

9. Dodatne vježbe

1. Objašnjenje primjera s predavanja (slajd 74)

Primjer: Učitavati pozitivne cijele brojeve sve dok njihova suma ne premaši dopušteni raspon za short int. Ispisati posljednju valjanu sumu.

Rješenje ovog problema je jednostavno ako se unaprijed može pretpostaviti da najveći pozitivni `short int` koji se može prikazati iznosi 32767. Najbolje je primijeniti petlju s ispitivanjem uvjeta na kraju, jer se tijelo petlje (tijelo petlje sadrži učitavanje broja s tipkovnice) **mora obaviti barem jednom**.

```
#include <stdio.h>
int main() {
    int x, suma = 0, gotovo = 0;
    do {
        printf("Unesite broj : ");
        scanf("%d", &x);
        if (suma + x <= 32767) {
            /* nema opasnosti, moze se uvecati */
            suma = suma + x;
        } else {
            /* ne smije se uvecati jer ce se prekoračiti dopustena granica */
            gotovo = 1;
        }
    } while (!gotovo);
    printf("Suma: %d\n", suma);
    return 0;
}
```

Logička varijabla `gotovo` se koristi u logičkom uvjetu petlje: dok je vrijednost varijable `gotovo` jednaka `false`, petlja se nastavlja. Vrijednost varijable `suma` nije se mogla koristiti kao uvjet nastavka petlje (npr. `while (suma <= 32767)`) jer `suma` nikad ne smije poprimiti vrijednost veću od 32767. Suma nikad ne smije postati veća od 32767 jer je potrebno sačuvati posljednju "ispravnu" vrijednost sume, tj. petlja se mora prekinuti PRIJE nego `suma` prekorači tu vrijednost.

Slično, ali neznatno lošije rješenje od prethodnog, može se napisati korištenjem petlje s ispitivanjem uvjeta na početku (takva je petlja korištena u primjeru na predavanju). Ovo rješenje se smatra neznatno lošijim jer se koristi vrsta petlje koja je pogodna u onim slučajevima kada postoji mogućnost da se tijelo petlje neće trebati obaviti niti jednom):

```
#include <stdio.h>
int main() {
    int x, suma = 0, gotovo = 0;
    while (!gotovo) {
        printf("Unesite broj : ");
        scanf("%d", &x);
        if (suma + x <= 32767) {
            suma = suma + x;
        } else {
            gotovo = 1;
        }
    }
    printf("Suma: %d\n", suma);
    return 0;
}
```

U primjeru s predavanja se pretpostavilo da najveća pozitivna vrijednost za tip podatka `short int` **nije unaprijed poznata**. Umjesto tipa podatka `int`, za varijable `x` i `suma`, koristi se tip podatka `short int`. Prvi korak prilagodbe rješenja izgleda ovako:

```
#include <stdio.h>
int main() {
    short int x, suma = 0, gotovo = 0;

    while (!gotovo) {
        printf("Unesite broj : ");
        scanf("%hd", &x);          /* paznja: u scanf za short format %hd, a ne %d */
        if (suma+x >= suma) {
            suma = suma + x;
        } else {
            gotovo = 1;
        }
    }
    printf("Suma: %d\n", suma);
    return 0;
}
```

Ideja je u tome da se sumiranje obavlja dok god je vrijednost sume uvećane za `x`, veća od prethodne vrijednosti sume. Naime, ako se uvećanjem sume za `x` dobije broj manji od stare sume, to znači ili da je `x` bio negativan broj (što ne može biti jer se u zadatku tražilo da uneseni `x` bude pozitivan), ili je `suma + x` postala negativna jer je prekoračena najveća dopuštena vrijednost sume.

