

# **Základy programovania 25ZS**

## **While, Switch,**

## **a ako pekne programovať**

Prednášajúci: prof. Gabriel Juhás  
Cvičiaci: Milan Mladoniczky



# While cyklus

- While cyklus sa používa ak je potrebné vykonať kód opakovane kým je podmienka splnená.
- Podmienka: Podstatná časť výrazu, ktorá rozhoduje o tom, či sa kód v tele výrazu má vykonať. Iterácia cyklu sa vykoná iba vtedy ak je podmienka vyhodnotená ako pravdivá.
- Telo cyklu: Kód, ktorý sa má vykonať každou iteráciou ak je vyhodnotená podmienka pravdivo.

```
while (<podmienka>) {  
    <telo podmienka>  
}
```

```
int i = 0;
```

```
while(i < 10) { i++; }
```



# Switch

- Príkaz switch case je alternatívou k postupnosti príkazov if - else if, ktorý možno použiť na vykonanie podmieneného kódu na základe hodnoty premennej uvedenej v príkaze switch.
- Blok výrazu switch pozostáva z prípadov (case), ktoré sa majú vykonať na základe hodnoty premennej v switch.

```
int var = 2;

switch (var) {

    case 1:

        printf("Case 1 is executed");

        break;

    case 2:

        printf("Case 2 is executed");

        break;

    default:

        printf("Default Case is executed");


        break;

}
```



## Zapúzdzrenie (Enkapsulácia)

- Ak máme časť kódu, ktorá vykonáva nejakú činnosť, je vhodné vytvoriť z nej funkciu.
- Tento proces nazývame enkapsulácia / zapúzdzrenie (encapsulation).
- Zapúzdzrenie je proces, pri ktorom zapúzdzrime ("zabalíme") kód do funkcie.
- Následne môžeme v programe namiesto pôvodného kódu používať príslušnú funkciu, čo okrem iného zlepšuje čitateľnosť kódu.

- 
- Ak zapúzdrame kód, ktorý vypíše párne čísla, dostaneme funkciu, ktorú môžeme následne použiť.
  - Výhodou zapúzdrenia je navyše to, že ak sa vhodne zvolí samotný identifikátor funkcie, vieme z neho vydedukovať, čo funkcia robí!


```
void vypis_parnych_cisel() {  
    for(int i = 0; i <= 10; i++){  
        if(i % 2 == 0) printf("%d", i);  
    }  
}
```

```
vypis_parnych_cisel();
```



## Zovšeobecnenie (generalizácia)

- Ak máme funkciu, ktorá realizuje nejakú činnosť, niekedy je vhodné zamyslieť sa, či nevieme vytvoriť jej všeobecnejšiu verziu, ktorá bude vedieť vykonávať danú činnosť všeobecnejšie.
- Tento proces nazývame zovšeobecnenie / generalizácia (generalization).
- Pri zovšeobecnení funkcie spravidla pridáme do hlavičky funkcie nový vstupný parameter pomocou ktorého vieme parametrizovať typ činnosti, ktorý robí daná funkcia. Zároveň vždy platí, že pre vhodnú voľbu tohto parametra vie funkcia realizovať aj pôvodnú činnosť!
- V predošlom príklade sme zostrojili funkciu, ktorá vypíše párne čísla v intervale od 0 do 10. Preto by sme sa mohli pokúsiť funkciu generalizovať, t.j. zostrojiť funkciu, ktorá bude vedieť vypísať párne čísla pre ľubovoľný interval, pričom hranice intervalu budú nové parametre funkcie.

- 
- Zovšeobecníme predošlú funkciu `vypis_parnych_cisel()` tak, aby sme pomocou jej vstupných parametrov vedeli zmeniť interval čísel na výpis.
  - Pridáme teda do hlavičky parametre `od` a `do` a v tele funkcie tento parameter použijeme ako začiatok a koniec intervalu výpisu čísel. Všimnite si, že pre voľbu vstupu 2 a 20 sa funkcia `vypis_parnych_cisel(2, 20)` správa ako pôvodná funkcia, ktorú sme zovšeobecnil!

```
void vypis_parnych_cisel(int od, int do) {  
    for(int i = od; i <= do; i++){  
        if(i % 2 == 0) printf("%d", i);  
    }  
}  
  
vypis_parnych_cisel(2, 20);
```

---

Praktická ukážka →





<https://zapr.interes.group/exercises/exercise>