Dynamická pole

Jiří Zacpal

DEPARTMENT OF COMPUTER SCIENCE
PALACKÝ UNIVERSITY, OLOMOUC

KMI/ZP2 Základy programování 2

Statické dvourozměrné pole – připomenutí

- příklad deklarace:

```
int poleA[2][3];
```

- nejjednodušší způsob vytvoření pole
- rozměry pole jsou dány již při kompilaci
- pole je vždy „pravoúhlé“
- prvky jsou v paměti uloženy za sebou uspořádané po jednotlivých řádcích
- pole je statické

Dvourozměrné pole jako pole ukazatelů

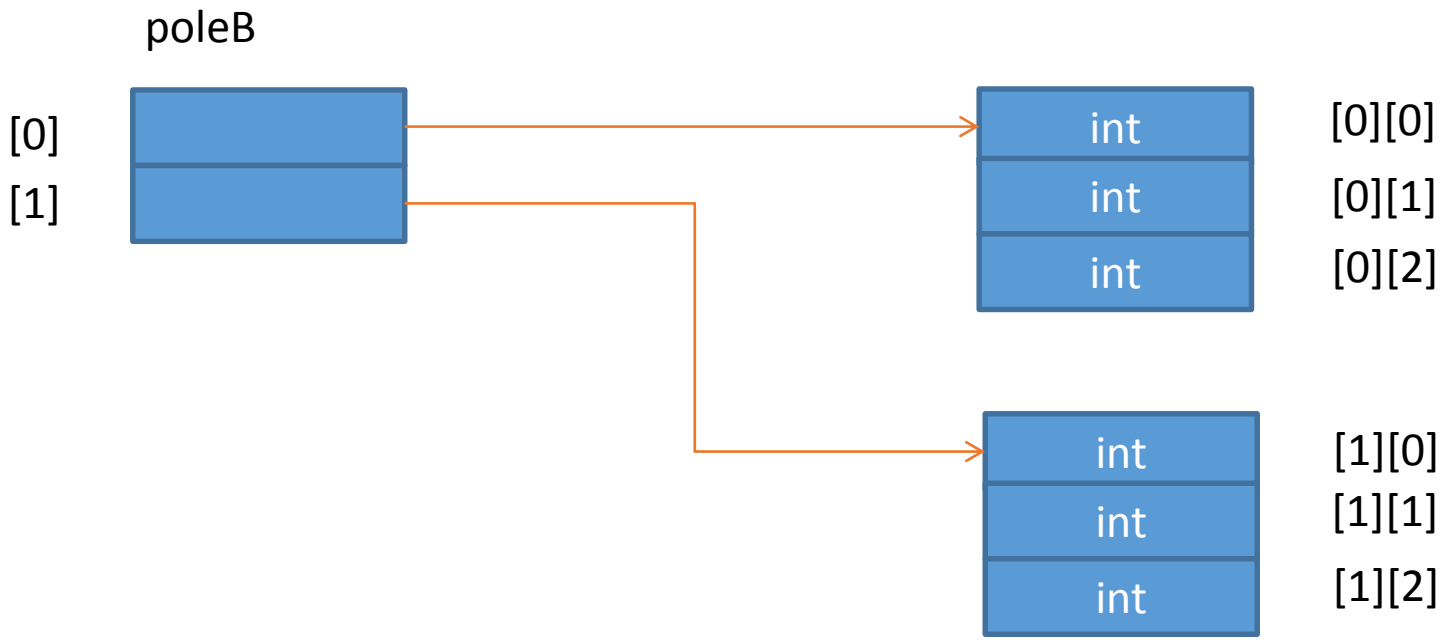
- příklad vytvoření:

```
int *poleB[2];
poleB[0]=(int *)malloc(3 * sizeof(int));
poleB[1]=(int *)malloc(3 * sizeof(int));
```

