Dynamické premenné

Pomocou dynamických premenných si počas behu programu ukrajujeme z pamäte– podľa potreby (alokujeme si pamäť). Táto pamäť sa nazýva **heap (halda).**

Najčastejšie používanou funkciou pre vytvorenie dynamickej premennej je funkcia **malloc.** Jediným parametrom tejto funkcie je počet bajtov, ktoré chceme alokovať. Počet bajtov zvoleného typu nám zistí funkcia **sizeof**.

Funkcia **malloc** vráti ukazovateľ na vytvorenú dynamickú premennú, presnejšie **pointer na void** (t.j. neurčený prázdy údajový typ)**, ktorý je potrebné pretypovať** na pointer na požadovaný typ.

```
int *p;
p = (int*)malloc(sizeof(int));
```

Ak chceme vytvoriť dynamické pole 10 prvkov typu float, musíme počet bajtov zistených funkciou **sizeof** vynásobiť požadovaným počtom prvkov poľa, teda 10:

```
float *pole;
int n;
n = 10;
pole = (float*)malloc(n * sizeof(float));
```

V prípade, že sa nepodarilo prideliť pamäť požadovanej dĺžky, vráti funkcia **malloc** hodnotu **NULL**. Je dobrým zvykom pri každom prideľovaní pamäti testovať návratovú hodnotu a nespoliehať sa na to, že pamäte bude vždy dosť.

```
if ((pole = (float*)malloc(n * sizeof(float)))== NULL){
    printf("Nedostatok pamati!");
    exit(1);
}
```

Uvoľňovanie pamäte je opačná akcia než jej prideľovanie. Platí všeobecná zásada, že dynamicky alokovanú pamäť, teda dynamickú premennú, ktorú už nepotrebujeme, uvoľníme ihneď a nečakáme až na koniec programu.

Ak sme dynamickú premennú vytvorili pomocou funkcie **malloc,** na jej zrušenie je potrebné použiť funkciu **free**. Predtým ju ešte pretypujeme na **pointer na void**. Zrušením dynamickej premennej vrátime už nepotrebnú pamäť späť do heapu.

Ukazovateľ nastavíme na **NULL**, aby neukazoval na už uvoľnené miesto v pamäti heap.

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

Program na nájdenie maxima v poli upravený cez dynamické pole:

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int najdenie_maxima (int* pole, int pocet_prvkov);

int main ()
{
    int n, i, poz,*a;

    do{
        printf("Zadaj pocet prvkov pola = ");
        scanf("%d", &n);
    }
    while((n<=0)||(n>100));

    if ((a=(int *)malloc (n*sizeof(int))) == NULL)
    {
        printf ("Nedostatok miesta");
        return 0;
    }
        printf("\n\n...generujem cisla...\n\n");
        srand(time(0));

        for (i = 0; i < n; i++)
        {
            *(a+i) = rand() % 101 ;
            printf ("%d ", *(a+i));
        }

        poz = najdenie_maxima (a,n);

        printf("\n\nmax = %d, pozicia maxima v poli = %d\n\n",
*(a+poz), poz +1 );

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

    return 0;
}

int najdenie_maxima (int* pole, int pocet_prvkov)
{
    int i, maximum, pozicia_maxima;

     maximum = *(pole+0);
     pozicia_maxima = 0;

     for (i = 1; i < pocet_prvkov; i++)
    {
        if (*(pole+i) > maximum)
        {
            maximum = *(pole+i);
            pozicia_maxima = i;
        }
    }
    return pozicia_maxima;
}
```