Zapiski s priprav

Pogojni stavki

ACM priprave na tekmovanja

Različica z dne 22. marec 2025

Kazalo

1 Osnovna struktura pogojnega stavka

Pogosto želimo, da računalnik izvaja drugačno kodo glede na vrednost ene ali več spremenljivk, npr. da nam pokaže drugačno vsebino, če smo napisali pravilno ali napačno geslo, da računalo sešteva, če smo pritisnili gumb za seštevanje, oz. odšteva, če smo pritisnili gumb za odštevanje. Z drugimi besedami, želimo upravljati potek programa (torej izbrati, katera koda naj se izvede) glede na vrednosti spremenljivk. Angleško takemu upravljanju pravimo *control flow*, najpogosteje pa ga izvajamo s t.i. *pogojnimi stavki*. Osnovna struktura je sledeča:

```
if (pogoj) {
    // koda, ki se izvede, ce pogoj velja
} else {
    // koda, ki se izvede, ce pogoj ne velja
}
```

Pogoj je nov pojem. Označuje neke vrste račun, katerega rezultat ni število, vendar *logična vrednost*. Tu sta možni vrednosti le dve: pravilno (angl. `true`) in napačno (angl. `false`). Če bo rezultat računa, navedenega v običajnih oklepajih v zgornjem `if` stavku, `true`, se bo izvedla koda znotraj prvih zavitih oklepajev, če pa je rezultat računa `false`, pa se bo izvedla koda v drugih zavitih oklepajih (tistih za besedo `else`). Drugega dela, tj. `else` in oklepaje za njim, ni treba pisati, če tega ne želimo.

Kako pa zapišemo pogoj? Za to uporabimo posebne *logične operatorje*. Pri delu s številkami so nam na voljo naslednji:

- `==`: primerja dve številski vrednosti. Rezultat je `true`, če sta vrednosti enaki.
- `!=`: primerja dve številski vrednosti. Rezultat je `true`, če sta vrednosti različni.
- `<`: primerja dve številski vrednosti. Rezultat je `true`, če je vrednost na levi manjša od vrednosti na desni.
- `>`: deluje podobno kot `<`, le da v drugo smer; rezultat je `true`, če je vrednost na desni manjša od vrednosti na levi.
- `<=`: primerja dve številski vrednosti. Rezultat je `true`, če sta vrednosti enaki, ali če je vrednost na levi manjša od vrednosti na desni.
- `>=` deluje podobno kot `<=`, le da v drugo smer.

Poglejmo si primer uporabe pogojnega stavka.

```
#include <stdio.h>

int main() {
```

```
4       int a, b;
5       scanf("%d%d", &a, &b);
6       if (a == b) {
7           printf("Stevili sta enaki.\n");
8       }
9       if (a != b) {
10          printf("Stevili sta razlicni.\n");
11      }
12      if (a < b) {
13          printf("Prvo stevilo je manjse od drugega.\n");
14      }
15      if (a > b) {
16          printf("Prvo stevilo je vecje od drugega.\n");
17      }
18      if (a <= b) {
19          printf("Prvo stevilo je manjse ali enako drugemu.\n");
20      }
21      if (a >= b) {
22          printf("Prvo stevilo je vecje ali enako drugemu.\n");
23      }
24      return 0;
25  }
```

Program v zgornjem primeru primerja dve števili z vsemi naštetimi operatorji. Če v program vpišemo npr. števili 3 in 7, vstopimo v drugi, tretji in peti pogojni stavek, zaradi česar se izpišejo naslednje vrstice:

```
Stevili sta razlicni.
Prvo stevilo je manjse od drugega.
Prvo stevilo je manjse ali enako drugemu.
```

Če pa v program dvakrat vnesemo število 12 (ali katerokoli drugo število), pa dobimo naslednji izhod:

```
Stevili sta enaki.
Prvo stevilo je manjse ali enako drugemu.
Prvo stevilo je vecje ali enako drugemu.
```

Pozorni moramo biti, da pri primerjavi enakosti dveh števil napišemo dva enačaja, ==. Če zapišemo le en enačaj, kot v matematiki (torej =), se bo program sicer zagnal, vendar ne bo deloval pravilno. Enojnega enačaja nikoli ne uporabljamo v pogoju `if` stavka!

