

WEB DEVELOPER

Fondamenti di Programmazione

Massimo PAPA

Le funzioni

- Approfondimenti -

Il punto della situazione

Ripasso:

- Tipi di funzioni c++
 - ✓ Funzioni standard
 - ✓ Funzioni definite dall'utente
- Le funzioni
 - ✓ Dichiarazione
 - ✓ Implementazione

Cosa faremo:

- Condivisione dati tra funzioni attraverso i parametri
 - ✓ Passaggio per valore
 - ✓ Passaggio per riferimento
 - ✓ Passaggio per riferimento costante
 - ✓ Parametri di default
- Visibilità (Scope) delle variabili
 - ✓ Variabili Globali
 - ✓ Variabili Locali

Intervento realizzato da

Funzioni standard C++

- Il linguaggio C++ viene fornito con molte funzioni note come funzioni standard
- Queste funzioni standard sono raggruppate in diverse librerie che possono essere incluse nel programma C++, ad es
 - ◆ Le funzioni matematiche sono dichiarate nella libreria `<cmath>`
 - ◆ Le funzioni di manipolazione dei caratteri sono dichiarate nella libreria `<cctype>`
 - ◆ C++ viene fornito con più di 100 librerie standard, alcune delle quali sono molto popolari come `<iostream>` e `<stdlib.h>`, altre sono molto specifiche per determinate piattaforme hardware, ad esempio `<limits.h>` e `<largeInt.h >`

Esempio di utilizzo

Funzioni matematiche C++ standard

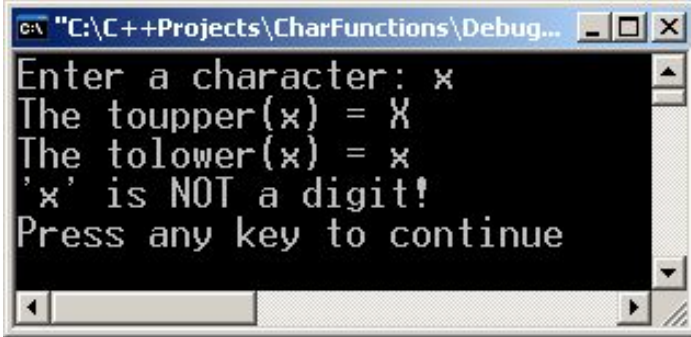
```
#include <iostream>
#include <cmath>
int main()
{
    // Getting a double value
    double x;
    cout << "Please enter a real number: ";
    cin >> x;
    // Compute the ceiling and the floor of the real
    number
    cout << "The ceil(" << x << ") = " << ceil(x) << endl;
    cout << "The floor(" << x << ") = " << floor(x) <<
    endl;
}
```

Esempio di utilizzo

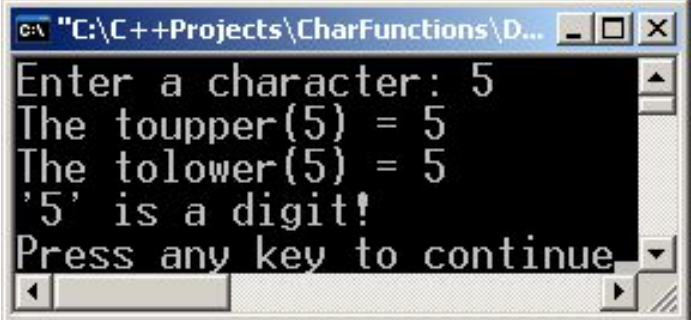
Funzioni dei caratteri C++ standard

```
#include <iostream> // input/output handling
#include <cctype> // character type functions
void main()
{
    char ch;
    cout << "Enter a character: ";
    cin >> ch;
    cout << "The toupper(" << ch << ") = " << (char) toupper(ch) << endl;
    cout << "The tolower(" << ch << ") = " << (char) tolower(ch) << endl;
    if (isdigit(ch))
        cout << "" << ch << "" is a digit!\n";
    else
        cout << "" << ch << "" is NOT a digit!\n";
}
```

