Vícerozměrná pole

Doposud jsme pracovali s poli, jejichž prvky byly základní datové typy. Takovým polím se říká *jednorozměrná*.Pole, jehož prvky jsou jednorozměrná pole, je *dvourozměrné*. Taková pole si můžeme představit jako tabulku. Samozřejmě mohou být pole i vícerozměrná, než jen dvou.

K prvkům dvourozměrných polí přistupujeme pomocí dvou indexů, jeden určuje ve kterém jednorozměrném poli se prvek vyskytuje, druhý index je indexem prvku v tomto poli. Pokud si představíme dvourozměrné pole jako tabulku, tak jeden z indexů by představoval index řádku a druhý sloupce.

1 Definice

Definice vícerozměrného pole je obdobná, jako definice pole jednorozměrného. V hranatých závorkách je potřeba specifikovat velikosti všech dimenzí daného pole. Každá dimenze se píše zvlášť do hranatých závorek.

```
typ jmeno[v1]...[vn]
```

Příklad použití:

```
/* definice dvourozmerneho pole */
int pole[2][3];

/* definice ctyrrozmerneho pole */
int pole2[2][3][4][5];
```

Při definici pole je možné pole rovnou inicializovat obdobně, jako u jednorozměrných polí. I zde se hodnoty píšou do složených závorek. Každý prvek je inicializací pole o jeden rozměr menší, než právě inicializované pole.

```
// jednorozmerne pole
int pole1[4] = {1, 2, 3, 4};

// dvourozmerne
int pole2[2][3] = {{1, 2, 3}, {1, 2, 3}};

// trojrozmerne
int pole3[2][3][4] = {{{1, 2, 3, 4}, {1, 2, 3, 4}, {1, 2, 3, 4}},
                      {{5, 6, 7, 8}, {5, 6, 7, 8}, {5, 6, 7, 8}}};
```

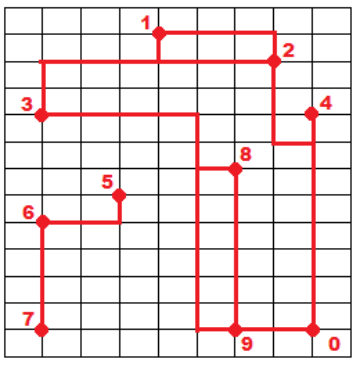
Přístup k jednotlivým prvkům vícerozměrného pole pomocí indexů je naprosto stejný, jako přístup do jednorozměrného pole, jen je potřeba specifikovat potřebný počet indexů.

```
// prvek jednorozmerneho pole pole1
pole1[2] = 3;

// prvek dvourozmerneho pole pole2
pole2[1][1] = 2;

// prvek trojrozmerneho pole pole3
pole3[0][1][2] = 3;
```

2 Příklad



Ukážeme si použití dvourozměrného pole na příkladu. Předpokládejme, že máme na mapě 10 zajímavých míst. Vzdálenosti mezi jednotlivými místy můžeme uchovávat v dvourozměrném poli 10x10. Prvek na indexech `i, j` pak představuje vzdálenost mezi místem `i` a `j`. Pokud je prvek roven -1, pak se z místa `i` nedá dostat do místa `j`.

Cesta průchodem míst na mapě pak může být reprezentovaná výčtem navštívených míst v pořadí, v jakém jsme je navštívili (jednorozměrným polem). Pokud pole bude obsahovat popořadě prvky `1, 2, 4`, tak to znamená, že jsme vycházeli z místa `1`, pak šli do místa `2` a skončili v místě `4`.

Z obrázku je vidět, že třeba vzdálenost mezi místem `0` a `4` je 8. Ale cesta mezi `0` a `7` neexistuje. Prvek na indexech `0 a 7` by tedy byl roven -1.

Následující kód pak představuje funkci pro výpočet délky cesty na mapě. Funkce bere 3 argumenty: pole vzdáleností mezi místy (`p`), pole představující cestu (`cesta`) a počet navštívených míst (`pocet`), přestavující velikost pole `cesta`.

Funkce vrátí délku cesty, nebo -1 pokud taková cesta neexistuje.

```
int delka(int p[][10], int cesta[], int pocet)
{
    int r=0;
    int i;

    /*z mista cesta[i-1] do cesta[i] */
    for(i=1;i<pocet;i++)
    {
        int m1 = cesta[i-1];
        int m2 = cesta[i];

        if(p[m1][m2]==-1)
            return -1;

        r+=p[m1][m2];
    }
    return r;
}
```

U argumentu `p` je potřeba specifikovat velikost druhé dimenze pole. Pokud je vícerozměrné pole argumentem funkce, můžeme vynechat pouze velikost první dimenze. Souvisí to s reprezentací vícerozměrných polí v paměti (jsou reprezentovaná jako jednorozměrná).

3 Reprezentace v paměti

Dvourozměrné pole je uloženo v paměti po řádcích. Tj. pro pole

```
int x[2][3];
```

se v paměti alokuje `2*3*sizeof(int)` bytů.

