Adresy a ukazatele

Základy programování 2

Mgr. Markéta Trnečková, Ph.D.

Palacký University, Olomouc

Ukazatele

- **paměť** = paměťové buňky 1 byte
- **adresa paměti** = ukazatel (pointer)
- typ pointeru
- `typ *jmeno = init;`
- **operátory**: `*, &`
- **operátor adresy &**
- **operátor dereference ***
- `printf("%p", pointer);`
- **nulový ukazatel** = NULL

Ukazatele

Example

```
int a = 3, b, *ptr1, *ptr2;

/* ukazatel ptr1 ukazuje na promennou a  */
ptr1 = &a;

/* hodnota b je 5 (*ptr1 je rovna 3) */
b = *ptr1 + 2;

/* ptr1 a ptr2 ukazuji na stejne misto */
ptr2 = ptr1;

/* zmenime hodnotu na miste, kam ukazuje ptr2 */
*ptr2 = 5;

/* hodnota b bude 8 */
b = a + 3;
```

Scanf

Example

```
int a;

scanf("%i", &a);
```

Co se zde děje?

Pointerová aritmetika

- **statické pole**: `int pole[] = 1,2,3`
- **konstantní pointer** - nelze mu přiřadit jinou hodnotu

Pointerová aritmetika

- K adrese lze přičíst nebo od ní odečíst nezáporné celé číslo
- Lze spočítat rozdíl adres stejného typu
- Adresy lze porovnávat stejně jako čísla

Přístup k prvkům pole

Example

```
int pole[] = {1, 2, 3};

/* pole[0] = 5 */
*pole = 5;

/* pole[1] = 5 */
*(pole + 1) = 5;

/*  chyba: pole[0] + 1 = 2 */
*pole + 1 = 5;
```

Napište program, který pomocí pointerové aritmetiky vypíše všechny prvky pole.

Přístup k prvkům pole

- Mějme pole: `int pole[]= {1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };`
- Jak zjistíte adresu 4. prvku pole?
- Máme ukazatel, který ukazuje na některý prvek pole pole. Jak zjistíte index tohoto prvku v poli?
- Máme 2 ukazatele do stejného pole. Jak zjistíme, který z ukazatelů ukazuje na prvek, který je v poli dříve?

Příklad převrácení pořadí prvků pole

Example

```
void otoc_pole(int *p, int velikost)
{
    int i;
    int foo;
    for(i=0; i<velikost/2; i++)
    {
        foo = *(p + i);
        *(p + i) = *(p + velikost - 1 - i);
        *(p + velikost - 1 - i) = foo;
    }
}
```

Cvičení

1. Naprogramujte předchozí funkci bez pointerové aritmetiky (přístup k prvkům pole přes hranaté závorky) a srovnejte dobu běhu těchto dvou programů.
2. Bez použití funkce sizeof zjistěte, jak velkou část paměti zabírají typy char, int a double.
3. Vytvořte libovolnou strukturu a zjistěte, jak velkou část paměti tato struktura zabírá. Bez použití funkce sizeof. Vyzkoušejte pro různé datové typy položek struktury.
4. Naprogramujte funkci `void vypis(int *pole, int zacatek, int krok, int konec)`, která vypíše prvky pole od indexu zacatek po index konec s krokem krok. Například pro pole = {1,2,3,4,5,6,7,8,9,10} a zacatek =0, krok = 2, konec = 9 vypíše prvky 1, 3, 5, 7, 9.