Sviluppo di programmi strutturati

- Fasi di un programma
- La struttura sequenziale
- La struttura di selezione - If/Else
- La struttura di ripetizione - While, For, Do/While
  - Caso 1 - Ripetizioni controllate da un contatore
  - Caso 2 - Ripetizioni controllate da un valore “sentinella”
  - Caso 3 - Strutture di controllo annidate
- Operatori di assegnamento
- Operatori di incremento e decremento
- Operatori logici

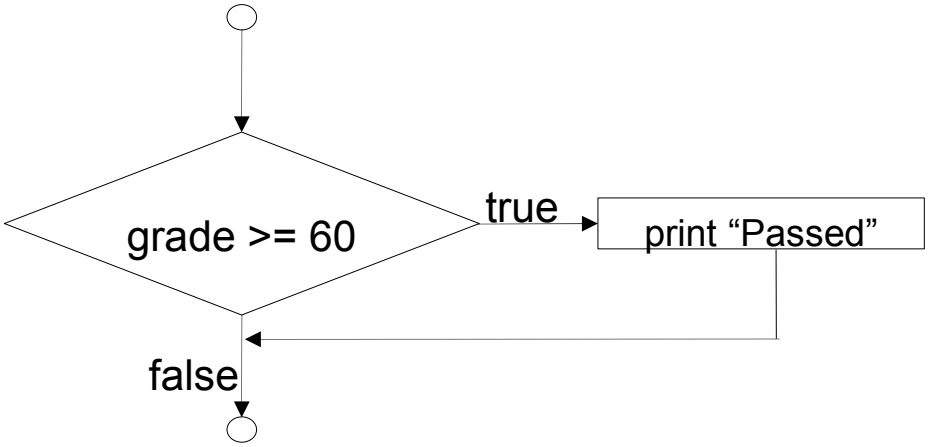
FASI DEL PROGRAMMA

Quasi tutti i programmi sono costituiti da tre fasi:

- Inizializzazione: inizializza le variabili del programma
- Elaborazione: ingresso dei dati e loro eventuale modifica a seconda delle operazioni da svolgere

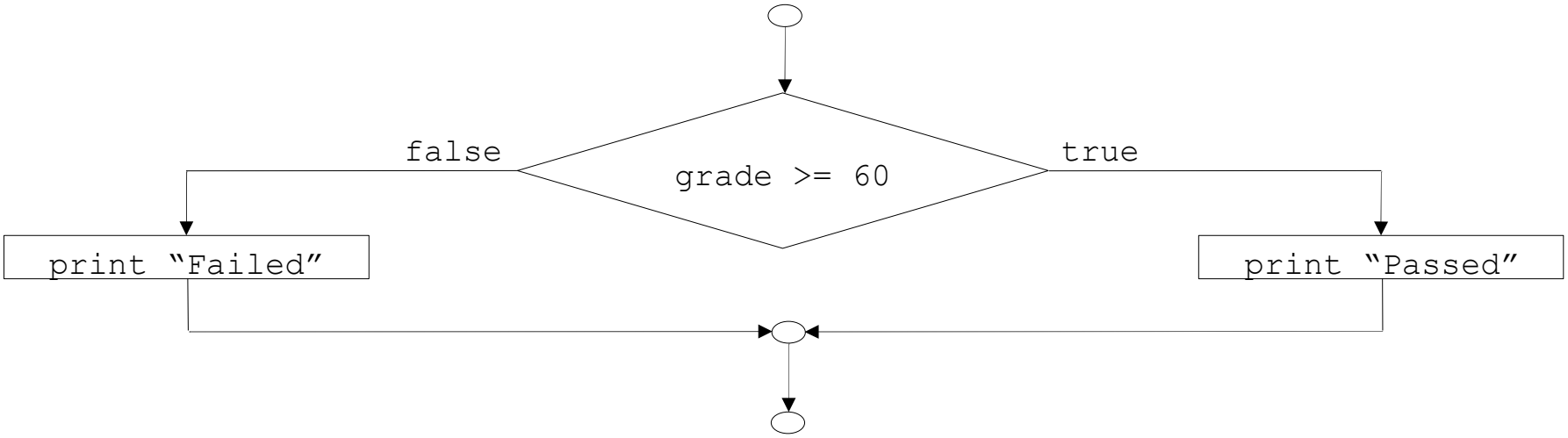
  Questa fase comprende **selezioni e cicli**
- Terminazione: calcolo e stampa dei risultati finali

FLOW-CHART IF E IF/ELSE



IF

IF/ELSE



IF/ELSE ANNIDATI

Effettuano il test di casi multipli. Strutture if/else vengono poste all'interno di altre strutture if/else.

Pseudocodice:

```
If student's grade is greater than or equal to 90
        Print "A"
   else
        If student's grade is greater than or equal to 80
                Print "B"
          else
                If student's grade is greater than or equal to 70
                        Print "C"
                else
                        If student's grade is greater than or equal to
   60
                                Print "D"
                        else
                                Print "F"
```

- Quando la condizione è verificata, si saltano le altre istruzioni.
- In pratica non si usano indentazioni troppo "profonde".

ISTRUZIONI COMPOSTE

Sono **blocchi di istruzioni comprese fra parentesi graffe**.

Esempio:

```
if ( grade >= 60 )
  printf( "Passed.\n" );
else {
  printf( "Failed.\n" );
  printf( "You must take this course again.\n" );
}
```

Senza parentesi graffe:

```
printf( "You must take this course again.\n" );
```

verrebbe eseguito comunque, indipendentemente dal valore di "grade". In questo caso infatti questo printf sarebbe fuori dall' else.

Scrivere un programma che determina se un anno è bisestile o meno (un anno è bisestile se è divisibile per 4 ma non per 100 oppure è divisibile per 400)

```
#include<stdio.h>
void main()
{
        int anno;
        printf("Inserire l'anno: ");
        scanf("%d", &anno);
        if ((anno % 4) != 0)
         printf("Anno non bisestile");
         else                                          /* e' divisibile per 4 */
         if ((anno % 100) != 0)
                        printf("Anno bisestile");
         else                                           /* e' divisibile per 4 e per 100 */
                        if ((anno % 400) == 0)
                                        printf("Anno bisestile");
                        else                                /* divisibile per 4, 100 ma non 400 */
                                                printf("Anno non bisestile");
}
```


Operatori logici

Questi operatori sono utili come condizioni di fine ciclo

- `&&` (**AND** logico)
  - Restituisce il valore **`vero`** (true) se sono vere entrambe le condizioni
- `||` (**OR** logico)
  - Resitutisce il valore **`vero`** se una delle condizioni è vera
- `!` (**NOT** logico, negazione logica)
  - Inverte la verità/falsità della condizione
  - E' un operatore unario, cioè ha un solo operando

Expressione	Risultato
`true && false`	`false`
`true || false`	`true`
`!false`	`true`

Scrivere un programma che determina se un anno è bisestile o meno (un anno è bisestile se è divisibile per 4 ma non per 100 oppure è divisibile per 400)

```
#include<stdio.h>
void main()
{
        int anno;
        printf("Inserire l'anno: ");
        scanf("%d", &anno);
        /* parentesi più interne (blu) per la prima condizione (anno
           divisibile per 4 ma non per 100)/*
        /* parentesi intermedie (verdi) per la seconda condizione (anno
           divisibile per 400) */
        if (((anno % 4 == 0) && (anno % 100 != 0)) || (anno % 400 == 0))
         printf("Anno bisestile");
       else
         printf("Anno non bisestile");
}
```


Ripetizioni controllate da un contatore

Per questo tipo di ripetizioni, sono necessari:

- *Il **nome*** di una variabile di controllo (o contatore del ciclo).
- *Un **valore iniziale*** della variabile di controllo.
- Una ***condizione*** che controlla il *valore finale* della variabile di controllo (cioè se il ciclo deve proseguire o no).
- *Un **incremento*** (o *decremento*) che modifica la variabile di controllo a tutti i passi del ciclo.

