Tipi di dato e istruzioni elementari in C++

Docente: Ing. Edoardo Fusella
Dipartimento di Ingegneria Elettrica e Tecnologie dell'Informazione
Via Claudio 21, 4° piano – laboratorio SECLAB
Università degli Studi di Napoli Federico II
e-mail: edoardo.fusella@unina.it

Dichiarazioni di variabili: identificatori

- In C++, una variabile viene **dichiarata** con la sintassi:

  Tipo <nome_variabile>;

  - <*nome_variabile*> è un nome scelto dal programmatore formato da una qualunque sequenza di lettere (maiuscole o minuscole) e di cifre numeriche, senza altri caratteri o spazi bianchi a parte l'underscore (che è considerato una lettera), che deve cominciare con una lettera;
  - Gli identificatori possono contenere un numero qualsiasi di caratteri ma solo i primi 31 caratteri sono significativi per il compilatore;
  - Il C++ è case-sensitive: significa che il compilatore considera le lettere maiuscole e minuscole come caratteri distinti;
  - L'identificatore scelto rappresenta un'area di memoria la cui dimensione dipende dal tipo della variabile;
  - E' conveniente usare nomi significativi per le variabili;

- Esempi di dichiarazioni

```
int x2;        //x2 è un intero
char c;        //c è un carattere
```

Definizione di una variabile

- La variabile è **definita** quando le viene assegnato un valore, che viene quindi scritto nella cella di memoria
- Una variabile deve essere sempre dichiarata prima di essere definita;
- La definizione di una variabile viene effettuata con *istruzioni di assegnazione*

variabile = espressione;

- Esempi di definizioni

  x2=3;

  c='a';
- Dichiarazione e definizione possono avvenire in un sol colpo:

  int x2=3;

Costanti

- Le costanti esprimono dei valori prefissati non alterabili e possono essere di tipo numerico ed alfanumerico
- Le costanti numeriche si distinguono in:
  - Costanti intere: sequenze di cifre eventualmente precedute da un segno
  - Costanti reali: si esprimono sia in virgola fissa esplicitando la posizioni del punto che in virgola mobile indicando l'esponente con la lettera e (minuscola o maiuscol)
    - 10.23, -0.11
    - 0.1e-23, -10.0003E10
- Le costanti alfanumeriche si distinguono in:
  - Costanti carattere: caratteri ASCII racchiusi tra apici
    - 'A', 'c', '?', '2'
  - Costanti stringhe di caratteri: sequenze di caratteri ASCII racchiusi tra doppie virgolette
    - "questa è una stringa"

Costanti stringhe di caratteri

- La stringa "" è una stringa di lunghezza nulla
- Con il backslash \ si può inserire qualsiasi carattere ASCII specificandone il codice in ottale o esadecimale
  - \101 è la lettera A codificata come 101 in ottale
  - \x41 è la lettera A codificata come 41 in esadecimale
- Una stringa di caratteri può essere spezzata su più linee usando il backslash
  - " questa è una stringa \

    molto lunga "
- E' possibile inserire caratteri speciali sia all'interno delle stringhe di caratteri che nelle costanti carattere

Carattere speciale	Descrizione
\n	Newline o anche ritorno a capo
\r	Carriage return o ritorno a inizio rigo
\t	Tab
\v	Vertical tab
\b	Backspace
\f	Form feed
\a	Alert o beep
\'	Singolo apice
\"	Singole virgolette
\?	Punto interrogativo
\\	Backslash

Dichiarazione di costanti

- Il linguaggio consente di assegnare alle costanti un nome da usare al loro posto nel programma per migliorare la leggibilità e la parametricità
- Per dare un nome ad una costante si usa la direttiva **#define**

#define *nome_costante* valore_costante

  - Nota: nome_costante non è una variabile a cui viene assegnato spazio in memoria!
- E' possibile anche dichiarare costanti tipizzate, usando il prefisso const

const nome_tipo *nome_costante* = valore;

