Ukazatele a předávání parametrů

IZP-cv06

Ing. Jakub Husa

Vysoké Učení Technické v Brně, Fakulta informačních technologií
Božetěchova 1/2. 612 66 Brno - Královo Pole
ihusa@fit.vut.cz

6. listopadu 2023

Ukazatele

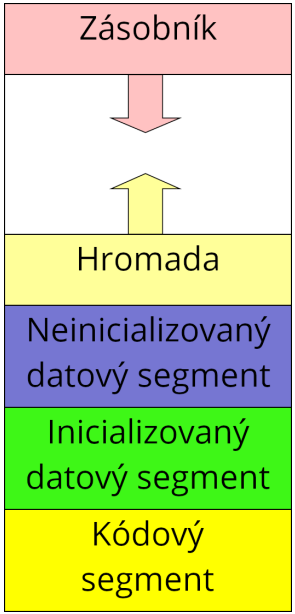
Paměťový prostor

Paměťový prostor programu se skládá z pěti segmentů:

- Zásobník obsahuje automaticky alokované proměnné vytvářené uvnitř funkcí, parametry předávané při volání, a roste směrem shora dolů.
- Hromada obsahuje dynamicky alokované proměnné vytvářené funkcí malloc a roste směrem zdola nahoru.
- Neinicializovaný datový segment obsahuje globální proměnné bez počáteční hodnoty.
- Inicializovaný datový segment obsahuje globální proměnné s počáteční hodnotou.
- Kódový segment obsahuje a zdrojový kód programu, a za běhu programu do něj nelze zapisovat.

Operační paměť počítače má omezenou velikost:

- Když se zásobník s hromadou potkají program havaruje!
- Více si spolu vysvětlíme příští semestr v předmětu ISU.



Velikost datových typů

Různé datové typy zabírají v paměti počítače různé množství místa:

- Velikost zjistíme příkazem – `sizeof` – který vrací velikost v bajtech (byte).
- Velikost datových typů **NENÍ** pevně definována a závisí na operačním systému.

```
1 printf("int      = %i\n", sizeof(int));      //cele cislo
2 printf("float    = %i\n", sizeof(float));    //desetinne cislo
3 printf("double   = %i\n", sizeof(double));   //s dvojnasobnou presnosti
4 printf("char     = %i\n", sizeof(char));     //znak
```

Položky pole jsou v paměti ukládány jako jeden souvislý blok dat:

- Velikost pole je tedy násobkem počtu a velikosti jeho položek.

Položky struktury jsou v paměti automaticky zarovnávány na velikost slova:

- Velikost struktury tedy může být větší než je suma velikostí jejích položek.

Veškerá data uložená v paměti počítače mají nějakou adresu:

- Adresu proměnné získáme operátorem reference (&).

```
1 int x;                //promenna typu "cele cislo" jmenem "x"
2 scanf("%i", &x);      //na adresu promenne "x" nacitame vstup
```

Adresy ukládáme do proměnných datového typu ukazatel (pointer), který vytvoříme přidáním hvězdičky (*) za jméno nějakého datového typu:

- Hodnotu z adresy získáme operátorem dereference (*).

```
3 int y = 10;           //"cele cislo" jmenem "y" s hodnotou 10
4 int* z;               //"ukazatel na cele cislo" jmenem "z"
5 z = &y;               //do "z" ukladame adresu promenne "y"
6 printf("%i\n", y);   //vypis hodnotu "y" (cislo 10)
7 printf("%i\n", z);   //vypis hodnotu "z" (adresa "y")
8 printf("%i\n", *z); //vypis hodnotu ulozenou na adrese "z" (cislo 10)
```

Ukazatel na ukazatel

Ukazatel může ukazovat i na další ukazatel:

```
1  int a = 10;           //cele cislo 10
2  int* b = &a;          //ukazatel na cele cislo 10
3  int** c = &b;         //ukazatel na ukazatel na cele cislo 10
4  int*** d = &c;        //ukazatel na ukazatel na ukazatel na cele cislo 10
5  printf("%i", ***d); //vypis hodnotu z adresy z adresy z adresy "d" (10)
```

Vyzkoušejte si:

- Vytvořte si pole celých čísel, a vypište jeho adresu a velikost.
- Vypište index, adresu a velikost všech položek pole.
- Vytvořte si datový typ xxx pro strukturu obsahující znak, znak a celé číslo.
- Vytvořte si datový typ yyy pro strukturu obsahující znak, celé číslo a znak.
- Vypište velikost obou těchto typů.

