FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

POUŽITÍ A VYUŽITÍ UKAZATELŮ

Autor textu:

Ing. Miloslav Richter, Ph. D.

Květen 2014

Komplexní inovace studijních programů a zvyšování kvality výuky na FEKT VUT v Brně
OP VK CZ.1.07/2.2.00/28.0193

INVESTICE DO ROZVOJE VZDĚLÁVÁNÍ

Ukazatel jako parametr funkce

- slouží k předání výsledku mimo funkci. Funkce má možnost předat jednu návratovou hodnotu. Další návratové hodnoty je možné předat pomocí ukazatelů v poli parametrů. Ukazatel funkci odkáže na místo, kam má uložit výsledek.
-

Napište funkci, která načítá data z klávesnice / souboru a vrací minimální a maximální hodnotu. Návratová hodnota obsahuje počet hodnot, ze kterých byly určeny.

```
//hlavička říká, že předáváme ukazatele/adresy, na
// kterých jsou hodnoty typu double
int MinMax(double *aMax, double *aMin)
{
 int Minimum, Maximum;
 ...
 *aMax = Maximum;
// na předanou adresu se zapíše nalezená hodnota
*aMin = Minimum; // před přiřazením dojde ke konverzi
 return Pocet; // vrátí se počet hodnot
}

volání:
double mi, ma;
long kolik;

kolik = MinMax( &ma, &mi); //ukazatele musí být na stejný
// typ jako v prototypu fce. návratová hodnota je pro
// přirazení konvertována – nemusí být stejný typ
```

Ukazatel jako návratová hodnota

Který z následujících předávání ukazatele jako návratové hodnoty je možný? A proč?

```
double *
Funkce(double hodnota, double *adresa, double *adresa 2)
{
 double vysledek;

// možné výsledky

return *adresa;
return  adresa;
return  hodnota;
return &hodnota;
return &adresa;
return  vysledek;
return &vysledek;
}
```

```
double *
Funkce(double hodnota, double *adresa, double *adresa2)
{
 double vysledek;

// možné výsledky

return *adresa;  // nesouhlasí typ - double
return adresa;   //typ souhlasí, adresa je předána
                 // „z venku“->existuje, může být vrácena
return hodnota;  // nesouhlasí typ
return &hodnota; //nelze předat „ven“ adresu lokální
                 // proměnné, která zanikne na konci
return &adresa;  // nesouhlasí typ – double **
return vysledek; // nesouhlasí typ - double
return &vysledek;// nelze předat adresu lokální proměnné
}
```

Napište funkci tak, aby vrátila minimum ze tří hodnot typu int. Zároveň je dán požadavek, aby bylo možné tuto proměnnou nahradit daným čísle – tj. aby mohl být nalevo od znaménka rovná se.

```
int *Min3(int *a1, int *a2, int *a3)//předávají se adresy
{
 if ((*a1 < *a2) && (*a1 < *a3)) //porovnávají se hodnoty
     return a1; //  vrací se ukazatel (adresa)
 if (*a2 < *a3)
     return a2;
 return a3;
}
```

volání:

```
int x1=10, x2=20, x3=-10, r;
r = *Min3(&x1,&x2,&x3); // předávají se adresy,
  // vrací se adresa. Přístup dereferencí. Pomocí
  // dereference se dostaneme k hodnotě na vrácené adrese
*Min3(&x2,&x3,&x1) = 10;
  // parametry jinak umístěny, výsledek musí být stejný
  // ve funkci Min3 bude ale jiný průběh při krokování
  // návratovou adresu (místo) je možné použít i k zápisu
r = *Min3(&x1,&x1,&x1);
  //důležitá „ladící dovednost“, všechny parametry stejné
```

pole a/versus pointer

- z uvedených vlastností ukazatelů a typů je výjimka – proměnná typu pole
- pole a ukazatel mají natolik podobný přístup, že pokud je to možné, pole se konvertuje na ukazatel a dále se s nimi manipuluje stejně
- základní vlastností je, že název pole se konvertuje na ukazatel na první prvek pole („ztrácí“ se tím pojem o délce (?), kontrola opuštění mezí není ani v jednom případě)
- další vlastností je možnost „oindexování“ ukazatele (k adrese se přičte hodnota indexregistru – dojde k posunu na jinou adresu).
- díky znalosti typu ukazatele a tím i velikosti daného typu je krokem posunu v poli (i adresy ukazatele) vzdálenost rovná rozměru daného typu -> posunujeme se po prvcích pole
- první prvek má offset nula, druhý jedna ...
- index může být i záporný
- ukazatel je možné měnit (posunout), „hodnota“ pole je konstanta