Předpokládejme, že `sizeof(int) = 2`, pak by uložení v paměti mohlo vypadat následovně:

adresa	100	102	104	106	108	110	112
prvek	x[0][0]	x[0][1]	x[0][2]	x[1][0]	x[1][1]	x[1][2]	volno

Pro efektivní práci s vícerozměrnými poli je dobré si uvědomit, co reprezentují proměnné `x, x[0] a x[0][0]`. Všechny tyto výrazy přísluší stejnému místu v paměti. Uvědomme si, že dvourozměrné pole je jednorozměrné pole, jehož prvky jsou pointery. Tedy prvek `x[0]` je ukazatel na první "řádek"dvourozměrného pole `x`. Jinými slovy, typ jednorozměrného pole `x` je tříprvkové pole prvků typu `int`.

Tedy:

adresa	100	102	104	106	108	110	112
prvek	x[0][0]	x[0][1]	x[0][2]	x[1][0]	x[1][1]	x[1][2]	volno
	x[0]			x[1]			
	x						

`x` a `x[0]` sice představují tutéž adresu, ale jsou to pointery jiných typů. Použijeme-li pointerovou aritmetiku, tak

`x + 1` reprezentuje adresu 106

`x[0] + 1` reprezentuje adresu 102

4 Alokace

Alokace a dealokace na zásobníku se děje automaticky (jak jsem říkala na dřívější přednášce).

O alokaci a dealokaci na heapu se stará programátor. Pří alokaci postupujeme analogicky, jako při alokaci jednorozměrných polí. Dvourozměrná pole jsou pole, jejichž prvky jsou jiná jednorozměrná pole.

Postup při alokaci dvourozměrného pole o rozměrech `m,n`:

1. Alokujeme pole pointerů o `m` prvcích.
2. Alokujeme m jednorozměrných polí o `n` prvcích, přičemž pointery uložíme do pole alokovaného v prvním kroku.

Konkrétní příklad je v následujícím kódu.

```c
int i;

/* 1. krok */
int **pole2d=malloc(m*sizeof(int*));

/* 2. krok */
for(i=0;i<m;i++)
    pole2d[i]=malloc(n*sizeof(int));
```

Pointer může ukazovat na další pointer. Například pole2d ukazuje na pointer, který ukazuje na prvek typu `int`. Při deklaraci takového pointeru se používají dvě `*`. U vícenásobných pointerů se používá více `*`.

Při dealokaci je potřeba postup obrátit:

1. Dealokujeme `m` jednorozměrných polí.
2. Dealokujeme pole pointerů.

```c
int i;

/* 1. krok */
for(i=0;i<m;i++)
    free(pole2d[i]);

/* 2. krok */
free(pole2d);
```

5 Jiný způsob reprezentace vícerozměrných polí

Princip si ukážeme na dvourozměrných polích. Příklad si můžeme vzít z toho, jak je dvourozměrné pole uloženo v paměti. Vezmeme jednotlivé řádky vícerozměrného pole a naskládáme je za sebe. Dvourozměrné pole s rozměry `m, n`, reprezentujeme jednorozměrným polem o `m*n` prvcích. Prvek na indexu `i,j` se v tomto poli nachází na indexu `i*n + j`.

dvourozmerne pole	x[0][0]	x[0][1]	x[0][2]	x[1][0]	x[1][1]	x[1][2]
jednorozmerne pole	y[0*3 + 0]	y[0*3 + 1]	y[0*3 + 2]	y[1*3 + 0]	y[1*3 + 1]	y[1*3 + 2]

Toto pole je jednodušší na alokaci a dealokaci, ale hůře se s ním pracuje, než s dvourozměrným polem.

Příklad práce s tímto polem:

```c
/* alokace */
int *pole2d = malloc(m*n*sizeof(int));

/* pristup k prvku na indexech i,j */
pole2d[i*n + j] = 12;

/* dealokace */
free(pole2d);
```

6 Cvičení

1. Vytvořte funkci, beroucí celočíselné argumenty m a n, která alokuje a vrátí dvourozměrné pole o velikosti $m \times n$. Prvky pole budou reprezentovat násobky. Tedy prvek pole na indexu i, j bude roven $i \cdot j$.

   Ve funkci main vypište toto pole.

2. Přepište předchozí funkci s použitím reprezentace jednorozměrným polem.

3. Pomocí dvourozměrného pole lze reprezentovat hrací pole při piškvorkách (prázdné políčko = 0, křížek = 1, kolečko = 2). Napište funkci, která prohledá toto dvourozměrné pole a vrátí nejdelší souvislou posloupnost křížků nebo koleček

   (a) na řádku

   (b) ve sloupci

   (c) diagonálně

4. Naprogramujte hru piškvorky pro dva hráče.

   Dokud v poli nebude posloupnost 5 stejných znaků (využijte předchozí funkci) se budou hráči střídat a umisťovat svůj znak do pole (na střídačku budou hráči vyzváni, aby zadali 2 souřadnice, kam chtějí umístit znak).