- přístup k prvkům pole je stejný jako u statického pole
- tento typ pole se používá poměrně často
  (pole textových řetězců, ...)
- při kompilaci je nutné znát pouze první rozměr pole
- jednotlivé řádky nemusí mít stejnou délku
- řádky nejsou v paměti uloženy za sebou

Dvourozměrné pole jako pole ukazatelů

```
int *poleB[2];
poleB[0]=(int *)malloc(3 * sizeof(int));
poleB[1]=(int *)malloc(3 * sizeof(int));
```



Dvourozměrné pole jako ukazatel na pole

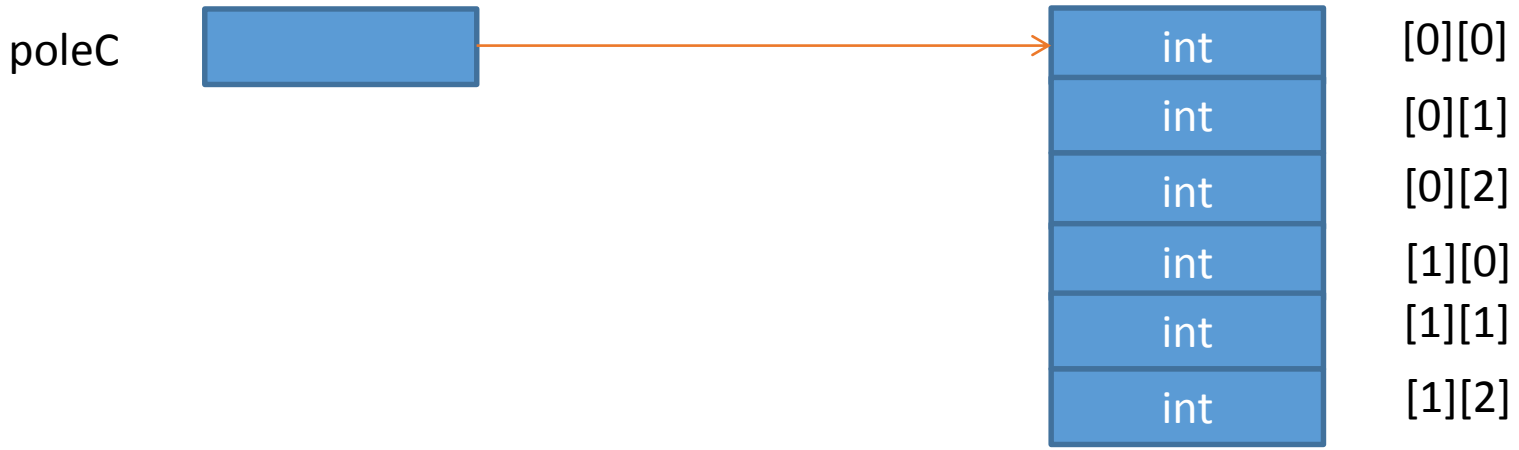
- příklad vytvoření:

```
int (*poleC)[3];
poleC=(int(*)[3])malloc(2*3*sizeof(int));
```

- přístup k prvkům pole je stejný jako u statického pole
- tento typ pole se příliš často nepoužívá, ale občas se může hodit
- při kompilaci je nutné znát pouze druhý rozměr pole
- jednotlivé řádky mají stejnou délku
- řádky jsou v paměti uloženy za sebou jako u statického pole (pouze jsou v dynamicky alokované paměti)

Dvourozměrné pole jako ukazatel na pole

```
int (*poleC)[3];
poleC=(int(*)[3])malloc(2*3*sizeof(int));
```



