

Vícerozměrná pole

Jiří Zacpal



DEPARTMENT OF COMPUTER SCIENCE
PALACKÝ UNIVERSITY, OLOMOUC

KMI/ZP2 Základy programování 2

Jednorozměrné pole – připomenutí

- deklarace:
`int pole[5]={1,2,3,4,5};`
- přístup k prvkům:
`pole[4]=3;`
`printf("%i", pole[2]);`
- vztah pointerů a polí:
`*(pole+i)` odpovídá zápisu `pole[i]`
- typ pole (resp. prvků v poli) může být libovolný

Příklad 1



```
#define N 11
```

```
main()
{
    int i, p[N],temp;
    for(i=0;i<N;p[i++]=i+1);
    printf("Puvodni pole:");
    for(i=0;i<N;printf("%d, ",p[i++]));
    for(i=0;i<N/2;i++)
    {
        temp=p[i];
        p[i]=p[N-1-i];
        p[N-1-i]=temp;
    }
    printf("\nObracene pole:");
    for(i=0;i<10;printf("%d, ",p[i++]));
}
```

Deklarace vícerozměrného pole

- nabízí se tedy následující otázky:
 - Lze vytvořit pole, jehož prvky budou pole?
 - Jak s tímto „vícerozměrným“ polem pracovat?
- bez inicializace (obecně):
`typ identifikátor[rozmer1]...[rozmerN];`
- příklady:
`int moje_matice[3][4];`
`float trojrozmerne[3][4][10];`
- s inicializací:
`typ id[r1]...[rN]={hodnoty v blocích};`
- příklady:
`int matice[2][3]={{1,2,3},{4,5,6}};`
`int`
`m[2][3][4]={{1,2,3,4},{5,6,7,8},{9,10,11,12}},{{13,14,15,16},{17,18,19,20},{21,22,23,24}}};`

Příklad 2



```
main()
{
    int i,j, p[N][N],temp;
    for(i=0;i<N;i++)
    for(j=0;j<N;j++)
    p[i][j]=10*i+j;
    printf("Puvodni pole:\n");
    for(i=0;i<N;i++)
    {
        for(j=0;j<N;j++)
            printf("%d, ",p[i][j]);
        printf("\n");
    }
    for(i=0;i<N/2;i++)
    for(j=0;j<N/2;j++)
    {
        temp=p[i][j];
        p[i][j]=p[N-1-i][N-1-j];
        p[N-1-i][N-1-j]=temp;
    }
    printf("\nObracene pole:\n");
    for(i=0;i<N;i++)
    {
        for(j=0;j<N;j++)
            printf("%d, ",p[i][j]);
        printf("\n");
    }
}
```

Dvourozměrné pole v paměti



- prvky pole `int x[2][3]` jsou v paměti uloženy v pořadí: `x[0][0]`, `x[0][1]`, `x[0][2]`, `x[1][0]`, `x[1][1]`, `x[1][2]`
- na dvourozměrné pole se lze také dívat jako na jedno-rozměrné pole jednorozměrných polí daného typu (čili ukazatel na jednorozměrné pole daného typu)
- `x` je adresa dvourozměrné pole (typ `int[2][3]` nebo `int*[3]`)
- `x[0]` je adresa prvního řádku, `x[1]` je adresa druhého řádku (oba typ `int[3]` nebo `int*`)
- `x+1` a `x[0]+1` jsou tedy různé adresy

Příklad 3



```
main()
{
    int i,j, p[N][N],temp;
    for(i=0;i<N;i++)
        for(j=0;j<N;j++)
            *(p[i]+j)=10*i+j;
    printf("Puvodni pole:\n");
    for(i=0;i<N;i++)
    {
        for(j=0;j<N;j++)
            printf("%d, ",*(p[i]+j));
        printf("\n");
    }
    for(i=0;i<N/2;i++)
        for(j=0;j<N/2;j++)
        {
            temp=p[i][j];
            p[i][j]=p[N-1-i][N-1-j];
            p[N-1-i][N-1-j]=temp;
        }
    printf("\nObracene pole:\n");
    for(i=0;i<N;i++)
    {
        int *pp=p[i];
        for(j=0;j<N;j++)
            printf("%d, ",*(pp+j));
        printf("\n");
    }
}
```

Pole jako parametr funkce

- jednorozměrné pole:

```
int maximum(int ciska[], int pocet){...}
```

...

```
int ciska[10]={1,45,21,5,7,2,3,35,47,4};  
max = maximum(ciska, 10);
```

- dvourozměrné pole:

```
int maxim(int ciska[][3], int radku){...}
```

...

```
int ciska[2][3]={{1,45,21},{5,7,2}};  
max = maxim(ciska, 2);
```


Příklad 4



```
int maximum(int prvky[][4], int radku)
{
    int max=prvky[0][0],i,j;
    for (i=0;i<radku;i++)
        for (j=0;j<4;j++)
            if (prvky[i][j]>max) max=prvky[i][j];
    return max;
}

void tisk(int prvky[][4], int radku)
{
    int i;
    for (i=0;i<radku;i++)
        printf("%5d %5d %5d %5d \n", *prvky[i],*(prvky[i]+1),*(prvky[i]+2),*(prvky[i]+3));
}

main()
{
    int pole[3][4]={10,2,15,-2},{-52,41,0,12},{15,3,1,-8}};
    tisk(pole,3);
    printf("Maximum je: %d\n",maximum(pole,3));
}
```