Laboratorio di Informatica

1. Introduzione al C++

Corso di Laurea in Ingegneria Elettronica e Telecomunicazioni
A.A. 2013-2014
2° Semestre
Prof. Giovanni Pascoschi

Storia del C e del C++

- Linguaggio C++
  - Inizio anni '80: Bjarne Stroustrup (Bell Laboratories)
- Estensione del C
  - Fornisce funzionalità per la programmazione orientata agli oggetti
- Oggetti: componenti software riusabili
  - Modellano entità del mondo reale
- Programmi object-oriented
  - Migliore stile di programmazione
  - Facilità nel modificare/correggere programmi
- Linguaggio-ibrido
  - Si può usare uno stile C-like
  - Si può usare uno stile Object-oriented
  - Si possono usare entrambi gli stili

Un primo esempio in C++

```
// Programma di Esempio
#include <iostream>
using namespace std;
int main( ) {
cout << "Hello,World!\n";
return 0;
}
```

OUTPUT

```
Hello, World!
```


Un primo esempio in C++

```
// Programma di Esempio
#include <iostream>
using namespace std;
int main( ) {
cout << "Hello,World!\n";
return 0;
}
```

Commento

File header libreria funzioni I/O

Contenitore dei nomi degli identificatori standard C++

Un primo esempio in C++

```
// Programma di Esempio
#include <iostream>
using namespace std;
int main( ) {
cout << "Hello,World!\n";
return 0;
}
```

Inizio del programma principale

Operazione di output

Ritorna '0' al sistema operativo

Fine del programma principale

Commenti in C++

Singola riga

```
// Programma di Esempio
```

Multi riga

```
/* Programma per il calcolo dell'integrale
 definito della funzione cos(x) */
```


Un secondo esempio in C++

```
#include <iostream>
using namespace std;
int main()
{
int a = 8;
int b = 4;
int c = 3;
double tot = a * 0.01 + b * 0.10 + c * 0.25; /* valore totale */
cout << "Valore Totale = " << tot << "\n";
return 0;
}
```


Un secondo esempio in C++

OUTPUT

```
Valore Totale = 1.230000
```

Variabili in C++

- Identificatore: sequenza di lettere, numeri e simbolo di sottolineatura "_"
- Deve iniziare con una lettera (p.e. non va bene "2perim")
- Non puo' essere una parola o simbolo riservato del linguaggio (p.e. non va bene "for")
- Non puo' contenere spazi
- Es. Nomi validi : pippo, pluto, ciao5, salve_a_tutti
- E' bene scegliere nomi che descrivano adeguatamente il contenuto della variabile
- "Case sensitive": maiuscole e minuscole vengono considerate diverse (p.e. Pippo e pippo sono due variabili diverse)

Tipi di variabili numeriche in C++

Tipo predefinito	Denominazioni alternative
char signed char unsigned char signed short int signed int signed long int unsigned short int unsigned int unsigned long int float double long double	signed short, short signed, int long int, signed long, long unsigned short unsigned unsigned long

Tipi base

char	caratteri
int	interi
float	reali
double	reali in precisione doppia

Qualificatori

short	condizionano lo spazio allocato per le variabili
long	
signed	condizionano i valori assumibili dalle variabili
unsigned	

Variabili - Differenze rispetto al C

Variabili booleane

bool	Variabile booleana (p.e. 'true'/'false' oppure '1'/'0')

Variabili stringa

Stringhe → nel linguaggio C : char nome[10];

Stringhe → nel linguaggio C++ : string nome;

Tipi di variabili : stringhe in C++

```
// Esempio in C++

#include <string>
using namespace std;

string nome;
string siglaprov = "TO";
```

Dichiarazione variabili in C++ (differenza rispetto al C)

Esempio:

```
#include <iostream>
using namespace std;
int main()
{
int a;
a = 8;
.......
int b;
b = 4;
.......
return 0;
}
```


Costanti in C++

- **Una costante è una espressione che ha un valore prefissato** : Numeri interi, Numeri in virgola mobile, Caratteri, Stringhe

1. Modalità : istruzione in C++

- Esempi:

```
const int  A = 24;
const float PIGRECO = 3.14;
const float NUM = 1.6e-19;
const int A = 0x13; → in esadecimale
const int B = 075; → in ottale
```

