

**Язык C++** - это  
компилируемый, статически (*или строго*) типизированный  
язык программирования общего назначения

### **Поддерживает парадигмы программирования**

(набор свойств и понятий, определяющий стиль программирования):

- процедурное программирование,
- объектно-ориентированное программирование,
- обобщённое программирование (*шаблоны классов*).

### **Областью применения языка является создание:**

- операционных систем,
- прикладных программ,
- драйверов устройств,
- приложений для встраиваемых систем,
- высокопроизводительных серверов,
- игр и пр.;

Синтаксис **C++** унаследован от языка **C**. Одним из принципов разработки было сохранение совместимости с **C**. Тем не менее, **C++** не является дополнением языка **C**.

*Разработал язык сотрудник фирмы Bell Labs **Бьёрн Страуструп** в начале 1980-х годов под первоначальным названием «**C с классами**». Первая коммерческая версия языка вышла в 1985 г., и только в 1998 г. язык был стандартизирован.*

## Создание проекта в *Visual Studio*:

**Файл -> Создать проект -> Установленные -> Visual C++ -> Консольное приложение Windows**

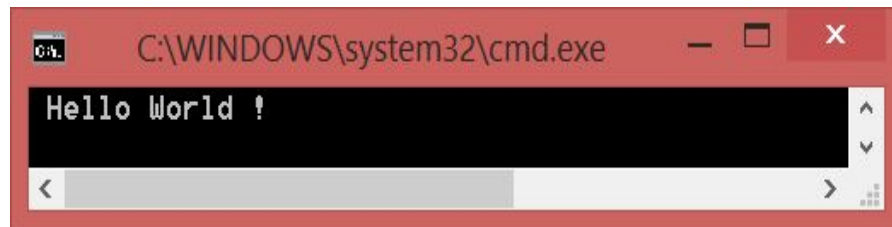
```
#include "stdafx.h"
int main()
{
    return 0;
}
```



```
using namespace System;
void main() // можно и оставить int !
{
    Console::WriteLine(" Hello World !");
    Console::ReadLine();
}
```

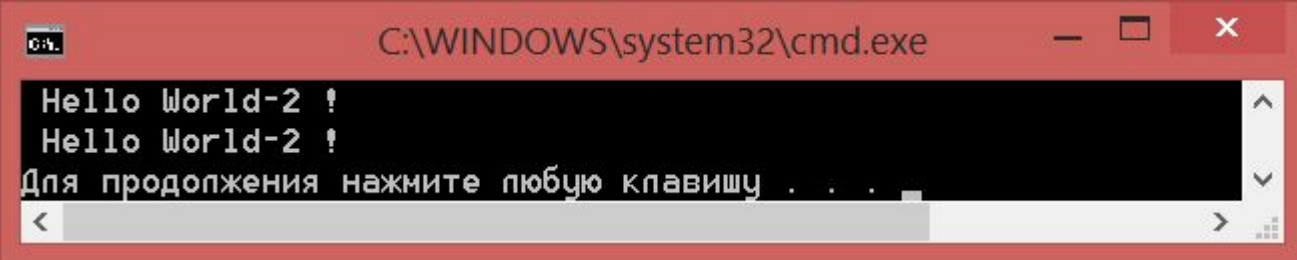
### При этом (в VS2017):

- в свойствах проекта должна быть выбрана опция «поддержка clr-среды»(проект-свойства-общие),
- отменена прекомпиляция заголовочных файлов (проект-свойства - C/C++ - Предварительно откомпилированные заголовки),
- можно также из проекта удалить неиспользуемые \*.h и \*.cpp файлы.



