Lekcija 3 – Razvoj programa

Outline

Ciljevi lekcije

- U ovoj lekciji:
  - Podsjećanje na osnovne algoritamske tehnike.
  - Koristićete `if` i `if…else` naredbu za selekciju akcija.
  - Koristićete `while` naredbu za ponovljeno izvršavanje.
  - Shvatićete counter-controlled i sentiel-controlled ponavljanja.
  - Razumijećete struktuirano programiranje.
  - Naučićete da koristite operatore inkrementiranja, dekrementiranja i dodjeljivanja.

3.1 Uvod

- Prije pisanja programa:
  - Shvatite dubinski problem
  - Pažljivo napravite plan za njegovo rješavanje
- Dok pišete program:
  - Imajte na umu da postoje gotove funkcije-“building blocks”
  - Koristite dobre programerske principe

3.2 Algoritmi

- Izračunljivi problemi
  - Mogu biti riješeni izvršavanjem niza akcija u određenom redosledu
- Algoritam: procedure u terminima
  - Akcija koje treba izvršiti
  - Redosleda kojim te akcije treba izvršavati
- Programska kontrola
  - Specificira redosled kojim akcije treba izvršavati

3.3 Pseudo kod

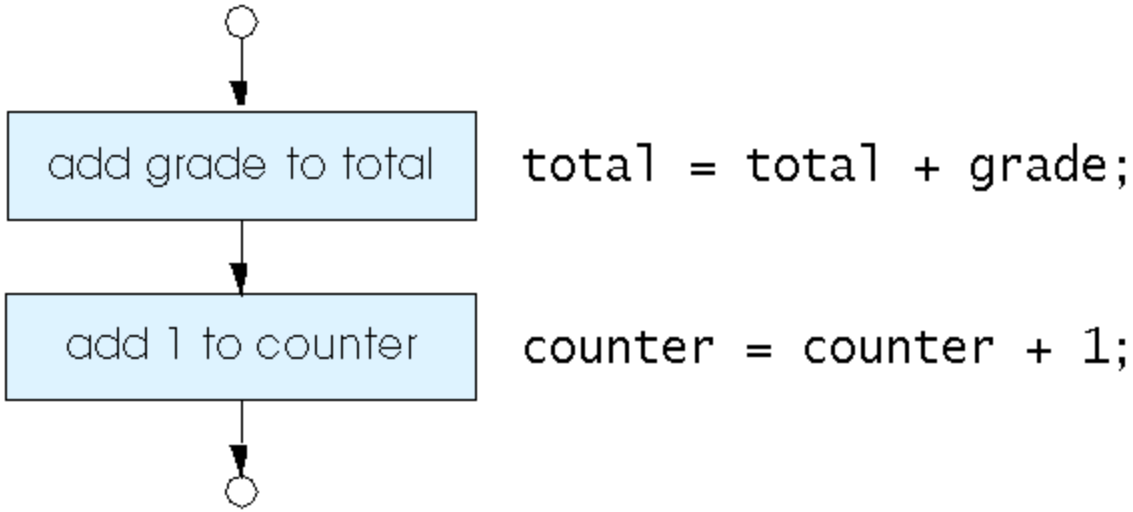
- Pseudocode
  - Vještački, neformalni jezik pri razvoju algoritama
  - Sličan svakodnevnom engleskom
  - Ne izvršava se na računaru
  - Pomaže pri “razmišljanju o programu” prije pisanja programa
    - Lako se konvertuje u odgovarajući C program
    - Sastoji se samo od izvršnih naredbi

3.4 Kontrolne strukture

- Sekvencijalno izvršavanje
  - Naredbe se izvršavaju jedna za drugom u redosledu kako su napisane
- Transfer kontrole
  - Kada sledeća naredba nije sledeća u nizu
- Bohm i Jacopini
  - Svi programi su napisani u terminima 3 kontrolne strukture
    - Sekvencijane strukture: Ugrađene u C. Programi se izvršavaju sekvencijano (po default-u)
    - Strukture selekcije : C ima 3 tipa: `if`, `if…else`, i `switch`
    - Strukture ponavljanja (repeticije): C ima 3 tipa: `while`, `do…while` i `for`

3.4 Kontrolne strukture

Figure 3.1 Flow chart sekvencijalna struktura u C-u.