Přístup k prvkům pole

- pro přístup k prvkům pole slouží unární operátor [ ] – parametrem je celočíselná hodnota udávající prvek od počátku (vztažná adresa)
- další možností je využití tzv. ukazatelové aritmetiky. Přičte-li (odečte-li) se k ukazateli celé číslo, výsledkem je adresa tolikátého prvku pole.

```
int Pole[10]={0,1,2,3,4,5,6,7,8,9};
   // pole o deseti prvcích, indexy 0-9
int *pi = NULL; // pomocný ukazatel na int

pi = Pole; // díky konverzi ukazuje nyní pi na první
   // (nultý) prvek Pole
Pole[2] = -2;
   // přístup ke třetímu prvku pomocí operátoru [ ] –
   // pracuje s hodnotou na dané pozici
*(Pole + 2) = -2; // ekvivalentní zápis
   //– posun ukazatele o dvě „délky“ typu int
   // a přístup k dané adrese.
   // Závorky () jsou nutné kvůli větší prioritě * před +

*(pi + 2) = -2;
pi[2]=-2;  // ekvivalent předchozího
// přístup pomocí ukazatele je stejný.
// Nedochází ke konverzi z typu pole
```

S ukazateli je spojen pojem *ukazatelová aritmetika*

- přičteme-li (odečteme) k ukazateli celé číslo, „posune“ se adresa o daný počet prvků typu, na který ukazatel ukazuje (tj. N * sizeof(typ pole) bytů)
- přičtením (odečtením) jedničky se dostáváme na další (předchozí) prvek
- odečteme-li dva ukazatele (musí patřit ke stejnému poli/paměťovému celku) získáme počet prvků, které mezi nimi leží
- další matematické operace nejsou pro ukazatele definovány

```
int Pole[20]={0};
int *první, *aktualni,*poslední,i,pocet;
       // tři ukazatele na int a dva inty
int suma=0;
for ( aktualni=první=Pole, i=0; i<20 ; i++, aktualni++ )
          // aktualni ++ je posun ukazatele na další prvek
   suma += *aktualni;  // součet prvků v poli
// vyjádřete totéž pro Pole (typ pole) a prvni (ukazatel)
pocet = aktualni – první; //počet prvků mezi ukazateli.
 // Prvky jsou typu, na který ukazují ukazatele (zde int)
```

- přístup k poli přes ukazatel pro Pole a prvni

  v jazyce C se posouváme v paměti o prvky, ne byty

```
suma += *(první + i);

suma += Pole[i]; // v [] je skrytá * (to je přístup)
```

pole – předávání do funkce a z funkce – může být problém při kombinacích [] a * například pole[] na zásobníku – krátký relativní od zásobníku, ukazatel – dlouhý od počátku, nemusí dojít ke správné konverzi – dlouhý ukazatel na krátký od zásobníku

Zjištění velikosti pole

- pro zjištění velikosti typu se používá sizeof
- pokud je parametrem sizeof pole, vrací velikost pole, pokud je parametrem ukazatel, vrací velikost ukazatele (tj. adresy)

Co se stane v následujícím případě? Předává se pole jako pole nebo pomocí ukazatele?:

```
int Test(int pole[])// stejné s int pole[10] či int *pole
{
  int i;
  pole[0] = 10; // dojde ke změně hodnoty i mimo funkci?
  i = sizeof(pole);  // i nabývá hodnoty  ???
  return i;
}

int  Pole[20]= {1},i;
i = sizeof(Pole); // i nabývá hodnoty=?; Pole[0] nabývá ?
i = Test(Pole); // i nabývá hodnoty ?;
    //Pole[0] nabývá po ukončení funkce hodnotu ?
```