Prethodno rješenje još uvijek nije ispravno! Izraz `suma + x` se izračunava tako da se vrijednosti `suma` i `x` prvo implicitnom konverzijom pretvore u tip podatka `int`. Dobiveni broj je stoga uvijek pozitivan, te uvjet **nije dobro** napisan. Potrebno je `int` rezultat izraza `suma + x` eksplicitno pretvoriti u tip `short int`. U tu svrhu se koristi cast operator `short int` ili `signed short int` ili `short` ili `signed short`.

```
#include <stdio.h>
int main() {
    short int x, suma = 0, gotovo = 0;

    while (!gotovo) {
        printf("Unesite broj : ");
        scanf("%hd", &x);
        if ((signed short)(suma+x) >= suma) {
            suma = suma + x;
        } else {
            gotovo = 1;
        }
    }
    printf("Suma: %d\n", suma);
    return 0;
}
```

Završni komentar: prikazano rješenje demonstrira neke efekte koji se pojavljuju pri korištenju tehnike dvojnog komplementa, ali je nepotrebno komplicirano. Daleko bolji pristup je ako se takav zadatak riješi na manje kompliciran način:

```

#include <stdio.h>
#include <math.h>

int main() {
    short int x, najveciShort, gotovo = 0;
    int suma = 0;
    najveciShort = (short int)pow(2, sizeof(x)*8 - 1) - 1;
    do {
        printf("Unesite broj : ");
        scanf("%hd", &x);
        if (suma + x <= najveciShort)
            suma = suma + x;
        else
            gotovo = 1;
    } while (!gotovo);
    printf("Suma: %d\n", suma);
    return 0;
}

```

2. Dopunsko objašnjenje primjera s predavanja (slajd 110)

Primjer: Napisati program koji će ispisivati prvih 0-N Fibonaccijevih brojeva

Potrebno je uočiti da za računanje člana niza f_i , treba poznavati članove niza f_{i-1} i f_{i-2} . U svakom koraku petlje u kojem se računa član niza f_i , iskoristit će se članovi f_{i-1} i f_{i-2} , a zatim će se pripremiti "nove" vrijednosti za f_{i-1} i f_{i-2} , tako što će se stari član f_{i-1} zapisati u član f_{i-2} , a netom izračunati član f_i zapisati u član f_{i-1} .

U programu se član f_i čuva u varijabli f , član f_{i-1} u varijabli $f1$, a član f_{i-2} u varijabli $f0$. Naredbom za selekciju osigurava se da se članovi počinju izračunavati prema opisanom principu tek nakon što se obavi ispis članova niza f_0 i f_1 .

Djelovanje programa može se lakše shvatiti ako se ispiše tablica koja pokazuje kako se varijable mijenjaju u pojedinim koracima petlje. Prikazan je primjer za učitanu vrijednost $n==5$:

i	f0 nakon koraka petlje	f1 nakon koraka petlje	f nakon koraka petlje	Ispis u koraku petlje
0	1	1	1	Fibonnaci (0) = 1
1	1	1	1	Fibonnaci (1) = 1
2	1	2	2	Fibonnaci (2) = 2
3	2	3	3	Fibonnaci (3) = 3
4	3	5	5	Fibonnaci (4) = 5
5	5	8	8	Fibonnaci (5) = 8

ZADACI ZA VJEŽBU

3. U varijablu tipa `int` učitati nenegativni cijeli broj (ne treba kontrola je li broj nenegativan). Na zaslon ispisati binarni sadržaj registra te varijable (binarne znamenke se moraju ispisati ispravnim redoslijedom). Za izdvajanje bitova koristiti operaciju broj $\& 2^n$, pri čemu se n mijenja od 31 do 0 (može se pretpostaviti da se za pohranu `int` tipa podatka koristi 32 bita).

Primjer: ako je korisnik upisao broj 18, na zaslon treba ispisati

[illegible]

4. S tipkovnice učitati cijeli broj n koji mora biti između 0 i 16 (uključivo s granicama). Ako broj nije ispravan, ispisati odgovarajuću poruku. Nakon toga učitati n binarnih znamenki i ispisati dekadski ekvivalent učitano binarnog broja (ne primjenjuje se tehnika dvojnog komplementa, pa je dekadski ekvivalent sigurno pozitivan broj).

