

Увод в програмирането

Типове указател и псевдоним

2017-2018 г.

ФМИ, специалност „Софтуерно инженерство“

Тип указател ...

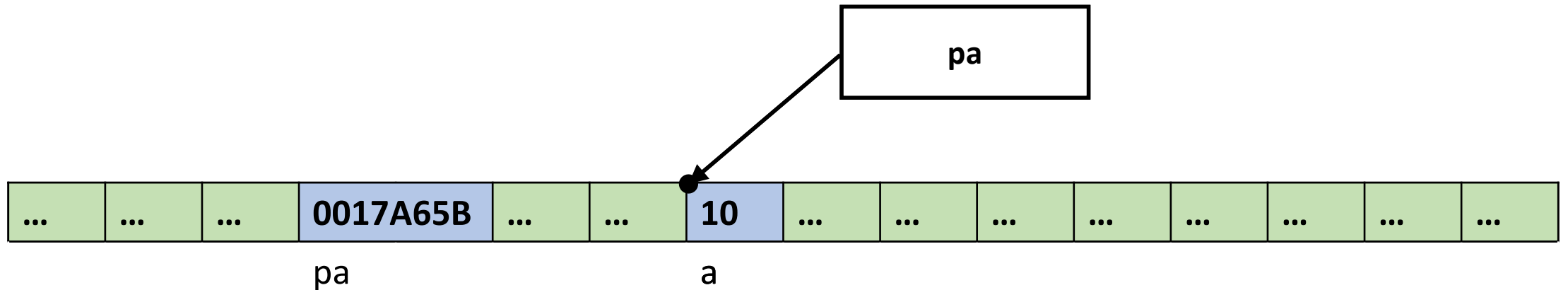
T* pa;

- Дефинира указател с име **pa** от тип **T**
- Указателят практически е (не)обикновена променлива, чиято стойност се интерпретира като адрес в паметта
- **T** определя типа на данните, които указателят адресира, а също и начина на интерпретацията им.

Указател

- Променлива, стойността на която се интерпретира като адрес в паметта на някакъв друг обект

```
int a = 10;  
int *pa = &a;
```

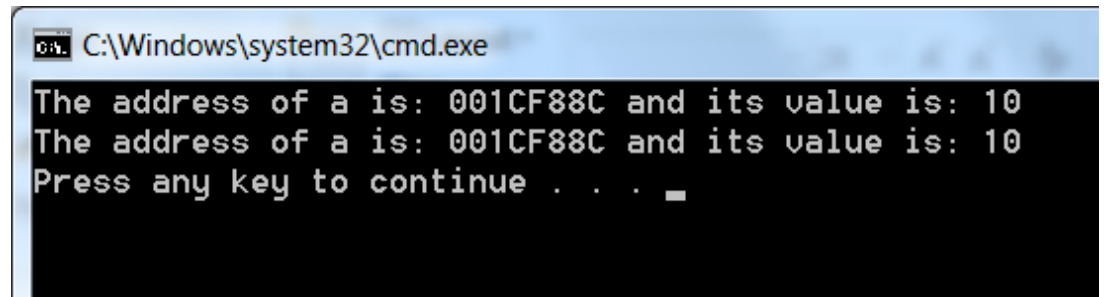


Указатели и адреси

```
int main()
{
    int a = 10;
    int *pa = &a;

    cout << "The address of a is: " << pa
         << " and its value is: " << *pa << endl;
    cout << "The address of a is: "
         << &a << " and its value is: " << a << endl;

    return 0;
}
```