Například:

- Adresa celeho pole je 6422256, velikost je 12 bajtu
  Adresa polozky 0 je 6422256, velikost je 4 bajtu
  Adresa polozky 1 je 6422260, velikost je 4 bajtu
  Adresa polozky 2 je 6422264, velikost je 4 bajtu
- Velikost typu xxx je 8 bajtu
  Velikost typu yyy je 12 bajtu

Předávání parametrů

Předávání parametrů hodnotou

Při volání funkce se hodnoty předávaných parametrů kopírují:

- Pokud hodnotu předaných parametrů ve volané funkci změníme originální hodnoty ve volající funkci zůstanou stejné.

```
1  void zdvojnasob(int a);     //deklarace funkce (parametrem je hodnota)
2
3  int main()                  //zacatek programu
4  {
5      int x = 10;             //cele cislo "x" s hodnotou 10
6      zdvojnasob(x);          //volani funkce (predavani hodnoty)
7      printf("x = %i\n", x); //vypis hodnoty promenne "x" (stale 10)
8      return 0;               //konec programu
9  }
10
11 void zdvojnasob(int a)      //definice funkce (parametrem je hodnota)
12 {
13     a = a*2;                //zmena hodnoty parametru (kopie)
14     return;                 //konec funkce
15 }
```

Předávání parametrů odkazem

Volané funkci můžeme místo hodnoty předat adresu nějaké proměnné:

- Kopie adresy ukazuje na původní data, a pokud je ve volané funkci změníme originální hodnoty ve volající funkci se tím změní také.

```
1  void rozpul(int* b);          //deklarace funkce (parametrem je ukazatel)
2
3  int main()                    //zacatek programu
4  {
5      int y = 10;               //cele cislo "y" s hodnotou 10
6      rozpul(&y);               //volani funkce (predavani adresy)
7      printf("y = %i\n", y);    //vypis hodnoty promenne "y" (jen 5)
8      return 0;                 //konec programu
9  }
10
11 void rozpul(int* b)           //definice funkce (parametrem je ukazatel)
12 {
13     *b = *b / 2;              //zmena hodnoty na adrese (original)
14     return;                   //konec funkce
15 }
```

Vyzkoušejte si:

- Vytvořte si dvě celá čísla (X a Y) a pole celých čísel (Z).
- Napište funkci vymen která vymění hodnoty dvou čísel předaných odkazem.
- Napište funkci preskladej která prvky pole čísel přeskládá do opačného pořadí.
- Funkce jsou deklarovány tímto způsobem:

```
1 void vymen(int* a, int* b);
2 void preskladej(int delka, int pole[]);
```

- Ve funkci main vypište původní a vyměněné hodnoty proměnných X a Y, a původní a přeskládané hodnoty prvků pole Z.

Například:

- (10, 20, 11 22 33 44 55) => X = 10, Y = 20
  => X = 20, Y = 10
  => Z = 11 22 33 44 55
  => Z = 55 44 33 22 11

Předávání polí

Hodnota pole (bez hranatých závorek) je adresou jeho prvního prvku:

- Pole je tedy vždy předáváno odkazem a pokud ve volané funkci změníme hodnotu jeho položek originální hodnoty ve volající funkci se tím změní také.

```
1  void negace(int delka, int pole[]); //deklarace (parametrem je pole)
2
3  int main()                          //zacatek programu
4  {
5      int z[2] = {10, 20};            //pole cisel "z"
6      negace(2, z);                   //volani funkce (predavani pole)
7      for (int i = 0; i < 2; i++)              //vsechny polozky pole
8          printf("pole[%i] = %i\n", i , z[i]); //vypis (-10, -20)
9      return 0;                       //konec programu
10 }
11
12 void negace(int delka, int pole[])  //definice (parametrem je pole)
13 {
14     for (int i = 0; i < delka; i++) //vsechny polozky pole
15         pole[i] *= -1;              //vynasob hodnotou "-1"
16     return;                         //konec funkce
17 }
```

