



UNIVERSITY OF TRENTO - Italy

Laboratorio 2

Pierluigi Roberti
Carmelo Ferrante

DISI – aa 2025-26

Università degli Studi di Trento

pierluigi.roberti@unitn.it

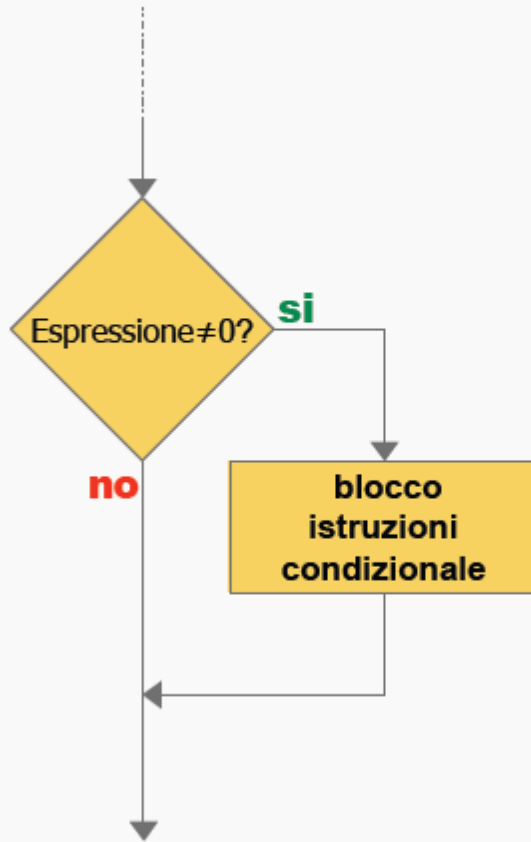
Il controllo del flusso di esecuzione

- In un programma scritto in **C/C++** le istruzioni che si incontrano vengono eseguite in maniera sequenziale ed ordinata, nell'ordine in cui sono scritte.
- Tuttavia, in molti casi è necessario modificare questo naturale modo di procedere, cioè modificare il flusso d'esecuzione del programma.
- Lo scopo della modifica del flusso d'esecuzione è quello di poter scrivere un programma che - a fronte dei particolari valori dei dati di input e a seconda del risultato di determinati calcoli intermedi (valori e risultati non noti in generale al tempo di stesura del codice) - possa comportarsi in maniera flessibile e dunque eseguire in pratica una sequenza di istruzioni che non è la semplice lista sequenziale dei "comandi" che si trovano nel testo del programma.
- Il **C/C++** mette a disposizione degli opportuni costrutti per soddisfare queste necessità.