## Ещё одна «Первая программа»

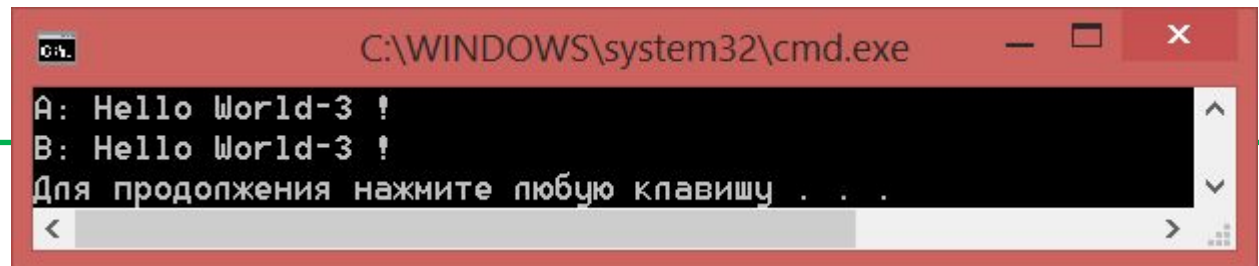
```
using namespace System;
class A
{
    public: void helloworld()
    {
        Console::WriteLine(" Hello World-2 !");
    }
};
void main()
{
    A a;
    a.helloworld ( );
    A *pa = &a;
    pa -> helloworld ( );
}
```



A screenshot of a Windows command prompt window. The title bar shows the path "C:\WINDOWS\system32\cmd.exe". The window contains the following text: "Hello World-2 !", "Hello World-2 !", and "Для продолжения нажмите любую клавишу . . .". The text is displayed in a monospaced font on a black background. There are standard window controls (minimize, maximize, close) in the top right corner and a scroll bar on the right side.

## «Первая программа», версия 3

```
using namespace System;
class B {
    public: void helloworld();
};
void B::helloworld() { Console::WriteLine("B: Hello World-3 !"); }
class A {
    public: void helloworld()
        { Console::WriteLine("A: Hello World-3 !"); }
};
int main()
{
    A a;
    B b;
    a.helloworld();
    b.helloworld();
    return 0;
}
```



A screenshot of a Windows command prompt window. The title bar shows the path C:\WINDOWS\system32\cmd.exe. The window contains the following text: "A: Hello World-3 !", "B: Hello World-3 !", and "Для продолжения нажмите любую клавишу . . .". The window has a red border and standard Windows window controls (minimize, maximize, close) in the top right corner.

```
C:\WINDOWS\system32\cmd.exe
A: Hello World-3 !
B: Hello World-3 !
Для продолжения нажмите любую клавишу . . .
```

## Встроенные типы данных

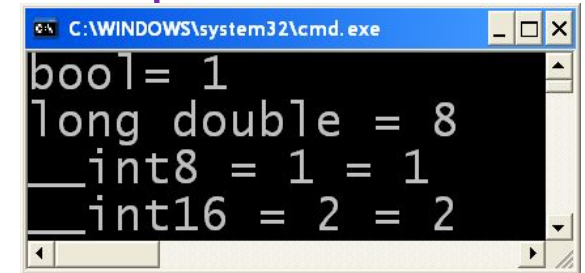
- `bool` - 1 байт;
- `char` - 1 байт;
- `unsigned char` - 1 байт;
- `short` - 2 байта;
- `unsigned short` - 2 байта;
- `int` - 4 байта;
- `unsigned int` - 4 байта;
- `long` - 4 байта;
- `unsigned long` - 4 байта;

С плавающей точкой:

- `float` - 4 байта;
- `double` - 8 байт;
- `long double` – вещественный размером 10 байт;

Специальные:

```
using namespace System;
int main()
{
    bool b;
    long double ld;
    __int8 i8;
    __int16 i16;
    Console::WriteLine("bool= {0}", sizeof b);
    Console::WriteLine("long double = {0}", sizeof ld);
    Console::WriteLine("__int8 = {0} = {1}", sizeof i8, sizeof(__int8)
);
    Console::WriteLine("__int16 = {0} = {1}", sizeof i16,
sizeof(__int16));
}
```