- Esempio (struttura while):

```
int counter =1; /*inizializzazione*/
while (counter <= 10){     /*condizione di
                             ripetizione*/
        printf( "%d\n", counter );
        counter=counter+1; /*incremento*/
    }
```


LA STRUTTURA DI RIPETIZIONE WHILE

Struttura di ripetizione:

Il programmatore definisce un'azione che deve essere ripetuta finchè (while) una certa condizione rimane vera (**`true`**)

Pseudocodice:

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

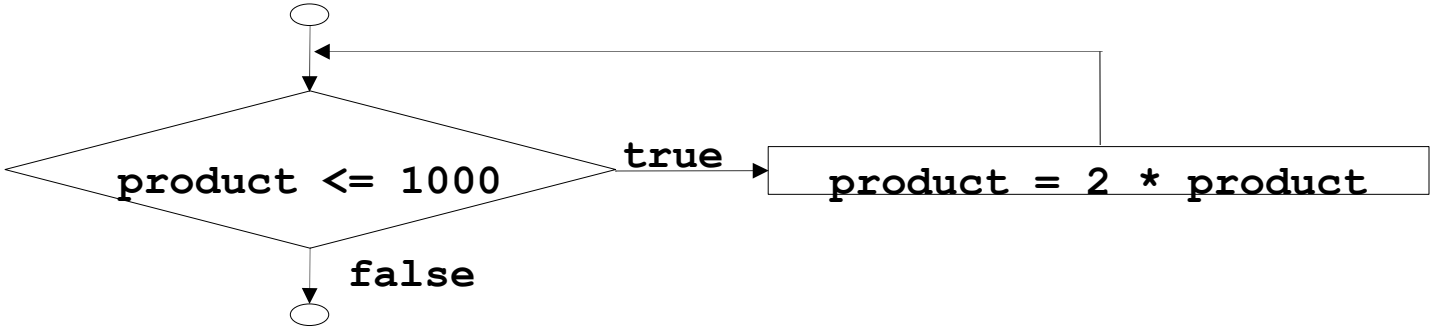
–**Il ciclo while** è ripetuto finchè la condizione "*there are more items on my shopping list*" è falsa.

–*"cross it off my list"* consente di raggiungere la condizione di fine ciclo.

ESEMPIO

Si moltiplica per due la variabile product (inizializzata a 2) finchè non rimane <1000.

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



CICLI - RIPETIZIONI CONTROLLATE DA UN CONTATORE

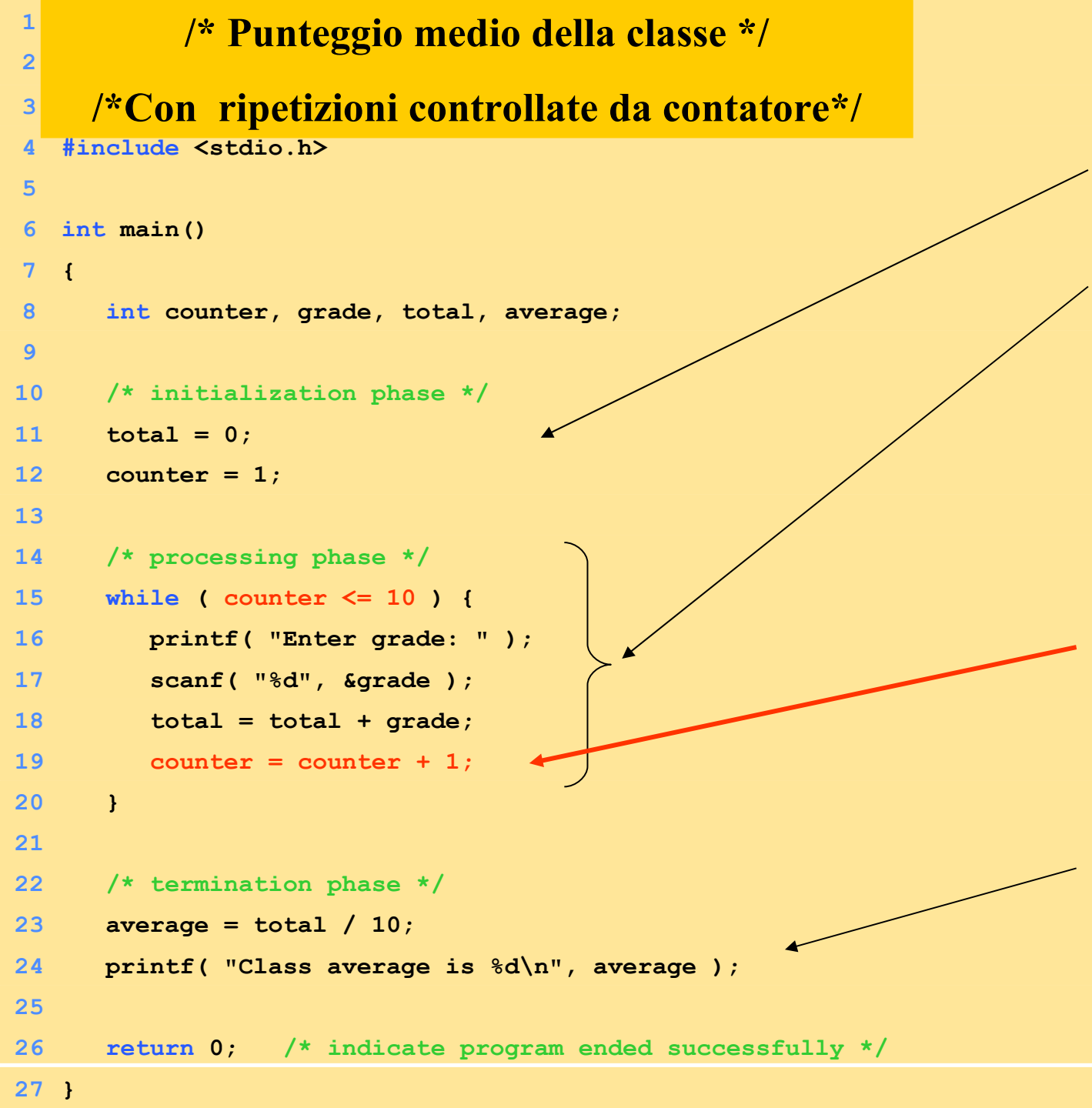
Ripetizioni controllate da un contatore

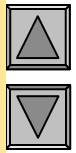
- Il ciclo è ripetuto finchè il contatore non raggiunge un determinato valore.
- Sono ripetizioni definite: il numero delle ripetizioni è noto.
- Esempio: Una classe di 10 studenti effettua un test. Il punteggio è variabile da 0 a 100 ed è inserito dall'utente. Determinare il punteggio medio della classe.

