FONDI STRUTTURALI E DI INVESTIMENTO EUROPEI 2014/2020
fondo sociale europeo
ITS TECNOLOGIE DELL'INFORMAZIONE E DELLA COMUNICAZIONE

FONDAMENTI DI PROGRAMMAZIONE

C / C++

Docente: Armando Valentino

in collaborazione con:
ISTITUTI TECNICI SUPERIORI
Ministero dell'Istruzione
UNIONE EUROPEA
REGIONE PIEMONTE
per una crescita intelligente, sostenibile ed inclusiva
www.regione.piemonte.it/europa2020
INIZIATIVA CO-FINANZIATA CON FSE

LINGUAGGIO C++

INPUT/OUTPUT, VARIABILI, OPERATORI

Prof. Armando Valentino

POR Piemonte
FSE 2014-2020

Struttura di un programma

```
#include <iostream>
using namespace std;
int main()
{
        //dichiarazione delle variabili
        …
        //istruzioni programma
        …
        return 0;
}
```

// dichiarazione di uso della libreria iostream

// dichiarazione dello spazio dei nomi usato, in questo caso **std**

clausola per evitare di ripetere il namespace cioè il contenitore detto "***spazio dei nomi" std*** che contiene *identificatori standard del linguaggio C++* ad esempio invece di std :: cout si scriverà solo **cout**

Corpo del programma

Per inserire dei commenti (non vengono compilati) si usa // o /* …*/

Variabili e tipi di dati

C++ è un linguaggio tipizzato. Tutte le variabili devono essere dichiarate prima di usarle.
C++ è è case-sensitive, quindi fa differenza tra maiuscole eminuscole

```
tipo nomevariabile;
```

Più variabili dello stesso tipo si possono scrivere su un unico rigo.

```
tipo nome1, nome2, .. nomeN;
```

Tipi di dati numerici Interi

TIPO	RANGE	BYTE	VALORE MAX
short	-32.768 .. +32768	2	SHRT_MAX
Unsigned short	0 .. 65635	2	USHRT_MAX
Int	come short o long (dipende dal compilatore)	2 o 4	INT_MAX
unsigned int	come unsigned short o unsigned long	2 o 4	UINT_MAX
long	- 2147.483.648 .. + 2147.483.648	4	LONG_MAX
unsigned long	0 .. 4294.967.295	4	ULONG_MAX

Variabili e tipi di dati

Tipi di dati numerici reali

TIPO	RANGE	BYTE
float	-3.4 E38 .. – 1.5 E-45 valori negativi 1.5 E-45 .. 3.4E38 valori positivi	4
double	-1.7E308 .. -5.0E-324 valori negativi 5.0E-324 .. 1.7E308 valori positivi	8
Long double	-1.1°4932 .. -3.4E-4932 valori negativi 3.4E-4932 .. 1.1°4932 valori positivi	12

Float, double e long double rappresentati con notazione IEEE 754

Altri Tipi di dati

TIPO	RANGE	BYTE
bool	True o False	1
char	Insieme dei caratteri ascii	1
string	Sequenza di caratteri del codice ascii	

Il tipo string non è un tipo di dato primitivo, ma una classe

Tipi di dati

Char

```
char c = 'a';  // Carattere letterale
char k = 32;  // Conversione da codice ascii
int c = int('$');  // Conversione in codice ascii
if( isupper(c) ) ..          // Esempio di funzioni
bool x = isdigit(k);         // k e' una cifra (0-9)
```

Virgola mobile

```
float  x = 1;              // Singola precisione
double y = 3.14;           // Doppia precisione
double z = 1.2e10;         // Notazione esponensiale
                  // Espressioni e funzioni
double t = 3*7-sqrt(k)+sin(9);
double z = 3^y;            // Esponenziale
int round = int(z+0.5);    // Arrotondamento ad intero
int trunc = int(z);        // Troncamento ad intero
```

Interi

```
int a = 3;                 // Intero standard
short int s;               // Intero
short s;                   // Scrittura abbreviata
long int l;                // Intero 4 byte
long l;                    // Scrittura abbreviata
unsigned int u;            // Intero senza segno
unsigned u;                // Scrittura abbreviata
unsigned long lu;          // Intero lungo positivo

int x = 3*5+7-4/2;         // Espressioni
int k = 15%7;              // =1, operatore modulo = resto
      divisione
int k = 0xFE;              // Costanti esadecimali
```