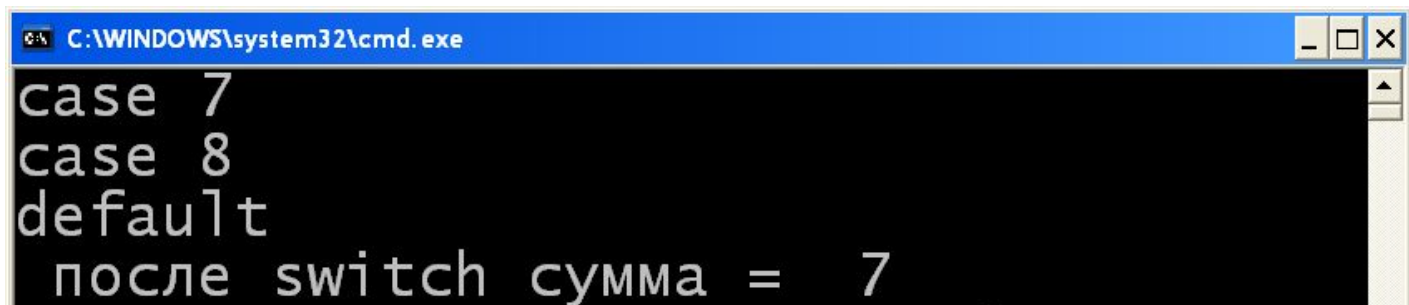
```
C:\WINDOWS\system32\cmd.exe
bool = 1
long double = 8
__int8 = 1 = 1
__int16 = 2 = 2
```

`sizeof` – операция получения размера объекта или типа, измеренных в байтах

## Особенности `switch`

```
using namespace System;
```

```
void main ()
{
    int i = 1, j = 2, k=3;
    switch( i + j + ++k)
    {
        case 5: Console::WriteLine ("case 5 "); break;
        case 6: Console::WriteLine ("case 6 "); break;
        case 7: Console::WriteLine ("case 7 ");
        case 8: Console::WriteLine ("case 8 ");
        default: Console::WriteLine ("default ");
    }
    Console::WriteLine(" после switch сумма = {0} ", i+j+k );
}
```



The screenshot shows a Windows command prompt window with the title bar "C:\WINDOWS\system32\cmd.exe". The window contains the following output:

```
case 7
case 8
default
после switch сумма = 7
```

## Подробнее об указателях

**Указатели** – это переменные, предназначенные для размещения адресов областей памяти.

**Указатели** бывают трёх видов: на объект, на функцию, и на ***void***.

**Указатель** не является ***самостоятельным*** типом, он всегда определяется типом объекта (*или функции*), адрес которого будет ***содержать***.

**Указатель** на функцию предназначен для вызова соответствующего типа функции (*во многом аналогичен событию на C#*).

**Указатель на объект** и **указатель на *void*** предназначены для работы с ***объектами данных***.

Указатель на ***void*** применяется в случае, когда конкретный тип не определён. Ему можно присвоить значение указателя любого типа, но перед его использованием в большинстве случаев требуется ***явное приведение типа***.

В дальнейшем речь пойдет об указателях ***на объект*** и ***на void***.

## Объявление и инициализация указателей

Указатель на объект объявляется с помощью конструкции

**Тип \*имя ;**

и может быть проинициализирован одним из следующих вариантов:

- **операцией** получения адреса: **имя = &ИмяОбъекта;**
- с помощью другого **указателя**: **имя = ДругойУказатель;**
- с помощью **имени массива**: **имя = ИмяМассива;**
- **адресом** памяти: **имя = (Тип\*) 0xШестнадцатеричнаяКонстанта;**
- **нуль-адресом** или просто нулём: **имя = NULL ; //или имя = 0;**
- с помощью операции **new**: **имя = new Тип;** – для размещения одного объекта;
- с помощью операции **new**: **имя = new Тип[N];** – для размещения **N объектов**;

## Операции с указателями

- **разъадресации** **\* имя** (доступ к значению по адресу, размещённому в указателе);
- **явное приведение** к типу указателя (**\* имя**);
- **сложения** (с переменными, константами, с другими указателями);
- **сравнения** (с другим указателем);