Casting
esplicito



```
C:\C++Projects\CharFunctions\Debug...
Enter a character: x
The toupper(x) = X
The tolower(x) = x
'x' is NOT a digit!
Press any key to continue
```



```
C:\C++Projects\CharFunctions\D...
Enter a character: 5
The toupper(5) = 5
The tolower(5) = 5
'5' is a digit!
Press any key to continue
```

Funzioni C++ definite dall'utente

- Sebbene C++ sia fornito con molte funzioni standard, queste funzioni non sono sufficienti per tutti gli utenti, quindi C++ fornisce ai suoi utenti un modo per definire le proprie funzioni (o funzioni definite dall'utente)
- Ad esempio, la libreria `<cmath>` non include una funzione standard che consente agli utenti di arrotondare un numero reale alla i^{esima} cifre, quindi, dobbiamo dichiarare e implementare noi stessi questa funzione

Come definire una funzione C++?

- In generale, definiamo una funzione C++ in due passaggi (preferibilmente ma non obbligatori)
 - Passaggio #1: dichiarare la *firma della funzione* in un file di intestazione (file .h) o prima della funzione principale del programma
 - Passaggio 2: implementare la funzione in un file di implementazione (.cpp) o dopo la funzione principale

Qual è la struttura sintattica di una funzione C++?

- Una funzione C++ è composta da due parti
 - ◆ L'intestazione (*header*) della funzione, e
 - ◆ Il corpo (*body*) della funzione
- L'intestazione della funzione ha la seguente sintassi
<valore restituito> <nome> (<elenco parametri>)
- Il corpo della funzione è semplicemente un codice C++ racchiuso tra { }

Esempio di Definito dall'utente

Funzione C++

```
double computeTax(double income)  
{  
    if (income < 5000.0)  
        return 0.0;  
    double taxes = 0.07 * (income-5000.0);  
    return taxes;  
}
```

Esempio di funzione definita dall'utente

Header della funzione

```
double computeTax(double income)
```

```
{  
    if (income < 5000.0)  
        return 0.0;  
    double taxes = 0.07 * (income-5000.0);  
    return taxes;  
}
```

Esempio di funzione definita dall'utente

Header della
funzione

Corpo della
funzione

```
double computeTax(double income)
```

```
{  
    if (income < 5000.0)  
        return 0.0;  
    double taxes = 0.07 * (income-5000.0);  
    return taxes;  
}
```

Condivisione dati tra funzioni definite dall'utente

- Esistono due modi per condividere i dati tra diverse funzioni
 - ◆ Utilizzo di variabili globali (pratica pessima!)
 - ◆ Passaggio di dati attraverso parametri di funzione
 - Parametri del valore
 - Parametri di riferimento
 - Parametri di riferimento costanti

Variabili C++

- Una variabile è un posto nella memoria che ha
 - ◆ Un nome o un identificatore (ad es. reddito, tasse, ecc.)
 - ◆ Un tipo di dati (es. int, double, char, ecc.)
 - ◆ Una dimensione (numero di byte)
 - ◆ Un ambito (la parte del codice del programma che può utilizzarlo)
 - Variabili globali: tutte le funzioni possono vederlo e usarlo
 - Variabili locali: solo la funzione che dichiara le variabili locali vede e usa queste variabili
 - ◆ Un tempo di vita (la durata della sua esistenza)
 - Le variabili globali possono vivere finché il programma viene eseguito
 - Le variabili locali vivono solo quando vengono eseguite le funzioni che definiscono queste variabili

I. Utilizzo di variabili globali

```
#include <iostream>
using namespace std;
int x = 0;
void f1() { x++; }
void f2() { x+=4; f1(); }
int main()
{
    f2();
    cout << x << endl;
}
```

