;
	a = 2.78;
	cout << "a=" << a << " and b = " << b << endl;
	b *=2;
	cout << "a=" << a << " and b = " << b << endl;
	substract(b);
	cout << "a=" << a << " and b = " << b << endl;
	substract(a);
	cout << "a=" << a << " and b = " << b << endl;
	double d = 1;
	b = d; // Здесь мы присваиваем значение!
		// Сделать так, чтобы b стало ссылкой на d - невозможно
	cout << "a=" << a << " and b = " << b <<" and d = " << d << endl;
	return 0;
}
```

Ссылки

```
Starting /Users/amakashov/Documents/Work/test/build-test_proj-Desktop-Debug/test_proj...
a=3.14 and b = 3.14
a=2.78 and b = 2.78
a=5.56 and b = 5.56
a=4.56 and b = 4.56
a=3.56 and b = 3.56
a=1 and b = 1 and d = 1
/Users/amakashov/Documents/Work/test/build-test_proj-Desktop-Debug/test_proj exited with code 0
```

Функции и их синтаксис

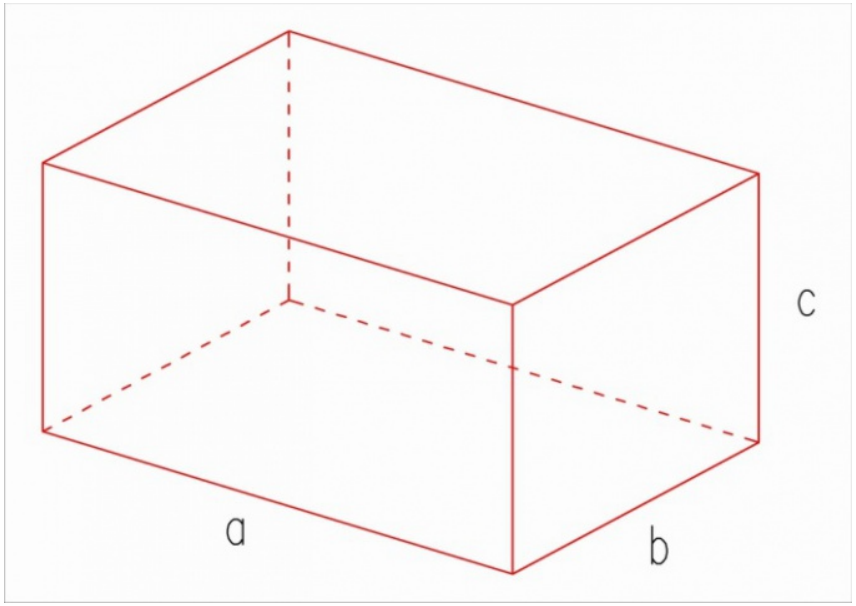
```
void function1(){;}

void function2(int x, int* y){;}

double function3(double& r, double h)
{
            return M_PI*r*r*h;
}
int main(int argc, char *argv[])
{
            int x=0, y=0;
            double r = 1, h=2;
            function1();
            function2(x, &y);
            double volume = function3(r, h); // Присваиваем возвращаемое
                                             //          значение
            function3(r, h); // Просто ничего не присвоится
            return 0;
}
```

Перегрузка функций

```
#include <iostream>
#define USE_MATH_DEFINES
#include <cmath>
using namespace std;

double Area(double r)
{
        cout << "Circle area function" << endl;
        return M_PI*r*r;
}

double Area(double a, double b)
{
        cout << "Rectangle area function" << endl;
        return a*b;
}

int main(int argc, char *argv[])
{
        double r = 1, h=2;
        double circleArea = Area(r);
        cout << "Circle area is " << circleArea << endl;
        double rectArea = Area(r, h);
        cout << "Rectangle area is " << rectArea << endl;
        return 0;
}
```

Перегрузка функций

Добавьте ещё одну функцию, которая будет вычислять площадь поверхности прямоугольного параллелепипеда



Значения по-умолчанию

```
Starting /Users/amakashov/Do
Rectangle area function
First rectangle area is 2
Rectangle area function
Second rectangle area is 1
/Users/amakashov/Documents/W
```

```cpp
#include <iostream>

using namespace std;

double Area(double a, double b = 1.0)
{
        cout << "Rectangle area function" << endl;
        return a*b;
}

int main(int argc, char *argv[])
{
        double a = 1, b=2;
        double rectArea1 = Area(a, b);
        cout << "First rectangle area is " << rectArea1 << endl;
        double rectArea2 = Area(a);
        cout << "Second rectangle area is " << rectArea2 << endl;
        return 0;
}
```

Значения по-умолчанию

```
double Area(double a, double b = 1.0)
        // так можно
double Area(double a = 2.0, double b = 1.0)
        // и так тоже
double Area(double a = 2.0, double b) // нельзя

double Area(double a = 2.0, double b = 1.0)
{
        cout << "Rectangle area function" << endl;
        return a*b;
}

int main(int argc, char *argv[])
{
        double x = Area(3.0, 2.5);
        double y = Area(7.0); // a=7.0, b=1.0
        double z = Area(); // a=2.0, b=1.0
}

double function (int a, int b=1, int c=2, int d=3);
```