  - Nota: in questo caso a nome_costante viene attribuito uno spazio di memoria non modificabile

Tipi delle variabili C++

- I tipi delle variabili c++ possono essere:
  - **Semplici/standard/nativi :** sono atomici ovvero non suddividibili e ci permettono di definire alcune tipologie base di variabili.
    - Esempi : interi, booleani, caratteri
  - **Strutturati :** sono composti da più tipi semplici o strutturati stessi (quindi suddivisibili)
    - Esempi : vettori di interi, vettori di stringhe

Tipi semplici C++

Tipo	Applicato in questa situazione:	Byte	Range
unsigned char	Numeri piccoli e insieme completo dei caratteri	1	0 ÷ 255
char	Numeri molto piccoli e caratteri ASCII	1	-128 ÷ 127
enum	Insiemi ordinati di valori	2	-32.768 ÷ 32.767
unsigned int	Numeri grandi e cicli	2	0 ÷ 65.535
short int	Contatori, numeri piccoli e cicli	2	-32.768 ÷ 32.767
int	Contatori, numeri piccoli e cicli	2	-32.768 ÷ 32.767
unsigned long	Distanze astronomiche	4	0 ÷ 4.294.967.295
long	Numeri grandi, popolazioni	4	-2.147.483.648 ÷ 2.147.483.647
float	Scientifico (precisione a 7 cifre)	4	$3,4 * 10^{-38} \div 3,4 * 10^{38}$
double	Scientifico (precisione a 15 cifre)	8	$1,7 * 10^{-308} \div 1,7 + 10^{308}$
long double	Finanziario (precisione a 19 cifre)	10	$3,4 * 10^{-4932} \div 3,4 * 10^{4932}$

- Nota: I range indicati sono prescritti dallo standard ISO e indicano gli intervalli minimi da garantire; l'effettiva dimensione dei tipi dipende esclusivamente dalla piattaforma utilizzata
- Con l'istruzione **sizeof(type**) è possibile verificare la dimensione di un tipo sulla propria architettura

- A questi tipi si aggiunge il tipo **booleano** (bool), che occupa un byte e può assumere i valori true (1) e false (0);
  - Solitamente qualsiasi valore diverso da 0 viene considerato true

Tipo enumerativo

- Il C++ consente di dichiarare un nuovo tipo elencando i valori costanti che lo compongono con l'operatore **enum;** le variabili a cui viene assegnato questo tipo potranno assumere uno dei valori indicati
- **Esempio:**

  enum *colore* {bianco, rosso, verde};
  colore bandiera;

- Nota: le costanti definite nel tipo enum sono ordinate in base alla loro posizione (rosso>bianco e rosso<verde); il compilatore associa ad ogni costante il valore intero corrispondente alla posizione occupata (partendo da 0)

- E' possibile specificare dei valori interi in maniera esplicita:
- Esempio:

  enum *colori* {bianco=3, rosso=5, verde=6};
  enum *risposte* {si='s', no='n'};
- I tipi definiti con enum possono essere usati per dichiarare variabili di tipo intero

Commenti

- In C++ le frasi di commento si indicano anteponendo alla frase la sequenza // quando il commento è su un solo rigo;
- Quando la frase occupa più righe la si può racchiudere fra /* e */
- I commenti possono cominciare subito dopo un'istruzione (dopo il «;»)

Operatori (1/3)

Operatori aritmetici

Operatore	Descrizione
+	Somma
-	Sottrazione
*	Moltiplicazione
/	Divisione
%	Modulo che restituisce il resto della divisione tra due interi

Operatori logici

Operatore	Descrizione
&&	AND o anche prodotto logico
||	OR o anche somma logica
!	NOT o anche negazione logica

Operatori bitwise

Operatore	Descrizione
&	AND bit a bit
|	OR bit a bit
^	OR esclusivo bit a bit
~	Complementazione bit a bit o anche inversione dei bit (zero al posto di uno e viceversa)
<<	Shift left
>>	Shift right