Dvourozměrné pole jako pointer na pointer

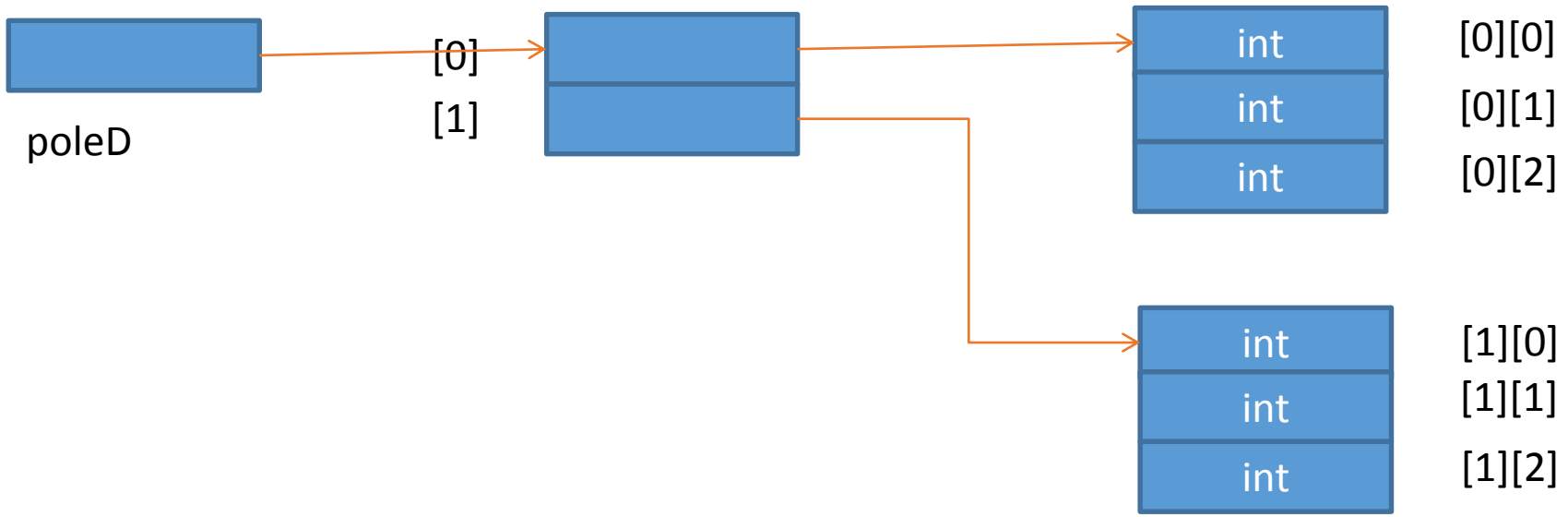
- příklad vytvoření:

```
int **poleD;
poleD=(int **)malloc(2*sizeof(int *));
poleD[0]=(int *)malloc(3*sizeof(int));
poleD[1]=(int *)malloc(3*sizeof(int));
```

- přístup k prvkům pole je stejný jako u statického pole
- tento typ pole se používá poměrně často
- při kompilaci není nutné znát žádný rozměr pole
- jednotlivé řádky nemusí mít stejnou délku
- řádky nejsou v paměti uloženy za sebou
- pole je celé uloženo v dynamicky alokované paměti

Dvourozměrné pole jako pole ukazatelů

```
int **poleD;
poleD=(int **)malloc(2*sizeof(int *));
poleD[0]=(int *)malloc(3*sizeof(int));
poleD[1]=(int *)malloc(3*sizeof(int));
```



Výhody a nevýhody jednotlivých typů polí

- paměťové nároky:
  - statické pole `poleA` je paměťově nejvýhodnější
    (žádné pomocné ukazatele)
  - dynamická pole vyžadují paměť na uložení ukazatelů a navíc každá alokace dynamické paměti potřebuje
    nějaké místo na „administrativu”
- rychlost přístupu k prvkům
  - přístup k prvkům statického pole `poleA` je nejrychlejší
  - téměř stejně rychlý bude přístup k prvkům u typu `poleC`
  - nejpomalejší pravděpodobně bude přístup k prvkům `poleD`
    (přístup k dynamicky alokované paměti přes 2 ukazatele)
- rozměry pole (viz dříve)

Pole textových řetězců

- používá se často v programech, které pracují s textem
- jde o pole typu poleB (případně poleD)
- příklad:

```
char* p_text[4];
p_text[0]="prvni";
p_text[1]="druhy";
p_text[2]=(char *)malloc(6*sizeof(char));
strcpy(p_text[2], "treti");
p_text[3]="ctvrty";
```

- pokud použijeme za identifikátorem pole pouze jeden index, pracujeme s celým řetězcem

```
printf("%s\n", p_text[1]);
```

Jak číst složité deklarace?