Il controllo del flusso di esecuzione

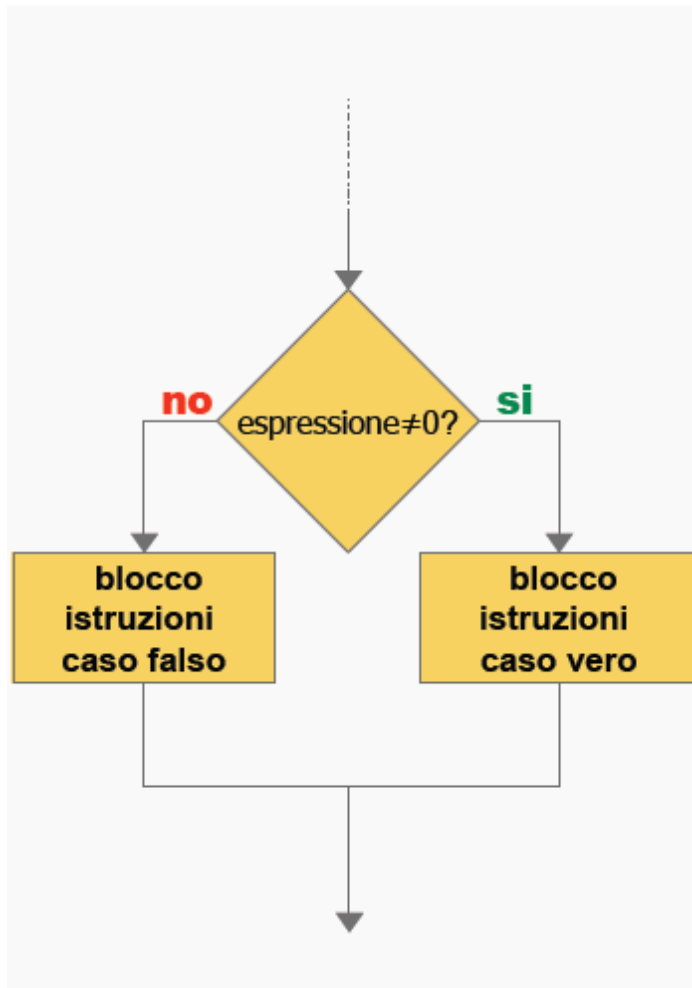
- Quattro tipiche necessità di modifica del flusso d'esecuzione di un programma (con i relativi costrutti **C/C++**) sono le seguenti:
 - Esiste un blocco di istruzioni che va eseguito soltanto se una certa condizione è vera, mentre se è falsa non si deve fare niente:
 - si usa il **costrutto if**
 - Esiste un blocco di istruzioni che va eseguito se una certa condizione è vera, mentre se è falsa esiste un altro blocco di istruzioni da eseguire in alternativa:
 - si usa il **costrutto if-else**
 - Esiste un'espressione che può dare un certo numero di risultati, e a seconda del risultato va eseguito un blocco di istruzioni specifico per quel risultato:
 - si usa il **costrutto switch**
 - Esiste un blocco di istruzioni che va eseguito più di una volta di seguito, e la differenza tra le successive esecuzioni è solo nel valore di alcune variabili presenti nel codice eseguito, e non nel codice medesimo:
 - si usa uno dei **costrutti do, while, for**.

Costrutto if



```
...  
  
if (espressione) {  
    blocco  
istruzioni  
condizionale  
}  
  
...
```

Costrutto if - else



```
...  
if (espressione) {  
    blocco  
istruzioni  
caso vero  
} else {  
    blocco  
istruzioni  
caso falso  
}  
...
```

Espressioni logiche

- Le espressioni logiche generano come risultato un valore **VERO** o **FALSO**
 - Valore uguale a 0 → **FALSO**
 - qualsiasi valore diverso da 0 → **VERO**
- Le espressioni di controllo
 - non inizializzano una variabile ad un valore, ma
 - valutano una condizione.

Espressioni logiche

- Gli operatori di confronto
 - > (maggiore),
 - >= (maggiore o uguale),
 - < (minore),
 - <= (minore o uguale)
 - == (uguale, ATTENZIONE: non é l'operatore di assegnamento),
 - != (diverso)

Operatori booleani e tavole di verità

- In algebra di Boole, AND rappresenta la moltiplicazione e OR l'addizione
- Due espressioni booleane sono equivalenti se hanno la medesima tavola di verità
- Leggi di De Morgan
 - $A \text{ AND } B = \text{NOT} ((\text{NOT } A) \text{ OR } (\text{NOT } B))$
 - $A \text{ OR } B = \text{NOT} ((\text{NOT } A) \text{ AND } (\text{NOT } B))$

A	B	A AND B
0	0	0
0	1	0
1	0	0
1	1	1

A	B	A OR B
0	0	0
0	1	1
1	0	1
1	1	1

A	A NOT B
0	1
1	0

Condizioni in C++

- Si costruiscono usando

- operatori *relazionali*

== != < > <= >=

- operatori logici (booleani)

! (NOT) && (AND) || (OR)

- Esempi:

- **X == 0**

- **X > 0 && A != 3**

- **! ((x+5)*10 >= (ALFA3/BETA_DUE))**

- **(x+5)*10 < (ALFA3/BETA_DUE)**

C++: statement condizionali

- Statement condizionale **if-then**

```
if ( <condizione> ) {  
    // Blocco istruzioni se la <condizione> è vera  
}
```