I. Utilizzo di variabili globali

```
#include <iostream>
using namespace std;
int x = 0;
void f1() { x++; }
void f2() { x+=4; f1(); }
int main()
{
    f2();
    cout << x << endl;
```

x 0



I. Utilizzo di variabili globali

```
#include <iostream>
using namespace std;
int x = 0;
void f1() { x++; }
void f2() { x+=4; f1(); }
int main()
{
    f2();
    cout << x << endl;
```

x 0

1

```
void main()
{
    f2();
    cout << x << endl ;
}
```


I. Utilizzo di variabili globali

```
#include <iostream>
using namespace std;
int x = 0;
void f1() { x++; }
void f2() { x+=4; f1(); }
int main()
{
    f2();
    cout << x << endl;
}
```

x

4

2

```
void f2()
{
    x += 4;
    f1();
}
```

1

```
void main()
{
    f2();
    cout << x << endl ;
}
```

I. Utilizzo di variabili globali

```
#include <iostream>
using namespace std;
int x = 0;
void f1() { x++; }
void f2() { x+=4; f1(); }
int main()
{
    f2();
    cout << x << endl;
}
```

X 5

4

```
void f1()
{
    x++;
}
```

3

```
void f2()
{
    x += 4;
    f1();
}
```

1

```
void main()
{
    f2();
    cout << x << endl ;
}
```

I. Utilizzo di variabili globali

```
#include <iostream>
using namespace std;
int x = 0;
void f1() { x++; }
void f2() { x+=4; f1(); }
int main()
{
    f2();
    cout << x << endl;
}
```

X 5

void f1()
{
 x++;
}

void f2()
{
 x += 4;
 f1();
}

void main()
{
 f2();
 cout << x << endl ;
}

I. Utilizzo di variabili globali

```
#include <iostream>
using namespace std;
int x = 0;
void f1() { x++; }
void f2() { x+=4; f1(); }
int main()
{
    f2();
    cout << x << endl;
```

x

5

```
void f2()
{
    x += 4;
    f1();
}
```

6

```
void main()
{
    f2();
    cout << x << endl ;
}
```

1

I. Utilizzo di variabili globali

```
#include <iostream>
using namespace std;
int x = 0;
void f1() { x++; }
void f2() { x+=4; f1(); }
int main()
{
    f2();
    cout << x << endl;
```

x

5



```
void main()
{
    f2();
    cout << x << endl ;
}
```



7

I. Utilizzo di variabili globali

```
#include <iostream>
using namespace std;
int x = 0;
void f1() { x++; }
void f2() { x+=4; f1(); }
int main()
{
    f2();
    cout << x << endl;
```

x 5



```
int main()
{
    f2();
    cout << x << endl ;
}
```

8

I. Utilizzo di variabili globali

```
#include <iostream>
using namespace std;
int x = 0;
void f1() { x++; }
void f2() { x+=4; f1(); }
int main()
{
    f2();
    cout << x << endl;
}
```



A screenshot of a Windows command prompt window. The title bar shows the path "C:\C++Projects\CharFunctions\Debug\C...". The window contains the output of the program: the number "5" on the first line, followed by the text "Press any key to continue" on the second line. The window has standard Windows controls (minimize, maximize, close) in the top right corner and a scrollbar at the bottom.

Cosa c'è di male nell'usare? Variabili globali?

- Non sicuro!

- ◆ Se due o più programmatori stanno lavorando insieme in un programma, uno di loro può cambiare il valore memorizzato nella variabile globale senza dirlo agli altri. I quali potrebbero dipendere nel loro calcolo dal vecchio valore memorizzato!

- Contro il principio dell'information hiding!

- ◆ L'esposizione delle variabili globali a tutte le funzioni è contro il principio dell'occultamento delle informazioni poiché ciò dà a tutte le funzioni la libertà di modificare i valori memorizzati nelle variabili globali in qualsiasi momento (non sicuro!)

