Pole a pointery

Pole

Pole slouží k uchování více proměnných stejného typu. V jazyce C je jeho velikost definována předem a lze klasicky vytvořit i pole vícerozměrné (takové pole prakticky do svých hodnot přiřazuje další pole o zvolené velikosti – dvojrozměrné by tedy vytvářelo 2D prostor).

Deklarace v jazyce C se provádí stejně jako deklarace proměnné, akorát je za název vložen popis pole s rozměry:

```
int pole[10];

char dvojpole[10][6];
```

Přístup k jednotlivým hodnotám se provádí obdobně (je třeba dbát na to, že hodnoty začínají nulou):

```
printf(„Paty prvek pole celych cisel: %d“, pole[4]);
```

Pointery (ukazatele)

Pointery jsou velmi podobné normálním proměnným, ale jejich hodnotou je nějaká adresa v paměti. Adresu můžeme z normálních proměnných získat přidáním referenčního operátoru **&** před danou proměnnou (takto ji můžeme například předat ukazateli). K přístupu k hodnotám na dané adrese pak slouží operátor dereference *.

Deklarace

Pointery využívají podobné deklarace jako normální proměnné, za jejich typ je však připojen znak *. Využívá se tedy formy **typ* identifikátor**:

```
char* ukazatel;
```

Pro (dynamickou) deklaraci je také důležitá funkce **malloc()**. Touto funkcí lze alokovat místo v paměti (bez alokace ukazuje pointer na **NULL** kvůli ochraně skryté paměti). K alokaci využíváme konverzi na zvolený typ pointeru (malloc vrací **void***) a pro univerzálnost i **sizeof()**:

```
ukazatel = (char*) malloc(sizeof(char));
```

Uvolnění paměti provádíme funkcí **free()**, které musíme předat adresu ukazatele.

Propojení pole a ukazatele

Mezi polem a ukazatelem je velmi úzký vztah – jsou teoreticky zaměnitelné.

Součet ukazatele a celého čísla

Pokud k pointeru přičteme celé číslo, posune se nám jeho adresa o stejný počet hodnot zvoleného typu dále (například pro ukazatele **int*** by došlo k posunu o čtyři bajty). Dereferencí pak můžeme s pointery pracovat podobně jako s polem (dokonce lze využít i zápis s hranatými závorkami).

Dynamická alokace dvojrozměrného pole

Ukazatel na začátek pole se dá díky výše zmíněnému přičítání využít přímo jako pole. Dvojrozměrné **[10][10]** pole znaků bychom tedy vytvořili takto (vynechána kontrola návratu **NULL**, která by se měla ošetřit pomocí **free()**):

```
char* * array;

array = (char* *) malloc(10 * sizeof(char*));

for (i = 0; i < x; i++) { array[i] = (char*) malloc(10 * sizeof(char)); }
```