- Statement condizionale **if-then-else**

```
if ( <condizione> ) {  
    // Blocco istruzioni se la <condizione> è vera  
} else {  
    /* Blocco istruzioni se la <condizione> è falsa  
       (OPZIONALE) */  
}
```

Condizioni in C++

```
int dato;  
printf("dato=");  
scanf("%d",&dato);  
  
//verifico appartenenza intervallo [3,9)  
if (dato>=3 && dato<9){  
    printf("compreso\n");  
}else{  
    printf("non compreso\n");  
}  
  
//condizione uguale  
if ( !(dato<3 || dato>=9) ){  
    printf("compreso\n");  
}else{  
    printf("non compreso\n");  
}
```

C++: statement condizionali

- Statement condizionale **if-then-else** innestati

// Attenzione alle parentesi: i due frammenti di codice sotto sono diversi

```
if ( <condizione1> ) {  
    // Blocco istruzioni se <condizione1> è vera  
    if ( <condizione2> ) {  
        // Blocco istruzioni se la <condizione2> è vera  
    }  
    else {  
        // Blocco istruzioni se la <condizione2> è falsa  
    }  
}
```

```
if ( <condizione1> ) {  
    // Blocco istruzioni se <condizione1> è vera  
    if ( <condizione2> ) {  
        // Blocco istruzioni se la <condizione2> è vera  
    }  
}  
else {  
    // Blocco istruzioni se la <condizione1> è falsa  
}
```

Istruzione condizionale: `if-else`

- È possibile modificare il ***flusso di controllo*** del programma (cioè quali istruzioni vengono eseguite) in funzione del risultato della valutazione di una espressione condizionale

```
int x; if (x>0) {printf("%d",x) ; }  
else {printf("%d",-x) ; }
```

- Per maggior leggibilità, è bene usare regole di incolonnamento (***indentazione***)

```
if (x>0) {  
    printf ("%d" , x) ;  
} else {  
    printf ("%d" , -x) ;  
}
```

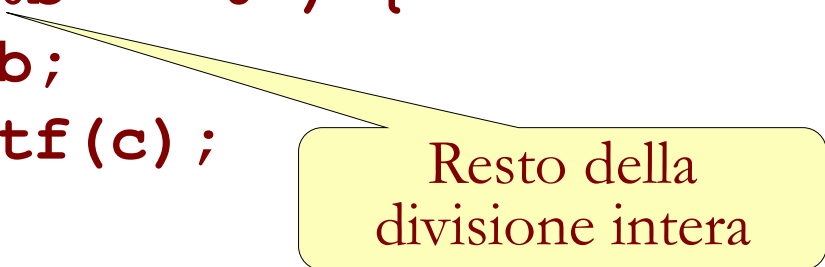
Esercizio 1

```
int dato;  
printf("dato="); scanf("%d",&dato);  
  
//codice non ottimale: sempre 3 confronti  
if (dato>0) { printf("maggiore 0\n"); }  
if (dato<0) { printf("minore 0\n"); }  
if (dato==0) { printf("uguale 0\n"); }  
  
//codice ottimale al massimo 2 confronti  
if (dato>0){  
    printf("maggiore 0\n");  
} else {  
    if (dato<0){  
        printf("minore 0\n");  
    } else {  
        printf("uguale 0\n");  
    }  
}
```

Istruzione composta

- Si possono raggruppare più istruzioni in sequenza tra **{** e **}** a costituire un blocco del tutto equivalente a una singola istruzione
 - Analogamente al blocco **main()**...
- Non è necessario il **;** dopo **}**, in quanto quest'ultimo carattere fa già da delimitatore
 - Se si aggiunge, non viene segnalato errore
- Esempio:

```
if ( a%b == 0 ) {  
    c=a/b;  
    printf(c) ;  
}
```



Resto della
divisione intera

Esercizio 2

- Leggere due variabili di tipo intero denominate **x** e **y** in input e stampare a video un messaggio che dice quale delle due variabili è la maggiore
- **Da sapere/ricordare:**

```
if (<condizione>) {  
    <istruzioni>;  
}else{  
    <istruzioni>;  
}
```