Conversioni di tipo: CAST

I dati possono essere convertiti da un tipo ad un altro. La conversione si dice anche CAST
La **conversione** può essere **implicita** o **esplicita.**
La conversione implicita avviene in modo automatico. Quando le conversioni implicite non sono ammissibili**,** in fase di compilazione vengono segnalati gli errori, e in questo caso si deve usare la conversione esplicita.
Le conversioni esplicite richiedono l'uso di funzioni c++ che fanno il cast.
Il cast può avvenire con perdita di precisione.
L'operazione di conversione può generare errori di oveflow o underflow

Es1:
```
double a=5.2;
int n = (int) a;
int n = int(a);
```

Es2:
```
Short int i;
i=40000;
cout <<i;
```

errore

Risultato è -13108
Valore max di short int è 32768

Alle variabili booleane si può dare il valore 0 o 1 oppure true o false oppure espressioni il cui risultato è vero o falso.

COSTANTI

I dati costanti posso essere degli stessi tipi già visti in precedenza.
I dati costati si dichiarano con la parola chiave **const**

```
Const tipo nomecostane=valore;
```

Es:

```
const double a=5.2;
const int n = 10+2;
const char c=‘a’;
const string s = “Ciao mondo”;
```

Le **costanti** possono essere dichiarate anche con la direttiva **#define**

Es:

```
#define a 5.2;
#define n 10+2;
#define c ‘a’;
#define s “Ciao mondo”;
```

STRUTTURE DI CONTROLLO IN C++

STRUTTURA DI SELEZIONE

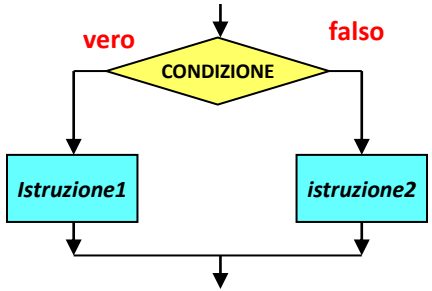
SELEZIONE BINARIA

È una struttura che valuta una condizione e a secondo del suo valore (vero o falso) viene svolta una istruzione o un'altra diversa.

```
SE condizione ALLORA
        Istruzione1
ALTRIMENTI
        Istruzione 2
FINESE
```

traduzione

```
if (<condizione>)
{
        istruzione1 ;
}
else
{
        istruzione2;
}
```



La selezione può diventare UNARIA se il ramo ALTRIMENTI non presenta istruzioni

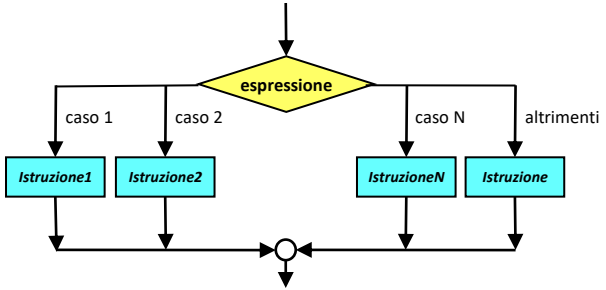
STRUTTURE DI CONTROLLO IN C++

SELEZIONE MULTIPLA

```
NEL CASO CHE espressione SIA
        caso 1:
                istruzione1
        caso 2:
                istruzione 2
        ……..
        caso N:
                istruzione N
        ALTRIMENTI
                Istruzione
FINECASO
```

traduzione

```
switch espressione {
        case caso 1:
                istruzione1;
                break;
        case caso 2:
                istruzione 2
                break;
                …….
        case caso N:
                istruzione N;
                break;
        default:
                Istruzione;
}
```



Funzionamento:
Viene valutata l'espressione. Se il valore vale caso1 viene eseguita l'istruzione, o se il valore vale ccaso2 vie eseguita l'istruzione2, o se il valore casoN viene eseguita l'istruzione N. Nse non rientra in nuìesusn caso viene eseguita l'istruzione in Altrimenti. Solo un caso viene eseguito tra tutti quelli presenti