Npr., ako je korisnik upisao

$$\begin{matrix} 4 \\ 1 \\ 1 \\ 0 \\ 1 \end{matrix}$$

program treba ispisati 13.

Npr., ako je korisnik upisao

0

program treba ispisati 0.

Riješiti pomoću petlje **s unaprijed poznatim brojem ponavljanja**. Je li bolje ovaj zadatak rješavati pomoću petlje s unaprijed poznatim brojem ponavljanja ili pomoću petlje s ispitivanjem uvjeta na početku.

- Isto kao prethodni zadatak, ali riješiti pomoću petlje s ispitivanjem uvjeta na kraju. Zašto takva vrsta petlje nije pogodna za rješavanje ovog zadatka?
- Načinite program koji će s tipkovnice učitati nenegativni cijeli broj iz intervala [0, 4294967295]. Učitani broj treba ispisati u oktalnom obliku. Npr. za učitani broj 250 treba ispisati 00000000372; za učitani broj 4294967295 treba ispisati 3777777777. Zadatak riješite tako da grupe od po tri bita pretvarate u oktalne znamenke. Za određivanje grupa po tri bita koristite operator posmaka u desno (za tri mjesta) i bitovni operator &. **Uputa:** vrijednost tipa podatka `unsigned` može se učitati po formatu `%u` (umjesto po formatu `%d`).
- Isto kao prethodni zadatak, ali učitani dekadski broj treba pretvoriti u heksadekadski.
- Napišite program koji učitava dva znaka te ispisuje sve znakove ASCII tablice koji se nalaze između ta dva znaka. Npr., ako se učitaju znakovi `d i k`, program ispisuje `defghi jk`.

9. Načinite program koji će ispisati sljedeću tablicu:

```
A. a b c d e f .F
B. b c d e f g .G
C. c d e f g h .H
D. d e f g h i .I
... itd.
S. s t u v w x .X
T. t u v w x y .Y
U. u v w x y z .Z
```

Očekuje se da zadatak riješite pomoću dvije ugniježdene petlje, a ne npr. ovako:

```
printf("A. a b c d e f .F\n");
printf("B. b c d e f g .G\n");
printf("C. c d e f g h .H\n");
printf("D. d e f g h i .I\n");
... itd.
```

10. Prepravite program iz prethodnog zadatka tako da se na mjestima gdje bi se ispisao "mali" samoglasnik, umjesto toga ispiše znak '?'.

11. Načinite program za izračunavanje "m povrh n".

$$m! / (n! \cdot (m - n)!)$$

Vrijednosti za m i n učitati s tipkovnice uz kontrolu jesu li te vrijednosti ispravno zadane (cijeli brojevi veći ili jednaki 0, m je veći ili jednak n).

12. Ispišite sve pitagorine trojke čiji su članovi veći od 0 i manji ili jednaki 100. Ispis treba izgledati ovako (objašnjenje: oznaka 3^2 u sljedećem ispisu ima značenje 3^2):

```
1. trojka: 3^2 + 4^2 = 5^2
2. trojka: 4^2 + 3^2 = 5^2
3. trojka: 5^2 + 12^2 = 13^2
4. trojka: 6^2 + 8^2 = 10^2
5. trojka: 7^2 + 24^2 = 25^2
... itd.
101. trojka: 80^2 + 60^2 = 100^2
102. trojka: 84^2 + 13^2 = 85^2
103. trojka: 84^2 + 35^2 = 91^2
104. trojka: 96^2 + 28^2 = 100^2
```

Uputa: zadatak možete riješiti tako da pomoću tri ugniježdene petlje testirate svaku kombinaciju 3 cijela broja: 1 1 1; 1 1 2; 1 1 3; ... 1 1 99; 1 1 100; 1 2 1; 1 2 2; ...; 1 2 100; 1 3 1; ... Ispišite samo one kombinacije 3 cijela broja koji zadovoljavaju "uvjet pitagorine trojke".