Pseudocodice:

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

```
1  /* Punteggio medio della classe */
2
3  /*Con  ripetizioni controllate da contatore*/
4  #include <stdio.h>
5
6  int main()
7  {
8     int counter, grade, total, average;
9
10    /* initialization phase */
11    total = 0;
12    counter = 1;
13
14    /* processing phase */
15    while ( counter <= 10 ) {
16       printf( "Enter grade: " );
17       scanf( "%d", &grade );
18       total = total + grade;
19       counter = counter + 1;
20    }
21
22    /* termination phase */
23    average = total / 10;
24    printf( "Class average is %d\n", average );
25
26    return 0;   /* indicate program ended successfully */
27 }
```





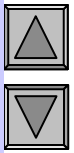
Outline

- Inizializzazione delle variabili

4. Esecuzione del ciclo

CONSENTE DI RAGGIUNGERE LA CONDIZIONE DI FINE CICLO

3. Output



Outline

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
```

Ripetizioni controllate da un valore “sentinella”

- Problema :

  Sviluppare un programma per il calcolo del punteggio medio della classe, tale da elaborare un numero arbitrario di punteggi ogni volta che va in esecuzione.

  - Il numero di studenti non è noto
  - Come fa il programma a saper quando deve terminare?

- Uso di un valore “sentinella”
  - Detto anche valore segnale, valore muto, valore flag
  - Indica “fine dell’input dei dati.”
  - Il ciclo termina quando si inserisce il valore sentinella
  - Il valore sentinella deve essere scelto in modo da non essere confuso con un ingresso regolare (ad es. `-1` in questo caso)

CALCOLO DELLA MEDIA DEI PUNTI SU UNA CLASSE - PSEUDOCODICE

- Fase di *Inizializzazione delle variabili*:

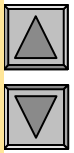
  Initialize total to zero
  Initialize counter to zero

- Fase di *input, somma e conteggio dei punteggi:*

  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)

- Fase del *calcolo e stampa del valor medio :*

  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

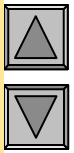
```
1  /* Fig. 3.8: fig03_08.c
2     Class average program with
3     sentinel-controlled repetition */
4  #include <stdio.h>
5
6  int main()
7  {
8     float average;              /* new data type */
9     int counter, grade, total;
10
11    /* initialization phase */
12    total = 0;
13    counter = 0;
14
15    /* processing phase */
16    printf( "Enter grade, -1 to end: " );
17    scanf( "%d", &grade );
18
19    while ( grade != -1 ) {
20       total = total + grade;
21       counter = counter + 1;
22       printf( "Enter grade, -1 to end: " );
23       scanf( "%d", &grade );
24    }
```

1. Inizializzazione delle variabili

2. input del dati

3 ciclo

```
25
26    /* termination phase */
27    if ( counter != 0 ) {
28       average = ( float ) total / counter;
29       printf( "Class average is %.2f", average );
30    }
31    else
32       printf( "No grades were entered\n" );
33
34    return 0;   /* indicate program ended successfully */
35 }
```



Outline

3. Calcolo della media

4. Stampa dei risultati

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
```

Scrivere un programma che somma le cifre che compongono il numero intero fornito dall'utente

```
#include<stdio.h>
void main()
{
      int n, somma;
      scanf("%d", &n);
      somma = 0;          /* variabile inizializzata a zero, altrimenti può
                                                contenere un qualsiasi valore*/
      while(n > 0)                    /*condizione di fine ciclo*/
      {
            somma = somma + (n % 10);      /* somma += n % 10; */
            n = n / 10;                    /* n /= 10; */
      }
      printf("%d", somma);
}
```


/* Calcola la somma dei valori interi passati dall'utente
Termina quando viene immesso il valore 0 (zero) */

```
#include <stdio.h>

main()
{
int somma, numero;
printf("SOMMA NUMERI\n");
printf("zero per terminare\n");
numero = 1;                          /*inizializzazione delle variabili*/
somma  = 0;                  /* altrimenti possono contenere qualsiasi valore*/
while(numero!=0) {                   /*condizione di fine ciclo*/
     printf("Inserisci un intero: ");
     scanf("%d", &numero);
     somma = somma+numero;
}
printf("Somma: %d\n",somma);
}
```


STRUTTURE DI CONTROLLO ANNIDATE

Problema

- Una scuola ha una lista di risultati di un test (1 = pass, 2 = fail) per 10 studenti.
- Scrivere un programma che analizza i risultati
  - Se ci sono più di 8 studenti promossi, stampa "Aumenta il livello del corso"
- Il programma deve elaborare i risultati di 10 test
  - Si userà un ciclo con controllo del contatore
- Si possono usare 2 contatori
  - Uno per i promossi e uno per i bocciati
- Ogni test può avere valore 1 o valore 2
  - Se il numero non è 1, si suppone che sia 2

Strutture di controllo annidate – approccio top-down

- Fase iniziale (top)

  Analyze exam results and decide if tuition should be raised

- Primo livello di dettaglio

  Initialize variables

  Input the ten quiz grades and count passes and failures

  Print a summary of the exam results and decide if tuition should be raised

- Secondo livello di dettaglio: *Inizializza le variabili:*

  Initialize passes to zero
  Initialize failures to zero
  Initialize student counter to one

Strutture di controllo annidate (cont.)

- Dettaglio successivo: *Input the ten quiz grades and count passes and failures:*

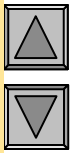
  While student counter is less than or equal to ten
  Input the next exam result
  If the student passed
  Add one to passes
  else
  Add one to failures
  Add one to student counter

- Dettaglio finale: *Print a summary of the exam results and decide if tuition should be raised*

  Print the number of passes
  Print the number of failures

  If more than eight students passed
  Print "Raise tuition"

```
1
2     /* Analisi dei risultati dell'esame */
3  #include <stdio.h>
4
5  int main()
6  {
7     /* initializing variables in declarations */
8     int passes = 0, failures = 0, student = 1, result;
9
10    /* process 10 students; counter-controlled loop */
11    while ( student <= 10 ) {
12       printf( "Enter result ( 1=pass,2=fail ): " );
13       scanf( "%d", &result );
14
15       if ( result == 1 )         /* if/else nested in while */
16          passes = passes + 1;
17       else
18          failures = failures + 1;
19
20       student = student + 1;
21    }
22
23    printf( "Passed %d\n", passes );
24    printf( "Failed %d\n", failures );
25
26    if ( passes > 8 )
27       printf( "Raise tuition\n" );
28
29    return 0;    /* successful termination */
30 }
```