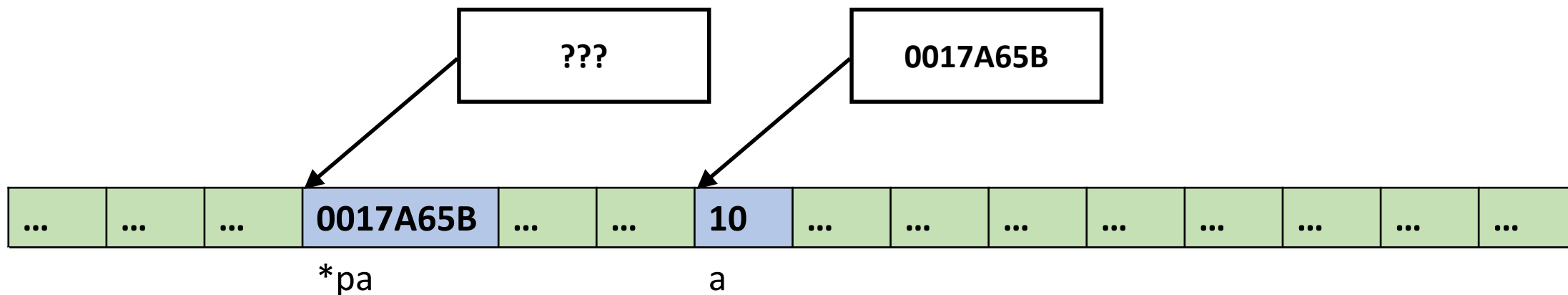
The screenshot shows a Windows command prompt window with the title bar "C:\Windows\system32\cmd.exe". The window contains the following text:

```
The address of a is: 001CF88C and its value is: 10
The address of a is: 001CF88C and its value is: 10
Press any key to continue . . .
```

Указател

- Променлива, стойността на която се интерпретира като адрес в паметта на някакъв друг обект

```
int a = 10;  
int *pa = &a;
```



```
int main()
```

```
{
```

```
    int a = 10;
```

```
    int *pa = &a;
```

```
    int **ppa = &pa;
```

```
    cout << "The address of    a is: "  
          << *ppa << " and its value is: "  
          << **ppa << endl << endl;
```

```
    cout << "The address of   pa is: " << ppa  
          << " and its value is: " << pa << endl << endl;
```

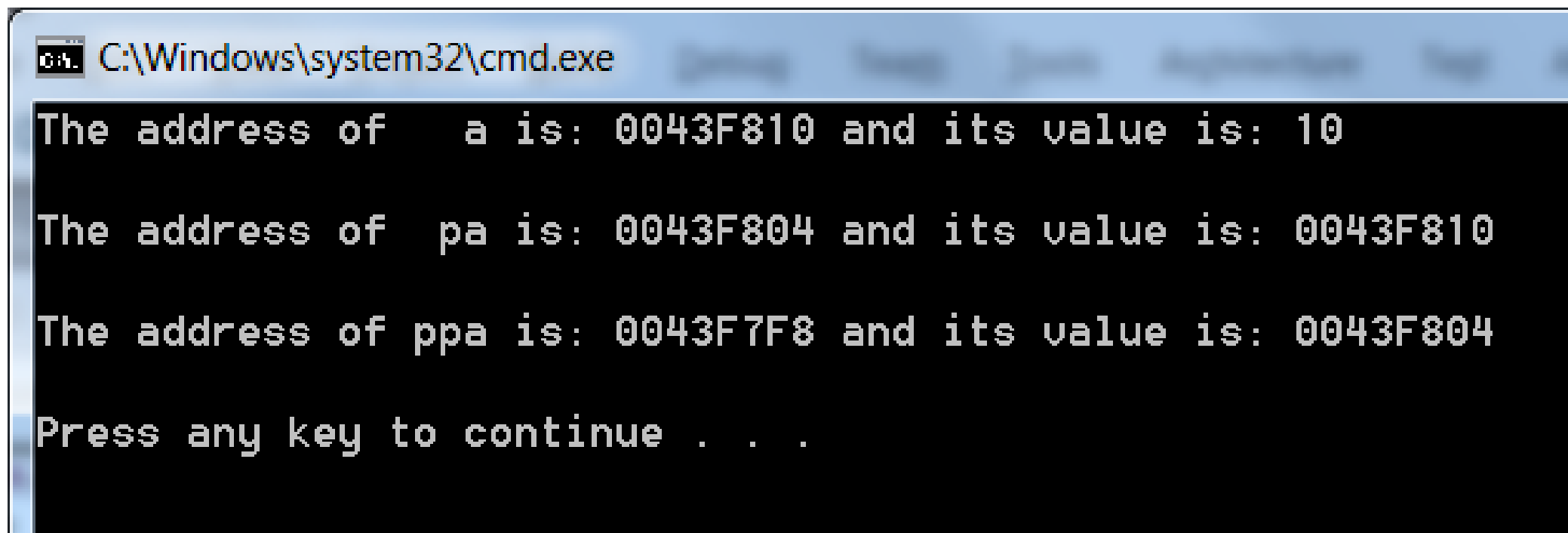
```
    cout << "The address of ppa is: " << &ppa  
          << " and its value is: " << ppa << endl << endl;
```

```
    return 0;
```

```
}
```