3.5 `if` naredba

- Struktura selekcije:
  - Izbor između 2 mogućnosti
  - Pseudocode:

    If broj bodova studenta veci ili jednak 45
    Print "Polozio"
- Ako je uslov `true`
  - Izvršava se Print naredba i program prelazi na sledeću naredbu
  - Ako je uslov `false`, print naredba se ignoriše i program prelazi na sledeću naredbu
  - Nazubljivanje koda čini ga čitljivijim
    - C ignoriče whitespace karaktere

3.5 `if` naredba

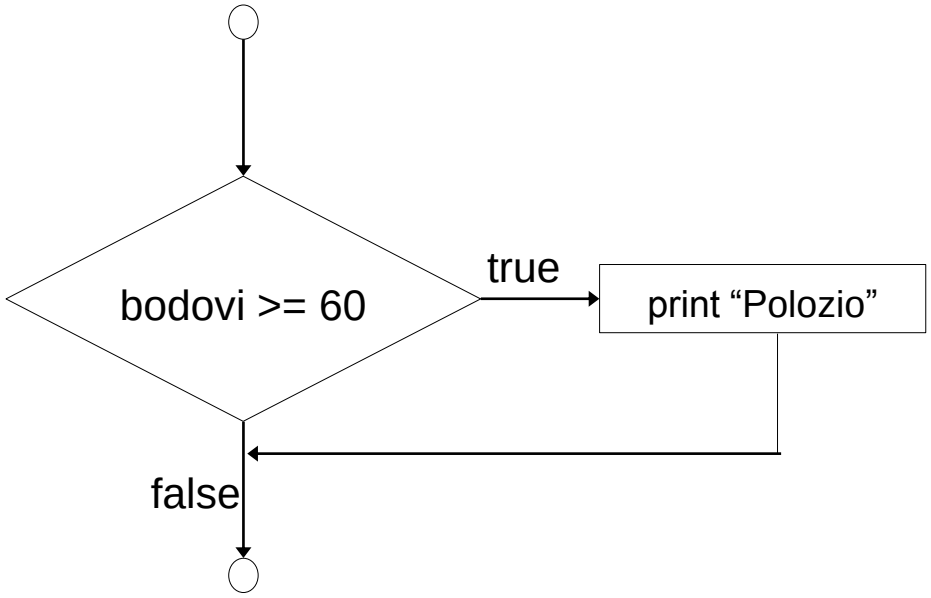
- Pseudocode naredba prevedena u C:

```
if ( bodovi >= 60 )
   printf( "Passed\n" );
```

  - C kod odgvovara pseudo kodu
- Simbol romba (decision symbol)
  - Označava da treba donijeti odluku
  - Sadrži izraz koji može biti `true` ili `false`
  - Testira se uslov i slijedi odgovarajuća staza

3.5 `if` naredba

- `if` naredba je single-entry/single-exit struktura



Odluka se može donijeti na svakom izrazu.

nula - `false`

nenula - `true`

Example:

`3 - 4` je `true`

3.6 `if...else` naredbaSelection Statement

- `if`
  - Akcija se izvršava ako je uslov tačan (`true`)
- `if...else`
  - Specificira akcije koje treba izvršiti kad je uslov `true` i kada je `false`
- Psuedocode:

  If broj bodova studenta veci ili jednak 45
  Print "Polozio"
  else
  Print "Pao"

  - Obratite pažnju na konvenciju o uvlačenju redova

3.6 The `if...else` Selection Statement

- C kod:

```
if ( bodovi >= 60 )
   printf( “Polozio\n");
else
   printf( “Pao\n");
```