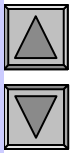


Outline

1. Inizializzazione delle variabili

2. Input dei dati e conteggio dei passes/failures

3. Stampa dei risultati



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
```

OPERATORI DI ASSEGNAMENTO

- L'assegnamento

  c = c + 3;

  Può essere abbreviato in

  c += 3;

  Con l'uso dell'*operatore di assegnamento* **+=**
- In generale, assegnamenti del tipo

  variabile = *variabile operatore espressione*;

  si possono riscrivere come

  variabile operatore= *espressione*;

- Esempi di altri operatori di assegnamento:

  d -= 4 (d = d - 4)

  e *= 5 (e = e * 5)

  f /= 3 (f = f / 3)

  g %= 9 (g = g % 9)

Operatori di incremento e decremento

- Operatore di incremento (**++**) - si può usare invece di **c+=1**
- Operatore di decremento (**--**) - si può usare invece di **c-=1**.
- Preincremento
  - L'operatore è posto prima della variabile (**++c** or **--c**)
  - Prima si modifica la variabile, poi si valuta l'espressione
- Postincremento
  - L'operatore è posto dopo la variabile (**c++** or **c--**)
  - Prima si esegue l'espressione, poi si modifica la variabile
- If **c = 5**, then

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

  - Stampa **6**

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

  - Stampa **5**
  - In entrambi i casi, **c** ha ora il valore **6**

```
/* Determina somma e maggiore dei valori immessi */
#include <stdio.h>
main()
{
int somma,numero,max,i;
printf("SOMMA E MAGGIORE\n");
printf("zero per finire\n");
numero = 1;
somma = 0;
max = 0;
i = 1;
while(numero!=0 && i<=10)
{
     printf("Valore int.: ");
     scanf("%d", &numero);
     if(numero>max)
        max = numero;
     somma = somma+numero;
     i++;
}
printf("Somma: %d\n", somma);
printf("Maggiore: %d\n", max);
}
```


Confondere l'operatore di uguaglianza (==) con quello di assegnamento (=)

E' un errore pericoloso, perchè non causa errori di sintassi. E' un errore logico.

- –Qualsiasi espressione che produce un valore può essere usata nelle strutture di controllo.
- –Valori diversi da zero significano **`true`**, valori uguali a zero significano **`false`**

Esempio:

```
if ( payCode == 4 )
  printf( "You get a bonus!\n" );
```

Valuta il valore di payCode: se è uguale a 4 viene assegnato un omaggio

```
if ( payCode = 4 )
printf( "You get a bonus!\n" );
```

Questa istruzione assegna a payCode il valore 4; Poichè 4 è diverso da zero, l'espressione è true e l'omaggio viene assegnato a prescindere dal valore di payCode.

I cicli for, switch, do-while

Schema

3. Nozioni base
4. La struttura di ripetizione for
5. Esempi
6. La struttura di selezione multipla switch
7. La struttura di ripetizione do-while

Concetti fondamentali sui cicli (loop)

- Loop
  - Gruppo di istruzioni che l'elaboratore esegue ripetutamente, finchè una certa condizione rimane vera
- Ripetizioni controllate da contatore
  - Sono ripetizioni definite, cioè è noto il numero di ripetizioni del ciclo
  - Si fa uso di una variabile di controllo per il conteggio delle ripetizioni
- Ripetizioni controllate da un valore "sentinella"
  - Sono ripetizioni indefinite, cioè non note a priori
  - Il valore sentinella indica la "fine dei dati"

LA STRUTTURA DI RIPETIZIONE FOR

Formato delle istruzioni di un ciclo **`for`**:

`for (` ***inizializzazione del contatore*** **`;`** ***test di continuazione del ciclo*** **`;`** ***incremento del contatore*** **`)`**
istruzioni ;

Non si mette il ; dopo l'ultima espressione

Esempio:

```
for( int counter = 1; counter <= 10;
counter++ )
    printf( "%d\n", counter );
```

- Stampa gli interi da 1 a 10.

FOR

L'istruzione for è utilizzata per effettuare un ciclo dal principio sino alla fine di un intervallo di valori. La sintassi è la seguente:

for(*espressione-iniziale*;*espressione-booleana*;*espressione-incremento*)
    istruzione

L'espressione-iniziale permette di inizializzare le variabili di ciclo, e viene eseguita una volta sola, prima di qualsiasi altra operazione.

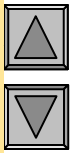
Successivamente ad essa, viene valutata l'***espressione booleana*** e, se questa ha valore diverso da 0 - è vera -, viene eseguita l'istruzione che costituisce il corpo del ciclo.

Al termine dell'esecuzione del corpo del ciclo, viene valutata l'***espressione-incremento***, di solito per poter aggiornare i valori delle variabili di ciclo.

Quindi, si valuta nuovamente l'*espressione* del ciclo e cosi via. Il ciclo si ripete finché non si valuta come falsa l'espressione del ciclo - valore 0-.

/* Esempio di utilizzo dell'istruzione for
Calcola la somma di cinque numeri interi immessi dall'utente */

```
#include <stdio.h>
int i, somma, numero;

main()
{
printf("SOMMA 5 NUMERI\n");
somma = 0;

for(i=1; i<=5; i=i+1) {
      printf("Inser. intero: ");
      scanf("%d", &numero);
      somma = somma + numero;
      }

printf("Somma: %d\n",somma);
}
```

Outline

PROGRAMMA PER LA SOMMA DEI NUMERI PARI FINO A 100

```
1   /*USO DEL CICLO FOR */
2
3   #include <stdio.h>
4
5   int main()
6   {
7      int sum = 0, number;
8
9      for ( number = 2; number <= 100; number += 2 )
10        sum += number;
11
12     printf( "Sum is %d\n", sum );
13
14     return 0;
15  }
```

```
Sum is 2550
```

Output

```
/* Calcolo di n! (n fattoriale) con ciclo for a decremento delle iterazioni*/
/*   n!=n*(n-1)*(n-2)*…*2*1  */
#include <stdio.h>

main()
{
int n, fat, m;

printf("CALCOLO DI N!\n\n");
printf("Inser. n: ");
scanf("%d", &n);

fat = n;
for(m=n; m>2; m--)
  fat = fat*(m-1);

printf("Il fattoriale di: %d ha valore: %d\n", n, fat);
}
```


```
/* Calcolo n! (n fattoriale) con ciclo for ad incremento delle iterazioni */
#include <stdio.h>

main()
{
int n, fat, aux;

printf("CALCOLO DI N!\n\n");
printf("Inser. n: ");
scanf("%d", &n);

fat = 1;
for(aux=2; aux<=n; aux++) fat = fat*aux;

printf("Il fattoriale di: %d ha valore: %d\n", n, fat);
}
```


```
#include <stdio.h>

main()      /* esempio cicli for annidati – matrice di + */
{
int n, m, i, j;

printf("Inserire il numero di linee: \n");
scanf("%d", &n);
printf("Inserire il numero di colonne: \n");
scanf("%d", &m);

for(i=1; i<=n; i++) { /* ciclo esterno: va a capo n volte */
  printf("\n");
     for(j=1; j<=m; j++)      /*ciclo interno:stampa di m volte + +/
            printf("+");
}                   /* fine blocco ciclo esterno */
}
```


Il ciclo for - note

I cicli for possono essere riscritti come cicli while:

inizalizzazione;
while (*Test di continuazione del ciclo***){**
istruzioni
incremento;
}

Inizializzazione e incremento

La parte di inizializzazione e quella di iterazione di un ciclo for possono essere costituite da elenchi di espressioni separate da virgole. Sono liste separate da ;