Předávání struktur

Struktury můžeme funkcím předávat hodnotou i odkazem:

- K položkám struktury předané odkazem přistupujeme operátorem šipka (->).
- Protože operátor položka (.) má vyšší prioritu než operátor dereference (*), bez použití šipky bychom přístup k položkám museli závorkovat.

```
1  void inkrementuj(bod* A); //deklarace funkce "inkrementuj"
2
3  int main()                //zacatek programu
4  {
5      bod A = {10.0, 20.0}; //vytvarime "bod" jmenem "A"
6      inkrementuj(&A);      //funkci predavame adresu bodu
7      printf("A.x = %f, A.y = %f\n", A.x, A.y); //vypis
8      return 0;             //konec programu
9  }
10
11 void inkrementuj(bod* A)  //definice funkce "inkrementuj"
12 {                         //parametrem je struktura predana odkazem
13     A->x = A->x + 1;      //pristup pomoci sipky
14     (*A).y = (*A).y + 1;  //pristup pomoci hvezdicky a tecky
15     return;               //konec funkce
16 }
```

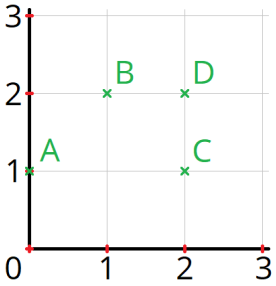
Vyzkoušejte si:

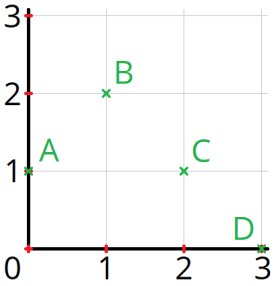
- Deklarujte datový typ bod pro strukturu obsahující dvě desetinná čísla (X a Y).
- Napište funkci jeFunkce která ověří jestli pole bodů (binární relace) představuje matematickou funkci (v poli nejsou žádné dva body se stejným X a rozdílným Y).
- Napište funkci maxFunkce která vrátí hodnotu X bodu s nejvyšším Y.
- Ve funkci main si vytvořte pole čtyř bodů, načtěte jejich souřadnice, ověřte zda jde o funkci, a pokud ano, vypište X bodu s nejvyšším Y.

```
1 bool jeFunkce(int delka, bod pole[]);
2 float maxFunkce(int delka, bod pole[]);
```

Například:

- ((0.0, 1.0), (1.0, 2.0), (2.0, 1.0), (2.0, 2.0))
  => Relace neni funkce
- ((0.0, 1.0), (1.0, 2.0), (2.0, 1.0), (3.0, 0.0))
  => Relace je funkce
  Maximum je pro X = 1.0





Na procvičení

Vyzkoušejte si:

- Napište funkci stred která spočítá geometrický střed množiny bodů (artimetický průměr souřadnic všech jejích prvků).
- Napište funkci nejblizsi která zjistí index nejbližšího prvku z množiny.
- Ve funkci main si vytvořte pole čtyř bodů, načtěte jejich souřadnice, a vypište souřadnice jejího středu a nejbližšího prvku množiny.

```
1 void stred(int delka, bod pole[], bod* S);
2 int nejblizsi(int delka, bod pole[], bod* S);
```

Například:

- ((0.0, 0.0), (1.0, 1.0), (2.0, 2.0), (2.0, 0.0))
  => Souradnice stredu (1.25, 0.75)
  Nejblizsi prvek je (1.00, 1.00)
- ((0.0, 0.0), (1.0, 3.0), (3.0, 2.0), (3.0, 0.0))
  => Souradnice stredu (1.75, 1.25)
  Nejblizsi prvek je (3.00, 2.00)