## Указатель и одномерный массив

```
using namespace System;
void main()
{
    int *p = new int ( 2 ); // выделение памяти с инициализацией
    Console::WriteLine( "{0}", *p );
    if (p == 0 ) return;
    *p = 100/33;
    Console::WriteLine( "{0}", *p );
    delete p;
    int m [ 10 ] = { 2, 3, 4, 5, 6, 7, 10, 12 }; // остальные – нули
    int *pm = m;
    *pm = 10;
    pm++;
    *(pm+1) = 11;
    pm+=2;
    *(pm+1) = 20;
    *(pm+2) += *pm--;
    Console::WriteLine("{0} {1} {2} {3} {4}",
        *pm, *(pm + 1), *(pm + 4), *(pm + 5), *(pm + 1) + *pm );
}
```

|    |   |    |    |    |
|----|---|----|----|----|
| 2  |   |    |    |    |
| 3  |   |    |    |    |
| 11 | 5 | 10 | 12 | 16 |

**Выделенная** указателю память (с помощью `new` или `new [ ]`) должна быть в обязательном порядке **освобождена** (с помощью `delete` или `delete [ ]`)

Исключение из этого правила составляют локальные указатели в функции *main*.

# Указатель и матрица

```
using namespace System;
void main()
{ int m [4] [6] = {
    {2,3,4,5,6,7},
    {1,3,5,7,9,1},
    {7,6,5,4,3,2},
    {1,2,6,7,8,9}, };
```

```
int *pm = (int*) m;
```

```
int (*pmm) [6] = m; //указатель pmm на массив из шести int-ов
```

**//значение указателя - целое число:**

```
Console::WriteLine("№1 \tpm = {0:x}\t\tpmm = {1:x}\n\tsizeof(pm)={2} байта\tsizeof(pmm)={3} байта",
    (int)pm, (int)pmm, sizeof(pm),sizeof(pmm));
```

**//указатели можно складывать с константами:**

```
Console::WriteLine("№2 {0} {1} {2} {3}", *pm , *(pm+1) ,*(int*) pmm , *(int*) ( pmm+1) );
pm++;
```

**pmm++; //практически, то же самое, что и в №2, но после сдвига указателей:**

```
Console::WriteLine("№3 {0} {1} {2} {3}", *pm , *(pm+1) ,*(int*)pmm , *(int*)(pmm+1));
```

```
int s;
```

```
Console::WriteLine("№4 s = {0}", s ); //в стандарте языка авто переменные не обнуляются
```

**s= \*(++pm); //указатели можно сравнивать:**

```
while ( pm < (int*) pmm ) s += *pm++;
```

```
Console::WriteLine("№5 s = {0}", s ); // 26, потому что элемент m[0][2] суммируется дважды
```

```
pm = &m[3][4];
```

**pmm = ( int (\*)[6] ) pm; //также допустима операция вычитания:**

```
Console::WriteLine("№6 {0} {1} {2} {3}", *(pm-1) , *(int*)pmm ,*(int*)(pmm-1) , *(int*)(pmm-2));
```

```
}
```

|    |   |   |                    |   |                     |
|----|---|---|--------------------|---|---------------------|
| №1 |   |   | pm =12f34c         |   | pmm =12f34c         |
|    |   |   | sizeof(pm)=4 байта |   | sizeof(pmm)=4 байта |
| №2 | 2 | 3 | 2                  | 1 |                     |
| №3 | 3 | 4 | 1                  | 7 |                     |
| №4 | s | = | 0                  |   |                     |
| №5 | s | = | 26                 |   |                     |
| №6 | 7 | 8 | 3                  | 9 |                     |

## Составные описатели

Составными описателями называют идентификаторы (*имена*), дополненные более чем одним признаком :

- [ ]** - массив,
- ( )** - функция или изменение очередности,
- \*** - указатель.