```
for (int i = 0, j = 0;  j + i <= 10; j++, i++)
    printf( "%d\n", j + i );
```

Tali espressioni vengono valutate, come molti operatori, da sinistra a destra.

Ad esempio, per poter far procedere due indici in direzioni opposte, sarebbe appropriato scrivere del codice come:

```
for (i = 0, j = NumElem - 1; j >= 0; i++, j--)
 {
/* ... */
 }
```


Il ciclo for – note (cont.)

- Espressioni aritmetiche:
  - Inizializzazione, continuazione del ciclo e incremento possono contenere espressioni artimetiche.
  - Esempio: (supposti **`x = 2`** e **`y = 10`**)

```
for ( j = x; j <= 4 * x * y; j += y / x )
```

  è equivalente a

```
for ( j = 2; j <= 80; j += 5 )
```

- "Incrementi" possono essere negativi (decrementi)
- Se la condizione di continuazione del ciclo è inizalmente **`false`**
  - Il corpo della struttura **`for`** non è eseguito
  - Il controllo procede con la prima istruzione dopo il **`for`**

Scrivere un programma che legge un numero intero e stampa a video VERO o FALSO in base al fatto che il numero sia primo o meno

```
#include<stdio.h>
void main()
{
      int n, i;
      printf("Inserisci un numero intero ");
      scanf("%d", &n);
          for(div = 0, i = 2; (i < n) && (div < 1); i++)
                    if((n % i)==0)
                        div++;              /* se trova un divisore si sa che non e'
          primo */
                    if (div > 0)
                        printf("VERO");
                    else
                        printf("FALSO");
}
```


La struttura di selezione multipla switch

- **`switch`**
  - E' utile quando una variabile o espressione viene valutata per tutti i valori che essa può assumere, in corrispondenza dei quali vengono eseguite azioni diverse.

Formato dell'istruzione switch:

- Serie di etichette **`case`** e di un caso **`default`** opzionale

```
switch ( value ){
    case '1':
        azioni
    case '2':
        azioni
    default:
        azioni
    }
```

`break;` ha come risultato l'uscita dallo switch

switch

Il costrutto switch consente di trasferire il controllo di esecuzione all'interno di un blocco di istruzioni, in cui sia presente un'etichetta. Il punto scelto è determinato dal risultato della valutazione di una espressione *intera*. La forma generale del costrutto è:

switch (*espressione*) {

case *n*: *istruzioni*

case *m*: *istruzioni*

case *h*: *istruzioni*

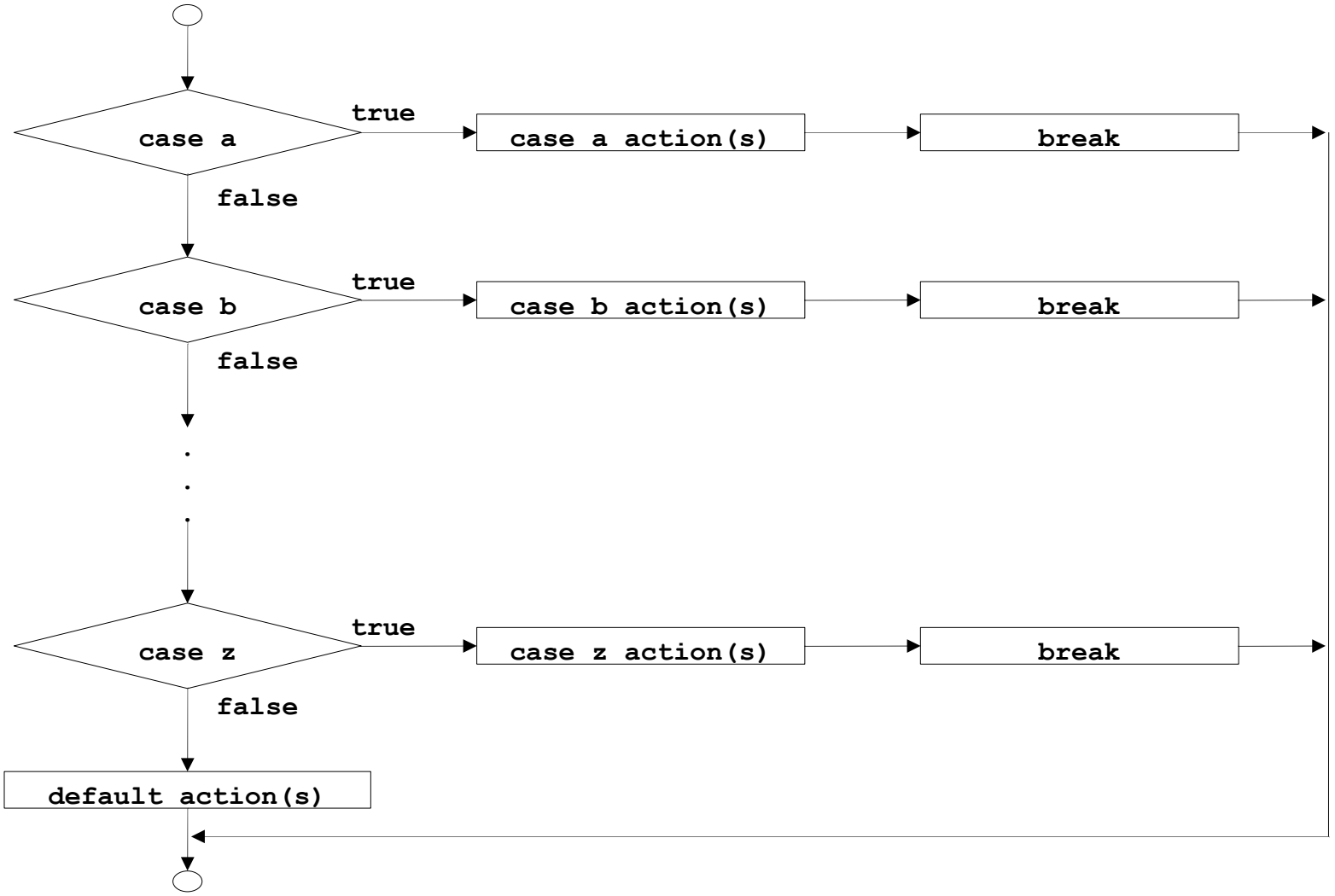
...

default: *istruzioni*

}

I valori *n*, *m*, ... sono delle costanti **intere**. Se il valore dell'espressione combacia con il valore di una delle etichette case, il controllo viene trasferito alla prima istruzione che segue tale etichetta. Se non vi è alcuna etichetta che combacia, allora il controllo viene trasferito alla prima istruzione dopo l'etichetta default, se esiste, altrimenti si salta tutta l'istruzione switch.

La struttura di selezione multipla switch - flow-chart



SWITCH - NOTE

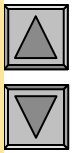
Una volta trasferito il controllo alla prima istruzione di quelle che seguono l'etichetta case che combacia, le istruzioni successive vengono eseguite una per volta, anche se sono associate ad un'altra etichetta case. Un'etichetta case o default **non** spinge ad uscire dallo switch.

Se si desidera arrestare l'esecuzione delle istruzioni all'interno del blocco switch è necessario utilizzare l'istruzione **break.** All'interno di un blocco switch, l'istruzione break trasferisce il controllo all'esterno del blocco, alla prima istruzione che segue lo switch.

L'espressione di switch deve essere di tipo char o int. Tutte le etichette case devono essere espressioni costanti. In tutte le istruzioni switch singole, ogni valore associato alle etichette case deve essere unico, e ci può essere al più una sola etichetta default.