STRUTTURE DI CONTROLLO IN C++

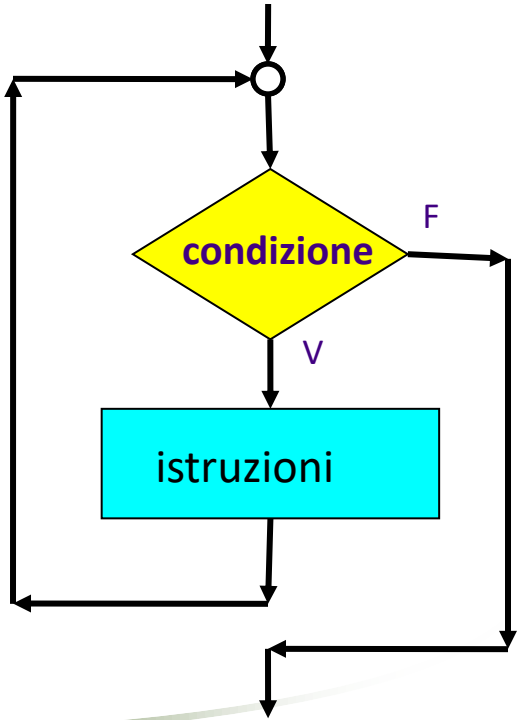
STRUTTURA DI ITERAZIONE

RIPETIZIONE CON CONTROLLO IN TESTA

```
MENTRE condizione
        Istruzioni
FINEMENTRE
```

traduzione

```
While (condizione){
        Istruzioni;
}
```



Funzionamento:
Viene valutata la condizione.
Se la condizione è vera vengono eseguite le istruzioni.
Poi viene di nuovo valutata la condizione, se è ancora vera si rieseguono le istruzioni. Le istruzioni si rieseguono sempre fino a quando la condizione è vera.
Il ciclo termina quando la condizione diventa falsa

STRUTTURE DI CONTROLLO IN C++

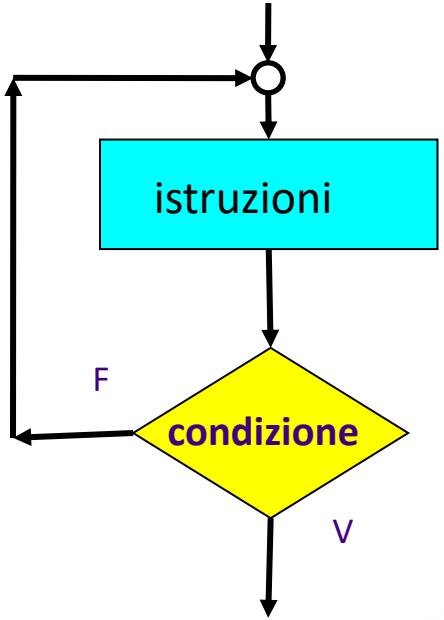
STRUTTURA DI ITERAZIONE

RIPETIZIONE CON CONTROLLO IN CODA

```
RIPETI
        Istruzioni
FICHE condizione
```

traduzione

```
do{
        Istruzioni;
} While (condizione);
```



Funzionamento:
Vengono svolte per prima le istruzioni.
Poi si valuta la condizione. Se la condizione è falsa vengono rieseguite le istruzioni.
Poi viene di nuovo valutata la condizione, se è ancora falsa si rieseguono le istruzioni. Le istruzioni si rieseguono sempre fino a quando la condizione è falsa.
Il ciclo termina quando la condizione diventa vera.

STRUTTURE DI CONTROLLO

STRUTTURA DI ITERAZIONE

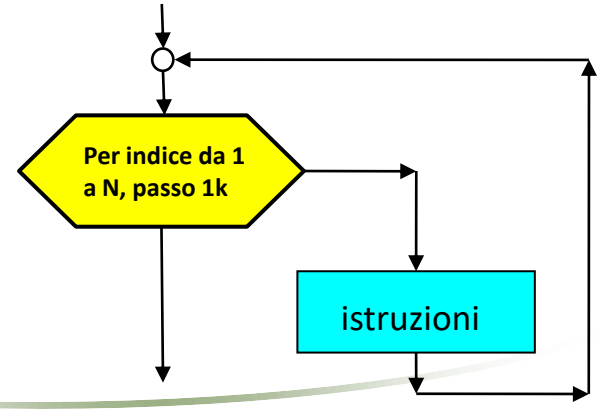
ITERAZIONE ENUMERATIVA

```
PER indice Da 1 a N, passo k
        Istruzioni
FINEPER
```

traduzione

```
for(indice=1; indice<=N;indice+=k){
        Istruzioni
}
```

Funzionamento:
l'indice ve posto uguale al primo valore, poi vengono svolte le istruzione . Poi l'indice viene incrementato di k volte, se il valore non supera quello massimo N, vengono rieseguite le istruzione e questo viene ripetuto fichè l'indice non supera il valore massimo.