Указател към
указател

Указател към указател



```
C:\Windows\system32\cmd.exe

The address of  a is: 0043F810 and its value is: 10

The address of  pa is: 0043F804 and its value is: 0043F810

The address of ppa is: 0043F7F8 and its value is: 0043F804

Press any key to continue . . .
```

The screenshot shows a Windows command prompt window with the title bar "C:\Windows\system32\cmd.exe". The window contains three lines of output, each showing the memory address and value of a variable. The first line shows "The address of a is: 0043F810 and its value is: 10". The second line shows "The address of pa is: 0043F804 and its value is: 0043F810". The third line shows "The address of ppa is: 0043F7F8 and its value is: 0043F804". The prompt is followed by "Press any key to continue . . .".

Типове указатели

```
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;
```

```
// 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;
```

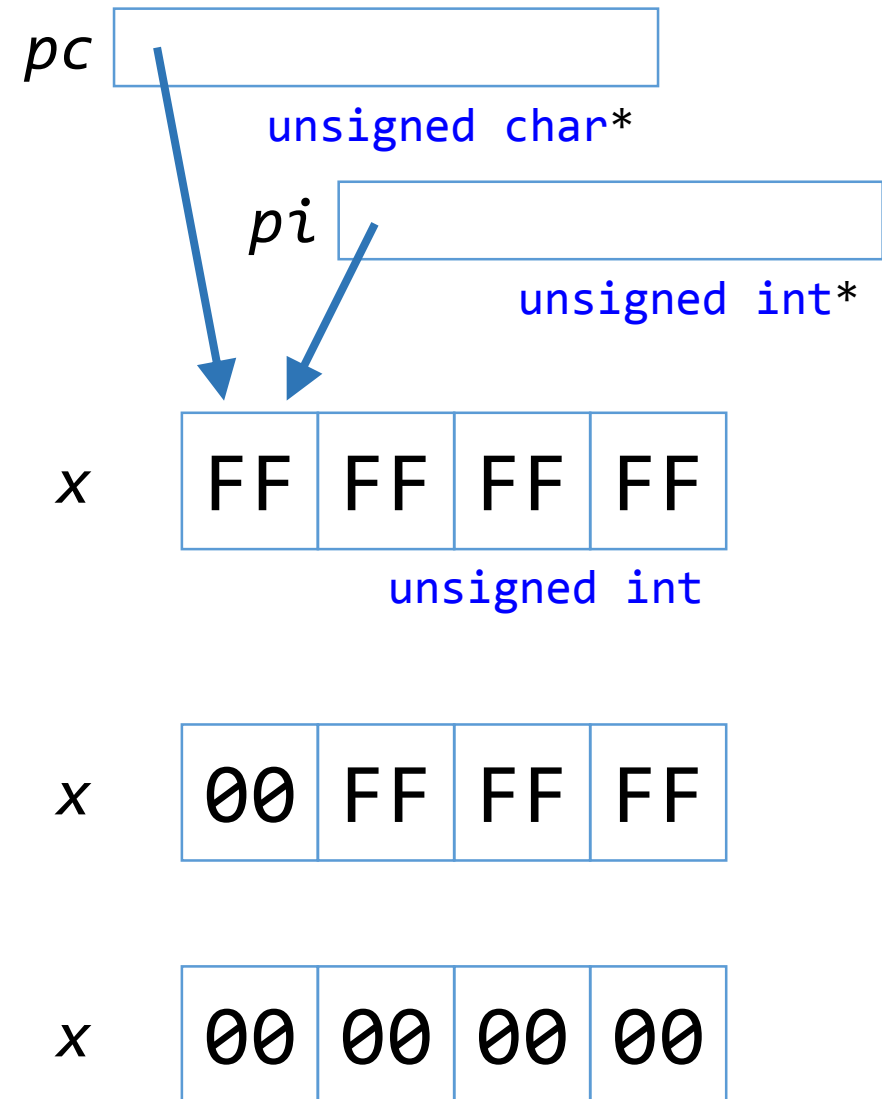
Работа с указателями

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

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

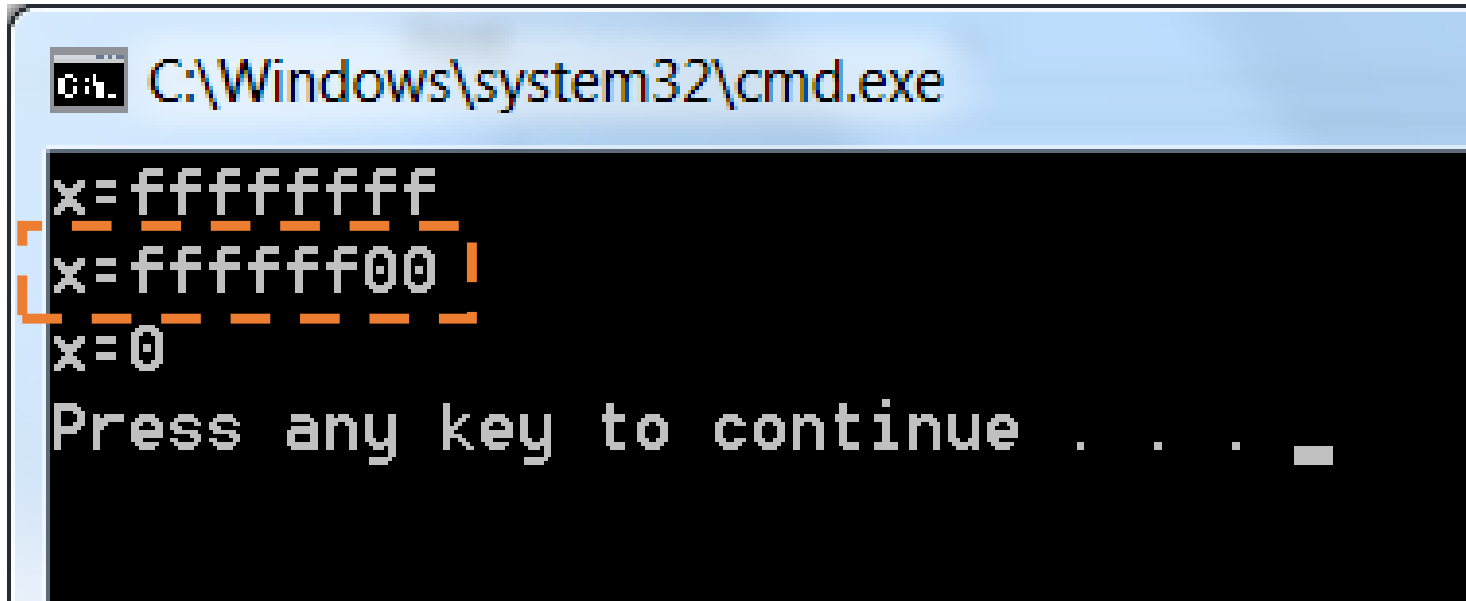
```
*pc = 0;  
cout << "x=" << hex << x << endl;
```

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



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

Little endian vs Big endian



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

The screenshot shows a Windows command prompt window with a blue title bar. The title bar text is "cmd C:\Windows\system32\cmd.exe". The command prompt has a black background with white text. The user has entered three lines of code: "x=ffffffff", "x=ffffff00", and "x=0". The second line is highlighted with a dashed orange box. The prompt "Press any key to continue . . ." is displayed at the bottom.

x

00	FF	FF	FF
----	----	----	----

Инициализиране на указателите!!!!