Esercizio 2 - Soluzione

```
#include <stdio.h>
```

```
int main() {  
    int x, y;  
    printf("valore di x:");  
    scanf("%d", &x);  
    printf("valore di y:");  
    scanf("%d", &y);  
    if (x > y){  
        printf("%d e' maggiore di %d\n", x, y);  
    }else{  
        if (x < y){  
            printf("%d e' minore di %d\n", x, y);  
        }else{  
            printf("i due numeri sono uguali\n");  
        }  
    }  
    system("PAUSE");  
}
```

if: regole sintattiche

- È possibile omettere la parte **else**
`if (x>0) printf(x) ;`
- Nel caso di **if** nidificati possono esserci delle ambiguità:
`if(n > 0) if(a>b) z = a; else z = b;`
 - A quale dei due **if** si riferisce l'**else**?
- Il C++ risolve associa **else** all'**if** più vicino
 - L'indentazione lo rende evidente (ma non al compilatore!)
`if(n > 0)
 if(a > b) z = a;
else z = b;`
 - Se incerti, usare le parentesi (processate dal compilatore)
`if(n > 0) {
 if(a > b) z = a;
 else z = b;
}`

Esercizio 3

- Scrivere un programma che legga da tastiera tre numeri (denominati n1, n2 e n3) e stampi a video il **minimo**, il **massimo** e la **media** di questi tre numeri.

- Esempio di interazione:

Inserisci il primo numero: **11**

Inserisci il secondo numero: **6**

Inserisci il terzo numero: **10**

Il valore minimo e': 6

Il valore massimo e': 11

La media aritmetica e': 9

- Cosa succede inserendo **2**, **6** e **8** come input?

Esercizio 3 – soluzione 1/2

```
int main() {  
    int n1, n2, n3;  
  
    printf("Inserisci primo numero: ");  
    scanf("%d", &n1);  
    printf("Inserisci secondo numero: ");  
    scanf("%d", &n2);  
    printf("Inserisci terzo numero: ");  
    scanf("%d", &n3);  
  
    // ...  
}
```

Esercizio 3 – soluzione 2/2

```
if (n1<n2 && n1<n3) {
    printf("Il valore minimo e' %d\n", n1);
} else {
    if (n2<n1 && n2<n3) {
        printf("Il valore minimo e' %d\n", n2);
    } else {
        printf("Il valore minimo e' %d\n", n3);
    }
}

if (n1>n2 && n1>n3) {
    printf("Il valore massimo e' %d\n", n1);
} else {
    if (n2>n1 && n2>n3) {
        printf("Il valore massimo e' %d\n", n2);
    } else {
        printf("Il valore massimo e' %d\n", n3);
    }
}

printf("La media aritmetica e' %f\n", (float)(n1+n2+n3)/3);
```

Esercizio 3 — soluzione alternativa

```
int main() {
    int n1, n2, n3, min, max;
    printf("Inserisci primo numero: ");
    scanf("%d", &n1);
    printf("Inserisci secondo numero: ");
    scanf("%d", &n2);
    printf("Inserisci terzo numero: ");
    scanf("%d", &n3);
    min=n1; max=n1;
    if (n2>max) { max = n2; }
    if (n3>max) { max = n3; }
    if (n2<min) { min = n2; }
    if (n3<min) { min = n3; }
    printf("Il valore minimo e' %d\n", min);
    printf("Il valore massimo e' %d\n", max);
    printf("La media aritmetica e' %f\n", (n1+n2+n3)/3.0);
}
```

Valutazione Condizioni

- Le condizioni sono espressioni che hanno valore *vero* o *falso*
- Convenzionalmente, vero (TRUE) è rappresentato dal bit 1, falso (FALSE) dal bit 0
 - I valori TRUE e FALSE (0 e 1) sono anche detti valori **booleani**
 - In C++ si usa una convenzione lievemente diversa
 - TRUE → qualunque valore intero diverso da 0
 - FALSE → il valore 0
- L'*algebra di Boole* è basata su operazioni *logiche* che si applicano a operandi che possono assumere solo i valori vero e falso