Variabili locali

- Le variabili locali sono dichiarate all'interno del corpo della funzione ed esistono finché la funzione è in esecuzione. Vengono distrutte all'uscita della funzione
- Devi inizializzare la variabile locale prima di usarla
- Se una funzione definisce una variabile locale ed esiste una variabile globale con lo stesso nome, la funzione usa la sua variabile locale invece di usare la variabile globale

Esempio di definizione e utilizzo di variabili globali e locali

```
#include <iostream>
using namespace std;
int x; // Global variable

void fun()
{
    int x = 10; // Local variable
    cout << x << endl;
}

int main()
{
    x = 4;
    fun();
    cout << x << endl;
}
```

Esempio di definizione e utilizzo di variabili globali e locali

```
#include <iostream>
using namespace std;
int x; // Global variable

void fun()
{
    int x = 10; // Local variable
    cout << x << endl;
}

int main()
{
    x = 4;
    fun();
    cout << x << endl;
}
```

X 0

**Le variabili globali vengono
inizializzate automaticamente a 0**

Esempio di definizione e utilizzo di variabili globali e locali

```
#include <iostream>
using namespace std;
int x; // Global variable
```

x

0

```
void fun()
{
    int x = 10; // Local variable
    cout << x << endl;
}
int main()
{
    x = 4;
    fun();
    cout << x << endl;
}
```

1

```
int main()
{
    x = 4;
    fun();
    cout<<x<<endl;
}
```

Esempio di definizione e utilizzo di variabili globali e locali

```
#include <iostream>
using namespace std;
int x; // Global variable
```

X

4

```
void fun()
{
    int x = 10; // Local variable
    cout << x << endl;
}

int main()
{
    x = 4;
    fun();
    cout << x << endl;
}
```

void fun ()

X

????

3

```
{
    int x = 10;
    cout<<x<<endl;
}
```

2

```
void main()
{
    x = 4;
    fun();
    cout<<x<<endl;
}
```

Esempio di definizione e utilizzo di variabili globali e locali

```
#include <iostream>
using namespace std;
int x; // Global variable
```

```
void fun()
{
    int x = 10; // Local variable
    cout << x << endl;
}
int main()
{
    x = 4;
    fun();
    cout << x << endl;
}
```

X 4

3

```
void fun()
{
    int x = 10;
    cout << x << endl;
}
```

2

```
void main()
{
    x = 4;
    fun();
    cout << x << endl;
}
```

Esempio di definizione e utilizzo di variabili globali e locali

```
#include <iostream>
using namespace std;
int x; // Global variable
```

X

4

```
void fun()
{
    int x = 10; // Local variable
    cout << x << endl;
}
int main()
{
    x = 4;
    fun();
    cout << x << endl;
}
```



4

```
void fun()
{
    int x = 10;
    cout << x << endl;
}
```

2

```
void main()
{
    x = 4;
    fun();
    cout << x << endl;
}
```

Esempio di definizione e utilizzo di variabili globali e locali

```
#include <iostream>
using namespace std;
int x; // Global variable
```

X

4

```
void fun()
{
    int x = 10; // Local variable
    cout << x << endl;
}
int main()
{
    x = 4;
    fun();
    cout << x << endl;
}
```



void fun()

X

10

```
{
    int x = 10;
    cout << x << endl;
}
```

5

void main()

```
{
    x = 4;
    fun();
    cout << x << endl;
}
```

2

Esempio di definizione e utilizzo di variabili globali e locali

```
#include <iostream>
using namespace std;
int x; // Global variable
```

X

4

```
void fun()
{
    int x = 10; // Local variable
    cout << x << endl;
}

int main()
{
    x = 4;
    fun();
    cout << x << endl;
}
```



```
void main()
{
    x = 4;
    fun();
    cout << x << endl;
}
```

6

Esempio di definizione e utilizzo di variabili globali e locali

```
#include <iostream>
using namespace std;
int x; // Global variable
```

4

```
void fun()
{
    int x = 10; // Local variable
    cout << x << endl;
}

int main()
{
    x = 4;
    fun();
    cout << x << endl;
}
```



```
void main()
{
    x = 4;
    fun();
    cout << x << endl;
}
```

II. Utilizzo dei parametri

- I parametri di funzione sono disponibili in tre modalità di passaggio:
 - ◆ ***Passaggio per valore*** – che copiano i valori degli argomenti della funzione
 - ◆ ***Passaggio per riferimento*** – che fanno riferimento agli argomenti della funzione con altri nomi locali e hanno la capacità di modificare i valori degli argomenti a cui si fa riferimento
 - ◆ ***Passaggio per riferimento costante*** – simile ai parametri di riferimento ma non si possono modificare i valori dei parametri.