Operatori relazionali

Operatore	Descrizione
==	Uguaglianza, per determinare se due valori sono uguali
!=	Diversità, per determinare se due valori sono diversi
>	Per determinare se il valore a sinistra precede quello di destra (o è più grande)
<	Per determinare se il valore a sinistra segue quello di destra (o è più piccolo)
>=	Per determinare se il valore a sinistra è più grande o uguale a quello di destra
<=	Per determinare se il valore a sinistra è più piccolo o uguale a quello di destra

Operatori (2/3)

Operatori di incremento e decremento unitario

++i	i++	i = i +1
--i	i--	i = i -1

B = 2;
A = ++B;
//B è uguale a 3
//A è uguale a 3
	B = 2;
	A = B++;
	//B è uguale a 3
	//A è uguale a 2

Operatori composti

```
var = var op espressione;
```

```
var op= espressione;
```

Assegnazione	Equivalente a
B = B * 4	B *= 4
C = C – (D * 4)	C -= D * 4
X = X + (++I)	X += ++I
X = X + 1	X += 1

Operatore condizionale ?

```
condizione ? risulato1 : risultato2
```

```
int valore = 3;
bool positivo = (valore>=0) ? True : false;
```

Operatori (3/3)

Regole di precedenza degli operatori

Priorità	Operatore	Descrizione
1	++ --	Incremento/ decremento postfisso
2	++ --	Incremento/decremento prefisso
2	+ - !	Operatori unari di segno e complemento
3	(tipo)	Conversione di tipo
4	* / %	Moltiplicativi
5	+ -	Additivi
6	<< >>	Shift
7	== != < > <= >=	Relazionali
8	&	AND bitwise
9	^	XOR bitwise
10	|	OR bitwise
11	&&	AND logico
12	||	OR logico
13	? :	Condizionale
14	= *= /= %= += -= >>= <<= &= ^= |=	Assegnazione semplice o composta

Libreria standard

- La **libreria standard** è una collezione di funzioni scritte in C++ che fanno parte dello standard e possono essere richiamate nei programmi per svolgere operazioni elementari e non
  - Funzioni per l'input/output sullo standard input/output
  - Funzioni matematiche
  - Funzione di supporto alla programmazione
- Per poter usare le funzione della libreria standard è necessario includere nel programma un riferimento ad esse con la direttiva:

  #include <nome_file> o #include "nome_file«
  - Se si usano le parentesi angolari, si intende che **nome_file** vada cercato nella ***directory di default*** del linguaggio; se invece si usano le virgolette, il file si trova nella ***directory*** del programma.
- <*nome_file*> è il nome di un *header file* che contiene la dichiarazione delle funzioni da includere (nome e parametri);
  - La direttiva viene letta e interpretata dal preprocessore, che sostituisce ad essa il contenuto del file indicato
  - Il codice sorgente così modificato è poi passato al compilatore, che genera un file oggetto
  - Il linker collega infine questo file oggetto con i file oggetto prodotti per la libreria stessa, e genera l'eseguibile

Namespace std (1/2)

- Tutte le entità (variabili, tipi, costanti, funzioni) dichiarate nella libreria standard fanno parte del ***namespace* std**
  - Un namespace consente di definire un ambito di visibilità per gli oggetti in esso definiti;
  - Due oggetti appartenenti a due namespace diversi possono avere lo stesso nome;
  - Se vogliamo usare due oggetti con lo stesso nome ma appartenenti a due namespace diversi dobbiamo indicare di quale namespace fanno parte con la forma **<namespace>::<var>**

```
namespace myNamespace
{
  int a, b;
}
```

Le variabili **a** e **b** sono normali variabili che possono essere richiamate con il loro nome all'interno del namespace myNamespace; se le si vuole usare fuori da std, è necessario usare la forma: myNamespace::a

Namespace std(2/2)