### Правило интерпретации:

1. Рассматриваются квадратные и круглые скобки (*одинаковый приоритет*) **справа от имени** в направлении **слева направо (-->)**;
1. Затем в направлении **справа налево (<--)** рассматриваются звёздочки, расположенные **слева от имени**;
1. Для изменения данного порядка могут быть использованы **круглые скобки**.

### Примеры:

`int *p [10];` - массив **p** указателей на значения *типа int*

`int (*pp) [10];` - указатель **pp** на массив из 10 значений *типа int*

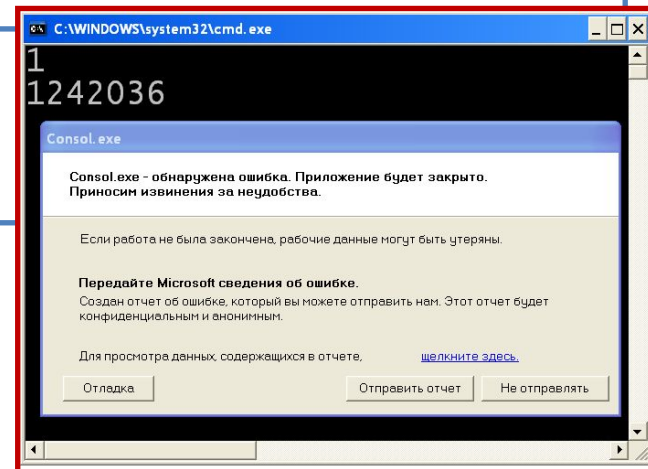
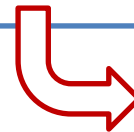
## Типичные ошибки с указателями

```
using namespace System;
int main()
{
    int *p; //значение - ноль
    Console::WriteLine ("{0} ", *p );
} // исключение NullReference
```

```
using namespace System;
int main()
{
    int *p;
    *p = 1;
    Console::WriteLine ("{0} ", *p );
    delete p;
} //освобождение не выделенной
памяти
```

```
using namespace System;
int main()
{
    int *p = new int;
    *p=1;
    Console::WriteLine("{0}", *p);
    p = new int; // потерян адрес, где был выделен первый фрагмент
    *p= 2 ;
    Console::WriteLine("{0}", *p);
    delete p; } //«висячая ссылка», система становится потенциально неустойчивой
```

```
using namespace System;
int main()
{
    int m[10] = {1, 2, 3, 4, 5, 6, 7, 8, 9};
    int *p = m;
    Console::WriteLine("{0}", *p);
    p += 10;
    Console::WriteLine("{0}", *p);
    p[0] = 100;
    Console::WriteLine("{0}", *p);
} // исключение NullReference
```



## Контрольная работа: нулевой билет

**Задание:** установите содержимое консольного окна в результате выполнения программы на языке C++, приведите решение в виде схем объектов:

```
using namespace System;
void main()
{ int m[3][4] = { {1,2,3,4},
                  {5,6,7,8},
                  {9,9,9,9} };
  int *p1      = (int*)m;
  int (*p2) [4] = m;
  int s       = *( ++p1 );
  while ( p1 < (int*) &m[1][0] )
    s += *(p1++) + *(int*) (p2++);

  Console::WriteLine("{0} {1} {2} {3}", *p1, *(p1-1), *(int*) (p2-1), s);
}
```

|   |   |   |    |
|---|---|---|----|
| 5 | 4 | 9 | 26 |
|---|---|---|----|

# Тема коллоквиума

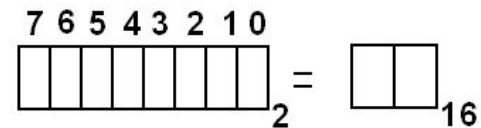
## обработка целочисленных объектов с помощью указателей

Варианты типов (задачи):

