Ukazatele, předávání parametrů, polí a struktur

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

23. října 2024

Ukazatele a předávání parametrů

Všechna data programu jsou uložena na nějaké adrese v operační paměti počítače:

- Adresu proměnné získáme operátorem reference (&).

```
1  int x;                    //promenna typu "cele cislo" jmenem "x"
2  scanf("%i", &x);          //na adresu promenne "x" nacitame vstup
3  printf("x = %i\n", x);    //vypisujeme hodnotu promenne "x"
```

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

- Hodnotu z adresy získáme operátorem dereference (*).

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

Ukazatel na ukazatel

Cílem ukazatele může být i 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)
```

Předávání parametrů hodnotou a odkazem

Hodnoty předávaných parametrů se do funkce při volání kopírují:

- Pokud hodnotu předaných parametrů ve volané funkci změníme, na původních hodnotách ve volající funkci se změna neprojeví.
- Pokud funkci předáme adresu, její kopie bude ukazovat na původní data, a pokud tato data změníme, ve volající funkci se změna projeví.

- Předávání parametrů hodnotou.

```
1  void foo(int a) //parametrem je
2  {               //cele cislo
3      a = a +10;  //menime pouze
4  }               //hodnotu kopie
5
6  int main()
7  {
8      int x = 10; //cele cislo
9      printf("x = %i\n", x); //10
10     foo(x);     //predavame cislo
11     printf("x = %i\n", x); //10
12     return 0;
13 }
```

- Předávání parametrů odkazem.

```
14 void bar(int*b) //parametrem je
15 {               //ukazatel
16     *b= *b +10; //menime hodnotu
17 }               //puvodnich dat
18
19 int main()
20 {
21     int y = 10; //cele cislo
22     printf("y = %i\n", y); //10
23     bar(&y);    //predavame adresu
24     printf("y = %i\n", y); //20
25     return 0;
26 }
```

Na procvičení

Vyzkoušejte si:

- Vytvořte si pole znaků a ze vstupu do něj načtěte řetězec.
- Napište funkce které vrátí adresu prvního a posledního velkého písmene v načteném řetězci, nebo NULL, pokud řetězec velká písmena neobsahuje.
- Napište funkci která vymění (swap) dva znaky předané odkazem.
- Vyměňte první a poslední velké písmeno řetězce a vypište upravený řetězec.

```
1 char* najdiPrvniVelke(char str[]);
2 char* najdiPosledniVelke(char str[]);
3 void vymenZnaky(char* a, char* b);
```

Například:

- (”AhojSvete”) => ShojAvete
- (”aBcDeFg”) => aFcDeBg
- (”hello”) => hello

```c
char* najdiPrvniVelke(char str[]);        //deklarace funkce
char* najdiPosledniVelke(char str[]);     //deklarace funkce
void vymenZnaky(char* a, char* b);        //deklarace funkce

int main()                                //zacatek programu
{
    char arr[101];                        //vytvarime si pole znaku
    scanf("%100s", arr);                  //ze vstupu nacitame retezec

    char* prvni = najdiPrvniVelke(arr);       //hledame prvni    velke
    char* posledni = najdiPosledniVelke(arr); //hledame posledni velke

    if (prvni!=NULL && posledni!=NULL)    //pokud jsme je nasli
        vymenZnaky (prvni, posledni);     //velka pismena spolu vymenime

    printf("%s\n", arr);                  //vypisujeme upraveny retezec
    return 0;                             //konec programu
}

//TODO definice funkci
```

Předávání polí

Předávání polí

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

- Pole jsou tedy vždy předávána odkazem, a pokud hodnotu jejich prvků ve volané funkci změníme, ve volající funkci se změna vždy projeví.

```
1  void naVelka(char str[])              //parametrem funkce je pole
2  {
3      int delka = strlen(str);          //zjistujeme delku retezce
4      for (int i = 0; i < delka; i++) //vsechny znaky retezce
5      {
6          str[i] = toupper(str[i]);     //prevedeme na velke pismeno
7      }
8  }
9
10 int main()                            //zacatek programu
11 {
12     char arr[6] = "hello";            //pole znaku obsahujici retezec
13     printf("%s\n",arr);               //vypise "hello"
14     naVelka(arr);                     //volani funkce (predavame pole)
15     printf("%s\n",arr);               //vypise "HELLO"
16     return 0;                         //konec programu
17 }
```

Vícerozměrná pole

Pole mohou mít v jazyku C více rozměrů (dimenzí):

- Vícerozměrné pole vytvoříme použitím vícero hranatých závorek ([] []).

```
1  int a;           //cele cislo
2  int b[4];        //1D pole  4 celych cisel (                  4 sloupce)
3  int c[3][4];     //2D pole 12 celych cisel (         3 radky, 4 sloupce)
4  int d[2][3][4]; //3D pole 24 celych cisel (2 patra, 3 radky, 4 sloupce)
```

K prvkům pole přistupujeme vícero indexy a hranatými závorkami ([] []):

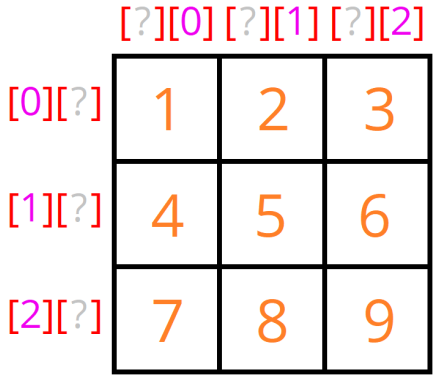
- Pole můžeme inicializovat pomocí vnořených složených závorek ({{}}):

```
5  int arr[2][2] = {{1,2}, {3,4}};   //dvourozmerne pole cisel
6
7  for(int i = 0; i < 2; i++)        //pro kazdy radek
8  {
9      for(int j = 0; j < 2; j++)    //pro kazdy sloupec
10     {
11         printf("%i ", arr[i][j]); //vypis jeden prvek pole
12     }
13     printf("\n");                 //vypis znak konce radku
14 }
```

Vyzkoušejte si:

- Vytvořte si inicializované jednorozměrné pole celých čísel (vektor).
- Vytvořte si inicializované dvourozměrné pole celých čísel (matice).
- Napište funkci vymenCisla která vymění dvě čísla předaná odkazem.
- Napište funkci obratVektor která prvky pole přeskládá do opačného pořadí.
- Napište funkci transponujMatici která provede transpozici matice.