- Per usare gli oggetti definiti nel namespace std si può usare la forma std::<nome_oggetto> ogni volta che l'oggetto viene richiamato, oppure si può inserire, in testa al programma, l'istruzione:

  using namespace std;

- In questo modo, tutti gli oggetti usati nel programma e non direttamente definiti dal programmatore saranno considerati come provenienti dalla libreria standard

Istruzioni di input

- Permettono di prelevare un valore dallo standard input e di assegnarlo ad una variabile

- In C++ si usa l'istruzione : **cin>>** x;
- Tale istruzione fa parte della libreria **iostream**, che va quindi inclusa nel programma con la direttiva #include
- **L'istruzione cin legge il primo valore in coda dallo standard input (tastiera) e lo assegna alla variabile x**

```
1. {

2.  int x;
3. cin>>x;
4.  …

5. }
```

Nota: la variabile in cui viene salvato lo standard input deve essere prima dichiarata. A seconda del tipo della variabile, il valore letto sarà interpretato come un numero o un carattere

Istruzioni di output

- Permettono di porre un valore dallo standard output

- In C++ si usa l'istruzione: **cout<< x;**
- Tale istruzione fa parte della libreria **iostream**, che va quindi inclusa nel programma con la direttiva #include
- **L'istruzione cout scrive il valore memorizzato nella variabile x sullo standard output (console mostrata a video)**

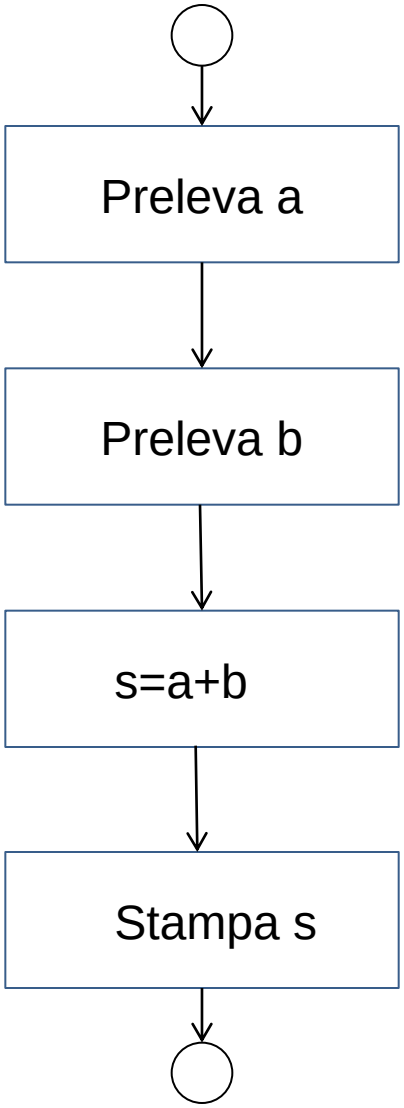
```
1. {
2.  …
3. char x ='c';
4. cout<<x;
5.  …
6. }
```

Nota: il valore scritto sull'output può essere un numero (es. cout<<10;), un carattere (es. cout<<'a';) o una stringa di caratteri (es. cout<<"ciao a tutti";) oppure può essere una variabile che contiene una di queste cose.