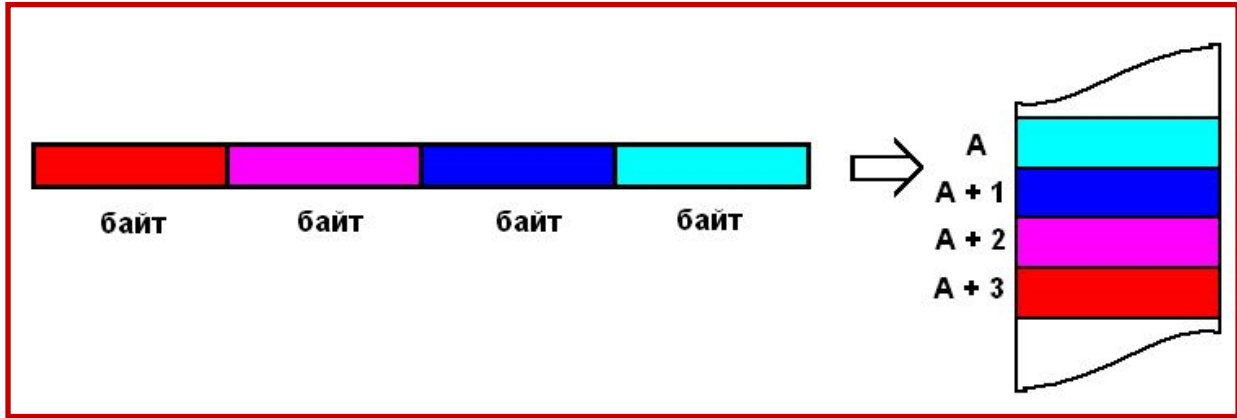
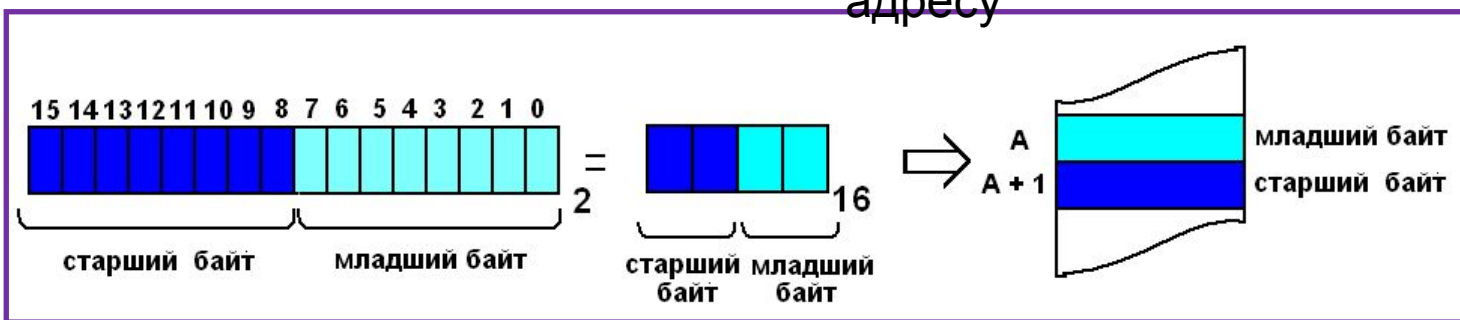
**unsigned char** - 1 байт;

**unsigned short** - 2 байта;

**unsigned int** - 4 байта;



**Правило intel:**  
старший байт размещается по большему адресу



# Пример 1

```
using namespace System;
int main()
{
    unsigned char m1[10] = { 2, 1, 3, 2, 4, 1, 5, 2, 3, 1 };
    unsigned short *p1 = (unsigned short*) m1;
    Console::WriteLine ( "Первый пример:" );
    Console::WriteLine ( "{0} {1} {2} {3}", m1[0], m1[1], m1[2], m1[3]);
    Console::WriteLine ( "{0} {1} {2} {3}", *p1, *(p1+1), *(p1+2), *(p1+3));
}
```