2. Modalità : direttiva/comando pre-processore

- Esempi:

```
# define A 24
# define PIGRECO 3.14
```


Costanti in C++

- Costanti di tipo stringhe:
  Caratteri ‘x’, ‘T’,’1’
  Stringhe “Salve gente”, “Hello, World”

- I letterali che rappresentano caratteri sono racchiusi da carattere apice (‘)

- Sequenze di letterali che rappresentato stringhe sono racchiusi da tra due caratteri doppio apice (“)

Operatori built-in in C++

Operatori built-in

=	Assegnamento	
+	Somma	Operatori artitmetici
-	Sottrazione	
*	Moltiplicazione	
/	Divisione	
%	Modulo (solo per tipo “int”)	
==	Relazione di uguaglianza	Operatori relazionali
!=	Relazione di diversità	
<	Relazione “minore di”	
>	Relazione “maggiore di”	
<=	Relazione “minore o uguale a”	
>=	Relazione “maggiore o uguale a”	

Operatori built-in in C++

Altri operatori di assegnamento

Operatore	Esempio	Istruzione equivalente
++	conta++;	conta = conta + 1;
--	conta--;	conta = conta - 1;
+=	x+=y;	x = x + y;
-=	x-=y;	x = x - y;
*=	x*=y;	x = x * y;
/=	x/=y;	x = x / y;
%=	x%=y;	x = x % y;

Operatori built-in in C++

Operatori logici built-in

&&	AND logico su due variabili	Operatori logici
||	OR logico su due variabili	
!	NOT logico su una variabile	
&	AND logico bit a bit	Operatori logici bit a bit
|	OR logico bit a bit	
^	XOR bit a bit	
~	complemento a uno dell'operatore	
>>	shift a destra di un certo numero di bit	
<<	shift a sinistra di un certo numero di bit	

Istruzioni ingresso/uscita in C++

```
using namespace std;
cin >> raggio;          // operazione di input dallo "stream" cin
cout << raggio;         // operazione di output allo "stream" cout
```

Alternativa all'uso del comando "using namespace std";

```
std::cin >> raggio;     // operazione di input dallo "stream" cin
std::cout << raggio;    // operazione di output allo "stream" cout
```

:: operatore risolutore di visibilità (scope resolution operator)

Istruzioni ingresso/uscita in C++

Esempi di input

```
cin >> raggio;
cin >> a >> b;          // non si usa la virgola
```

Istruzioni ingresso/uscita in C++

Esempi di output

```
cout << "La circonferenza è uguale a" << circonf;
cout << "Lati del rettangolo :" << a <<' '<< b; // non si usa la virgola

cout << "Area del cerchio :" << area << endl;
cout << "Area del cerchio :" << area << '\n';
```

endl → ritorno a capo inviando '\n' al buffer + svuota il buffer in uscita

'\n' → ritorno a capo inviato al buffer senza svuotare il buffer di uscita

Per forzare lo svuotamento del buffer:

```
cout << flush;
```


Istruzioni ingresso/uscita in C++

Sequenza di escape	Descrizione	Terminologia inglese
\n	A capo riga	New line
\t	Tabulazione	Tab
\r	Ritorno a capo della stessa riga	Carriage return
\"	Doppi apici	Double quote
\\	Barra contraria	Backslash
\?	Punto di domanda	Question mark
\'	Apice singolo	Single quote
\a	Segnalazione acustica	Bell
\f	Salto pagina	Form feed

Manipolatori I/O in C++

```
#include <iomanip>
```

Manipolatori	Descrizione
endl	Carattere di fine riga (simile a '\n')
dec	Imposta la base decimale
oct	Imposta la base ottale
hex	Imposta la base esadecimale
setw(b)	Imposta il numero di battute (default b=0, senza formattazione)
setfill(c)	Imposta il carattere di riempimento (default c=spazio)
setprecision(d)	Imposta il numero di decimali (default d=6)

setw rimane valido solo per l'output successivo, mentre gli altri operatori permangono fino a nuova impostazione

Manipolatori I/O in C++

Esempio 1:

```
#include <iostream>
using namespace std;
int main()
{
int i;
cout << "Introdurre un numero decimale: ";
cin >> dec >> i;
cout << "dec/oct/hex:\t";
cout << dec << i << '\t'
      << oct << i << '\t'
      << hex << i << endl;
return 0;
}
```