```
/* Esempio di utilizzo di switch-case */
#include <stdio.h>
int x;
main()
{
printf("Digita una cifra: ");
scanf("%d", &x);
switch(x) {
  case 0:
    printf("zero\n");
    break;
  case 1:
    printf("uno\n");
    break;
  case 2:
    printf("due\n");
    break;
  case 3:
    printf("tre\n");
    break;
  case 4:
    printf("quattro\n");
    break;
  case 5:
    printf("cinque\n");
    break;
  default:
    printf("non compreso\n");
    break;
}
}
```


```c
/* CONTEGGIO DI VOTI ESPRESSI IN LETTERE

#include <stdio.h>

int main()
{
   int grade;
   int aCount = 0, bCount = 0, cCount = 0,
       dCount = 0, fCount = 0;

   printf(  "Enter the letter grades.\n"  );
   printf(  "Enter the EOF character to end input.\n"  );

   while ( ( grade = getchar() ) != EOF ) {

      switch ( grade ) {    /* switch annidato in un while */

         case 'A': case 'a':  /* il voto può essere a oppure A */
            ++aCount;
            break;

         case 'B': case 'b':  /* il voto può essere b oppure B */
            ++bCount;
            break;

         case 'C': case 'c':  /* il voto può essere c oppure C */
            ++cCount;
            break;

         case 'D': case 'd':  /* il voto può essere d oppure D */
            ++dCount;
            break;
```



Outline

1. Inizializzazione delle variabili

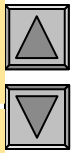
2. Input dei dati

Uso del ciclo switch per aggiornare count

EOF=“end of file”

E’ una sequenza di tasti che dipende dal sistema. In genere corrisponde a -1

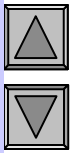
Sotto DOS si ottiene premendo ctrl-z



Outline

```
33
34          case 'F': case 'f':  /* il voto può essere f oppure F F */
35             ++fCount;
36             break;
37
38          case '\n': case' ':  /* ignora questo carattere in ingresso
39             break;
40
41          default:         /* tutti i restanti caratteri */
42             printf( "Incorrect letter grade entered." );
43             printf( " Enter a new grade.\n" );
44             break;
45       }
46    }
47
48    printf( "\nTotals for each letter grade are:\n" );
49    printf( "A: %d\n", aCount );
50    printf( "B: %d\n", bCount );
51    printf( "C: %d\n", cCount );
52    printf( "D: %d\n", dCount );
53    printf( "F: %d\n", fCount );
54
55    return 0;
56 }
```

Stampa dei risultati



Outline

Esempio

```
Enter the letter grades.
Enter the EOF character to end input.
A
B
C
C
A
D
F
C
E
Incorrect letter grade entered. Enter a new grade.
D
A
B

Totals for each letter grade are:
A: 3
B: 2
C: 3
D: 2
F: 1
```

Input dei dati e
Output dei risultati

while e do-while

Una struttura di iterazione consente di specificare un'azione, o un insieme di azioni, che dovrà essere ripetuta più volte. Si parla in questi casi di ***ciclo***.

Il ciclo while è generalmente strutturato come segue:

while (*espressione*)

Istruzione

Si tratta di un ciclo a ***condizione iniziale***: prima di eseguire il ciclo si valuta la condizione.

L'espressione viene valutata e, se ha valore diverso da 0 (*vero*) viene eseguita l'istruzione successiva (che può anche essere un intero blocco di istruzioni, opportunamente delimitato dalle parentesi graffe). Una volta che quest'ultima è terminata, l'espressione viene valutata nuovamente e se è nuovamente vera, si ripete l'istruzione. Ciò si ripete fino a quando l'espressione ha valore 0 (*falso*), nel qual caso il controllo si trasferisce all'istruzione successiva al while.

Un ciclo while può essere eseguito 0 o più volte, poiché l'espressione potrebbe essere falsa già la prima volta.

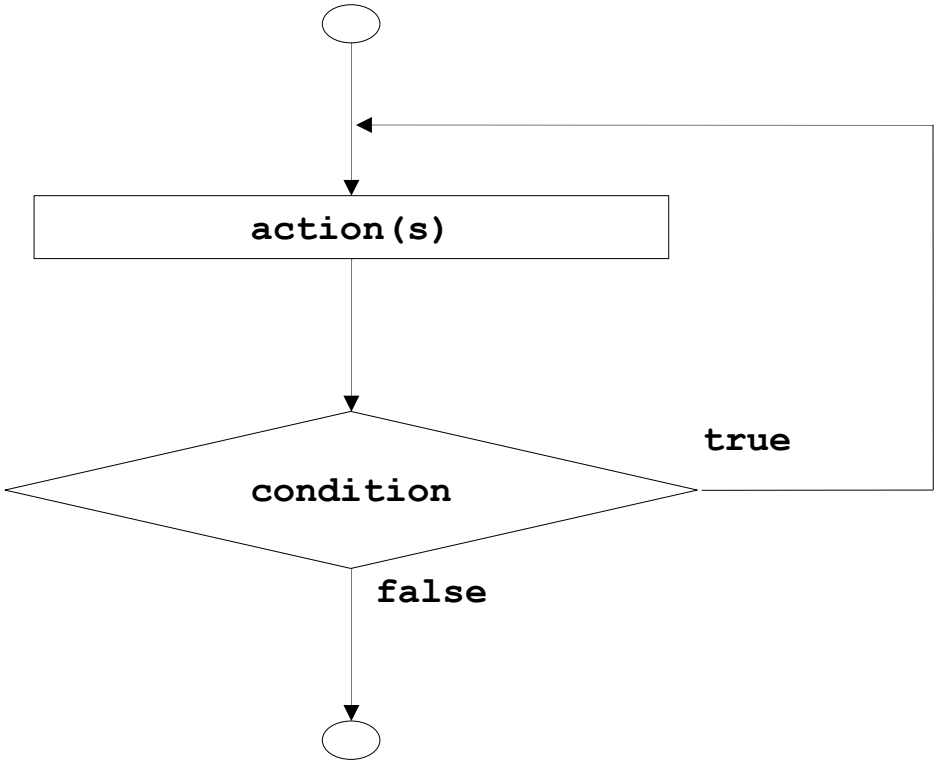
do-while

A volte si desidera eseguire il corpo di un ciclo almeno una volta. In tal caso si utilizza un ciclo do-while, ossia un ciclo a ***condizione finale***. La struttura è la seguente:

do *istruzione*
while (*espressione*);

In questo caso, l'espressione viene valutata al termine dell'esecuzione dell'istruzione (o del blocco di istruzioni). Fino a quando l'espressione è vera, l'istruzione viene ripetuta.

LA STRUTTURA DI RIPETIZIONE DO-WHILE



Esempio

/*Stampa gli interi da 1 a 10*/

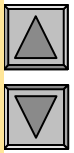
```
counter = 1;

      do {
           printf( "%d  ", counter );
      } while (++counter <= 10);
```

Esempio

