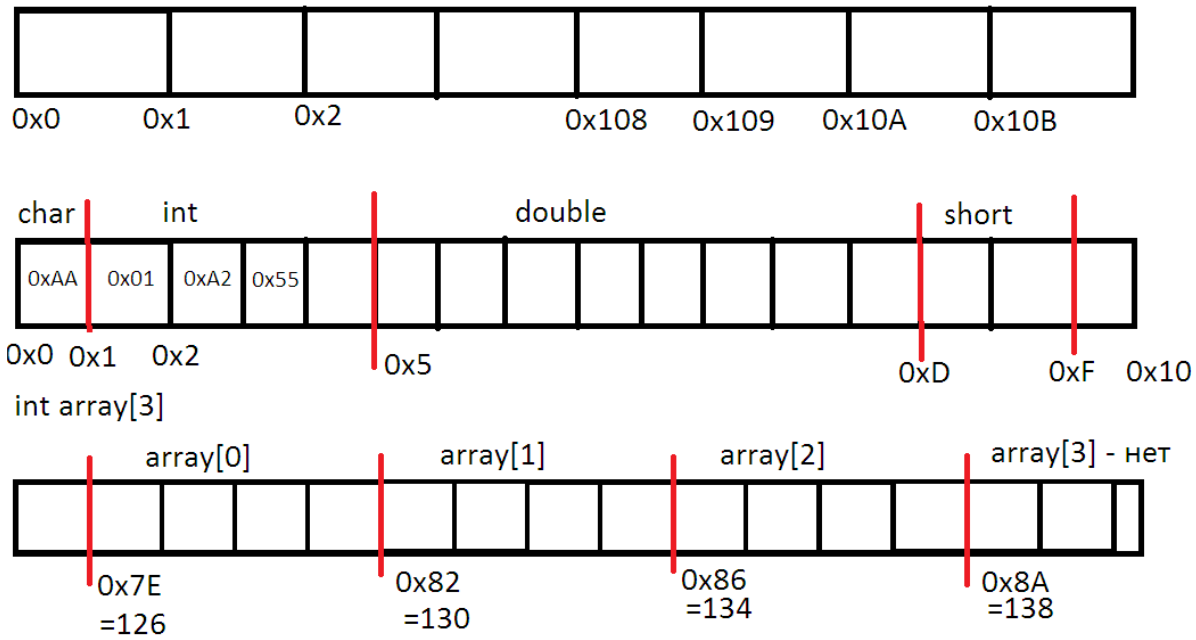
Научно-исследовательская практика

МОДЕЛЬ ПАМЯТИ. УКАЗАТЕЛИ. МАССИВЫ.

Плоская модель памяти



Функция sizeof()

sizeof(value) – стандартная функция, которая возвращает нам размер элемента

Например, для переменной

```
int a = 0;

cout << sizeof(a) << endl; //должно быть 4

cout << sizeof(float) << endl; // тоже 4
```

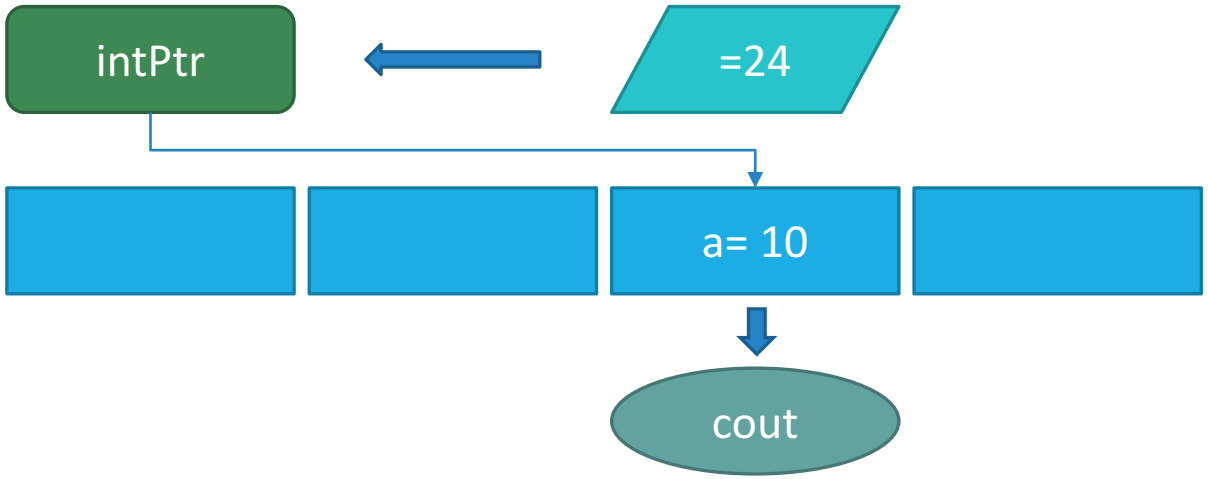
Функция sizeof()

```
#include <iostream>

using namespace std;

int main(int argc, char *argv[])
{
    cout << "Size of int="<< sizeof(int) << endl;
    int b = 0;
    cout << "Size of int variable="<< sizeof(b) << endl;
    cout << "Size of unsigned short="<< sizeof(unsigned short) << endl;
    cout << "Size of bool="<< sizeof(bool) << endl;

    return 0;
}
```