// p1 не е инициализиран

int *p1;

*p1 = 500; // Какво ли ще се случи?

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

error C4700: uninitialized local variable 'p1' used

```
int *p1 = 0;  
int *p2 = NULL;
```

```
// 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 = NULL; // Ако вече не ни трябва, нулираме указателя
```

void указател

- Използва се, когато е важна стойността на променливата от тип указател (т.е. адресът на данните към които сочи), а не нейното съдържание.
- Този случай е предвиден с цел една и съща променлива - указател да може в различни моменти да сочи към данни от различен тип.
- При опит да се използва съдържанието на променливата от тип указател, ще се предизвика грешка.
- Съдържанието на променлива - указател към тип `void` може да се извлече само след преобразуване на типа на указателя (`void*`) до типа на съдържанието.

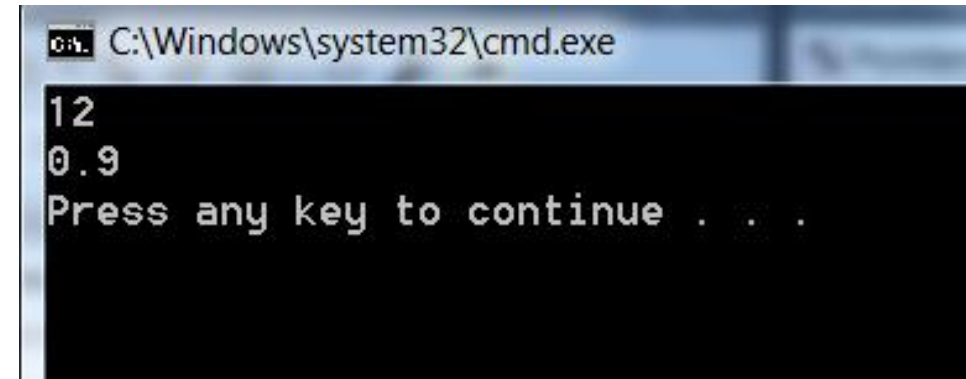
```
int main()
{
    long a = 12;
    double d = 0.9;

    void *pv;

    pv = &a;
    cout << *(int*)pv << endl;

    pv = &d;
    cout << *(double*)pv << endl;

    return 0;
}
```



A screenshot of a Windows command prompt window. The title bar shows the path "C:\Windows\system32\cmd.exe". The window has a black background with white text. It displays the output of a program: the number "12" on the first line, "0.9" on the second line, and "Press any key to continue . . ." on the third line.

```
C:\Windows\system32\cmd.exe
12
0.9
Press any key to continue . . .
```

Псевдоним (reference)

```
int a;  
int &ra = a;
```

- **ra** е псевдоним (alias, reference) на променливата **a**
- Свързва се с адреса на променливата **a**
- Чрез псевдонимите се задават алтернативни имена на обекти в общия смисъл на думата (променливи, константи и др.).

Псевдоним (reference)

- **Дефиницията на променлива от тип псевдоним задължително е с инициализация – дефинирана променлива от същия тип като на базовия тип на типа псевдоним.**
- Променливата от тип псевдоним не може да получи нова стойност
 - Променя се променливата, към която е псевдонимът

```
int ii = 0;  
int& rr = ii;  
rr++;  
int* pp = &rr;
```

Псевдоним (reference)

- **&**<име> —указател към променливата <име>
- *****<указател> —мястото в паметта, сочено от <указател>
- ***** и **&** всъщност са операции в езика с висок приоритет
 - Дуални са една на друга и се унищожават взаимно
- Операторът **&** не може да се прилага върху константи и изрази
 - **&100** и **&(i+5)** са недопустими обръщения

