UPR
Cvičení 4

Pole

- Zatím jsme pracovali jen s několika proměnnými
- Co když chceme uložit např. průměrné denní teploty za celý měsíc?
- Museli bychom vytvořit proměnné:

```
float tep_01, tep_02, …, tep_31;
```

- To je však nepraktické - místo toho použijeme pole

https://mrlvsb.github.io/upr-skripta/c/pole/pole.html

Statické pole

- Pole celých čísel o 3 prvcích:

```
int pole[3]; // v tuto chvíli jsou tam náhodné hodnoty
pole[0] = 1; // nastavíme hodnotu na 0
pole[1] = 2;
pole[2] = 3;
```

nebo

```
int pole[3] = {1, 2, 3};
int pole[] = {1, 2, 3};
```

https://mrlvsb.github.io/upr-skripta/c/pole/staticke_pole.html

Statické pole

- Přístup k prvkům pole:

  první prvek: `int p = pole[0];`

  změna hodnoty druhého prvku pole: `pole[1] = 1;`

- Hromadně lze nastavit pomocí cyklu:

```
for(int i = 0; i < 3; i++)
{
  pole[i] = 0;
}
```

Pole

- Úkol č.1

  Načtěte celá čísla do pole a vytvořte funkci, která vrátí součet čísel v poli
- Počet čísel zadejte předem pomocí proměnné

```
int soucet_pole(int *pole, int pocet)
```

Funkce + ukazatel + pole

- Úkol č.2
  - Vytvořte pole o 20 prvcích a naplňte ho náhodnými čísly
  - Jak na náhodná čísla?

```
#include <stdlib.h>
#include<stdio.h>
#include <time.h>
int main() {
  srand(time(NULL)); // na začátek main
  int r = rand() % 30; // náhodné číslo z intervalu 0 až 29
}
```

  - Vytvořte funkci, která vrátí maximální a minimální hodnotu z pole a hodnoty vytiskněte

Paměť

- Dosud jsme naše proměnné vytvářeli v rámci funkcí
  - Tyto proměnné existují pouze v rámci funkce, poté jsou uvolněny z paměti
  - Ukládají se do paměti zvané stack (zásobník)
    - Omezená velikost paměti (zkuste vytvořit pole o velikosti 1 000, 100 000, 10 000 000)

Paměť

- Proměnné lze ukládat do dynamické paměti, která má větší kapacitu
- Je třeba alokovat paměť a po konci používání ji dealokovat

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

free(num);
```

https://mrlvsb.github.io/upr-skripta/c/prace_s_pameti/dynamicka_pamet.html

Dynamické pole

- pole o třech prvcích

```
int *arr = (int*) malloc(3*sizeof(int));
```

- Je třeba dealokovat (vymazat z paměti)!

```
free(arr);
```

- práce je stejná jako se statickým polem

```
arr[0] = 1
...
```

https://mrlvsb.github.io/upr-skripta/c/pole/dynamicke_pole.html

Paměť

- Úkol č.3
  - Vytvořte dynamické pole o vybrané velikosti a naplňte ho znaky (typ char)
  - Vytvořte funkci `int najdi(char znak, char *pole, int velikost)`, která vrátí první pozici znaku v poli, pokud v poli znak není, tak -1