Oglejmo si še primer uporabe stavka `else`. Spodnji program bo uporabnika vprašal za PIN, in mu napisal, če je bil PIN pravilen oziroma napačen.

```c
#include <stdio.h>

int main() {
    // Vprasaj uporabnika za PIN, in preveri, ce je enak 42
    int pin;
    scanf("%d", &pin);
    if (pin == 42) {
        printf("PIN je pravilen!");
    } else {
        printf("PIN je napacen!");
    }
    return 0;
}
```

Pogojne stavke lahko tudi gnezdimo, torej vstavimo enega v drugega. Spodnji program od uporabnika sprejme naročilo v restavraciji, kjer ponujajo dve vrsti hrane; juhe in sendviče. Na voljo sta dve vrsti juhe, in dve vrsti sendvičev. Za izbiro kosila uporabnik prvo izbere med juho in sendvičem, nato pa še okus.

```c
#include <stdio.h>
int main() {
    // Uporabnik naj napise 1, ce zeli juho, in 2, ce zeli sendvic.
    int zelja;
    scanf("%d", &zelja);
    if (zelja == 1) {
        // Uporabnik naj napise 1, ce zeli govejo juho,
        // in 2, ce zeli paradiznikovo.
        scanf("%d", zelja);
        if (zelja == 1) {
            printf("Ena goveja juha. Dober tek!\n");
        } else {
            printf("Ena paradiznikova juha. Dober tek!\n");
        }
    } else {
        // Uporabnik naj napise 1, ce zeli sendvic s sunko,
        // in 2, ce zeli vegeterjanski sendvic.
        scanf("%d", &zelja);
        if (zelja == 1) {
            printf("En sendvic s sunko. Dober tek!\n");
        } else {
            printf("En vegeterjanski sendvic. Dober tek!\n");
```

```
        }
    }
    return 0;
}
```

Pozorni bodimo na postavitev kode. Običajno kodo znotraj zavitih oklepajev `if` stavka pišemo tako, da je poravnana štiri presledke bolj desno od kode zunaj `if` stavka. V nekaterih programskih jezikih je taka poravnava obvezna, v C/C++ pa ne, vendar nezamaknjena koda že v majhnih programih postane popolnoma nepregledna. Branje in popravljanje kode je veliko lažje, če del kode znotraj zavitih oklepajev zamaknemo za štiri presledke. Za to lahko uporabimo tudi tipko Tab, ki se na tipkovnici nahaja levo od tipke Q. Kode kot spodaj nikoli ne pišemo!

```
// TAKO NIKOLI NE PIŠEMO!
#include <stdio.h>
int main() {
if (3 > 2) {
printf("Velja\n");
} else {
printf("Ne velja\n");
}
return 0;
}
```

Pogosto se srečamo s problemi, kjer je za rešitev potrebno upoštevati več kot en pogoj. V takih primerih želimo združiti več pogojnih stavkov tako, da se nek del kode izvede, če velja prvi pogoj, drugi del kode pa, če prvi pogoj ne velja, velja pa drugi pogoj. Z gnezdenjem stavkov lahko to v kodo vključimo na naslednji način:

```
if (prvi pogoj) {
    // koda, ki se izvede, če velja prvi pogoj
} else {
    if (drugi pogoj) {
        // koda, ki se izvede, če prvi pogoj ne velja,
        // velja pa drugi pogoj
    } else {
        // koda, ki se izvede, če ne veljata ne prvi ne drugi pogoj
    }
}
```

V takem primeru nam je na voljo bližnjica `else if`. Zgornja koda deluje popolnoma enako kot spodnja:

```
if (prvi pogoj) {
    // koda, ki se izvede, če velja prvi pogoj
} else if (drugi pogoj) {
    // koda, ki se izvede, če prvi pogoj ne velja,
    // velja pa drugi pogoj
} else {
    // koda, ki se izvede, če ne veljata ne prvi ne drugi pogoj
}
```

Prednost te bližnjice je, da je naša koda krajša in bolj razumljiva. Stavke `else if` lahko tudi verižimo; enemu `if` stavku lahko sledi poljubno mnogo stavkov `else if`. Pri tem bo računalnik pogoje preverjal po vrsti. Pri prvem veljavnem pogoju se bo ustavil in izvedel kodo v pripadajočih zavitih oklepajih, za čimer ne bo več preverjal pogojev, temveč bo izvajanje nadaljeval za zaključkom vseh nanizanih stavkov. Kakor nam v osnovnem `if` stavku ni bilo treba pisati dela z `else`, če ga nismo potrebovali, nam ga tudi pri uporabi `else if` ni treba.

