Dynamicky alokovaná pole

19. prosince 2011

Na minulé přednášce jsme si ukázali, že pomocí ukazatele je možné alokovat v paměti místo pro novou proměnnou. Obdobným způsobem můžeme alokovat místo i pro celé pole. Výsledkem bude **dynamicky alokované pole**, které se od staticky alokovaného pole liší ve dvou hlavních rysech:

- pro stanovení velikosti dynamicky alokovaného pole můžeme použít proměnnou, jejíž hodnotu získáme teprve v průběhu výpočtu,
- paměť obsazená dynamicky alokovaným polem není automaticky uvolňována, např. na konci funkce, ale je nutné ji uvolnit příslušným příkazem.

Alokace jednorozměrného dynamicky alokovaného pole o `n` prvcích tedy může vypadat následovně:

```
int n, *p;
n=10;
p = new int [n];
```

Práce s hodnotami v dynamicky alokovaném poli vypadá zcela stejně jako v případě staticky alokovaného pole. Zde je ukázka uložení naplnění pole v cyklu hodnotami od 1 do 10.

```
for ( i=0; i<n; i++ )
  p[i] = i+1;
```

Ve chvíli, kdy už pole nebudeme potřebovat, je dobré alokovanou paměť uvolnit. K tomu slouží opět příkaz `delete`, ale tentokrát je nutné před názvem pole uvést prázdné hranaté závorky na znamení, že se jedná o pole.

```
delete [] p;
```

V případě vícerozměrných dynamicky alokovaných polí je situace složitější. Problematiku si vysvětlíme na případu dvourozměrného pole. Tehdy se můžeme na jednotlivé řádky dívat jako na jednorozměrná dynamická pole, které každé z nich je reprezentováno ukazatelem na zvolený datový typ. Abychom tedy vytvořili matici o `m` řádcích, potřebujeme nejdříve vytvořit pole ukazatelů. Abychom i toto pole mohli vytvořit jako dynamicky alokované, musíme nejprve vytvořit proměnnou, která bude v tomto případě **ukazatel na ukazatel na datový typ**. Vytvoření dynamicky alokovaného dvourozměrného pole zahrnuje nejprve vytvoření pole ukazatelů a poté vytvoření polí odpovídajících jednotlivým řádkům pole. Dynamická alokace matice o `m` řádcích a `n` sloupcích tedy vypadá následovně:

```
int i, n=10, m=8, **p;
p = new int* [m];
for ( i=0; i<m; i++ )
  p[i] = new int [n];
```

Práce s vícerozměrnými dynamicky alokovanými poli je opět obdobná jako v případě staticky alokovaných polí. Uvolnění paměti je nutné provést také pro jednotlivé ukazatele. Postup je opačný než při alokaci, nejprve se uvolní jednorozměrná pole a pak teprve pole ukazatelů:

```
for ( i=0; i<m; i++ )
  delete [] p[i];
delete [] p;
```

Ačkoliv je alokace vícerozměrných dynamických polí komplikovanější, má ve srovnání se statickou alokací některé výhody:

- při posílání vícerozměrného pole do funkce, není třeba uvádět žádné konstantní rozměry pole a funkce tak může být zcela obecná,
- jelikož se jednotlivé řádky alokují postupně, je možné vytvářet i jiná než obdélníková pole, každý řádek může mít jinou velikost a je tedy možné vytvořit například trojúhelníkovou matici a tím šetřit místo v paměti.