Предаване на аргументи на функция по адрес

```
void swap(int* p, int* q) {  
    int tmp = *p;  
    *p = *q;  
    *q = tmp;  
}  
  
int main() {  
    int a = 5, b = 8;  
    swap(&a, &b);  
    cout << a << ' ' << b << endl;  
}
```

```
void swap(int& x, int& y) {  
    int tmp = x;  
    x = y;  
    y = tmp;  
}  
  
int main() {  
    int a = 5, b = 8;  
    swap(a, b);  
    cout << a << ' ' << b << endl;  
}
```

Тип указател и константи

- В дефиницията:

T* **const** <идентификатор>;

<идентификатор> е константен указател към тип T и не може да бъде променяна стойността му.

- В дефиницията:

const **T*** <идентификатор>;

<идентификатор> е указател към константа от тип T и не може да бъде променяно съдържанието му.

Константни указатели и указатели към константи

```
int var = 100, anotherVar = 100;
```

```
// 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!
```

```
// Тези указатели се инициализират
```

```
// още при създаването им и по-късно не
```

```
// могат да сочат към друга променлива
```

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

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

```
p3 = &anotherVar // грешка!
```

```
p4 = &anotherVar // грешка!
```

```
*p3 = 222222222;
```

```
*p4 = 2222; // грешка
```

Адресна аритметика

- Допустими операции с указателите
 - рефериране (<lvalue>)
 - дерефериране (*<указател>)
 - указателна аритметика (+,-,+=,-=,++,--)
 - сравнение (==, !=, <, >, <=, >=)
 - изход (<<)
- няма вход! (>>)

Типове от данни

- По-рано през семестъра въведохме типове данни за числа и символи. Елементите на тези структури се състоят от една компонента и се наричат **прости**, или **скаларни**.
- Структури от данни, компонентите на които са редици от елементи, се наричат **съставни**.
- Структури от данни, за които операциите добавяне и изтриване на елемент не са допустими, се наричат **статични**, в противен случай - **динамични**.

Структура от данни масив ...

- **Физическо представяне**

Елементите на масива се записват последователно в паметта на компютъра, като за всеки елемент на редицата се отделя определено количество памет.

В езика C++ структурата масив се реализира чрез типа масив

Тип масив ...

- *Примери:*

- `Int a[5]`

- дефинира масив от 5 елемента от тип `int`, индексирани от 0 до 4;

- `Double d[10]`

- дефинира масив от 10 елемента от тип `double`, индексирани от 0 до 9;

- `Bool b[4]`

- дефинира масив от 4 елемента от тип `bool`, индексирани от 0 до 3.

```
int x = 1;  
int array[10]; // int[10]
```

```
array[0] = 100; // Първата клетка  
array[1] = 100; // Втората клетка  
array[9] = 100; // Последната клетка
```

```
// Името на масива е указател към първия му елемент  
*array = 100; // Първата клетка, екв. на *(array+0)  
*(array+1) = 100; // Втората клетка  
*(array+9) = 100; // Последната клетка
```

```
void PrintArray(double arr[5])
{
    for (int i = 0; i < 5; i++)
    {
        std::cout << arr[i] << std::endl;
    }
}

void main()
{
    double data[5] = { 1., 2., 3., 4., 5. };

    PrintArray(data);
}
```

```
void PrintArray(double arr[5])  
{  
    for (int i = 0; i < 5; i++)  
    {  
        std::cout << arr[i] << std::endl;  
    }  
}
```

```
void main()  
{  
    double data[3] = { 1., 2., 3. };  
    PrintArray(data);  
}
```

```
void PrintArray(double arr[], int Size)
{
    for (int i = 0; i < Size; i++)
    {
        std::cout << arr[i] << std::endl;
    }
}

void main()
{
    double dataA[3] = { 1., 2., 3. };
    double dataB[5] = { 1., 2., 3., 4., 5. };

    PrintArray(dataA, 3);
    PrintArray(dataB, 5);
}
```

- За подготовката на тази презентация са използвани слайдове на:
 - Доц. Александър Григоров
 - Доц. Атанас Семерджиев
 - Доц. Трифон Трифонов