OUTPUT

```
Introdurre un numero decimale: 10
dec/oct/hex:	10	12	a
```

Manipolatori I/O in C++

Esempio 2:

```
#include <iostream>
using namespace std;
int main()
{
cout << setfill('*')
      << setw(4)
      << 25
      << '\t'
      << 25
      << endl;
return 0;
}
```

OUTPUT

```
**25      25
```


Manipolatori I/O in C++

Esempio 3:

```
#include <iostream>
using namespace std;
int main()
{
cout << 1./3.
      << '\t'
      << setprecision(3)
      << 1./3.
      << endl;
return 0;
}
```

OUTPUT

```
0.333333        0.333
```


Funzioni matematiche in C++

- Operazioni più complesse, e.g., la radice quadrata, il logaritmo vengano realizzare tramite funzioni pre-definite C++

  Le funzioni C++ saranno studiate in dettaglio in seguito

Esempio:

```
double x;
cin >> x;
cout << sqrt(x);
```

- Sintassi → nome_funzione(espr1,…,esprn)

Calcola il valore della funzione nome_funzione con argomenti/parametri espr1,…,esprn

Si dice che la funzione restituisce (return) il valore calcolato

Funzioni matematiche in C++

- Per invocare le funzioni matematiche bisogna includere il file di intestazione <cmath>

```
#include <iostream>
#include <cmath>
using namespace std;
int main()
{
int x;
cout << "Inserire un numero \n";
cin >> x;
double radice = sqrt(x);
cout << "Radice quadrata= " << radice << "\n";
return 0;
}
```

Funzioni matematiche principali in C++

Funzione	Descrizione
sqrt(x)	radice quadrata di x
pow(x,y)	x^y
sin(x)	sin x
cos(x)	cos x
tan(x)	tan x
exp(x)	e^x
log(x)	logaritmo naturale $\log_e(x)$
log10(x)	logaritmo naturale $\log_{10}(x)$
ceil(x)	il piu' piccolo numero intero >=x
floor(x)	il piu' grande numero intero di <=x
fabs(x)	valore assoluto di x

Alternativa in C++ (a una via)

```
#include <iostream>
using namespace std;
int main()
{
int voto;
cout << “Inserire il voto dell’esame \n”;
cin >> voto;
if (voto>=18)
cout << “Promosso \n”;
return 0;
}
```


Alternativa in C++ (a due vie)

```
#include <iostream>
using namespace std;
int main()
{
int voto;
cout << “Inserire il voto dell’esame \n”;
cin >> voto;
if (voto>=18)
cout << “Promosso \n”;
else {
cout << “Bocciato \n”;
cout << “Mi spiace \n”;
}
return 0;
}
```


Istruzione while()

- L’istruzione while() permette di implementare un ciclo di ripetizione pre-condizionale

Sintassi:

```
while(condizione) {
istruzioni
…….
}
```

- L’istruzione viene eseguita finche’ la condizione rimane vera
- Puo’ essere eseguita 0 o piu’ volte

Istruzione while() : esempi

- Una classe di 100 studenti ha sostenuto un test. Inserire i voti (compresi tra 0 e 30) e calcolare la media.
- Una classe di studenti (N non noto) ha sostenuto un test. Inserire i voti (compresitra 0 e 30) e calcolare la media.

Istruzione while() : esempio 1

```
#include <iostream>
using namespace std;
int main()
{
int totale=0, contatore=0, voto;
double media;
while(contatore<100) {
    cout << "Inserire il voto dell'esame \n";
    cin >> voto;
    totale=totale+voto;
    contatore=contatore+1;
}
media=totale/100.0;
cout << "La media è " << media;
return 0;
}
```


Istruzione while() : esempio 2

```
#include <iostream>
using namespace std;
int main()
{
int totale=0, contatore=0, voto;
double media;
cout << "Inserire il voto dell'esame o -1\n";
cin >> voto;
while(voto !=-1) {
    totale=totale+voto;
    contatore=contatore+1;
    cout << "Inserire il voto dell'esame o -1\n";
    cin >> voto;
}
if (contatore>0) {
    media=(double) totale/contatore;
    cout << "La media è " << media << "\n";
}
else
    cout << "Non è stato inserito alcun valore \n";
return 0;
}
```