Istruzione sequenziale

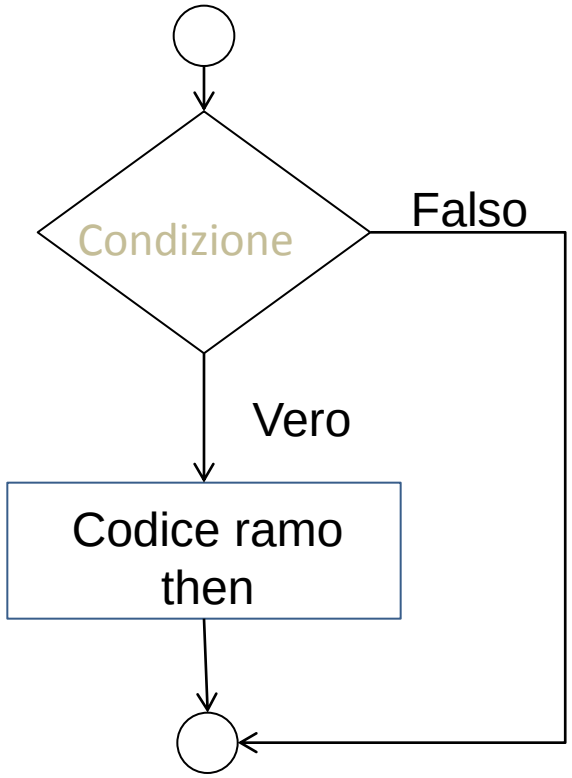
- Permette di eseguire le istruzioni una dopo l'altra
- E' composta da istruzioni semplici o complesse
- E' delimitata dalle parentesi graffe { }
- E' implicita in c++ (non prevede l'uso di parole riservate)
- Esempio : somma di due numeri

```
{
int a,b,s;
cin>>a;
cin>>b;
s=a+b;
cout<<s;
}
```



Istruzione di selezione if

Permette di decidere quale azione eseguire tra varie alternative



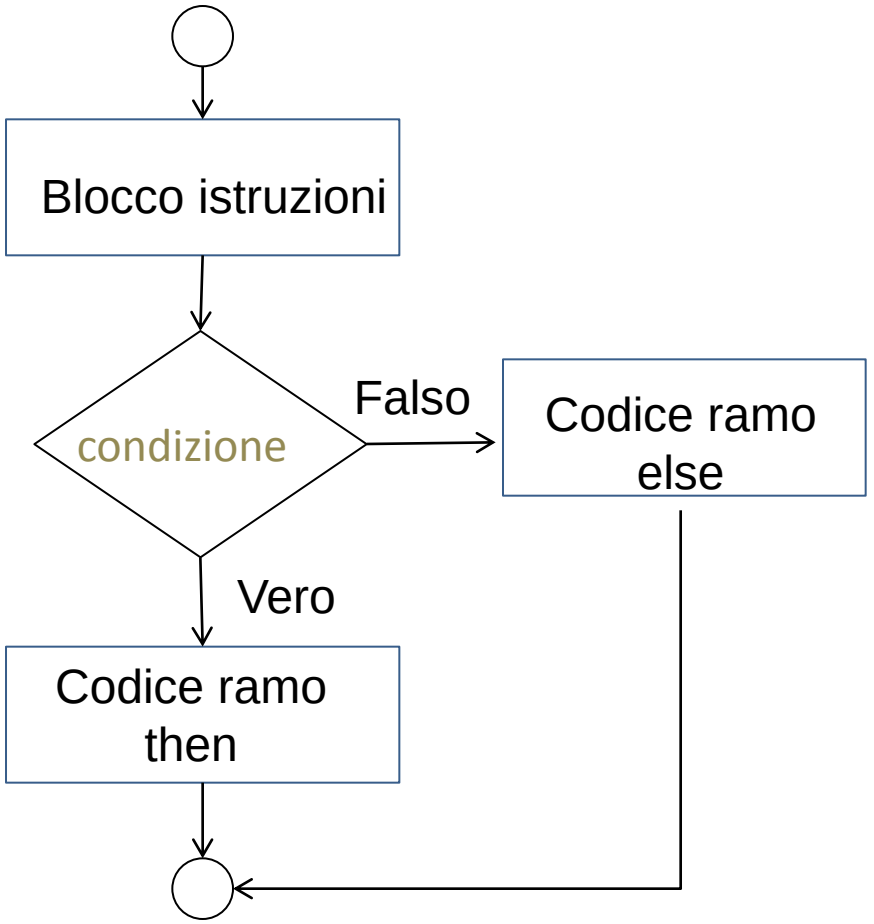

```
1. if(condizione) {
2.   codice ramo
   then;
3. }
```