Oglejmo si primer uporabe. Naslednji program prebere število in pove, če je večje, manjše ali enako 0.

```
#include <stdio.h>

int main() {
    int stevilo;
    scanf("%d", &stevilo);
    if (stevilo < 0) {
        printf("Stevilo je manjse od nic.\n");
    } else if (stevilo == 0) {
        printf("Stevilo je enako 0.\n");
    } else if (stevilo > 0) {
        printf("Stevilo je vecje od 0.\n");
    }
    return 0;
}
```

Naslednji primer prikaže, da se izvede samo koda pri prvem veljavnem pogoju, ne glede na to, koliko pogojev za tem je tudi veljavnih. Program izpiše le eno vrstico besedila — `a je 7`, kljub temu, da velja tudi `a > 1`.

```c
#include <stdio.h>

int main() {
    int a = 7;
    if (a < 3) {
        printf("a je manjsi od 3.\n");
    } else if (a == 7) {
        printf("a je 7.\n");
    } else if (a == 4) {
        printf("a je 4.\n");
    } else if (a > 1) {
        printf("a je vecji od 1.\n");
    } else {
        printf("Nic od nastetega ne velja.\n");
    }
    return 0;
}
```

2 Logični vezniki

Osnovni operatorji za primerjavo pogosto niso dovolj, da izrazimo željen pogoj. Če na primer želimo pogledati, ali je neko število med dvema drugima, tega ne moramo narediti samo z eno primerjavo. Pogoje združujemo s t.i. *logičnimi vezniki*, ki jim včasih pravimo tudi *logični operatorji*. Poznamo tri osnovne veznike:

- `and` oz. `&&` združi dva pogoja tako, da združeni pogoj velja samo v primeru, da veljata oba hkrati.
- `or` oz. `||` združi dva pogoja tako, da združeni pogoj velja v primeru, da velja katerikoli od dveh, ali da veljata oba.
- `not` oz. `!` sprejme samo en pogoj. Nov pogoj velja samo takrat, ko originalni pogoj ne velja.

Poglejmo si enostaven primer. Če želimo preveriti, ali je dano število med dvema drugima, uporabimo logični veznik `&&`.

```c
#include <stdio.h>

int main() {
    int n;
    scanf("%d", &n);
```

```
    if (3 < n && n < 9) {
        printf("n je med 3 in 9.\n");
    }
    return 0;
}
```

V matematiki pogosto napišemo dvojno primerjavo $a < b < c$. Če nekaj podobnega napišemo v C++ program, se bo le-ta sicer zagnal, vendar ne bo deloval pravilno. Kaj pričakujemo, da se zgodi, če v tako primerjavo zapišemo `3 < 2 < 1`? Preveri, kaj se dejansko zgodi!

Pri kombiniranju pogojev bodimo previdni glede pravil prednosti. Zanikanje (veznik `not` oz. `!`) ima namreč prednost pred primerjalnimi vezniki (`<`, `==`, …), ter drugima ločnima veznikoma. Če v takem primeru uporabljamo zanikanje, moramo zanikan izraz postaviti v oklepaje.

Za konec si oglejmo še malo bolj kompleksen primer. Napišimo program, ki preveri, ali je uporabnik vpisal prestopno leto. Leto je prestopno, če je deljivo s 4, razen če je hkrati deljivo s 100. Izjema so leta, deljiva s 400, ki so prestopna kljub temu, da so deljiva s 100.

Kako te pogoje zapišemo v program? Opazimo, da so leta, deljiva s 400, prestopna ne glede na druga pogoja. Če leto ni deljivo s 400, potem mora biti deljivo s 4 in ne sme biti deljivo s 100. Povedano krajše, leto mora biti deljivo s 400, ali pa s 4 in ne hkrati s 100. Tak pogoj lahko zapišemo z logičnimi vezniki.

```
#include <stdio.h>

int main() {
    int leto;
    scanf("%d", &leto);
    if (leto % 400 == 0 || (leto % 4 == 0 && !(leto % 100 == 0))) {
        printf("Leto je prestopno.\n");
    } else {
        printf("Leto ni prestopno.\n");
    }
    return 0;
}
```