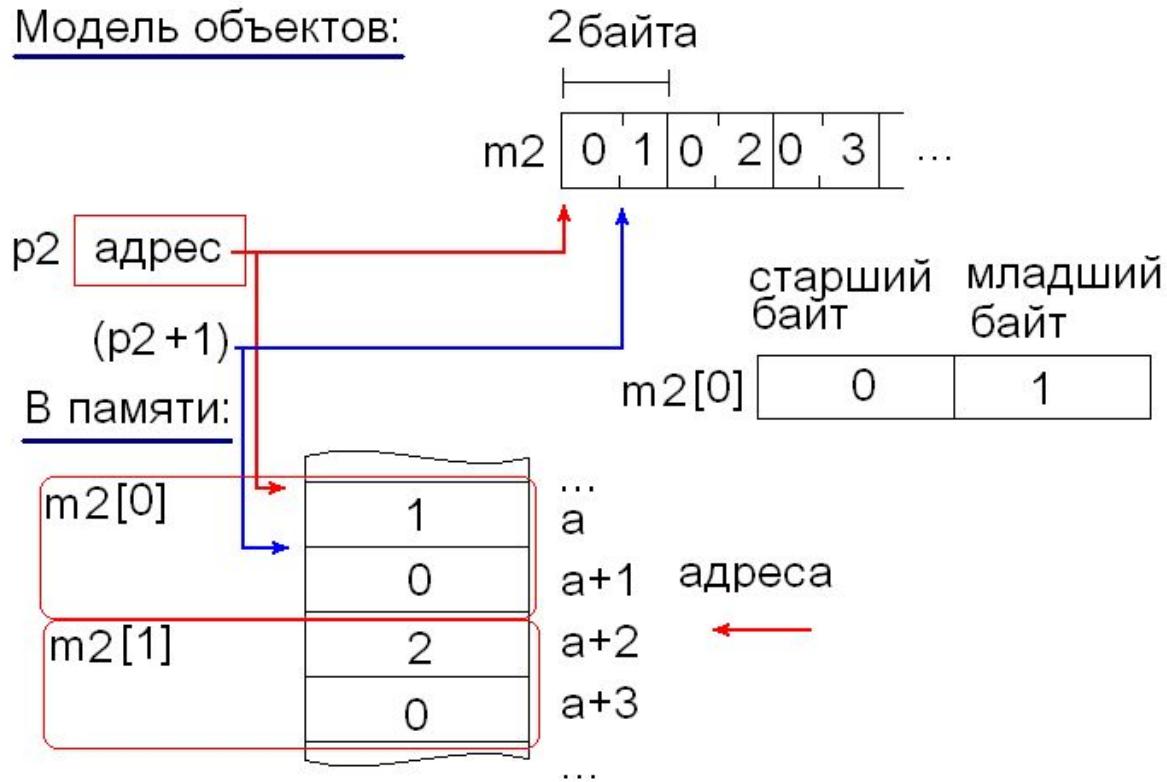
```
Первый пример:
2 1 3 2
258 515 260 517
```



## Пример 2

```
using namespace System;
int main()
{
    unsigned short m2[10] = {1,2,3,4,6,7,8,9,10};
    unsigned char *p2 = (unsigned char*) m2;
    Console::WriteLine("\n\nВторой пример:");
    Console::WriteLine("{0} {1} {2} {3}", m2[0], m2[1], m2[2], m2[3]);
    Console::WriteLine("{0} {1} {2} {3}", *p2, *(p2+1), *(p2+2), *(p2+3));
}
```

```
Второй пример:
1 2 3 4
1 0 2 0
```





## Коллоквиум: нулевой билет

*Коллоквиум по дисциплине «Информационные технологии в электроэнергетике и электротехнике», 2017/18 у.г.*

### Билет № 0

1. Установите содержимое консольного окна в результате выполнения программы, приведите решение в виде схемы объектов:

```
using namespace System;
int main()
{
    unsigned char m[12]= 0x11,1,0x12,2,0x13,1,0x14,1,0x15,2,0x16};
    unsigned short *p =(unsigned short *)m;
    p++;
    Console::WriteLine("{0}  {1}  {2}", *(p+1), *(p+2), *(p+3));
}
```

Ответ:        275    276    533

**B c ë !**