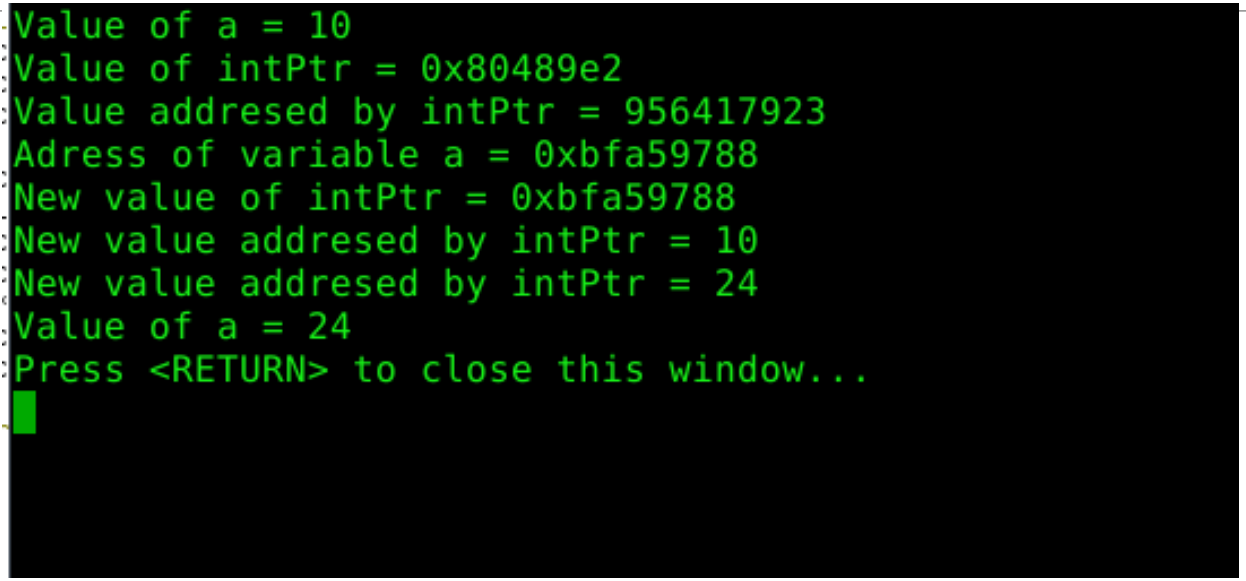
```
Starting /Users/amakashov/Documents/Work/test/build-test_proj-Desktop-Debug/test_proj...
Size of int=4
Size of int variable=4
Size of unsigned short=2
Size of bool=1
/Users/amakashov/Documents/Work/test/build-test_proj-Desktop-Debug/test_proj exited with code 0
```