➢Esempio : (**if**)Se il voto dello studente è maggiore o uguale a 18 **(then)** allora stampa “Promosso”

Istruzione di selezione if-then-else

Fornisce un'azione alternativa se la condizione dell'if non è vera

```
1. Blocco istruzioni;
2. if(condizione) {
3.   codice ramo then;
4. }else{
5.   codice ramo else;
6. }
```

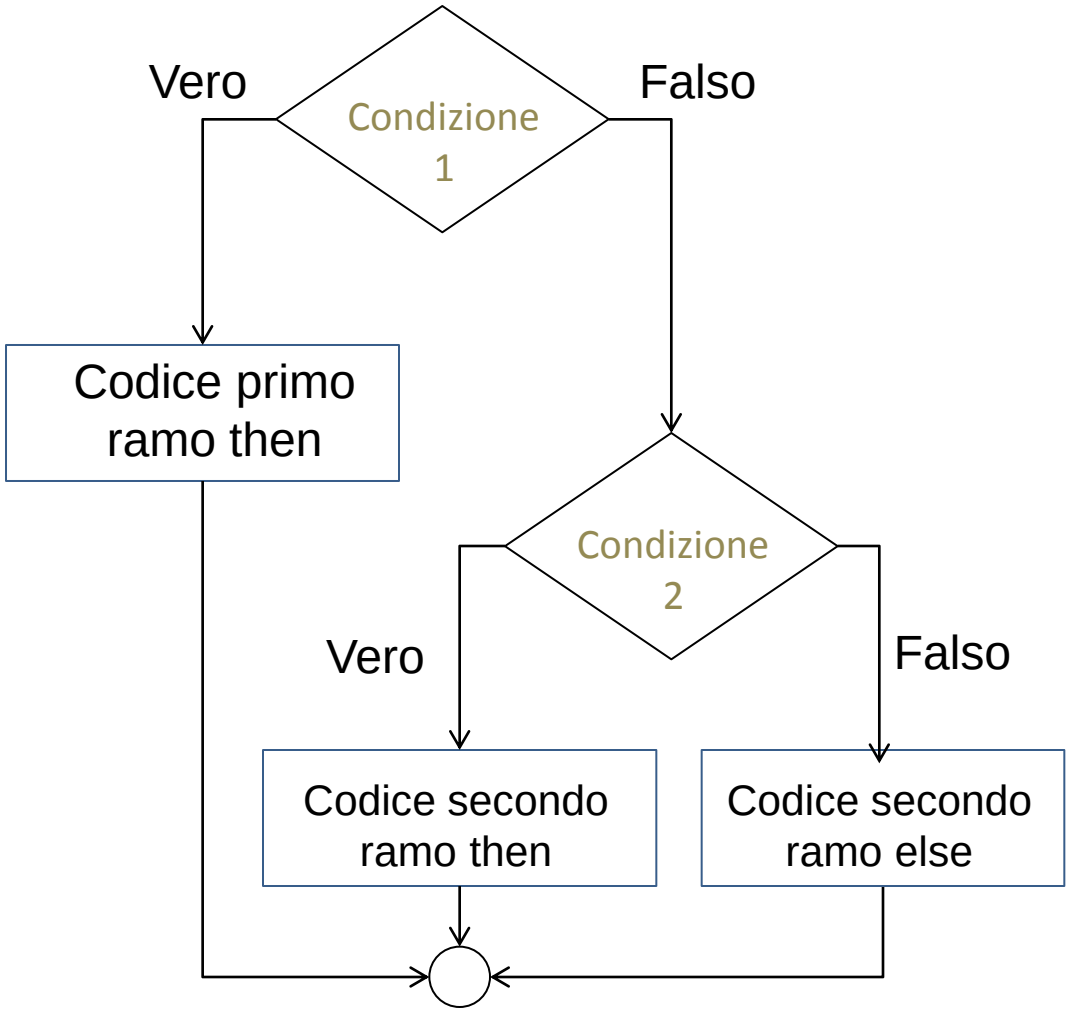


➢Esempio : (**if**)Se il raggio del cerchio è negativo **(then)** dà un messaggio di errore **(else)** calcola l'area