- Ternarni kondicionalni operator (`?:`)
  - Ima tri argumenta (uslov, value if `true`, value if `false`)
  - Naš primjer može biti zapisan kao:

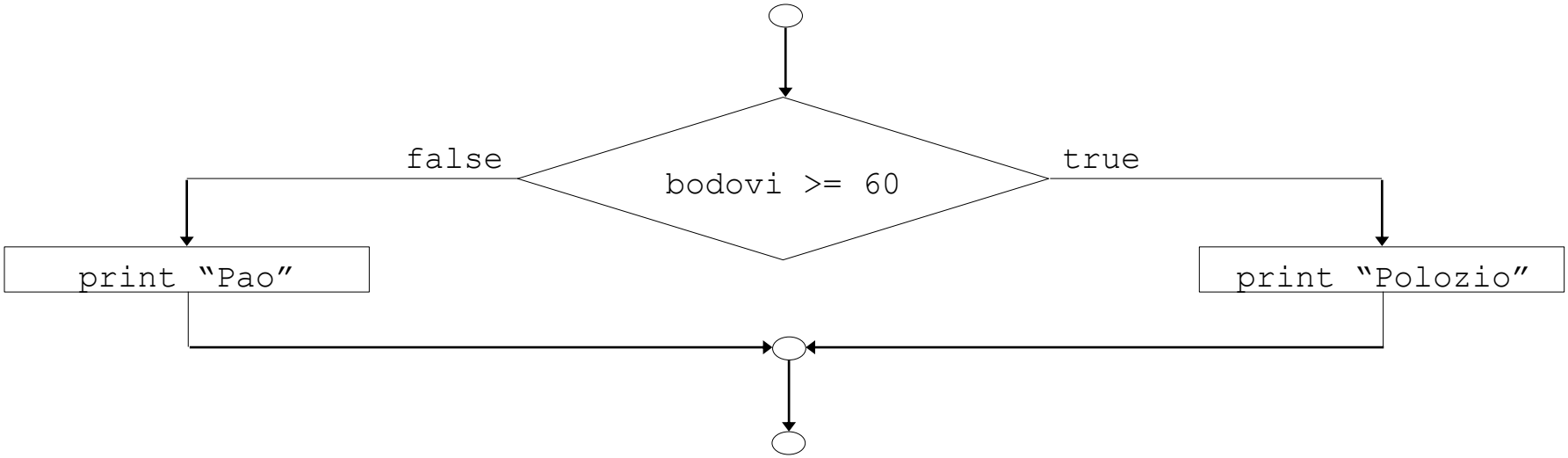
```
printf( "%s\n", bodovi >= 60 ? "Polozio" : “Pao"
  );
```

  - Ili kao:

```
bodovi >= 60 ? printf( “Polozio\n” ) : printf(
  “Pao\n” );
```

3.6 `if...else` naredba

- Flow chart za `if...else` naredbu



- Ugnježdene `if...else` naredbe
  - Test za više slučajeva postavljanjem `if...else` naredbe unutar `if...else` naredbe
  - Kada je ispunjen jedan od uslova, ostatak se preskače
  - Preduboko uvlačenje redova nije uobičajeno

3.6 `if...else` naredba

```
If broj bodova studenta veci ili jednak 85
    Print "10"
  else
    If broj bodova studenta veci ili jednak 75
      Print "9"
    else
      If broj bodova studenta veci ili jednak 65
        Print "8"
      else
        If broj bodova studenta veci ili jednak 55
          Print "7"
        else
          If broj bodova studenta veci ili jednak 45
            Print "6"
          else  Print "5"
```

3.6 if...else naredba

- Složena naredba:
  - Skup naredbi unutar para zagrada { i }
  - Example:

```
if ( bodovi >= 60 )
   printf( "Polozio.\n" );
else {
   printf( “Pao.\n" );
   printf( “Morate polagati cio ispit.\n" );
  }
```