Passaggio parametri per valore

- Questo è ciò che usiamo per dichiarare nella firma della funzione o nell'header della funzione, ad es:

int max (int x, int y);

- ◆ I parametri x e y sono parametri passati per valore
- ◆ Quando chiami la funzione *max(4, 7)*, i valori 4 e 7 vengono copiati rispettivamente in x e y
- ◆ Quando chiami la funzione *max(a, b)*, dove $a=40$ e $b=10$, i valori 40 e 10 vengono copiati rispettivamente in x e y
- ◆ Quando chiami la funzione *max(a+b, b/2)*, i valori 50 e 5 sono copiati rispettivamente in x e y
- Una volta che i parametri passati per valore accettano copie dei dati degli argomenti corrispondenti, agiscono come variabili locali!

Esempio di utilizzo di parametri passati per valore e variabili globali

```
#include <iostream>
using namespace std;
int x; // Global variable
void fun(int x)
{
    cout << x << endl;
    x=x+5;
}
int main()
{
    x = 4;
    fun(x/2+1);
    cout << x << endl;
}
```

x

0

1

```
int main()
{
    x = 4;
    fun(x/2+1);
    cout << x << endl;
}
```

Esempio di utilizzo di parametri passati per valore e variabili globali

```
#include <iostream>
using namespace std;
int x; // Global variable
void fun(int x)
{
    cout << x << endl;
    x=x+5;
}
int main()
{
    x = 4;
    fun(x/2+1);
    cout << x << endl;
```

X 4



3 → void fun(int x)
{
 cout << x << endl;
 x=x+5;
}

2 → int main()
{
 x = 4; 3
 fun(x/2+1);
 cout << x << endl;
}

Esempio di utilizzo di parametri passati per valore e variabili globali

```
#include <iostream>
using namespace std;
int x; // Global variable
void fun(int x)
{
    cout << x << endl;
    x=x+5;
}
int main()
{
    x = 4;
    fun(x/2+1);
    cout << x << endl;
```



X

4

```
void fun(int x 8 )
{
    cout << x << endl;
    x=x+5;
}
```

```
int main()
{
    x = 4;
    fun(x/2+1);
    cout << x << endl;
}
```

Esempio di utilizzo di parametri passati per valore e variabili globali

```
#include <iostream>
using namespace std;
int x; // Global variable
void fun(int x)
{
    cout << x << endl;
    x=x+5;
}
int main()
{
    x = 4;
    fun(x/2+1);
    cout << x << endl;
```

X 4



```
void fun(int x 8 )
{
    cout << x << endl;
    x=x+5;
}
```

5

```
int main()
{
    x = 4;
    fun(x/2+1);
    cout << x << endl;
}
```

2

Esempio di utilizzo di parametri passati per valore e variabili globali

```
#include <iostream>
using namespace std;
int x; // Global variable
void fun(int x)
{
    cout << x << endl;
    x=x+5;
}
int main()
{
    x = 4;
    fun(x/2+1);
    cout << x << endl;
}
```

X 4



6

```
int main()
{
    x = 4;
    fun(x/2+1);
    cout << x << endl;
}
```

Esempio di utilizzo di parametri passati per valore e variabili globali

```
#include <iostream>
using namespace std;
int x; // Global variable
void fun(int x)
{
    cout << x << endl;
    x=x+5;
}
int main()
{
    x = 4;
    fun(x/2+1);
    cout << x << endl;
```

4



```
int main()
{
    x = 4;
    fun(x/2+1);
    cout << x << endl;
}
```

Parametri passati per riferimento

- Come abbiamo visto nell'ultimo esempio, qualsiasi modifica ai parametri passati per valore non influisce sui valori degli argomenti passati alla funzione.
- A volte, vogliamo cambiare i valori degli argomenti della funzione originale o ritornare più di un valore dalla funzione, in questo caso utilizziamo i parametri passati per riferimento
 - ◆ Un parametro passato per riferimento è solo un altro nome della variabile argomento originale
 - ◆ Definiamo un parametro di riferimento aggiungendo & davanti al nome del parametro, ad es

double aggiornamento(double &x);