Указатели

```cpp
#include <iostream>

using namespace std;

int main()
{
    int a = 10; //   Переменная int
    int* intPtr;    // Указатель на переменную int
    cout << "Value of a = " << a << endl;
    cout << "Value of intPtr = " << intPtr << endl; // Здесь мы выводим адрес, на который указвает intPtr
    cout << "Value addresed by intPtr = " << *intPtr << endl; // А здесь - значение

    cout << "Adress of variable a = " << &a << endl;    //  Так мы можем вывести адрес переменной
    intPtr = &a;        //  Указателю можно присвоить адрес переменной
    cout << "New value of intPtr = " << intPtr << endl; //  Теперь указатель указывает на нашу переменную a
    cout << "New value addresed by intPtr = " << *intPtr << endl;   //  Значение по указателю - значение a
    *intPtr = 24;   //    теперь попробуем присвоить переменной, на которую указывает указатель, значение
    cout << "New value addresed by intPtr = " << *intPtr << endl;   //  Выведем значение по указателю
    cout << "Value of a = " << a << endl;   //  А теперь посмотрим, что поизошло с нашей переменной

    return 0;
}
```



Указатели


Value of a = 10
Value of intPtr = 0x80489e2
Value addresed by intPtr = 956417923
Adress of variable a = 0xbfa59788
New value of intPtr = 0xbfa59788
New value addresed by intPtr = 10
New value addresed by intPtr = 24
Value of a = 24
Press <RETURN> to close this window...

Чуть подробнее о синтаксисе

int var – объявление переменной

int* varPtr – объявление указателя

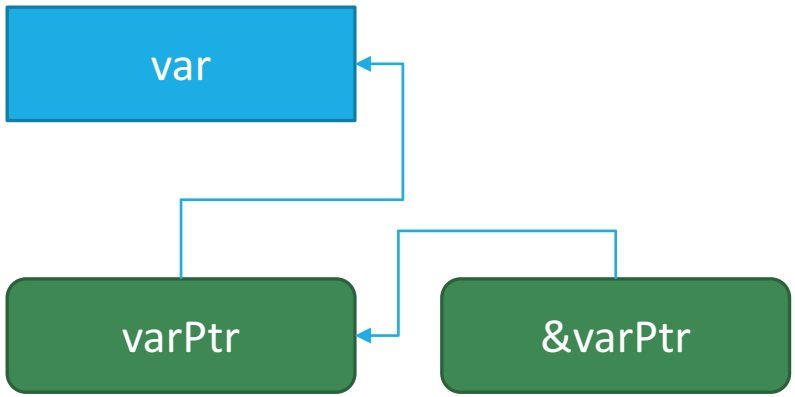
&var – адрес переменной

varPtr – адрес в указателе

*varPtr – значение по указателю

&varPtr – адрес самого указателя

```
int var;

int* varPtr;

varPtr = &var;

var = *varPtr;
```

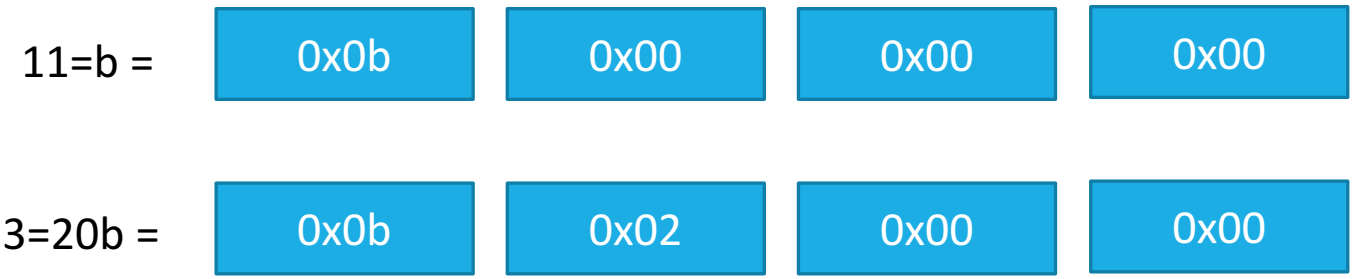


Указатели

```
#include <iostream>

using namespace std;

int main()
{
    int a = 11; //   Переменная int
    unsigned char* charPtr = 0;  //  казатель на переменную чар
                    //  Мы сразу инициализируем его нулём для безопасности
    cout << "Now some magic" << endl;
    //  charPtr = (char*)&a;    //  Преобразование типов в стиле C
    charPtr = reinterpret_cast<unsigned char*>(&a);  // Преобразование типов в стиле C++
    cout << "Size of int variable " << sizeof(int) << " bytes" << endl; // sizeof - размер элемента
    cout << "Size of char variable " << sizeof(unsigned char) << " bytes" << endl; // можно вызвать для типа
    cout << "Size of charPtr variable " << sizeof(*charPtr) << " bytes" << endl;   // а можно - для переменной
    // Мы получили указатель на переменную размером в 4 байты
    for (int i=0; i<sizeof(int); i++)   //  Выведем их в hex'е
        cout << std::dec << i <<"-th byte value " << std::hex<< (int)(*(charPtr+i)) << endl;
    a = 512+11; //  Слегка изменим переменную
    cout << "More magic with arrays" << endl;
    for (int i=0; i<sizeof(int); i++)   //  И выведем снова, но индексируя как массив
        cout << std::dec << i <<"-th byte value "<< std::hex << (int)charPtr[i] << endl;
    return 0;
}
```