  - Bez zagrada, naredba

```
    printf( “Morate polagati cio ispit.\n" );
```

    bi se izvršila automatski

3.6 `if...else` naredba

- Blok:
  - Složena naredba sa deklaracijama na početku
- Sintaksne greške
  - Prijavljuje ih prevodilac (kompajler)
- Logičke greške:
  - U vrijeme izvršavanja
  - Non-fatal: program radi ali daje pogrešan izlaz
  - Fatalne: program završava rad neočekivano

3.7 `while` naredba

- Strukture ponavljanja (repeticije)
  - Programer specificira akciju koja će se izvršavati dok je ispunjen neki uslov (`true`)
  - Psuedo kod:

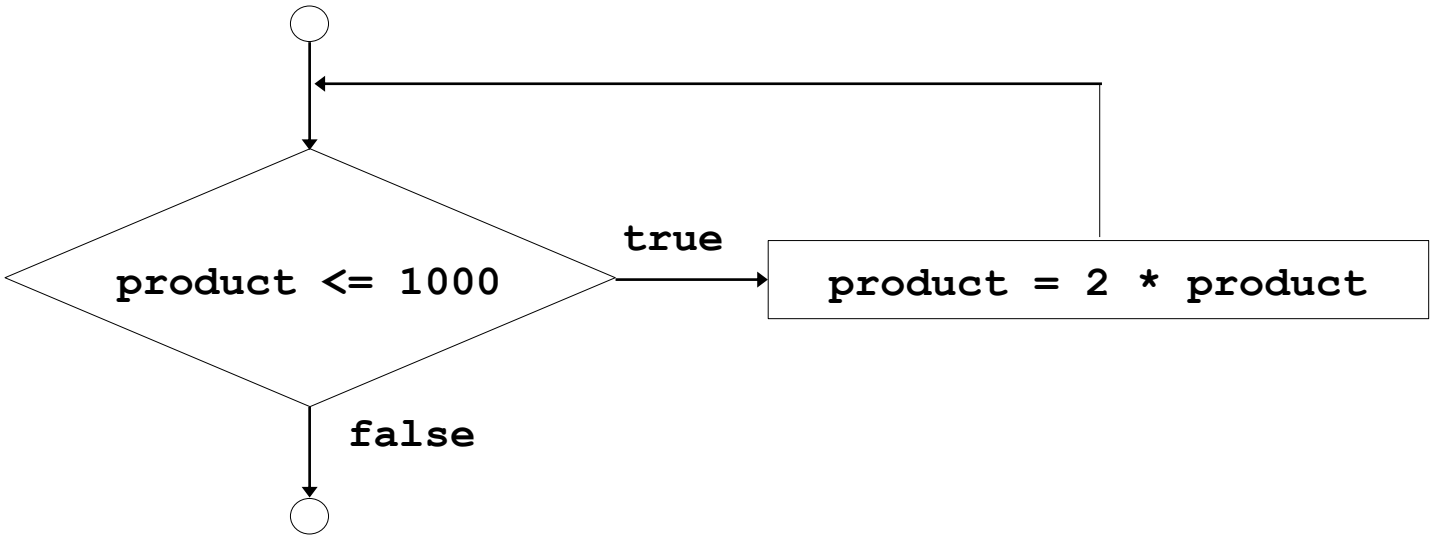
    While there are more items on my shopping list
    Purchase next item and cross it off my list

  - `while` petlja se ponavlja dok uslov ne postane `false`

3.7 while naredba

- Primjer:

```
int product = 2;
while ( product <= 1000 )
       product = 2 * product;
```



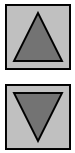
3.8 Formulisanje algoritma

- Ponavljanje kontrolisano brojačem (Counter-controlled repetition)
  - Petlja se ponavlja dok brojač ne dostigne određenu vrijednost
  - Konačna repeticija: broj ponavljanja je poznat
  - Primjer: 10 studenata rade test. Bodovi (između 0 i 100) su dati. Naći prosječan broj bodova na testu.
  - Pseudocode:

Set total to zero
Set grade counter to one

While grade counter is less than or equal to ten
Input the next grade
Add the grade into the total
Add one to the grade counter

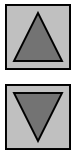
Set the class average to the total divided by ten
Print the class average



Outline

fig03_06.c (Part 1 of 2)

```
1  /* Fig. 3.6: fig03_06.c
2     Class average program with counter-controlled repetition */
3  #include <stdio.h>
4
5  /* function main begins program execution */
6  int main()
7  {
8     int counter; /* number of grade to be entered next */
9     int grade;   /* grade value */
10    int total;   /* sum of grades input by user */
11    int average; /* average of grades */
12
13    /* initialization phase */
14    total = 0;    /* initialize total */
15    counter = 1; /* initialize loop counter */
16
17    /* processing phase */
18    while ( counter <= 10 ) {     /* loop 10 times */
19       printf( "Enter grade: " ); /* prompt for input */
20       scanf( "%d", &grade );     /* read grade from user */
21       total = total + grade;     /* add grade to total */
22       counter = counter + 1;    /* increment counter */
23    } /* end while */
24
```



Outline

```
25    /* termination phase */
26    average = total / 10;         /* integer division */
27
28    /* display result */
29    printf( "Class average is %d\n", average );
30
31    return 0; /* indicate program ended successfully */
32
33 } /* end function main */
```

fig03_06.c (Part 2 of 2)

Program Output

```
Enter grade: 98
Enter grade: 76
Enter grade: 71
Enter grade: 87
Enter grade: 83
Enter grade: 90
Enter grade: 57
Enter grade: 79
Enter grade: 82
Enter grade: 94
Class average is 81
```

3.9 Formulisanje algoritma Top-Down metodom

- Preformulacija problema:

  Napisati program za izračunavnje srednje ocjene koji će obraditi proizvoljan broj studenata.

  – Nepoznat broj studenata

  – Kako će program znati kada da završi?

- Koristiti sentinel

  – Takođe poznata kao signal vrijednost, dummy vrijednost ili flag vrijednost

  – Označava kraj ulaznih podataka (“end of data entry.”)

  – Petlja završava kada korisnik unese sentinel

  – Sentinel se bira tako da ne može biti pomiješan sa regularnim ulazom (kao `-1` u našem slučaju)

3.9 Formulisanje algoritma Top-Down metodom

- Top-down, stepwise refinement
  - Počinjemo sa pseudo kodom na najvišem nivou:

    Determine the class average for the quiz

    Dijelimo problem na manje i dajemo njihov redosled:

    Initialize variables
    Input, sum and count the quiz grades
    Calculate and print the class average
- Skoro svi programi imaju 3 faze:
  - Inicijalizacija: inicijalizacija promjenljivih
  - Obrada: učitavanje ulaznih vrijednosti i prilagođavanje vrijednosti promjenljivih
  - Završetak: izračunavanje i štampanje krajnjih rezultata

3.9 Formulisanje algoritma Top-Down metodom

- Poboljšati fazu inicijalizacije sa:

  Initialize total to zero
  Initialize counter to zero

- Poboljšati drugu fazu sa

  Input the first grade (possibly the sentinel)
  While the user has not as yet entered the sentinel
  Add this grade into the running total
  Add one to the grade counter
  Input the next grade (possibly the sentinel)

3.9 Formulisanje algoritma Top-Down metodom

- Poboljšaj treću fazu sa

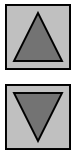
If the counter is not equal to zero
    Set the average to the total divided by the counter
    Print the average
else
    Print "No grades were entered"

3.9 Formulisanje algoritma Top-Down metodom

Initialize total to zero
Initialize counter to zero

Input the first grade
While the user has not as yet entered the sentinel
    Add this grade into the running total
    Add one to the grade counter
    Input the next grade (possibly the sentinel)

