

# Informační a komunikační technologie

## Funkce v C

**David Weber**

Kabinet K13

[weber3@spsejecna.cz](mailto:weber3@spsejecna.cz)

# Příklad na úvod

```
// Print array
for (int i = 0; i < size; i++) {
    printf("%d ", arr[i]);
}
printf("\n");
// Square array elements
for (int i = 0; i < size; i++) {
    arr[i] *= arr[i];
}
// Print array
for (int i = 0; i < size; i++) {
    printf("%d ", arr[i]);
}
```

# V čem je problém?

- Kód je funkční, ale část pro výpis pole je zde uvedena **dvakrát**.

```
for (int i = 0; i < size; i++) {  
    printf("%d ", arr[i]);  
}
```

- Opakující kód je nepraktický ✗
  - budeme-li chtít změnit nějakou jeho část, musíme změnu provést všude
- Použijeme funkci ✓

# Co je to funkce?

- Obecně se jedná o část programu, kterou je možné opakovaně “vyvolat” v různých místech programu.
- Motivace pro použití:
  - odstranění opakování kódu v programu,
  - rozklad složitých problémů na jednodušší,
  - znovupoužití v jiných programech, např. formou knihoven (to až ve druháku 🤖).

# Struktura funkce

```
navratovy_typ jmeno(parametry) {  
    <implementace>  
}
```

- U funkce je třeba specifikovat:
  - návratový datový typ,
  - jméno (identifikátor),
  - parametry,
  - tělo (implementace).
- Funkce uvádíme mimo tělo funkce `main`.

# Funkce bez parametrů

- Nejjednodušší typ funkce.
- Klíčové slovo `void` (prázdný datový typ), prázdné závorky `()`

```
void greet() {  
    printf("Hello World!");  
}
```

- **Důležité:** funkci musíme uvést nad `main`!
- Samotná definice funkce nic nedělá  $\Rightarrow$  je třeba ji tzv. **zavolat**.

# Volání funkce

- Je třeba specifikovat, kde v programu se má daná funkce provést.

```
void greet() {  
    printf("Hello World!");  
}
```

```
int main() {  
  
    greet();    // Prints out "Hello World!"  
  
    return 0;  
}
```

# Funkce s parametry

- Často budeme chtít činnost funkce zobecnit (výstup funkce bude na něčem záviset).
- $\Rightarrow$  k tomu použijeme tzv. **parametry**.
- Do kulatých závorek () uvádíme výčet parametrů (jejich datový typ a název), které funkce přijímá.

```
void fce(<datovy_typ> var1, <datovy_typ> var2, ...) {  
    ...  
}
```



# Příklad I (funkce s parametrem)

```
void square(float x) {  
    printf("Druha mocnina x je %g", x*x);  
}  
  
int main() {  
  
    float input;  
    scanf("%f", &input);  
  
    square(input);  
  
    return 0;  
}
```

# Příklad II (funkce s více parametry)

Funkce pro výpočet aritmetického průměru celých čísel  $a, b \in \mathbb{Z}$ :

```
void average(int a, int b) {  
    float avg = (a + b) / 2;  
    printf("%g", avg);  
}  
  
int main() {  
    float input1, input2;  
    scanf("%d %d", &input1, &input2);  
  
    average(input1, input2);  
  
    return 0;  
}
```

# Úloha I

Předělejte program pro výpis faktoriálu, tj.

$$n! = n(n-1)(n-2) \cdots 2 \cdot 1,$$

do funkce. (Viz domácí úkol 😊)

# Funkce s návratovou hodnotou

- **Problém:** s hodnotou, kterou funkce vypočte, nemůžeme dále pracovat (pouze ji na konci vypisujeme).
- Např. u funkcí `pow`, `sqrt`, ... (viz knihovna **math**) jsme mohli s vypočtenou hodnotou dále počítat

```
float hypotenuse = sqrt(pow(x, 2) + pow(y, 2));  
printf("%f", hypotenuse);
```

- $\Rightarrow$  využijeme tzv. **návratovou hodnotu**.

# Otázky?