Указатели

```
Now some magic
Size of int variable 4 bytes
Size of char variable 1 bytes
Size of charPtr variable 1 bytes
0-th byte value b
1-th byte value 0
2-th byte value 0
3-th byte value 0
More magic with arrays
0-th byte value b
1-th byte value 2
2-th byte value 0
3-th byte value 0
Press <RETURN> to close this window...
```



Массивы

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

using namespace std;

int main()
{
    const int size = 10;
    int array[size];
    for (int i=0; i<size; i++)
    {
        array[i] = size -i;
    }

    for (int i=0; i<size; i++)
    {
        cout << "Array[" << i << "] = " << array[i] << endl;
    }
}
```

Массивы

```cpp
#include <iostream>

using namespace std;

int main()
{
    const int size = 10;     //  Размер массива
    int array [size];        //  Создание массива заданого размера
    for (int i=0; i<size; i++)  //  В цикле заполняем значения
        array[i] = size - i;
    //  Давайте выведем размер массива и первого элемента
    cout << "Array size " << sizeof(array) << " and element size " << sizeof(array[0]) << endl;
    for (int i=0; i<size; i++)  //  Можно обращаться как с указателем
        cout << i << " element " << *(array+i) << endl;
    int* intPtr = array;     //  Создадим указател и укажем им на начало массива
    for (int i=0; i<size; i++)
        intPtr[i] *= 2;      //  Теперь мы работаем с указателем как с массивом
    cout << "Value of first element " << array[0] << endl;  //  Нумерация начинается с 0
    cout << "Value of last element " << array[size-1] << endl;  //  и заканчивается на size-1
    cout << "Element outside of array " << array[10] << endl;   //  А вот типичная ошибка
}
```

Массивы

```
Array size 40 and element size 4
0 element 10
1 element 9
2 element 8
3 element 7
4 element 6
5 element 5
6 element 4
7 element 3
8 element 2
9 element 1
Value of first element 20
Value of last element 2
Element outside of array -1073850140
Press <RETURN> to close this window...
```

Передача в функцию

```
#include <iostream>

using namespace std;
const int gSize = 10;

void fibonacci(int innerArray[], int size);

int main()
{
    int array [gSize];
    cout << "Array address " << array << endl;
    cout << "Gsize address " << &gSize << endl;
    for (int i =0; i<gSize; i++)
        array[i] = 0;
    fibonacci(array, gSize);
    for (int i =0; i<gSize; i++)
        cout << i << "-th element " << array[i] << endl;
    cout << "gSize=" << gSize << endl;
}
                        const char *

void fibonacci(int innerArray[], int size)
{
    cout << "Inner array adress "<< innerArray << endl;
    cout << "size address " << &size << endl;
    if (size<2)
        return;
    innerArray[0] = innerArray [1] =1;
    for (int i=2; i < size; i++)
    {
        innerArray[i] = innerArray[i-2]+innerArray[i-1];
    }
    size = 12;
}
```

Передача в функцию

```
Array address 0xbf969540
Gsize address 0x8048ac4
Inner array adress 0xbf969540
size address 0xbf969534
0-th element 1
1-th element 1
2-th element 2
3-th element 3
4-th element 5
5-th element 8
6-th element 13
7-th element 21
8-th element 34
9-th element 55
gSize=10
Press <RETURN> to close this window...
```

Передача в функцию