Input e output

Possono essere usate con la dichiarazione della libreria <iostream>

output Funzione **cout**

cout << valore1 << valore2<< " " << "costanteN";

Visualizza sul video le variabili valore1,valore2, un spazio e valoreN

<< è un operatore che **trasferisce il valore al dispositivo**. L'output standard è il video.
I valori vengono visualizzati accodati senza accapo. Per mandare a capo la stampa si deve inserire '\n' o **endl**

Es: ***cout*** *<<valore1<<endl;* *oppure* ***cout*** *<< valore1 << "\n";*

input Funzione **cin**

cin >> var1 >> var2>> varN;

Prende dalla tastiera tre variabili var1,var2, varN

>> è un operatore che **trasferisce il dati dalla tastiera alle variabili**
I valori devono essere inseriti uno dopo l'altro **separati da uno spazio o da un invio**

Lo spazio non può essere acquisito con **cin**, perché è usato come separatore, in questi casi si deve usare la funzione **getchar()**

Per prendere una sequenza di caratteri da tastiera si deve usare **getline()**
Es: *getline(cin, variabile);*

fflush(stdin); per svuotare il buffer di input

Formattazione input e output

```
#include <iostream>
using namespace std;
int main(int argc, char** argv) {
        int a=125300, aa=20;
        float b=12300.2691, d= 16.21;
        double c= 1234567890000.55;
        cout <<"int a=" << a << endl;
        cout <<"int aa=" << aa << endl;
        cout <<"float b=" << b << endl;
        cout <<"double c=" << c << endl;
        cout <<"float d=" << d << endl;
        getchar();
        return 0;
}
```

Codice C++

H:\2018-2019\3Binfo\LEZIONI\i-o-non formattato.exe

```
int a=125300
int aa=20
float b=12300.3
double c=1.23457e+012
float d=16.21
```

```
#include <iostream>
#include <iomanip>
using namespace std;
int main(int argc, char** argv) {
    int a=125300, aa=20;
    float b=12300.2691, d= 16.21;
    double c= 1234567890000.55;
    cout << "a=" << a << endl;
    cout <<"a=" << setw(7) << a << endl;
    cout <<"b=" << b << endl;
    cout <<"b=" << setw(20) << setfill('x') << b << endl;
    cout <<"c=" << c << endl;
    cout <<"c=" << setw(15) << c << endl;
    cout << "c=" << setprecision(15) << setfill(' ') << c << endl;
    cout << "aa=" << setiosflags(ios::showbase | ios::showpos) << hex << aa << endl;
    cout << "aa=" << oct << aa << resetiosflags(ios::showbase ) << endl;
    cout << "aa=" << oct << aa << endl;
    cout << "d=" << setprecision(7) << d << endl;
    cout << "d=" << setiosflags(ios::showpoint) << d << endl;
    cout << "d=" << setprecision(5) << d << resetiosflags(ios::showpoint) << endl;
    cout << "d=" << setprecision(8) << setiosflags(ios::fixed) << d << endl;
    cout << "700/3.0=" << 700/3.0 << endl;
    cout << "700/3.0="  << resetiosflags(ios::fixed) << 700/3.0 << endl;
    getchar();
    return 0;
}
```

H:\2018-2019\3Binfo\LEZIONI\opera

```
a=125300
a=  125300
b=12300.3
b=xxxxxxxxxxxxx12300.3
c=1.23457e+012
c=xxx1.23457e+012
c=1234567890000.55
aa=0x14
aa=024
aa=24
d=+16.21
d=+16.21000
d=+16.210
d=+16.20999908
700/3.0=+233.33333333
700/3.0=+233.33333
```

Input e output formattato: iomanip

- La libreria <iomanip> ha le specifiche per formattare l'input e l'output di cin e cout.