13. Napisati bolje rješenje (bez break i continue) zadatka sa slajda 121.

Rješenja svih zadataka provjeriti prevođenjem i testiranjem vlastitih programa!

Rješenja: NE GLEDATI prije nego sami pokušate riješiti zadatke

Rješenje 3. zadatka

```
#include <stdio.h>
#include <math.h>
int main () {
    int broj;
    int n, dva_na_ntu;
    scanf("%d", &broj);
    printf("Upisali ste broj %d\n", broj);
    for (n = 31; n >= 0; n--) {
        dva_na_ntu = (int)pow(2, n);
        /* ako u (broj & dva_na_ntu) postoji bit razlicit od 0,
           tada ce (broj & dva_na_ntu) biti razlicit od 0, tj. true */
        printf("%d", broj & dva_na_ntu ? 1 : 0);
    }
    printf("\n");
    return 0;
}
```

Petlja s unaprijed poznatim brojem ponavljanja je najpogodnija za ovaj slučaj jer je u trenutku kad petlja započinje poznato koliko puta se tijelo te petlje treba obaviti.

Rješenje 4. zadatka

```
#include <stdio.h>
int main () {
    int n, i, znamenka, dekadski = 0;
    scanf("%d", &n);
    if (n < 0 || n > 16) {
        printf("Upisali ste neispravan broj\n");
    }
    else {
        for (i = 0; i < n; i++) {
            scanf("%d", &znamenka);
            dekadski = dekadski*2 + znamenka;
        }
        printf("%d\n", dekadski);
    }
    return 0;
}
```

Petlja s unaprijed poznatim brojem ponavljanja je najpogodnija za ovaj slučaj jer je u trenutku kad petlja započinje poznato koliko puta se tijelo te petlje treba obaviti (uočite, to može biti i "nula puta", npr. ako u ovom slučaju korisnik za vrijednost varijable `n` upiše nulu).

Rješenje 5. zadatka

```
#include <stdio.h>
int main () {
    int n, znamenka, dekadski = 0;
    scanf("%d", &n);
    if (n < 0 || n > 16) {
        printf("Upisali ste neispravan broj\n");
    }
    else {
        do {
            if (n > 0) {
                scanf("%d", &znamenka);
                dekadski = dekadski*2 + znamenka;
                n--;
            }
        } while (n > 0);
        printf("%d\n", dekadski);
    }
    return 0;
}
```

Petlja s ispitivanjem uvjeta na kraju nije pogodna za rješavanje ovog zadatka, jer je moguće da tijelo petlje neće biti potrebno obaviti niti jednom (onda kada se za vrijednost varijable n učitava 0). Petlja "ne zna" da tijelo petlje ne smije obaviti sve do trenutka kada se ispituje uvjet (a to je na kraju), stoga je bilo nužno koristiti if naredbu.

Rješenje 6. zadatka

```
#include <stdio.h>

int main() {
    unsigned int a;
    int i;
    printf("Upisite nenegativni cijeli broj a: ");
    scanf ("%u", &a); /* za unsigned se kod čitanja koristi %u umjesto %d */
    for (i = 10; i >= 0; i--) {
        printf("%d", a >> 3*i & 0x7);
    }
    printf ("\n");
    return 0;
}
```

Rješenje 7. zadatka

```
#include <stdio.h>

int main() {
    unsigned int a;
    int i, broj;
    printf("Upisite nenegativni cijeli broj a: ");
    scanf ("%u", &a);
    for (i = 7; i >= 0; i--) {
        broj = a >> 4*i & 0xF;
        if (broj <= 9) {
            printf("%d", broj);
            /* ili printf("%c", broj + '0'); */
        }
        else {
            printf("%c", broj - 10 + 'A');
        }
    }
    printf ("\n");
    return 0;
}
```