Nesting degli if-then-else

- La parte **then** e la parte **else** possono a loro volta contenere delle istruzioni di selezione
- Una volta che una condizione risulta vera, le rimanenti sono saltate

```
1. if(condizione 1) {
2.   codice primo ramo then;
3. }
4. else if (condizione 2){
5.   codice secondo ramo then;
6. }else {
7.   codice secondo ramo else;
8. }
```

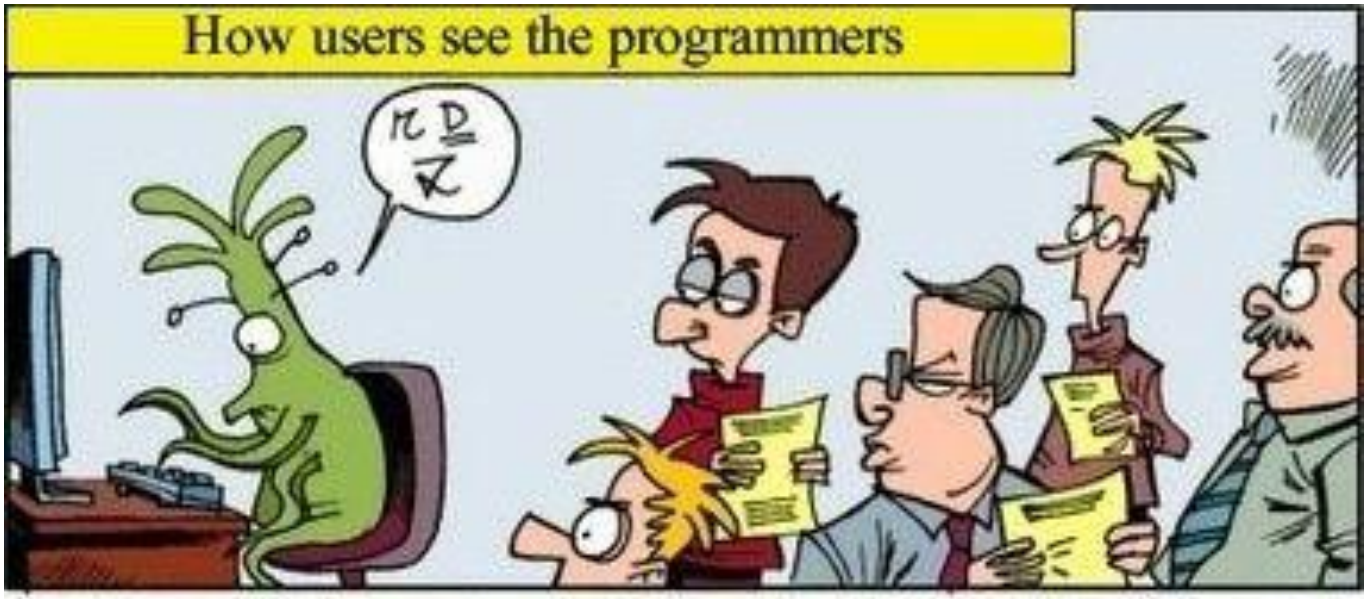


Switch-case

```
switch(<espressione intera>)
{
    case (<valore costante 1>):
        <sequenza di istruzioni 1>
        break;

    case (<valore costante 2>)
        <sequenza di istruzioni 2>
        break;

    ...

    case (<valore costante N>)
        <sequenza di istruzioni N>
        break;

    default:
        <sequenza di istruzioni N+1>
}
```