```
   do {
     printf("Inserisci un intero compreso tra 0 e 15, inclusi:");
     scanf("%d", num);
       }
   while (i<0 || i > 15);
```


```
1
2    /* uso della struttura do/while */
3  #include <stdio.h>
4
5  int main()
6  {
7     int counter = 1;
8
9     do {
10       printf( "%d  ", counter );
11    } while ( ++counter <= 10 );
12
13    return 0;
14 }
```

```
1  2  3  4  5  6  7  8  9  10
```



Outline

Inizializzazione delle variabili

Ciclo do-while

Stampa dei risultati

```
/* Determina somma e maggiore dei valori immessi    (esempio di uso do-while) */
#include <stdio.h>
main()
{
int somma,numero,max,i;
printf("SOMMA E MAGGIORE\n");
printf("zero per finire\n");
numero = 1;
somma = 0;
max = 0;
i = 1;
do {
      printf("Valore int.: ");
      scanf("%d", &numero);
      if(numero>max)
        max = numero;
      somma = somma+numero;
      i++;
}
while (numero!=0 && i<=10);     /*termina se si inserisce 0 oppure se i>10 */

printf("Somma: %d\n", somma);
printf("Maggiore: %d\n", max);
}
```


```
/* Determina lo zero della funzione f(x) = 2x^3-4x+1
                Metodo di bisezione*/
#include <stdio.h>
#include <math.h>
#define ERR 0.001
main()
{
float a, b, m;
float  fa, fb, fm;
char x;
/* lettura di a, b e controllo validità degli estremi*/
do {
    printf("Inserire a: ");
    scanf("%f", &a);
    printf("Inserire b: ");
    scanf("%f", &b);
/* Calcolo della funzione per x=a */
   fa = 2*a*a*a-4*a+1;
/* Calcolo della funzione per x=b */
   fb = 2*b*b*b-4*b+1;
}
while(fa*fb>0);
/* calcolo zero f */
do {
     m  = (a+b)/2;
/* Calcolo della funzione per x=m */
     fm = 2*m*m*m-4*m+1;
     if(fm!=0) {
 /* Calcolo della funzione per x=a */
    fa = 2*a*a*a-4*a+1;
/* Calcolo della funzione per x=b */
    fb = 2*b*b*b-4*b+1;
        if(fa*fm<0)
           b=m;
        else
            a=m;
/* Calcolo della funzione per x=m */
   fm = 2*m*m*m-4*m+1;
}
}
while(fabs(fm) > ERR);
printf("Zero di f in %7.2f\n", m);
}
```


L' ISTRUZIONE BREAK

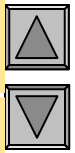
`break`

- Causa l'uscita immediata da cicli **`while`**, **`for`**, **`do`**/**`while`** o struttura **`switch`**
- L'esecuzione del programma prosegue con la prima istruzione dopo la struttura
- Usi comuni dell'istruzione **`break`**
  - Uscita da un ciclo prima della sua terminazione
  - Salto della parte restante di una struttura **`switch`**

L'ISTRUZIONE CONTINUE

continue

- Salta le istruzioni rimanenti nel corpo di strutture **while**, **for** o **do/while**
- Procede con l'iterazione successiva del ciclo **while** e **do/while**
- Il controllo sulla prosecuzione del ciclo è effettuato immediatamente dopo l'esecuzione dell'istruzione continue
- Nella struttura **for**, viene eseguita l'espressione di incremento e poi viene valutato il test di continuazione del ciclo



Outline

Uso dell'istruzione continue per saltare la stampa del valore 5

```
1
2 /* Uso della struttura continue in una struttura for */
3 #include <stdio.h>
4
5 int main()
6 {
7    int x;
8
9    for ( x = 1; x <= 10; x++ ) {
10
11      if ( x == 5 )
12         continue; /* salta la restante parte del ciclo se x==5 */
13
14
15      printf( "%d ", x );
16   }
17
18   printf( "\nUsed continue to skip printing the value 5\n" );
19   return 0;
20 }
```