MANIPOLATORE	DISPOSITIVO	SIGNIFICATO
oct	cout, cin	Imposta sistema ottale
dec	cout, cin	Imposta sistema decimale
hex	cout, cin	Imposta sistema esadecimale
ends	cout	Identifica la sequenza di escape \0 fine della stringa
setw(n)	cout	Lunghezza di stampa pari a n posizioni
setprecision(nd)	cout	Imposta il numero di cifre (compreso i decimali) pari a nd con cui sarà stampato il numero
setfill(car)	cout	Imposta il carattere car come carattere di riempimento; lo spazio è il default

Input e output formattato: iomanip

- Una volta impostato il **manipolatore rimane attivo per tutto il codice** (tranne setw)
- Esistono anche due funzioni per formattare o rimuovere formattazioni impostate **setiosflags** e **resetiosflags**, utilizzabili con un parametro
- ios è la classe che gestisce l'input /output e il parametro si scrive -> **ios::nomecostante**

COSTANTE IOS	SIGNIFICATO
ios::fixed	Imposta i numeri reali con un numero di decimali definito da setprecision
ios::showbase	Stampa prima i carateri 0 e 0x per i numeri ottali
ios::scientific	Stampa in formato scientifico
ios::showpos	Stampa i numeri positivi preceduti dal carattere +
ios::showpoint	Stampa i numeri reali con il . Decimalee cifre decimali(0)
ios::uppercase	Stampa in maiuscolole lettere esadecimali e i caratteri
ios::right	Allinea a destra e riempie lo spazio con ilcarattere impostato con setfill
ios::left	Allinea a sinistra e riempie lo spazio con setfill

Input e output formattato:stdio.h

Il formato di output e input può essere controllato con le funzioni **printf()** e **scanf()** contenute nella libreria <stdio.h> già inclusa in <iostream>.
La sintassi è la seguente:
printf(stringacontrollo, variabile)

E' possibile inserire un **numero** tra il simbolo % e la lettera specificata per indicare il **numero di posizioni di stampa** (se le posizioni di stampa non sono sufficienti comunque viene stampato completo).
Es: %15.3f riserva 15 posizioni di cui 3 per i decimali

Scanf(stringacontrollo, &variabili)

Si possono leggere più variabili contemporaneamente che devono essere immessi separati da uno spazio o da un invio

TRINGA DI CONTROLLO	COSA VIENE STAMPATO
%d, %i	Intero decimale
%f	Valore in virgola mobile
%c	Un carattere
%s	Una stringa di caratteri
%o	Numero ottale
%x, %X	Numero esadecimale
%u	Intero senza segno
%f	Numero reale (float o double)
%e, %E	Formato scientifico
%%	Stampa il carattere %

Operatori e priorità

- Gli operatori sono valutati in base alla priorità
- Per modificare la priorità si possono inserire parentesi

- Operatori matematici: +, - , * , /
- Uguaglianza: ==, !=
- Confronto: <, >, <=, >=
- Logici: &&, ||, & (and bit a bit), | (or bit a bit) , ^ (OR esclusivo bit a bit)
- Assegnazione: +=, -=, *=, /=, %=
- Operatori unari: ++, -- (in modo prefisso e postfisso) (incremento e decremento)
- Operatore concatenazione di stringhe: +
- Visibilità e riferimento globale: ::

Operatori e priorità

LIVELLO DI PRECEDENZA	OPERATORE	NOME
1	!	Not, negazione
2	*	Moltiplicazione
2	/	Divisione
2	%	Modulo
3	+	Addizione
3	-	Sottrazione
4	<	Minore
4	<=	Minore uguale
4	>	Maggiore
4	>=	Maggiore uguale
5	==	Uguale (confronto)
5	!=	Diverso
6	&&	AND
7	||	OR
8	=	Assegnamento

Esempi Operatori

- X=10;
- X+=2;

X vale 12

- X=1;
- y=x++;

y vale 1 e x vale 2

- X=1;
- y=++x;

y vale 2 e x vale 2

x++ equivale a x=x+1

x-- equivale a x=x-1

x+=2 equivale a x=x+2

- string s = “stringa”;
- char ch=’c’;
- String s1=s+” “+ch;

s1 vale “stringa c”

cout << x++;
Equivale a
cout<<x;
x=x+1;

cout << ++x;
Equivale a
x=x+1;
cout<<x;

Licenza