Esempio di parametri passati per riferimento

```
#include <iostream>
using namespace std;
void fun(int &y)
{
    cout << y << endl;
    y=y+5;
}
int main()
{
    int x = 4; // Local variable
    fun(x);
    cout << x << endl;
}
```

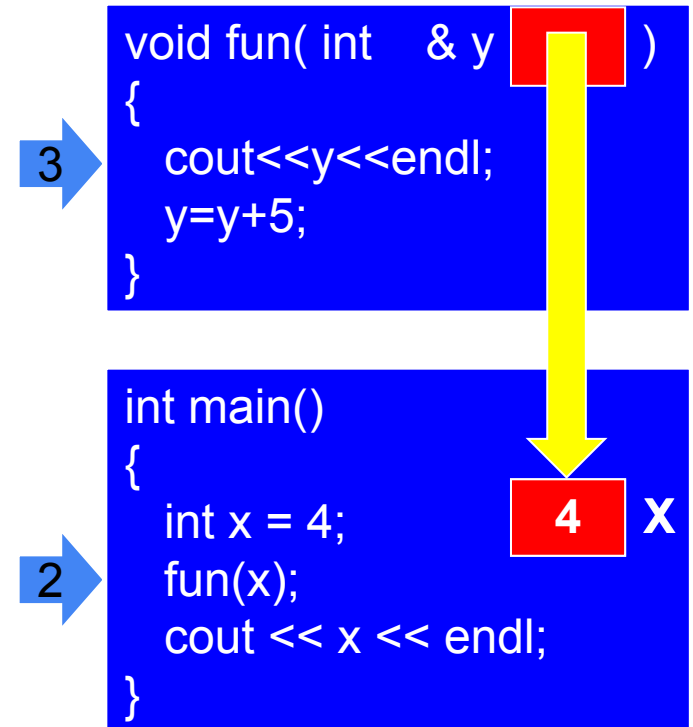


```
int main()
{
    int x = 4;
    fun(x);
    cout << x << endl;
}
```

4 **x**

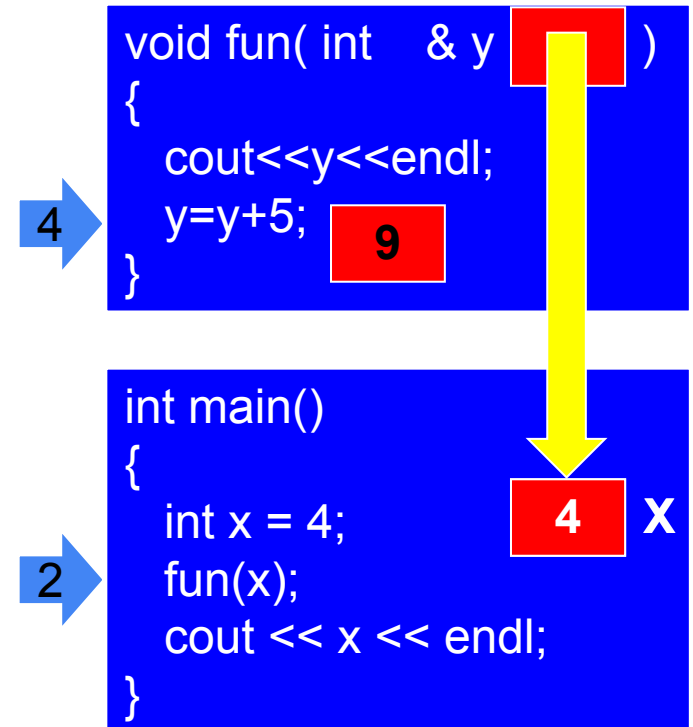
Esempio di parametri passati per riferimento

```
#include <iostream>
using namespace std;
void fun(int &y)
{
    cout << y << endl;
    y=y+5;
}
int main()
{
    int x = 4; // Local variable
    fun(x);
    cout << x << endl;
}
```



