Střední škola pedagogická, hotelnictví a služeb, Litoměřice, příspěvková organizace

Předmět:	**Algoritmizace a programování**
Téma:	**Programování**
Vyučující:	**Ing. Milan Káža**
Třída:	**EK2**
Hodina:	**27**
Číslo:	**V/5**

Programování v jazyce C a C#

11. Pointery

Pointery nebo ukazatel je srdcem i duší jazyka C. Tedy pokud jsou pochopeny pointery tak je možné říci:

„Umím dobře Céčko, protože pracuji s pointery “

Pozor pojme pionter se objevuje v jazyce C od firmy Borland v jazyce C# se tento pojem již nevyskytuje, proto se tato část týká pouze jazyka C.

Pointer je vždy svázán nějakým datovým typem, proto se uvádí: ***pointer na typ.*** Ukazatel (pointer) je v Céčku statická celočíselná proměnná obsahující adresu jiné statické či dynamické proměnné. Nese sebou současně informaci o datovém typu, který se na této adrese nachází. Pointer na proměnnou určitého datového typu se deklaruje pomocí operátoru dereference '*“ (hvězdička) a takto se deklaruje:

datový_typ *identifikátor_pointeru;

Uložení adresy do pointeru se liší podle toho zda pointer ukazuje na statickou či dynamickou proměnnou. Díky operátoru reference '&' si uloží do pointeru adresu statické proměnné:

identifikátor_pointeru=&identifikátor_proměnné;

A takto se do pointeru vloží adresa a vyhradí se prostor pro dynamickou proměnnou:

identifikátor_pointeru=(datový_typ *)malloc(sizeof(datový_typ));

Výše uvedený zápis se dá přeložit následovně:

Funkce **malloc()** je funkce, která alokuje paměťový prostor v haldě. Jejím parametrem je počet bytů potřebný pro alokaci. Ten byl zjištěn operátorem pro zjištění velikosti datového typu **sizeof().** Jelikož funkce **malloc()** vrací ukazatel na datový typ void

(generický pointer na libovolný datový typ) je nutné tento **pointer** přetypovat pomocí operátoru přetypování (datový_typ *). V případě, že se nepodaří požadovanou paměť alokovat, vrací funkce hodnotu NULL.
Pro ověření zda celá akce správně proběhla se provádí tento test:

if(identifikátor_pointeru==NULL) printf("Chyba alokace pameti neprobehla\n");

Pro samotné uložení hodnoty na místo kde ukazuje pointer se používá tento zápis:

***identifikátor_pointeru=hodnota;**

Pointer je možné jako dynamickou proměnnou a po ukončení práce s tímto pointerem se musí uvolněnit pamět, kterou si pointer zabral při práci to se provede tímto příkazem :

free(identifikátor_pointeru);

Aby se pointer neodkazoval na místo, které již neexistuje ta se ukotví tímto příkazem:

identifikátor_pointeru=NULL;

Několik příkladů pro práci s pointery:
Pro všechny platí definice: **int q,*qw;**
Statistické použití pointeru

```
int q, *qw;
   q=3;     do q vloží hodnotu 3
  *qw =4;  na adresu qw(kde je int) vloží hodnotu 4
   q=*qw;   do q  vloží obsah z adresy qw
```

***qw=q;** na adresu qw vloží obsah q
qw=&q; naplní qw adresou q

Dynamické přiřazení pointeru.

```
int q, *qw;
 q=4;
  qw=&qů
*qw=4;
```

před přiřazením hodnoty na adresu v **qw** musí být **qw** inicializován.

Následující program ukáže použití pointerů, program načítá pomocí funkce cti_radku řádky z klávesnice a počítá kolik bylo mezer a malých písmen. Funkce cti_radku vrací hodnotu 1, když na řádce byly nějaké znaky a hodnotu 0, pokud řádka byla prázdná.

*Osobní zkušenost: Tento program je opsán z knihy **Pavel Herout: Učebnice jazyka C**, já jsem si ho osobně vyzkoušel a program funguje, ale zkoušel jsem ho pod systémem MS-DOS. Pochopení pointerů není na jednu přednášku, zde si to skutečně musí každý odprogramovat, aby pochopil sílu pointerů. Až po delší době a studiu jsem sám pointery pochopil, proto jsem volil jednoduchý program, pro jeho přehlednost a pochopitelnost.*

```
#include <stdio.h>
```

```
  int cti_radek(int *p_mezera, int *p_male)
  {
   int c;
   int pocet=0;
    p_mezery=0; *p_male=0;
     while((c=getchar())!'\n') {
       pocet++;
    if(c==' ')
  (*p_mezery)++;
   else if (c>='a' && c<='z')
    (p_male)++ ;
 }
   return ((pocet==0) ? 0:1);
}
  main()
{
  int mezery, male;

   while(cti_radku(&mezery, &male)==1){
    printf(" naradce bylo  %d mezer a %d malych pismen \n", mezery, male);
}
}
```

Kontrolní otázky

1. Vysvětlete pojem pointer
2. Napište program, který bude pracovat s pointery.

Použitá literatura:

Miroslav Virius: C# pro zelenáče, 1.vydání, Neocortex, s.r.o Praha, 2002
Pavel Herout: Učebnice jazyka C, 3.vydání, KOOP České Budějovice, 1997
Miroslav Virius: od C++ k C# ,1.vydání, KOOP České Budějovice, 2002