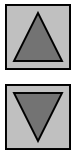
If the counter is not equal to zero
    Set the average to the total divided by the counter
    Print the average
else
    Print "No grades were entered"



Outline

fig03_08.c (Part 1 of 2)

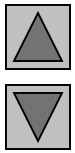
```
1  /* Fig. 3.8: fig03_08.c
2     Class average program with sentinel-controlled repetition */
3  #include <stdio.h>
4
5  /* function main begins program execution */
6  int main()
7  {
8     int counter;   /* number of grades entered */
9     int grade;     /* grade value */
10    int total;     /* sum of grades */
11
12    float average; /* number with decimal point for average */
13
14    /* initialization phase */
15    total = 0;     /* initialize total */
16    counter = 0;   /* initialize loop counter */
17
18    /* processing phase */
19    /* get first grade from user */
20    printf( "Enter grade, -1 to end: " );    /* prompt for input */
21    scanf( "%d", &grade );                   /* read grade from user */
22
23    /* loop while sentinel value not yet read from user */
24    while ( grade != -1 ) {
25       total = total + grade;                /* add grade to total */
26       counter = counter + 1;                /* increment counter */
27
```



Outline

```
28       printf( "Enter grade, -1 to end: " ); /* prompt for input */
29       scanf("%d", &grade);                  /* read next grade */
30    } /* end while */
31
32    /* termination phase */
33    /* if user entered at least one grade */
34    if ( counter != 0 ) {
35
36       /* calculate average of all grades entered */
37       average = ( float ) total / counter;
38
39       /* display average with two digits of precision */
40       printf( "Class average is %.2f\n", average );
41    } /* end if */
42    else { /* if no grades were entered, output message */
43       printf( "No grades were entered\n" );
44    } /* end else */
45
46    return 0; /* indicate program ended successfully */
47
48 } /* end function main */
```

fig03_08.c (Part 2 of 2)



Outline

Program Output

```
Enter grade, -1 to end: 75
Enter grade, -1 to end: 94
Enter grade, -1 to end: 97
Enter grade, -1 to end: 88
Enter grade, -1 to end: 70
Enter grade, -1 to end: 64
Enter grade, -1 to end: 83
Enter grade, -1 to end: 89
Enter grade, -1 to end: -1
Class average is 82.50
```

```
Enter grade, -1 to end: -1
No grades were entered
```

3.10 Ugnježdene kontrolne strukture

- Problem
  - Studentska služba ima listu rezultata ispita (1 = položio, 2 = pao) za 10 studenata
  - Napisati program koji analizira rezultate
    - Ako je više od 8 studenata položilo, štampati “Povecati platu profesoru"
- Obratiti pažnju na:
  - Program mora obraditi 10 rezultata
    - Koristiti petlju kontrolisanu brojačem
  - Potrebna su 2 brojača
    - Jedan za položio, drugi za pao
  - Rezultat testa je broj — ili 1 ili 2
    - Ako broj nije 1, pretpostavićemo da je 2

3.10 Ugnježdene kontrolne strukture

- Prvi nivo (Top level)

  Analiziraj rezultate i odluci da li treba povecati platu

- Razložiti prvi nivo na

  Inicijalizacija promjenjivih

  Unesi 10 rezultata testa i prebroj polozene i pale

  Stampaj sumarni izvjestaj i odluci da li treba povecati platu

- Razložiti *Inicijalizacija promjenljivih* u

  Inicijalizovati passes na nulu

  Inicijalizovati failures na nulu

  Inicijalizovati student counter na nulu

3.10 Ugnježdene kontrolne strukture

- Razložiti *Unesi 10 rezulatata testa i prebroj polozene i pale* u

  While student counter manji ili jednak od 10
  Unesi sledeci rezultat testa
  If student polozio
  Dodaj 1 na passes
  else
  Dodaj 1 na failures
  Dodaj 1 na student counter

