

Увод в програмирането

9: Указатели

доц. Атанас Семерджиев

```
// Намиране на адрес
```

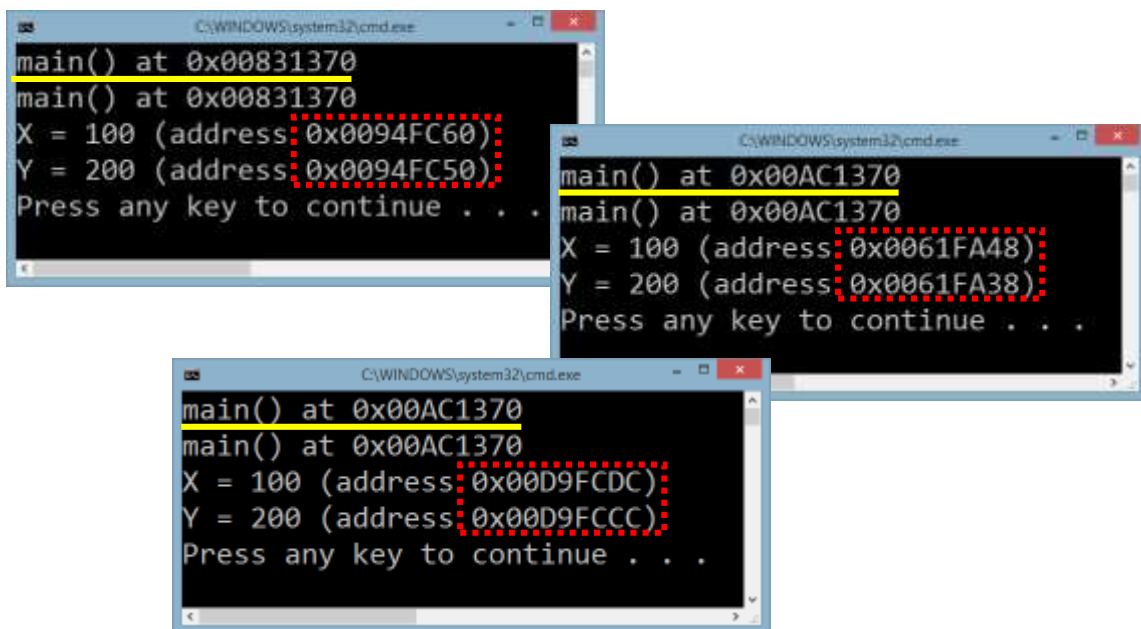
```
int main()
{
    int x = 100;
    double y = 200;

    std::cout
        << "main() at 0x" << main << "\n"
        << "main() at 0x" << &main << "\n";

    std::cout
        << "x = " << x << " (address 0x" << &x << ")\n"
        << "y = " << y << " (address 0x" << &y << ")\n";

    return 0;
}
```

2



Дефиниране и използване на указатели

```
int i = 100;
int * pi = &i; // Дефинираме указател

std::cout << "i=" << i << std::endl;

pi = &i; // Променяме указателя pi
*pi = 200; // Променяме променливата i чрез указателя pi

std::cout << "i=" << i << std::endl;

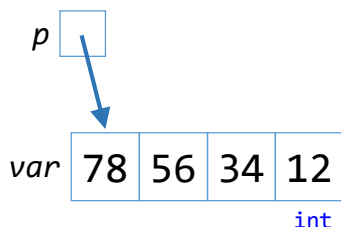
std::cout << "i=" << *pi << std::endl; // Достъпваме съдържанието
// на i чрез указателя pi
```

Графично представяне

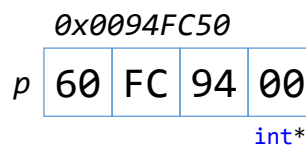
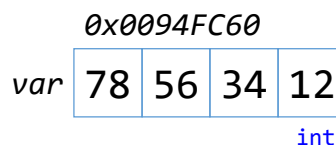
```
int var = 0x12345678;
```

```
int *p = &var;
```

Графично (опростено)



Физическо



* В примера допускаме, че системата е 32-битова, little-endian, а размерът на `int` е 32 бита

Типове

```
int i = 100;
```

```
double d = 1.5;
```

```
unsigned long long l = 200;
```

```
// Указател към даден тип T се дефинира, като към T
// се добави символът звезда.
```

```
int * pi = &i; // Казваме "pi е указател от тип int"
```

```
double * pd = &d;
```

```
unsigned long long * pl = &l;
```

```

// При дефиниране на няколко променливи на един ред
// има особеност!

// Три променливи от тип int
int var1, var2, var3;

// Грешка: pointer 1 е указател,
// но pointer2 е променлива от тип int!
int* pointer1, pointer2;

// Звездата се поставя пред всяка променлива,
// която бихме искали да бъде указатели. Това позволява
// на един ред да се дефинират променливи и указатели от
// един и същ тип.
int var4, var5, *pointer3, var6, *pointer4;