```
1  void vymenCisla(int* a, int* b);
2  void obratVektor(int n, int arr[n]);
3  void transponujMatici(int n, int arr[n][n]);
```



Například:

- (1, 2, 3, 4, 5) => 5 4 3 2 1
- (1, 2, 3) (4, 5, 6) (7, 8, 9) => 1 4 7 / 2 5 8 / 3 6 9

Šablona

```
1  //TODO deklarace funkci
2  int main()                                   //zacatek programu
3  {
4      int vektor[5] = {1,2,3,4,5};             //jedno-rozmerne pole
5      obratVektor(5,vektor);                   //volame funkci
6
7      for (int i = 0; i < 5; i++)              //pro vsechny prvky pole
8          printf("%i ", vektor[i]);            //vypis prvek pole
9      printf("\n");                            //vypis konec radku
10
11     int matice[3][3] = {{1,2,3},{4,5,6},{7,8,9}}; //dvou-rozmerne pole
12     transponujMatici(3,matice);                   //volame funkci
13
14     for (int i = 0; i < 3; i++)              //pro kazdy radek
15     {
16         for (int j = 0; j < 3; j++)          //pro kazdy sloupec
17             printf("%i ", matice[i][j]);     //vypis prvek pole
18         printf("\n");                        //vypis konec radku
19     }
20     return 0;                                //konec programu
21 }
22 //TODO definice funkci
```

Předávání struktur

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  typedef struct Sbod {      //definujeme datovy typ pro dstrukturu "Sbod"
2      float x, y;            //obsahujicci dve desetinna cisla "x" a "y"
3  } bod;                     //jmeno tohoto datoveho typu je "bod"
4
5  void zvojnasob(bod* A)     //parametrem funkce je ukazatel na strukturu
6  {
7      A->x *= 2.0;           //pristup pomoci sipky
8      (*A).y *= 2.0;         //pristup pomoci hvezdicky a tecky
9  }
10
11 int main()                 //zacatek programu
12 {
13     bod A = {10.0, 20.0}; //inicializujeme "bod" jmenem "A"
14     zvojnasob(&A);         //predavame adresu bodu
15     printf("A.x = %f, A.y = %f\n", A.x, A.y); //A.x = 20.0, A.y = 40.0
16     return 0;              //konec programu
17 }
```

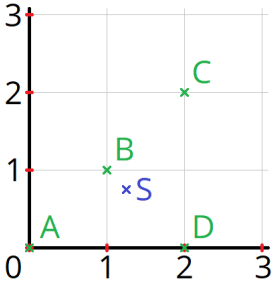
Vyzkoušejte si:

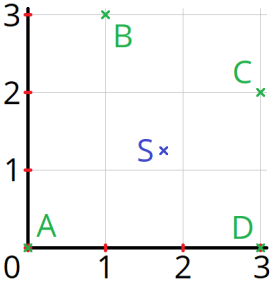
- Napište funkci která spočítá průměrnou vzájemnou vzdálenost bodů množiny.
- Napište funkci která spočítá souřadnice středu množiny bodů.
- Napište funkci která zjistí který z prvků množiny je k jejímu středu nejblíž.
- Vytvořte si pole čtyř bodů, načtěte jejich souřadnice, a vypište jejich průměrnou vzdálenost, souřadnice středu, a souřadnice nejbližšího bodu.

```
1 float prumernaVzdalenost(int delka, bod pole[]);
2 void stredMnoziny(int delka, bod pole[], bod* S);
3 bod* nejblizsiBod(int delka, bod pole[], bod S);
```

Například:

- ((0.0, 0.0), (1.0, 1.0), (2.0, 2.0), (2.0, 0.0))
  => Prumerna vzdalenost je 1.845178
  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))
  => Prumerna vzdalenost je 2.934908
  Souradnice stredu (1.75, 1.25)
  Nejblizsi prvek je (3.00, 2.00)





Šablona

```c
//TODO vytvoreni datoveho typu bod
//TODO deklarace funkci
int main()                                  //zacatek programu
{
    bod M[4];                               //pole ctyr bodu "M" (mnozina)
    for (int i = 0; i < 4; i++)             //pro vsechny prvky pole
        scanf("%f %f", &M[i].x, &M[i].y);   //nacitame souradnice bodu

    float d = prumernaVzdalenost(4, M);     //pocitame vzdalenost
    printf("Prumerna vzdalenost je %f\n", d);

    bod S;                                  //bod "S" (stred)
    stredMnoziny(4, M, &S);                 //pocitame souradnice stredu
    printf("souradnice stredu  (%.2f, %.2f)\n", S.x, S.y);

    bod* N = nejblizsiBod(4, M, S);         //zjistujeme neblizsi bod
    printf("Nejblizsi prvek je (%.2f, %.2f)\n", N->x, N->y);

    return 0;                               //konec programu
}
//TODO definice funkci
```