UPR
Cvičení 3

Funkce

- Blok kódu, který vykoná nějaké operace.
  - Tyto operace nemusíme psát pořád znova, ale jen zavoláme funkci

```
int obsah_obdelniku(int a, int b)
{
    int obsah = a*b;
    return obsah;
}
```

  - v mainu:

```
int strana_a = 4;
int strana_b = 5;
int obsah = obsah_obdelniku(strana_a,
strana_b);
```

Funkce

- Co když nechceme nic předat?

```
int vyzvi_a_nacti()
{
   printf(“Zadej cislo: “);
   int c;
   scanf(“%d”, &c);
   return c;
}
```

  - v mainu:

```
int a = vyzvi_a_nacti();
```

Funkce

- Co když nechceme nic vrátit?

```
void tiskni_cislo(int c)
{
   printf(“Tisknu cislo %d\n“, c);
}
```

  - v mainu:

```
tiskni_cislo(5);
```

Pointer (ukazatel)

- Proměnná uchovávající adresu do paměti
  - Data nemusíme kopírovat, ale odkážeme se na ně

```
int *a; // ukazatel na integer
float *b; // ukazatel na float
```

- Do ukazatele se ukládá adresa (jak zjistit adresu proměnné? Pomocí &)

```
int a = 5;
int *b = &a;
```

Pointer (ukazatel)

- Jak získat hodnotu pointeru?

```
printf("Adresa: %p, hodnota: %d\n", b, *b);
```

- K čemu je to dobré? Např. když chceme z funkce vrátit více hodnot.

Pointer (ukazatel)

- Úkol č.1
  - Minule jste vytvořili funkci pro součet dvou čísel
  - Upravte funkci, že vrátí součet i rozdíl těchto čísel
    - Buď vraťte jednu hodnotu pomocí `return` a druhou mocí ukazatele nebo obě pomocí ukazatele (funkce pak bude vracet `void`)

Pole

- Zatím jsme pracovali s dvěma až třemi proměnnými
- Co když jich bude více?
- Museli bychom mít:

```
int a, b, c, …, o, p, q;
```

- Místo toho použijeme pole

Pole

- Pole celých čísel o 3 prvcích:

```
int pole[3];
pole[0] = 1;
pole[1] = 2;
pole[2] = 3;
```

nebo

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

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.2 (= 4 z minula)

  Načtěte čísla do pole a vytvořte funkci, která vrátí součet čísel v poli
- Počet čísel zadejte předem pomocí `const int`
- `int soucet_pole(int *pole, int pocet)`

Pointer + pole

- Úkol č.3
  - Vytvořte pole o 20 prvcích a naplňte ho náhodnými čísly

```
#include <time.h>

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

Bodovaný úkol

- (2b) Vytvořte funkci `int faktorial(int cislo)`, která vrátí faktoriál čísla předaného v parametru

  `5! = 5*4*3*2*1, 0! = 1`

- (3b) Vytvořte funkci `float eulerovo_cislo(float epsilon)`, která vrátí hodnotu Eulerova čísla s přesností na epsilon – výpočet ukončete, pokud hodnota nového členu v nekonečné řadě bude menší než epsilon

$$e = \sum_{n=0}^{\infty} \frac{1}{n!} = \frac{1.0}{0!} + \frac{1.0}{1!} + \frac{1.0}{2!} + \ldots = 2.71828$$