N studenti non noto

valore di guardia ("tappo")

Istruzione do…while()

- L'istruzione do…while() permette di implementare un ciclo di ripetizione post-condizionale

Sintassi:

```
do{
istruzioni
…….
} while(condizione);
```

- L'istruzione viene eseguita finche' la condizione rimane vera
- Puo' essere eseguita 1 o piu' volte

Istruzione do…while() – esempio

```
#include <iostream>
using namespace std;
int main() {
int counter = 1;
do {
    cout << counter << " ";
    counter++;
 } while (counter <= 10 );
cout << endl;
return 0;
 }
```

1 2 3 4 5 6 7 8 9 10

Istruzione for

- L'istruzione **for** permette di implementare un ciclo di ripetizione enumerativo (con contatore)

Sintassi:

```
for(iniz; condizione; modifica){
istruzioni
…….
}
```

- L'istruzione viene eseguita finche' la condizione rimane vera

Istruzione for – esempio Calcolo Massimo

```
#include <iostream>
using namespace std;
int main( ) {
double next;
double highest;
for(int i=0; i<100; i++) {
   cout << "Please enter the temperature values:\n";
   cin >> next;
   if(i == 0) highest = next;
   if(highest < next) highest = next;
}
cout << "The highest temperature is " <<highest << "\n";
return 0;
}
```


Istruzione for – esempio 2

```
#include <iostream>
using namespace std;
int main()
{
int n, fattoriale=1;
cout << "Inserire Numero: ";
cin >> n;
for (int i =2; i<=n; i++)
   {
   fattoriale*=i;
   }
cout << n << "!= " << fattoriale;
return 0;
}
```

0!=1

Istruzione for – differenze C++

```
for( ; a<b;  )
   a +=2;
```

equivale a:

```
while (a < b)
   a +=2;
```

```
for( ; ;) {
}
```

equivale a:

ciclo infinito

Istruzioni break & continue

- L'istruzione **break** usata in un ciclo di ripetizione permette di uscire immediatamente dal ciclo di ripetizione stesso
- L'istruzione **continue** in un ciclo di ripetizione permette di saltare le successive istruzioni presenti nel ciclo, permettendo di andare alla successiva iterazione

Istruzione break (esempio)

```
#include <iostream>
using namespace std;
int main()
{
int count;
for ( count = 1; count <= 10; count++ )  {
   if ( count == 5 ) break;
   cout << count << " ";
}
cout << "\nBroke out of loop at count = " << count << endl
return 0;
}
```

```
1 2 3 4
Broke out of loop at count = 5
```


Istruzione continue (esempio)

```
#include <iostream>
using namespace std;
int main()
{
for ( int count = 1; count <= 10; count++ ) {
   if ( count == 5 ) continue;
   cout << count << " ";
}
cout << "\nUsed continue to skip printing 5" << endl;
return 0;
}
```

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

Istruzione switch

- L'istruzione switch permette di implementare una struttura a scelta multipla

Sintassi:

```
switch(variabile){
case valore1:
        istruzioni1;
        break;
case valore2:
        istruzioni2;
        break;
…….
default:
        istruzioni;
        break;
}
```


Funzioni : Passaggio di parametri in C++

- Il passaggio dei parametri tra funzione chiamante e funzione chiamata in C++ avviene secondo due modalità:
  - per valore o copia (by value)
    si passa il valore del parametro attuale
  - per referenza (riferimento) (by reference)
    si passa un riferimento (indirizzo) al parametro attuale

Esempio 1 (con prototipi)

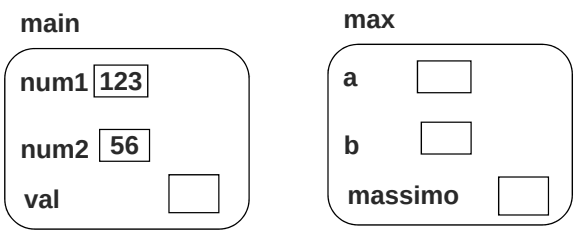
```
#include <iostream>
using namespace std;

int max(int , int );

int main ()
{
int num1, num2, val;
cout << "Inserire I due numeri \n";
cin >> num1 >> num2;
val = max(num1, num2);
cout << "Il valore piu' grande è " << val;
return 0;
}
```

```
int max(int a, int b) {
    int massimo;
    if( a > b ) {
        massimo = a;
    }
    else {
    massimo = b;
    }
    return massimo;
}
```