```
#include <iostream>

using namespace std;
int gSize = 10;

void fibonacci(int innerArray[], int* size);

int main()
{
    int array [gSize];
    cout << "Array address " << array << endl;
    cout << "Gsize address " << &gSize << endl;
    for (int i =0; i<gSize; i++)
        array[i] = 0;
    fibonacci(array, &gSize);
    for (int i =0; i<gSize; i++)
        cout << i << "-th element " << array[i] << endl;
    cout << "gSize=" << gSize << endl;
}

void fibonacci(int innerArray[], int *size)
{
    cout << "Inner array adress "<< innerArray << endl;
    cout << "size address " << size << endl;
    if (*size<2)
        return;
    innerArray[0] = innerArray [1] =1;
    for (int i=2; i < *size; i++)
    {
        innerArray[i] = innerArray[i-2]+innerArray[i-1];
    }
    *size = 12;
}
```

Передача в функцию

```
Array address 0xbfea04a0
Gsize address 0x8049e00
Inner array adress 0xbfea04a0
size address 0x8049e00
0-th element 1
1-th element 1
2-th element 2
3-th element 3
4-th element 5
5-th element 8
6-th element 13
7-th element 21
8-th element 34
9-th element 55
10-th element 1
11-th element 134514420
gSize=12
Press <RETURN> to close this window...
```

Сделай сам…

```
#include <iostream>

using namespace std;

const int vectorSize = 3;

void mul(float vector[], float multiplier);

int main(int argc, char *argv[])
{
    float vector[vectorSize];
    cout << "Size of vector is " << sizeof(vector) << endl;

    for (int i=0; i<vectorSize; i++)
    {
        vector[i] = i;
    }
    mul(vector, 3.14);
    for (int i=0; i<vectorSize; i++)
    {
        cout << "Vector["<<i<<"]="<<vector[i] << endl;
    }
    return 0;
}

void mul(float vector[], float multiplier) {...}
```

Сделай сам…

```
Starting /Users/amakashov/Documents/Work/test/build-test_proj-Desktop-Debug/test_proj...
Size of vector is 12
Size of vector is 8
Vector[0]=0
Vector[1]=3.14
Vector[2]=6.28
/Users/amakashov/Documents/Work/test/build-test_proj-Desktop-Debug/test_proj exited with code 0
```

Массивы и как всё сломать

```
cout << "array 0-th element " << &(intArray[0]) << " and 1-st element " << &(intArray[1]) << endl;
cout << "Difference between 0th and 1st " << &(intArray[1])-&(intArray[0]) << endl;
double doubleArray[2] = {3.14, 2.78};
cout << "Now for doubles:\narray 0-th element " << &(doubleArray[0]) << " and 1-st element " << &(doubleArray[1]) << endl;
cout << "Difference between 0th and 1st " << &(doubleArray[1])-&(doubleArray[0]) << endl;
char *c1=(char*)&(doubleArray[0]), *c2=(char*)&(doubleArray[1]);
cout << "Now char pointers:\c1="<< hex << (int)c1 << " c2=" <<  hex <<(int)c2 <<endl;
cout << "Difference between 0th and 1st " << c2-c1 << endl;
cout << "Now lets change char pointer value:" << endl;
cout << "Double values before: " << doubleArray[0] << "\t" << doubleArray[1] << endl;
c2 +=6;
*c2 += -127;
cout << "Double values after: " << doubleArray[0] << "\t" << doubleArray[1] << endl;
```

```
array 0-th element 0xbf86afd0 and 1-st element 0xbf86afd4
Difference between 0th and 1st 1
Now for doubles:
array 0-th element 0xbf86b020 and 1-st element 0xbf86b028
Difference between 0th and 1st 1
Now char pointers:c1=bf86b020 c2=bf86b028
Difference between 0th and 1st 8
Now lets change char pointer value:
Double values before: 3.14      2.78
Double values after: 3.14       743.68
```

Динамические массивы

```
//    double bigOne[10000000];
    unsigned int arraySize = 10000000;
    double step = 2*
            M_PI/(arraySize-1);
    double* doublePtr = new double [arraySize];
    for (int i=0; i<arraySize; i++)
        doublePtr[i] = sin (step*i);
    double summ = 0;
    for (int i=0; i<arraySize; i++)
        summ += doublePtr[i]*step;
    cout << "Summ on interval " << summ << endl;
    summ=0;
    for (int i=0; i<arraySize; i++)
        summ += fabs(doublePtr[i])*step;
    cout << "Abs summ on interval " << summ  << endl;

    return 0;
}
```