```
1 2 3 4 6 7 8 9 10
Used continue to skip printing the value 5
```

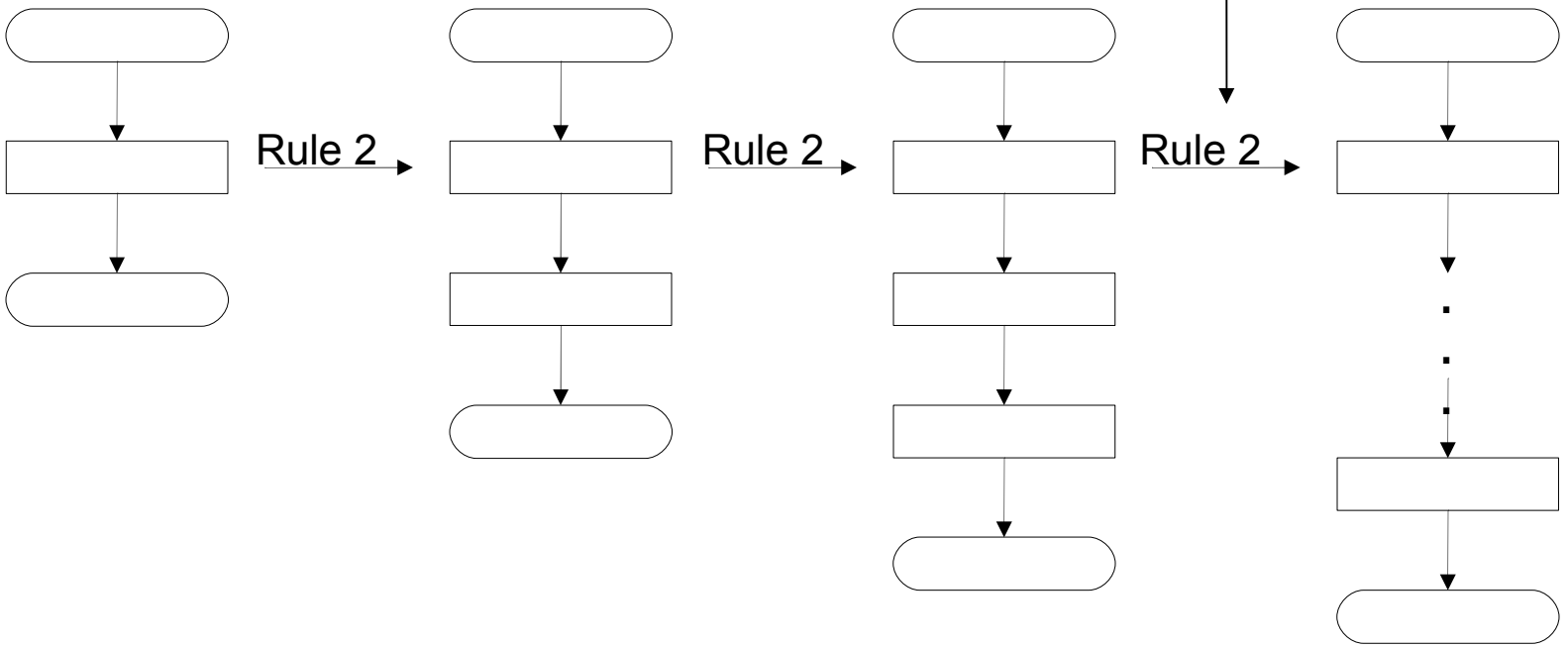
PROGRAMMAZIONE STRUTTURATA

- Programmazione strutturata
  - Rispetto alla programmazione non strutturata, i programmi sono più semplici da capire, testare e modificare
- Regole della programmazione strutturata
  - Si usano solo strutture ad un ingresso – un'uscita
  - Regole:
    1) Iniziare dal diagramma di flusso "più semplice possibile"
    2) Ogni rettangolo (azione) può essere sostituito da due rettangoli (azioni) in sequenza.
    3) Ogni rettangolo (azione) può essere sostituito da una qualsiasi delle strutture di controllo (sequenziale, **`if`**, **`if/else`**, **`switch`**, **`while`**, **`do/while`** o **`for`**).
    4) Le regole 2 e 3 si possono applicare in qualsiasi ordine e più volte.

Programmazione strutturata (cont.)

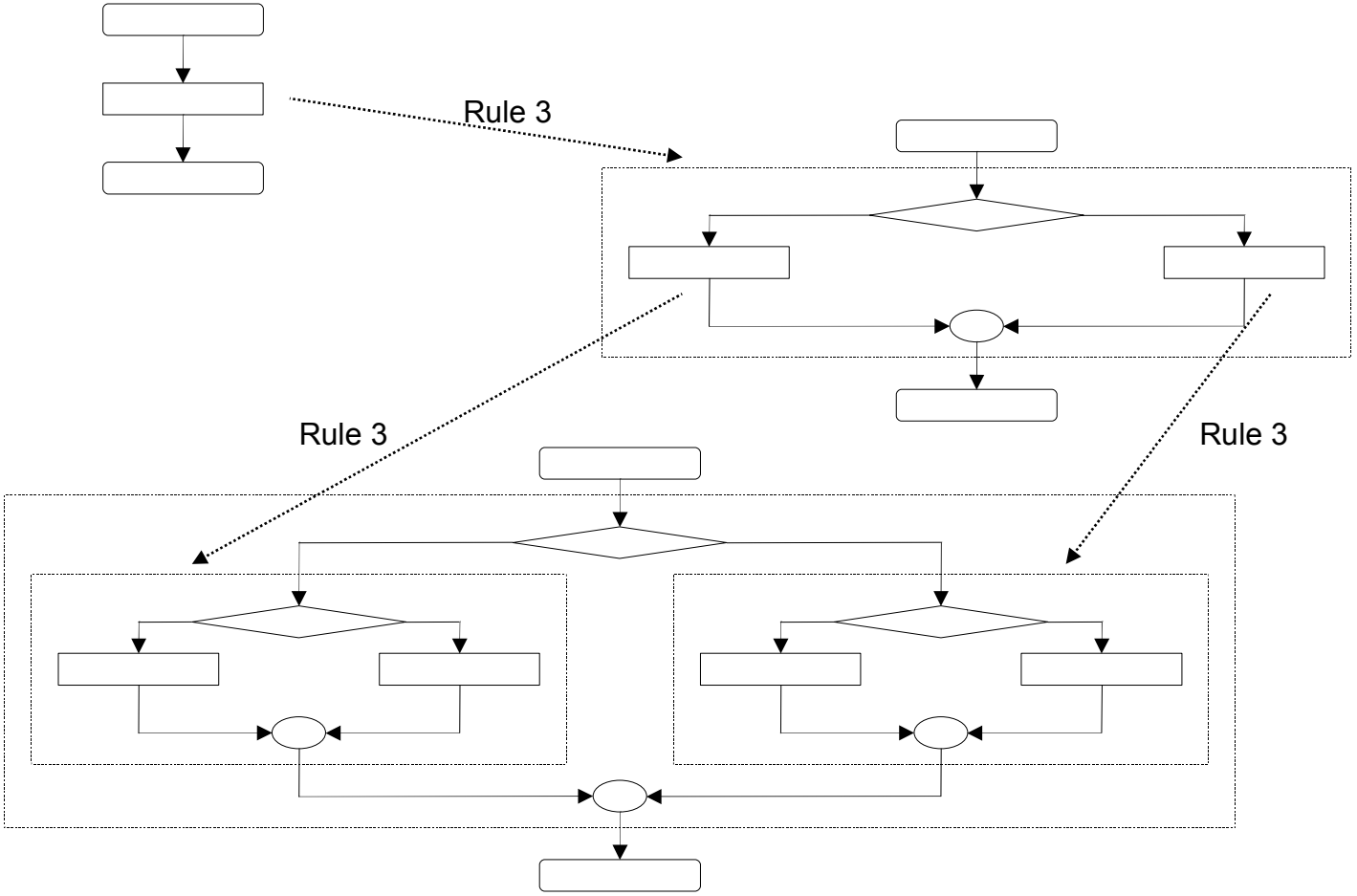
Regola 1 – Iniziare con il diagramma di flusso più semplice

Regola 2 - Ogni rettangolo può essere sostituito da due rettangoli in sequenza



Programmazione strutturata (cont.)

Regola 3 – Sostiture ogni rettangolo con uno schema di controllo



Programmazione strutturata (cont.)

- Tutti i programmi possono essere suddivisi in tre parti:
  - Sequenza -
  - Selezione - **`if`**, **`if/else`**, or **`switch`**
  - Ripetizione - **`while`**, **`do/while`**, or **`for`**

Ogni selezione può essere riscritta con un'istruzione ***`if`*** *e ogni ripetizione può essere riscritta con un'istruzione* ***`while`***

- I programmi possono essere ridotti a:
  - Sequenza
  - **`Struttura if`** (selezione)
  - **`Struttura while`** (ripetizione)

Le strutture di controllo si possono combinare in due soli modi: annidamento (regola 3) e sequenza (regola 2). Tutto questo è indice di semplicità