Rješenje 8. zadatka

```
#include <stdio.h>

int main() {
    char c1, c2;
    char i;
    scanf("%c %c", &c1, &c2);
    for (i = c1; i <= c2; i++)
        printf("%c", i);
    printf("\n");
    return 0;
}
```

Komentirajte: što će se dogoditi ako se umjesto znakova d i k učitaju znakovi k i d

Rješenje 9. zadatka

```
#include <stdio.h>

int main() {
    char i, j;
    for (i = 'A'; i <= 'U'; i++) {
        printf("%c. ", i);
        for (j = i + 32; j < i + 32 + 6; j++) {
            printf("%c ", j);
        }
        printf(":%c\n", i + 5);
    }
    return 0;
}
```

Vanjska petlja mijenja vrijednost varijable i od 'A' do 'U'. Na početku svakog retka se ispisuje vrijednost varijable i (naravno, u formatu %c), a na kraju retka ispisuje se slovo koje se u ASCII tablici nalazi "5 mjesta dalje" od slova koje se ispisalo na početku retka.

Unutarnja petlja mijenja vrijednost varijable j od "male verzije" slova koje je ispisano na početku retka, do slova koje se u ASCII tablici nalazi "6 mjesta dalje" od početnog malog slova.

Rješenje 10. zadatka

```
#include <stdio.h>

int main() {
    char i, j;
    for (i = 'A'; i <= 'U'; i++) {
        printf("%c. ", i);
        for (j = i + 32; j < i + 32 + 6; j++) {
            printf("%c ", j=='a' || j=='e' || j=='i' || j=='o' || j=='u' ? '?' : j);
        }
        printf(" %.5c\n", i + 5);
    }
    return 0;
}
```

ili

```
for (j = i + 32; j < i + 32 + 6; j++) {
    if (j=='a' || j=='e' || j=='i' || j=='o' || j=='u')
        printf("? ");
    else
        printf("%c ", j);
}
```

Rješenje 11. zadatka

```
#include <stdio.h>

int main() {
    int m, n, i;
    double brojnik, naziv1, naziv2, mpovrhn;

    /* unos vrijednosti za m i n */
    printf ("Unesite m i n:");
    scanf ("%d %d", &m, &n);
    if (m < 0 || n < 0 || m < n)
        printf("brojevi su neispravno zadani\n");
    else {
        brojnik = 1;
        for (i = 1; i <= m; i++)
            brojnik *= i;

        naziv1 = 1;
        for (i = 1; i <= n; i++)
            naziv1 *= i;

        naziv2 = 1;
        for (i = 1; i <= m-n; i++)
            naziv2 *= i;

        mpovrhn = brojnik/(naziv1*naziv2);
        printf("%d povrh %d iznosi = %g\n", m, n, mpovrhn);
    }
    return 0;
}
```

Rješenje 12. zadatka

```
#include <stdio.h>

int main() {
    int i, j, k;
    int n = 0;
    for (i=1; i <= 100; i++)
        for (j=1; j <= 100; j++)
            for (k=1; k <= 100; k++)
                if (i*i + j*j == k*k) {
                    n++;
                    printf("%d. trojka: %d^2 + %d^2 = %d^2\n", n, i, j, k);
                }
    return 0;
}
```

Rješenje 13. zadatka

```
#include <stdio.h>

int main() {
    int x;
    do {
        printf ("Upisite broj : \n");
        scanf ("%d", &x);
        if (x > 100)
            printf ("Zanemarujem vrijednost\n");
        else if (x >= 0)
            printf ("Upisani broj je : %d\n", x);
        else
            printf ("Nedopustena vrijednost\n");
    } while (x > 0);
    return 0;
}
```