Passaggio di parametri per valore

- la funzione chiamante passa alla funzione chiamata i valori dei parametri attuali
- i valori sono usati per inizializzare i parametri formali
- eventuali modifiche ai parametri formali a e b all'interno della funzione non hanno effetto sui parametri attuali num1 e num2

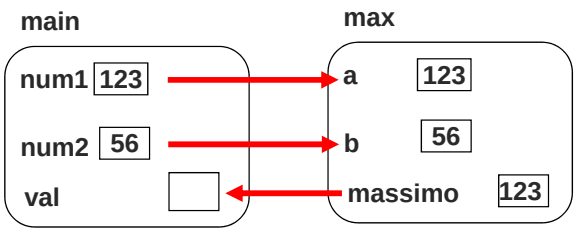
max (a,b)



Passaggio di parametri per valore

- la funzione chiamante passa alla funzione chiamata i valori dei parametri attuali
- al termine della funzione i parametri formali e tutte le variabili locali a e b cessano di esistere (vengono distrutte)
- l'unico effetto per il chiamante è dovuto al valore restituito massimo

max (a,b)



Algoritmo di swap

- Scrivere una procedura che scambia il contenuto/valore di due variabili a e b

```
#include <iostream>
using namespace std;

void swap (int, int);

int main ()
{
int a, b;
.........
swap(a, b);
.........
return 0;
}
```

```
void swap(int x, int y) {
    int temp;
    temp=x;
    x=y;
    y=temp;
}
```

- Vengono scambiati x e y?
- Vengono scambiati a e b?

Passaggio dei parametri per referenza in C++

- La funzione chiamante passa alla funzione chiamata dei riferimenti (indirizzi) ai parametri attuali
- sintassi

parametro per referenza = tipo& nome_variabile

- Al momento della chiamata il nome del parametro formale per referenza diventa un sinonimo (alias) del parametro attuale facente riferimento alla stessa locazione→ non si crea una nuova variabile ma un riferimento ad una variabile esistente
- Il parametro attuale deve essere una variabile (e non anche un'espressione come per il passaggio per valore)
- Una modifica al parametro formale all'interno della funzione rappresenta, in realtà una modifica al parametro attuale

Algoritmo di swap in C++

- Scrivere una procedura che scambia il contenuto/valore di due variabili a e b (versione corretta)

```
#include <iostream>
using namespace std;

void swap (int&, int&);

int main ()
{
int a, b;
.........
swap(a, b);
.........
return 0;
}
```

```
void swap(int& x, int& y) {
    int temp;
    temp=x;
    x=y;
    y=temp;
}
```

- Vengono scambiati x e y?
- Vengono scambiati a e b?

Namespace

- il C++ consente di dividere lo spazio globale di definizione delle variabili (e delle funzioni) in diversi contenitori detti namespace (spazio dei nomi) a ciascuno dei quali è associato un nome

- il vantaggio deriva dalla possibilità di sfruttare risorse globali (visibili a tutte le funzioni) senza entrare in conflitto con altre librerie che impiegano lo stesso nome di variabile

- il namespace è quindi un contenitore di dati e di funzioni che hanno un nesso logico tra loro

esempio:

```
namespace spazioglobale1 {
        //esempi di variabili e funzioni globali
        int totale;
        int Calcolo(int a, int b);
}
```


Namespace (utilizzo)

- per utilizzare le variabili di un namespace o una sua funzione si usa l'operatore :: Scope Resolution Operator (operatore risoluzione di visibilità)

esempio di utilizzo:

```
cout << spazioglobale1::totale<< endl;
```

oppure usando:

```
using namespace spazioglobale1;
cout << totale <<endl;
```


Namespace (esempio)