- Il **costrutto switch-case** serve a eseguire codice diverso a fronte di valori diversi assunti da una espressione.
  - Ad esempio risulta utile quando dobbiamo impostare il comportamento del programma in seguito alla scelta di un'opzione da parte di un utente.
- Il costrutto valuta la condizione intera passata a ***switch*** e rimanda lo svolgimento del programma al blocco in cui il parametro di ***case*** ha lo stesso valore di quello dell'istruzione ***switch***
- Se il blocco termina con ***break;*** allora il programma esce dallo switch, altrimenti vengono eseguiti anche i blocchi seguenti fino ad un break; o fino alla fine
- Se nessun blocco corrisponde ad un valore uguale a quello dell'istruzione switch viene eseguito il blocco in ***default:***

Hands on!



Struttura di un programma C++

- Per poter scrivere un programma in C++ bisogna definire un blocco di ingresso (**entry point**)..
  - Tale entry poin è il **main**
  - Il main ritorna un valore intero che indica il codice di errore (0 significa no error)
- .. e includere le librerie che ci servono
  - Per esempio la direttiva #include<iostream> ci serve per gestire standard input e output!

```
1. #include <iostream>

2. int main(){

3.   //istruzione 1;
4.   //istruzione 2;

5.   //istruzione n;

     return 0;

1. }
```

system("PAUSE")

- **system("PAUSE")**
  - Direttiva al compilatore che indica di sospendere l'esecuzione del programma fino a che non viene premuto un tasto
  - Fa apparire la frase "*Premere un tasto per continuare...*«
  - Bisogna includere la libreria *stdlib.h*
  - Se non si usa, la finestra di esecuzione viene chiusa subito e non si riesce a vedere l'output d un determinato programma

Hello World!

- Il classico «primo programma» di ogni linguaggio di programmazione
- Stampiamo sullo standard output la frase «Hello World!»

```
1. #include<iostream>
2. #include<stdlib.h>

3. using namespace std;
4. int main(){
5.         cout<<"Hello World! \n";
6.         system ("PAUSE");
7.         return 0;
8. }
```

Esercizio 1

Scrivere un programma che, dato un valore reale R inserito dall'utente, calcoli l'area del cerchio che ha come raggio R.
Nel caso in cui l'utente inserisca un numero negativo, il programma deve mostrare un messaggio di errore.

Esercizio 2

Scrivere un programma che, richiesti in ingresso da tastiera all'utente due numeri di tipo intero *a* e *b*, effettua un confronto tra essi usando gli operatori di confronto (==, <, >)

Il programma deve restituire a video una frase del tipo ‘risultato: a **op** b’, dove op è l’operatore che si applica al confronto

Esempio: a=4, b=5 => ‘risultato: a<5’

Esercizio 3

Scrivere un programma che, dati due valori numerici di tipo reale inseriti dall'utente *a* e *b*, effettui una tra le seguenti operazioni:

- La somma:a+b
- La sottrazione: a-b
- Il prodotto: a*b
- La divisione (**ricordando che non si divide per zero!):** a/b

Oltre ad inserire i due valori, l'utente deve anche indicare l'operazione da fare. Si indichi con *scelta* la variabile atta a contenere un valore intero (compreso fra 1 e 4) che indica l'operazione da fare.

Il programma deve restituire il risultato a video.

Esercizio 4

Scrivere un programma che, dati i coefficienti a,b,c di una equazione di secondo grado, calcoli le radici.

$$ax^2 + bx + c = 0 \qquad \Delta = b^2 - 4ac$$

Se $\Delta > 0$ $x_1 = \frac{-b + \sqrt{\Delta}}{2a}$ $x_2 = \frac{-b - \sqrt{\Delta}}{2a}$

- Il programma deve gestire le tre situazioni che si hanno con il discriminante minore, uguale o maggiore di zero
- Il programma deve poter calcolare anche le equazioni di primo grado
- Il programma deve mostrare un messaggio nel caso in cui le radici non possono essere calcolate in R