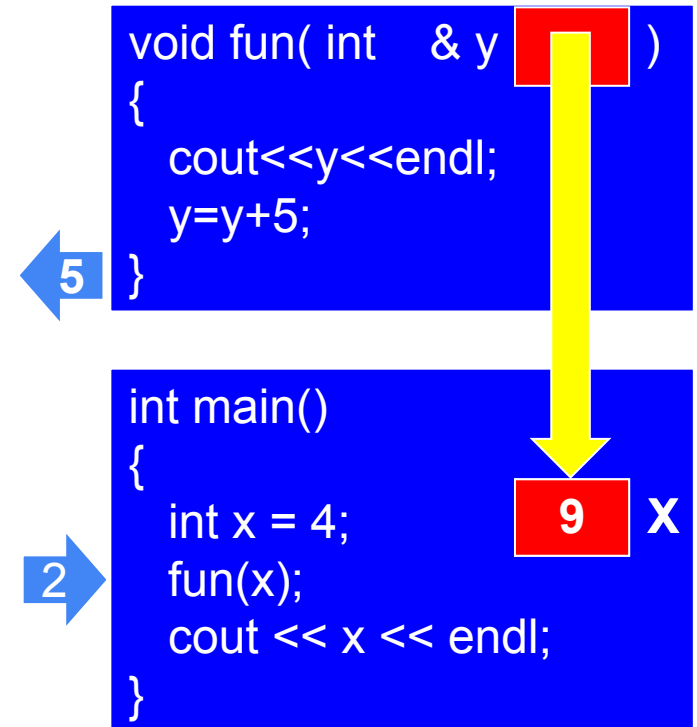
Esempio di parametri passati per riferimento

```
#include <iostream>
using namespace std;
void fun(int &y)
{
    cout << y << endl;
    y=y+5;
}
int main()
{
    int x = 4; // Local variable
    fun(x);
    cout << x << endl;
}
```



Esempio di parametri passati per riferimento

```
#include <iostream>
using namespace std;
void fun(int &y)
{
    cout << y << endl;
    y=y+5;
}
int main()
{
    int x = 4; // Local variable
    fun(x);
    cout << x << endl;
}
```



Esempio di parametri passati per riferimento

```
#include <iostream>
using namespace std;
void fun(int &y)
{
    cout << y << endl;
    y=y+5;
}
int main()
{
    int x = 4; // Local variable
    fun(x);
    cout << x << endl;
```



```
int main()
{
    int x = 4;
    fun(x);
    cout << x << endl;
}
```

9 X



Esempio di parametri passati per riferimento

```
#include <iostream>
using namespace std;
void fun(int &y)
{
    cout << y << endl;
    y=y+5;
}
int main()
{
    int x = 4; // Local variable
    fun(x);
    cout << x << endl;
}
```



```
int main()
{
    int x = 4;
    fun(x);
    cout << x << endl;
}
```

9 X

7

Parametri passati per riferimento costanti

- I parametri di riferimento costanti vengono utilizzati nelle due condizioni seguenti:
 - ◆ I dati passati sono così grandi e vuoi risparmiare tempo e memoria del computer
 - ◆ I dati passati non verranno modificati o aggiornati nel corpo della funzione

- Esempio:

void report (const string &prompt);

- Gli unici argomenti validi accettati dai parametri passati per riferimento e dai parametri passati per riferimento costanti sono i nomi delle variabili
 - ◆ È un errore di sintassi passare valori o espressioni costanti ai parametri di riferimento (const)

Parametri di default

- I parametri di default (o predefiniti) ci consentono di evitare di inserire tutti i parametri attuali di una funzione
- Questo ci permette di scrivere codice più chiaro e leggibile.
- Esempio di header:

```
void fun(int a, float b=2.5)
```

- Chiamata alla funzione:

```
fun(5); //Non dà errore  
fun(5, 6.4)
```