- Definire un namespace per gestire un rettangolo e le operazioni di calcolo su di esso

```
namespace Rettangolo {
    // dati
    int base, altezza;
    // funzioni
    int Area(int, int);
    int Perimetro(int, int);
}

int Rettangolo::Area(int b, int h) {
    return b * h;
}
int Rettangolo::Perimetro(int b, int h) {
    return (b + h) * 2;
}
```


Namespace (esempio)

- Questo file contenente il namespace Rettangolo dovrà essere salvato nella sottodirectory Include della directory di installazione del compilatore C++

```
#include < iostream >
using namespace std;
#include <rettangolo>

int main( ) {
    cout << "Base: ";
    cin >> Rettangolo::base;
    cout<< "Altezza:";
    cin >> Rettangolo::altezza;
    cout << Rettangolo::Area(Rettangolo::base, Rettangolo::altezza)<<endl;
    cout << Rettangolo::Perimetro(Rettangolo::base, Rettangolo::altezza)<<endl;
    return 0;
}
```

Namespace (esempio)

- oppure in maniera piu' comoda

```
#include < iostream >
#include <rettangolo>
using namespace std;
using namespace rettangolo;


int main( ) {
    cout << “Base: ";
    cin >> base;
    cout<< “Altezza:”;
    cin >> altezza;
    cout << Area(base, altezza) << endl;
    cout << Perimetro(base, altezza)<<endl;
    return 0;
}
```


Namespace (“hiding”)

- abbiamo ottenuto di nuovo un information hiding che permette di interagire a livello di interfaccia utilizzando le funzioni del namespace
- se dobbiamo cambiare le librerie contenute nei namespace “utente”, i programmi esterni che li utilizzano rimangono invariati

Librerie di inclusione C++

- inizialmente le librerie standard del C++ contenevano l’estensione .h→ iostream.h (header file come nel C)
- con il passare del tempo e con l’utilizzo dei namespace std (namespace delle principali librerie del linguaggio C++) è stato possibile eliminare l’estensione .h

```
#include <iostream.h>
```

oppure

```
#include <iostream>
using namespace std;
```


Librerie di inclusione C++

Operazioni di input/output	stdio.h	cstdio
Funzioni di uso comune	stdlib.h	cstdlib
Gestione delle stringhe come sequenze di caratteri	string.h	cstring
Gestione dei tipi di carattere	ctype.h	cctype
Funzioni matematiche in virgola mobile	math.h	cmath
Funzioni data e ora	time.h	ctime
Operazioni di input/output in sostituzione della libreria stdio.h in C	iostream.h	iostream
Input/output sui flussi	fstream.h	fstream
Output formattati	iomanip.h	iomanip

Overloading delle funzioni

- **overloading** delle funzioni indica la possibilita` di attribuire allo stesso nome di funzione piu` significati. Attribuire piu` significati vuol dire fare in modo che lo stesso nome di funzione sia in effetti utilizzato per piu` funzioni contemporaneamente.

- l'overloading si realizza cambiando sia il numero di parametri passati sia il loro tipo

- il compilatore riconoscerà in automatico la funzione che deve essere richiamata in base al tipo e al numero di parametri

Overloading delle funzioni (esempio)

```
#include <iostream>
using namespace std;

void f(int i) {
    cout << i << endl;
}

void f(char c) {
    cout << c << endl;
}

void f(int i, char c) {
    cout << i << '\t' << c << endl;
}
```

```
int main () {
    f(10);
    f('A');
    f(100, 'z');
}
```

```
10
A
100     z
```


Overloading delle funzioni (predefinite)

- double pow (double, double) //come nel C
- float pow (float, float);
- long double pow (long double, long double);

questi tre prototipi di funzione sono esempi di **function overloading predefinito** nel linguaggio C++

Riepilogo della lezione

Introduzione al linguaggio C++

- Tipi di dati predefiniti del C++
- Variabili, costanti, stringhe in C++
- Operatori aritmetici, di relazione e logici
- Operatori di assegnamento
- Istruzioni predefinite I/O e manipolatori
- Introduzione alle funzioni matematiche
- Struttura di alternativa
- Strutture di ripetizione (pre,post, enumerativa)
- Struttura di scelta multipla
- Passaggio di parametri (valore/referenza) in C++
- Algoritmo di swap
- Namespace e librerie di inclusione
- Function overload

Fine della lezione

- **Domande?**