- Razložiti *Stampaj sumarni izvjestaj i odluci da li treba povecati platu* u

  Stampaj broj polozenih (passes)
  Stampaj broj palih (failures)
  If vise od 8 studenata polozilo
  Print “Povecaj platu”

3.10 Ugnježdene kontrolne strukture

Inicijalizovati passes na nulu

Inicijalizovati failures na nulu

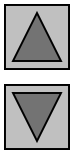
Inicijalizovati student counter na nulu

While student counter manji ili jednak od 10
  Unesi sledeci rezultat testa
  If student polozio
    Dodaj 1 na passes
  else
    Dodaj 1 na failures
  Dodaj 1 na student counter

Stampaj broj polozenih (passes)

Stampaj brol palih (failures)

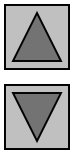
If vise od 8 studenat polozilo
  Print “Povecaj platu”



Outline

fig03_10.c (Part 1 of 2)

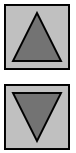
```
1  /* Fig. 3.10: fig03_10.c
2     Analysis of examination results */
3  #include <stdio.h>
4
5  /* function main begins program execution */
6  int main()
7  {
8     /* initialize variables in definitions */
9     int passes = 0;    /* number of passes */
10    int failures = 0; /* number of failures */
11    int student = 1;  /* student counter */
12    int result;       /* one exam result */
13
14    /* process 10 students using counter-controlled loop */
15    while ( student <= 10 ) {
16
17       /* prompt user for input and obtain value from user */
18       printf( "Enter result ( 1=pass,2=fail ): " );
19       scanf( "%d", &result );
20
21       /* if result 1, increment passes */
22       if ( result == 1 ) {
23          passes = passes + 1;
24       } /* end if */
```



Outline

```
25       else { /* otherwise, increment failures */
26          failures = failures + 1;
27       } /* end else */
28
29       student = student + 1; /* increment student counter */
30    } /* end while */
31
32    /* termination phase; display number of passes and failures */
33    printf( "Passed %d\n", passes );
34    printf( "Failed %d\n", failures );
35
36    /* if more than eight students passed, print "raise tuition" */
37    if ( passes > 8 ) {
38       printf( "Raise tuition\n" );
39    } /* end if */
40
41    return 0; /* indicate program ended successfully */
42
43 } /* end function main */
```

fig03_10.c (Part 2 of 2)



Outline

Program Output

```
Enter Result (1=pass,2=fail): 1
Enter Result (1=pass,2=fail): 2
Enter Result (1=pass,2=fail): 2
Enter Result (1=pass,2=fail): 1
Enter Result (1=pass,2=fail): 1
Enter Result (1=pass,2=fail): 1
Enter Result (1=pass,2=fail): 2
Enter Result (1=pass,2=fail): 1
Enter Result (1=pass,2=fail): 1
Enter Result (1=pass,2=fail): 2
Passed 6
Failed 4
```

```
Enter Result (1=pass,2=fail): 1
Enter Result (1=pass,2=fail): 1
Enter Result (1=pass,2=fail): 1
Enter Result (1=pass,2=fail): 2
Enter Result (1=pass,2=fail): 1
Enter Result (1=pass,2=fail): 1
Enter Result (1=pass,2=fail): 1
Enter Result (1=pass,2=fail): 1
Enter Result (1=pass,2=fail): 1
Enter Result (1=pass,2=fail): 1
Passed 9
Failed 1
Raise tuition
```

3.11 Operatori dodjeljivanja

- Operatori dodjeljivanja Izraz

```
c = c + 3;
```

može biti skraćen u `c += 3;`

- Izrazi oblika

variable = *variable operator expression*;

mogu biti napisani u kraćem obliku kao

variable `operator=` *expression*;