Valutazione Condizioni

- Esistono *regole di precedenza*, come per gli altri operatori
 - Es.: **! a || b && c**
 - Viene valutato prima NOT, poi AND, e ultimo OR
 - Se in dubbio, usare le parentesi tonde
- Gli operatori **&&** e **||** si valutano da sinistra a destra
- La valutazione termina quando si è raggiunta la certezza del valore di verità dell'espressione (lazy evaluation)
 - Esempio 1:
(i==0) || ((a/i)>5)
non genera un errore se **i=0**
 - Esempio 2:
((a/i)>5) || (i==0)
genera un errore se **i=0**

Esercizio 4

- Scrivere un programma che letti input i valori di 3 lati (denominati x , y e z); una volta verificato che siano tutti numeri positivi, verifica se tali valori possono rappresentare i lati di un triangolo (la somma di 2 lati qualsiasi deve essere maggiore del terzo).

In caso delle condizioni precedenti verificate verificare di che tipo di triangolo si tratta:

- equilatero,
- isoscele,
- scaleno

Soluzione esercizio 4

```
main() {  
    int x,y,z;  
    printf("x="); scanf("%d",&x);  
    printf("y="); scanf("%d",&y);  
    printf("z="); scanf("%d",&z);  
    if (x<=0 || y<=0 || z<=0){  
        printf("I valori devono essere positivi!");  
    } else {  
        if (x<y+z && y<x+z && z<x+y) {  
            if (x==y && y==z) {  
                printf("E' un triangolo equilatero");  
            } else {  
                if (x==y || y==z || x==z) {  
                    printf("E' un triangolo isoscele"); }  
                } else {  
                    printf("E' un triangolo scaleno"); }  
            }  
        } else {  
            printf("Non è un triangolo!"); }  
    }  
}
```

Esercizio 5

- Scrivere un programma che, ricevuto in ingresso un anno (per es. 2004), indichi a video se questo è bisestile.
 - Un anno è bisestile se è divisibile per 400 **oppure**
 - se è divisibile per 4 ma non per 100.

Esercizio 6

- Scrivere un programma che controlli gli ingressi ad una discoteca. Il programma riceve in input da tastiera il sesso ('m' o 'f') e l'età di una persona e restituisce a video un messaggio indicante il costo dell'ingresso (30 euro per i maschi, gratis per le femmine) nel caso in cui l'età sia almeno pari a 21 oppure il divieto di accedere (se l'età è minore di 21).

Quiz per verifica competenze

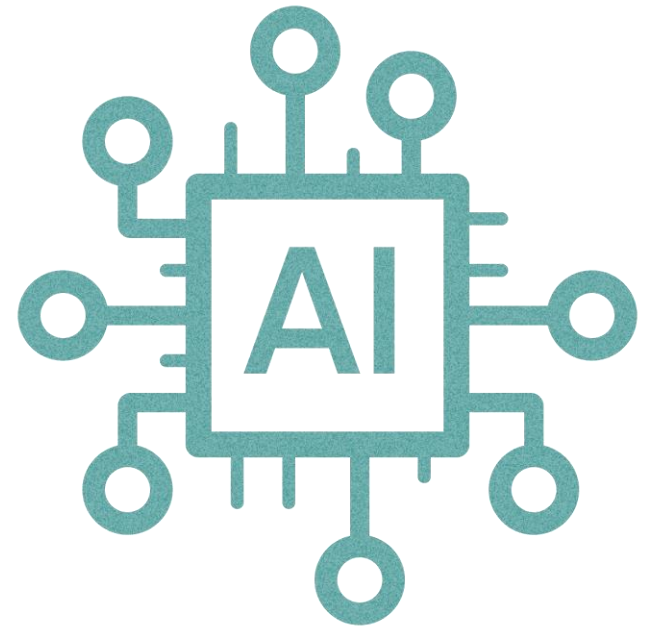
Quiz generati
automaticamente usando
l'applicazione:

Generative AI 4 Education
e supervisionati dal docente.

Il quiz è anonimo.

Bisogna essere autenticati nel dominio UNITN!

Non sono esempi di domande di esame!



<https://forms.gle/8EFr3GPRRk9iGNFm9>