```

Преобразуване на указатели

```

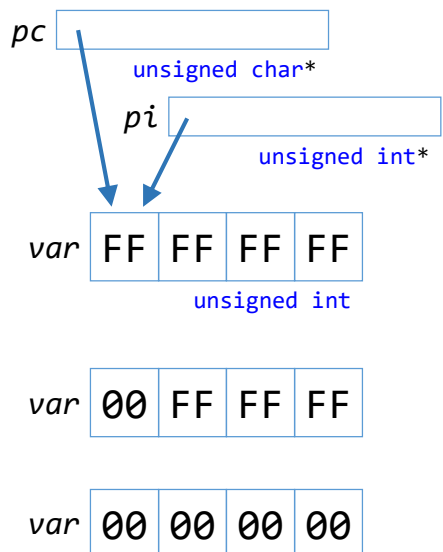
unsigned int x = 0xFFFFFFFF;
unsigned int *pi = &x;
unsigned char *pc = (unsigned char*)&x;

cout << "x=" << hex << x << endl;

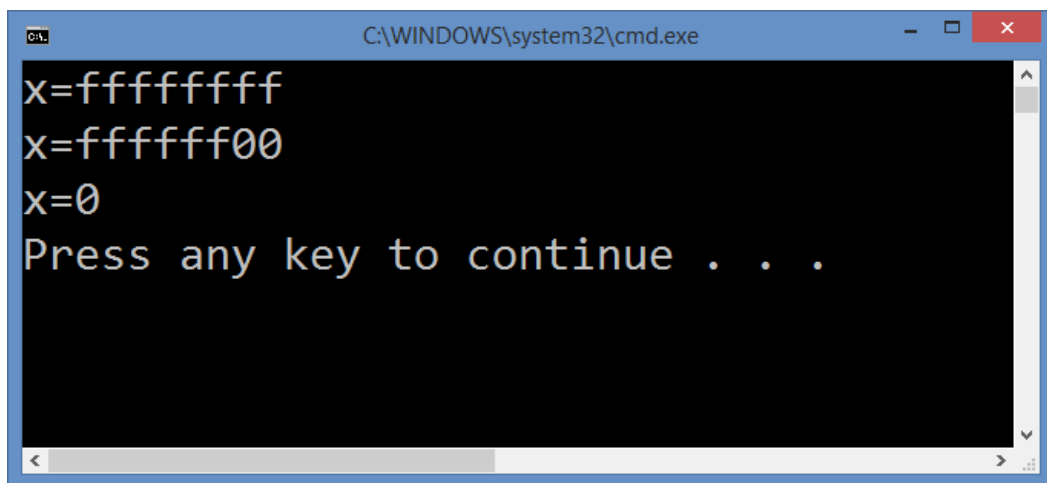
*pc = 0;
cout << "x=" << hex << x << endl;

*pi = 0;
cout << "x=" << hex << x << endl;

```



* В примера допускаме, че `unsigned int` заема 4B



```
C:\WINDOWS\system32\cmd.exe
x=ffffffff
x=ffffff00
x=0
Press any key to continue . . .
```

Блуждаещи указатели

```
// p1 не е инициализиран
```

```
int *p1;
```

```
*p1 = 500;           // Какво ли ще се случи?
```

```
std::cout << *p1;    // Какво ли ще се случи?
```

```
int *p1 = 0;
int *p2 = NULL;    // Еквивалентно с горното

// null-pointer assignment
*p1 = 500;          // грешка!
*p2 = 500;          // грешка!
std::cout << *p1;  // грешка!

int x;
p2 = &x;    // Инициализираме p2
*p2 = 500;  // Използваме го за нещо
p2 = 0;     // Ако вече не ни трябва, нулираме указателя
```

За повече информация: <http://c-faq.com/null/index.html>

Предаване на аргументи по указател

```
// Предаване по стойност
void f(int x)
{
    x = 100;
}

// Предаване по указател
void g(int* p)
{
    *p = 100;
}

int main()
{
    int var = 1;
    std::cout << var << std::endl;

    f(var);
    std::cout << var << std::endl;

    g(&var);
    std::cout << var << std::endl;
}
```

Константни указатели и указатели към КОНСТАНТИ

```
int var = 100, anotherVar = 100;           // Тези указатели се инициализират
                                           // още при създаването им и по-късно не
                                           // могат да сочат към друга променлива
// Тези указатели може да се пренасочат
// впоследствие
int *p1 = &var;
const int *p2a = &var;
int const *p2b = &var; // еквивалентно

p1 = &anotherVar; // OK
p2a = &anotherVar; // OK
p2b = &anotherVar; // OK
*p1 = 1000; // OK
*p2a = 1000; // грешка!

int * const p3 = &var;
const int * const p4 = &var;

p3 = &anotherVar // грешка!
p4 = &anotherVar // грешка!

*p3 = 222222222;
*p4 = 2222; // грешка
```