- Primjeri operatora dodjeljivanja:

```
d -= 4      (d = d - 4)
e *= 5      (e = e * 5)
f /= 3      (f = f / 3)
g %= 9      (g = g % 9)
```

3.11 Operatori dodjeljivanja

Pretpostavimo da je: `int c = 3, d = 5, e = 4, f = 6, g = 12;`

Operator dodjeljivanja:	Primjer izraza:	Objasnjenje:	Dodjeljivanje
`+=`	`c += 7`	`c = c + 7`	10 na `c`
`-=`	`d -= 4`	`d = d - 4`	1 na `d`
`*=`	`e *= 5`	`e = e * 5`	20 na `e`
`/=`	`f /= 3`	`f = f / 3`	2 na `f`
`%=`	`g %= 9`	`g = g % 9`	3 na `g`

Fig. 3.11 Aritmeticki operatori dodjeljivanja.

3.12 Operatori inkrementiranja i dekrementiranja

- Operator inkrementiranja (++)
  - Može da zamijeni `c+=1`
- Operator dekrementiranja (`--`)
  - Može da zamijeni of `c-=1`
- Prefiksni inkrement
  - Operator je ispred promjenljive (`++c` ili `--c`)
  - Promjenljiva je promijenjena prije nego što je izraz izračunat
- Postfiksni inkrement
  - Operator je ispred promjenljive (`c++` ili `c--`)
  - Promjenljiva je promijenjena poslije nego što je izraz izračunat

3.12 Operatori inkrementiranja i dekrementiranja

- Ako je c jednako 5, tada

```
printf( "%d", ++c );
```

  - Štampa 6

```
printf( "%d", c++ );
```

  - Štampa 5
  - U oba slučaja, c sada ima vrijednost 6

- Ako promjenljiva nije u izrazu
  - Prefiks i postfiks inkrementiranje imaju isti efekat

```
++c;
printf( “%d”, c );
```

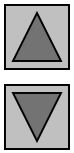
  - Ima isti efekat kao i

```
c++;
printf( “%d”, c );
```

3.12 Operatori inkrementiranja i dekrementiranja

Operator	Primjer izraza	Objasnjenje
++	++a	Povecati a za 1 i koristiti novu vrijednsot u izrazu u kom je promjenljiva.
++	a++	Koristi trenutnu vrijednost promjenljive u izraza, pa povacati a za 1.
--	--b	Umanjiti b za 1 i koristiti novu vrijednost u izrazu u kom je promjenljiva.
--	b--	Koristi trenutnu vrijednost promjenljive u izraza, pa umanjiti b za 1.

Fig. 3.12 Operetori inkrementiranja i dekrementiranja

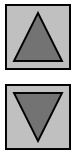


Outline

fig03_13.c

```c
/* Fig. 3.13: fig03_13.c
   Preincrementing and postincrementing */
#include <stdio.h>

/* function main begins program execution */
int main()
{
   int c;                /* define variable */

   /* demonstrate postincrement */
   c = 5;                /* assign 5 to c */
   printf( "%d\n", c );  /* print 5 */
   printf( "%d\n", c++ ); /* print 5 then postincrement */
   printf( "%d\n\n", c ); /* print 6 */

   /* demonstrate preincrement */
   c = 5;                /* assign 5 to c */
   printf( "%d\n", c );  /* print 5 */
   printf( "%d\n", ++c ); /* preincrement then print 6 */
   printf( "%d\n", c );  /* print 6 */

   return 0; /* indicate program ended successfully */

} /* end function main */
```

```
5
5
6

5
6
6
```



Outline

Program Output

3.12 Operatori inkrementiranja i dekrementiranja

Operators					Associativity	Type
++	--	+	-	(type)	right to left	unary
*	/	%			left to right	multiplicative
+	-				left to right	additive
<	<=	>	>=		left to right	relational
==	!=				left to right	equality
?:					right to left	conditional
=	+=	-=	*=	/=	right to left	assignment

Fig. 3.14 Precedence of the operators encountered so far in the text.