Podmínky

Podmínky umožňují větvení programu. Jsou jimi vyhodnocovány podmínky a podle toho buď vykonávány či nevykonávany určité funkce.

Syntaxe if a if-else

If je tím nejzákladnějším druhem podmínky, obsahuje nejdříve klíčové slovo `if`, samotnou podmínku (v kulatých závorkách `( )`)a dále tělo podmínky (ve složených závorkách `{}`), takto:

```
if (podmínka) {
   tělo podmínky //Vykoná se, pokud je podmínka pravdivá
}
```

Pokud je podmínka vyhodnocena jako pravdivá (je rovna 1), vykoná se tělo podmínky (tedy všechny příkazy a funkce ve složených závorkách), například:

```
int i;
if (i>5) {
   printf("Hodnota promenne i je vetsi nez 5.");
}
```

Složené závorky můžeme také vynechat, v takovém případě se vykoná pouze první funkce následující po kulatých závorkách.

If-else je jakési rozšíření podmínky If, začátek je stejný jako u If, ale dále následuje klíčové slov `else` a další tělo podmínky (opět ve složených závorkách`{}`), takto:

```
if (podmínka) {
   první tělo podmínky //Vykoná se, pokud je podmínka pravdivá
} else {
   druhé tělo podmínky //Vykoná se, pokud podmínka není pravdivá
}
```

Pokud je podmínka vyhodnocena jako pravdivá (je rovna 1), vykoná se první tělo podmínky (tedy všechny příkazy a funkce ve prvních složených závorkách), v opačném případě se vykoná druhé tělo podmínky (v druhých složených závorkách) například:

```
int i;
if (i>5) {
   printf("Hodnota promenne i je vetsi nez 5.");
} else {
   printf("Hodnota promenne i je mensi nebo rovna 5.");
}
```

Syntaxe Switch

Vysvětleno na Linuxsoftu.

Switch je pokročilejší podmínka, nejčastěji používaná jako zjednodušení zápisu několikanásobného If-else (tedy např. If-else if-else if-else). Zápis začíná klíčovým slovem `switch`, následuje nějaká *hodnota* v kulatých závorkách `()` a poté tělo podmínky ve složených závorkách. Tělo podmínky obsahují klíčová slova `case` a `default`. `Case` použijeme, když chceme porovnat *hodnotu v kulatých závorkách* s jinou hodnotou, kterou zapíšeme za daný `case` a dále napíšeme dvojtečku. Těchto *casů* můžeme napsat do Switche hned několik pokaždé platí, že pokud *hodnotu v kulatých závorkách* je rovna *hodnotě v casu*, vykonají se všechny příkazy a funkce až do konce těla podmínky Switche (tedy i přes další *casy* a `default`), popřípadě prvního příkazu `break`. Klíčové slovo `default` následované opět dvojtečkou má podobnou funkci jako *casy*, vyhodnotí se pozitivně ~~, pokud se pozitivně nevyhodnotil žádný case~~ vždy bez ohledu na hodnotu. Proto se *default* nepoužívá na začátku bloku s *case*, protože by ihned platil.

```
int i;
switch (i) {
   case 1: printf("i je rovno 1");
   case 2: printf("i je rovno 1 nebo 2"); //case 1 nebyl zakončen breakem, proto se vykoná i tento příkaz a i může být rovno 1
           break;
   case 3: printf("i je rovno 3");//case 1 a 2 byly ukončeny breakem, tento příkaz už se tedy nevykoná, pokud je i rovno 1 nebo 2
           break;
   default: printf("i neni rovno 1, 2 ani 3");
}
```

Ternární operátor

Ternární operátor je, jak už název vypovídá, operátor, který pracuje hned se třemi operandy; zapisuje se takto:

```
[první operand]?[druhý operand]:[třetí operand]
```

Je jakousi zkrácenou formou podmínky If-else, až na to, že je to výraz (což značně omezuje jeho použití). Tedy:

```
podmínka?hodnota když ano:hodnota když ne
```

Pokud je podmínka před otazníkem pravdivá (rovná 1), vrátí se hodnota výrazu za otazníkem a před dvojtečkou, v opačném případě se vrátí hodnota výrazu za dvojtečkou, např.:

```
int i;
printf(i>5?"Hodnota promenne i je vetsi nez 5.":"Hodnota promenne i je mensi nebo rovna 5.");
```

Líné vyhodnocování

Zdroj: Wikipedie

tedy

```
int function a() {printf("1"); return 1;}
int function b() {printf("2"); return 0;}
if(a() || b()) {...}
```

Toto na výstup vypíše 1, protože po vyhodnocení a() je jasné, že (PRAVDA nebo COKOLI) je PRAVDA, tedy není třeba vyhodnocovat b()