- příklad: `int *(*x)[3];`
- najdeme identifikátor „x“ a čteme: *„x je“*
- od identifikátoru čteme **doprava**, dokud nenarazíme na znak „)“ nebo „;“
- „)“ nás vrací na odpovídající „(“, od ní čteme **doprava** až po již přečtý text: *„ukazatel na“*
- přeskočíme již zpracovaný text a pokračujeme, dokud nenarazíme na „)“ nebo „;“: *„pole 3 prvků typu“*
- pokud narazíme na znak „;“, přesuneme se na začátek již zpracovaného textu a čteme **doleva**: *„ukazatel na int“*

Příklady

```
int *poleB[2];
```

„poleB je pole dvou prvků typu ukazatel na int”

```
int (*poleC)[3];
```

„poleC je ukazatel na pole tří prvků typu int”

```
long double *f(int, double);
```

„f je funkce s parametry typu int a double vracející ukazatel na long double”

```
long double *(* f)(int, double);
```

„f je ukazatel na funkci s parametry typu int a double vracející ukazatel na long double”

```
double *(*p_f[3])();
```

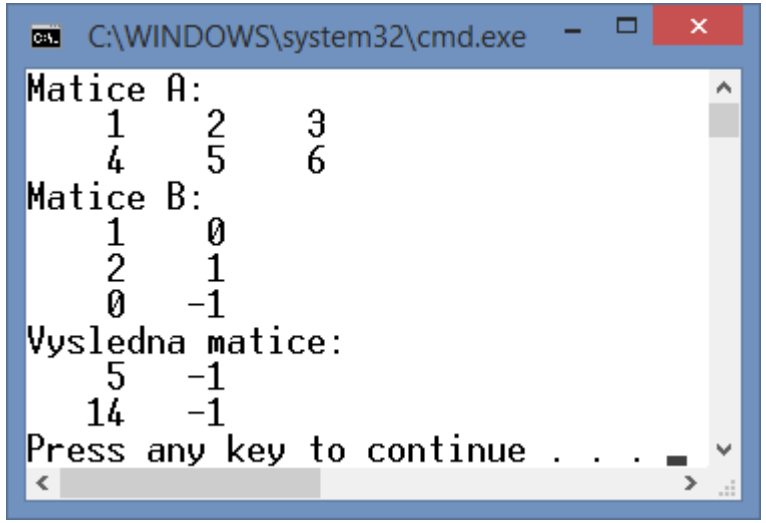
„p_f je pole tří prvků typu ukazatel na funkci vracející ukazatel na double”

Příklad

Napište v jazyku C funkci

```
double **soucin(int m, int n, int o, double **A, double **B),
```

která vypočítá součin matice A o rozměrech *m*x*n* a matice B o rozměrech *n*x*o*. Funkce vrací alokované dvojrozměrné pole s hodnotami výsledné matice.



Příklad 1

```
double **soucin(int m, int n, int o, double **A, double **B)
{
   int i,j,k;
   double **s;
   s=(double **)malloc(m*sizeof(double*));
   for(i=0;i<m;i++)
        s[i]=(double *)malloc(o*sizeof(double));

   for(i=0;i<m;i++)
        for(j=0;j<o;j++)
        {
            s[i][j]=0;
            for(k=0;k<n;k++)
                s[i][j]+=A[i][k]*B[k][j];
        }
   return s;
}
```