

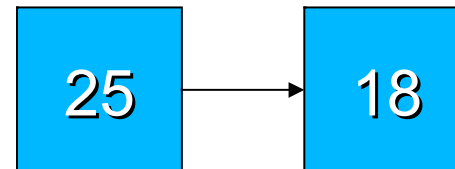
Programovací jazyk C

štvrtá část

Miroslav Melicherčík

Pointery

- pointer – predstavuje adresu v pamäti, na ktorej sa nachádza príslušná hodnota
- * dereferenčný operátor – vracia hodnotu nachádzajúcu sa danom mieste v pamäti
- & referenčný operátor – vracia adresu určitého miesta v pamäti
- definícia `int *p;`



Práca s pointermi

```
int i, *p_i;
```

Statické priradenie (ok pri preklade)

```
    i = 3;          p_i = 3;  
*p_i = 4;          i = p_i;  
    i = *p_i;       i = &p_i;  
*p_i = i;  
    p_i = &i;
```

Dynamické priradenie (ok pri preklade aj pri behu)

```
p_i = &i; *p_i = 4;
```

```
*p_i = 4;
```

4 je priradená na náhodnú adresu

Príklad

```
// priklad pointera
#include <stdio.h>

main()
{
    int i, j, *p_i;

    scanf("%d %d", &i, &j);
    p_i = (i > j) ? &i : &j;
    printf("Vacsie cislo z %d a %d je %d\n", i, j, *p_i);
    printf("Cislo i je ulozene na adrese %p", &i);
    printf("Cislo j je ulozene na adrese %p", &j);
    printf("Vacsie cislo je ulozene na adrese %p", p_i);
}
```

Alokácia pamäte

- statická alokácie pamäte

```
int a, *p_a;
```

```
a = 4;
```

```
p_a = &a;
```

```
*p_a = 3;
```

```
scanf("%d", p_a); scanf("%d", &a);
```

```
printf("%d", *p_a); printf("%d", a);
```

Alokácia pamäte

- dynamická alokácia pamäte

```
int *p_i;  
p_i = (int*) malloc(sizeof(int) * 1);  
free((void*) p_i);  
p_i = NULL;
```

- v C++

```
int *p_i;  
p_i = new int;  
delete p_i;  
p_i = NULL;
```

Operácie s pointermi

- porovnanie adries
 $p_i < p_j$
- porovnanie hodnôt
 $*p_i < *p_j$
- posun v adrese o jeden int vyššie
 $p_i + 1$
- pripočítanie 1 k hodnote na adrese p_i
 $(*p_i) + 1$
- hodnota na adrese $p_i + 1$
 $*(p_i + 1)$

Volanie hodnotou a odkazom

```
void vymen(int x, int y)
{
    int p;
    p = x;
    x = y;
    y = p;
}
```

```
volanie: vymen(a, b);
```

```
void vymen(int *x, int *y)
{
    int p;
    p = *x;
    *x = *y;
    *y = p;
}
```

```
volanie: vymen(&a, &b);
```


Jednorozmerné polia

- definícia jednorozmerného poľa (vektora)

```
int a[10] ; //pole s 10 prvkami typu int
int a[] = {3,4,7,8,10};
int a[10] = {1,2,[8] = 50};
```

- lokálna definícia jednorozmerného poľa
s neznámou veľkosťou (podľa normy "C99")

```
void pole(int n) {
    int p[n * 2];
    . . .
}
```

Jednorozmerné polia

- adresovanie prvkov poľa
- indexy začínajú číslom 0 až po n-1

`a[0] = *a //prvy prvok pola`

`a[1] = *(a + 1) //druhy prvok pola`

Jednorozmerné polia

- dynamická definícia jednorozmerného poľa

```
int *a;  
a = (int*) malloc(sizeof(int) * 10);
```

- zrušenie poľa

```
free((void*) a);  
a = NULL;
```

Jednorozměrné pole

- dynamická změna velikosti pole

```
int *a;  
a = (int*) malloc(sizeof(int) * 10);  
a = (int*) realloc(a, sizeof(int) * 20);
```

Reťazce

- reťazce ako také sú C reprezentované ako polia znakov

```
char s[20];
```

```
char s[] = "pomoc";
```

toto pole bude obsahovať 6 znakov



p o m o c \0

- reťazce sú ukončené znakom \0

Reťazce

```
char s[10] = "pomoc";
```

toto pole bude obsahovať 10 znakov

p o m o c \0

Reťazce

- dynamický reťazec

```
char *s;  
s = (char*) malloc(sizeof(char) * 20);  
s = (char*) realloc(s, sizeof(char) * 40);
```

zrušenie dynamického reťazca

```
free((void*) s);  
s = NULL;
```

Práca s reťazcami

- načítanie reťazca

`scanf ("%s", s);` //bez &
načíta reťazec po biely znak

`gets (s);`
načíta všetko po ENTER

- vypísanie reťazca

`printf ("%s", s);`

`puts (s);`

d'alšie príkazy: `fscanf()`, `fprintf()`, `fgets()`, `fputs()`

Práca s reťazcami

- knižnica pre prácu s reťazcami <string.h>
- dĺžka reťazca
`int strlen(char *s);`
- kopírovanie reťazca
`char strcpy(char *kam, char *co);`
- spájanie reťazcov
`char strcat(char *s1, char *s2);`
- hľadanie znaku v reťazci
`char strchr(char *s, char c);`
- porovnanie dvoch reťazcov
`int strcmp(char *s1, char *s2);`
- hľadanie podreťazca v reťazci
`char strstr(char *kde, char *co);`

Práce s řetězci

- převod řetězců na čísla `<stdlib.h>`
- celé číslo
`int atoi(char *string);`
- celé číslo typu long
`long atol(char *string);`
- reálné číslo
`double atof(char *string);`

Práca s reťazcami

- funkcie **sprintf** a **sscanf**
- pracujú rovnako ako printf a scanf
- ich výstup nie je vytlačený alebo načítaný zo štandardných zariadení pre vstup a výstup ale do/z nami zadaného reťazca

Práca s reťazcami

- pre prácu s viac bajtovými znakmi (UNICODE) existujú knižnice
- podľa normy "C99"

	ASCII znaky	UNICODE
znaky	<ctype.h>	<wctype.h>
reťazce	<string.h>	<wchar.h>

Viacrozmerné polia

- staticky

```
int m[5][3];
```

- dvojrozmerné pole ako pole pointerov

```
int *b[5];
```

```
for(int i=0; i<5; i++) b[i]=(int*) malloc(3 * sizeof(int));
```

- indexovanie polí

```
m[1][2]
```

znamená 2. riadok a 3. stĺpec

Viacrozmerné polia

- pointer na polia

```
int (*xc)[3];    //1 pointer na pole troch int  
xc = (int*[3]) malloc(sizeof(int) * 3 * 5);
```

- pointer na pointer

```
int **p;  
p = (int**) malloc(5 * sizeof(int*));  
for(int i=0; i<5; i++)  
    p[i] = (int*) malloc(3 * sizeof(int));
```

Viacrozmerné polia

- inicializácia viacrozmerného poľa

```
int f[][2] = {  
    {1, 2},  
    {3, 4},  
    {5, 6}  
};
```