

Introduzione a ROOT

Riccardo Lollini

riccardo.lollini@cern.ch

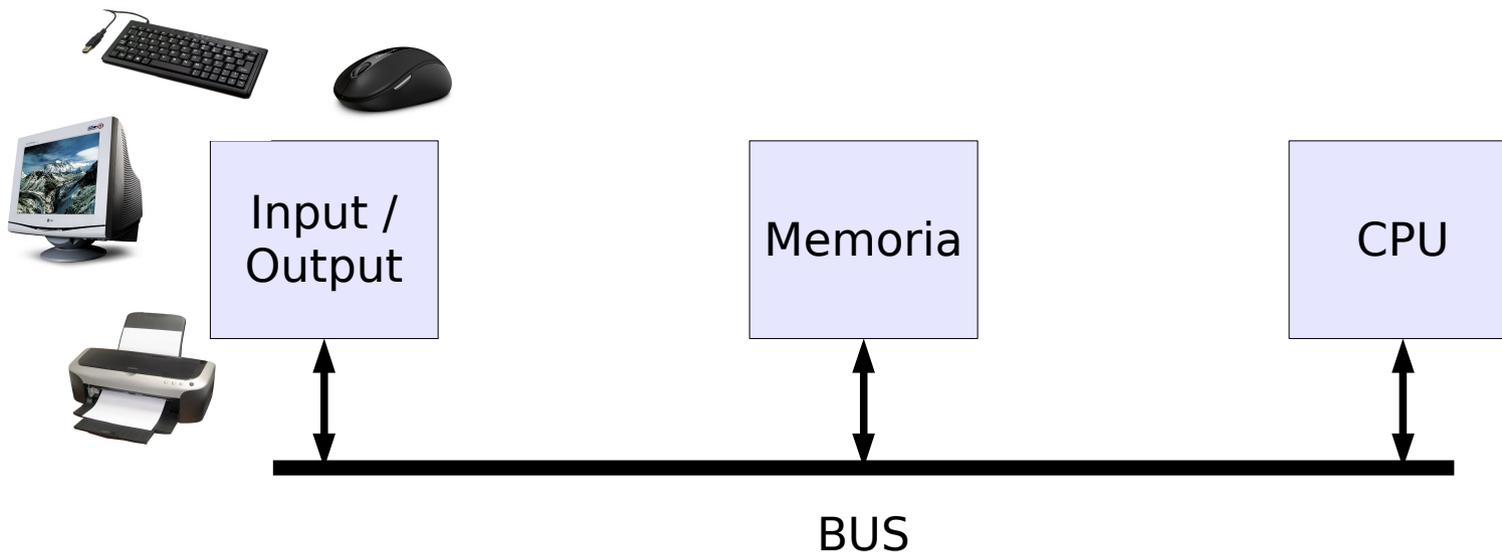
- Introduzione al linguaggio C++
- Introduzione al framework ROOT
- Esempi

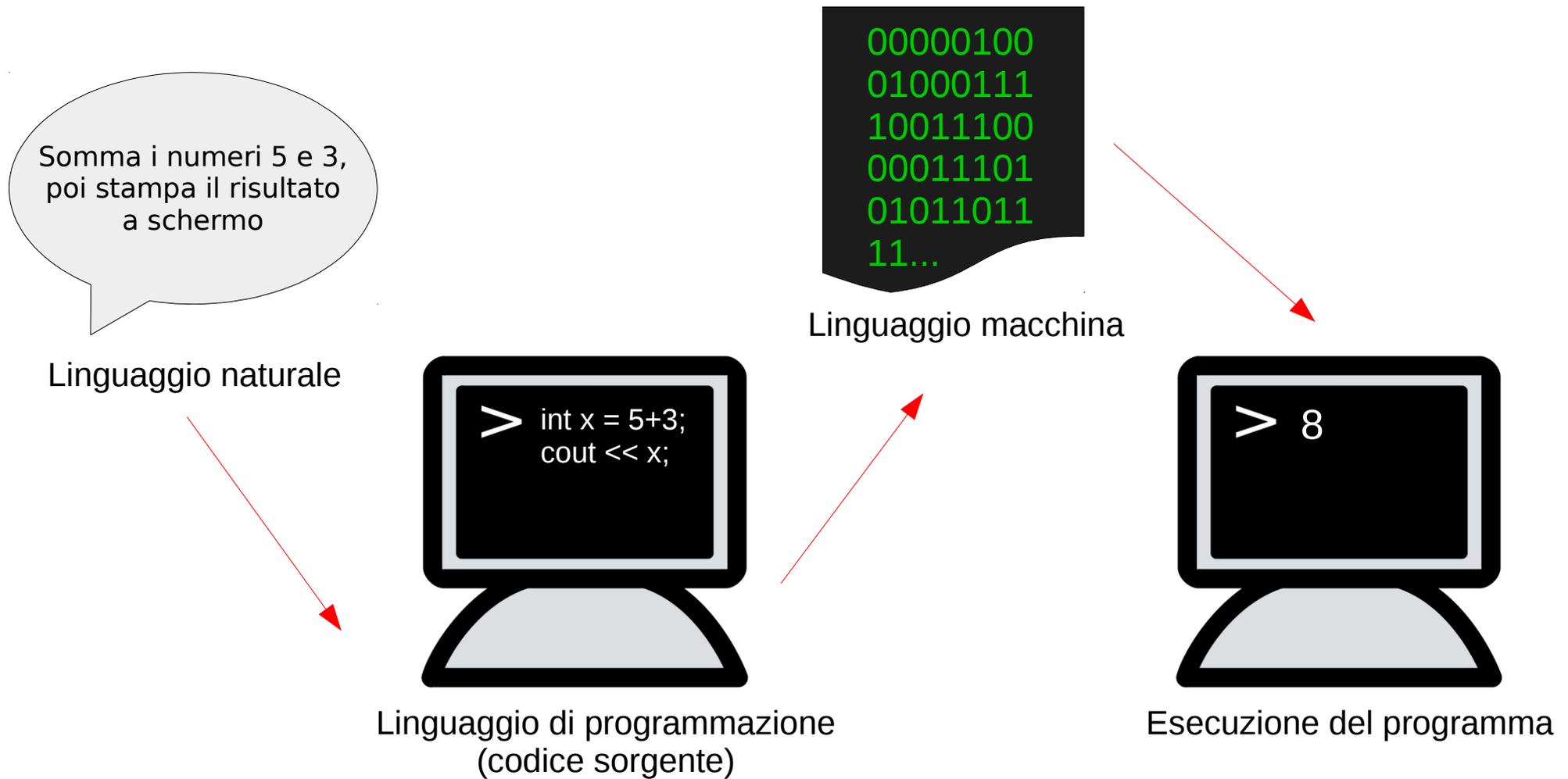


Programmazione: si richiede che un certo problema venga risolto da un calcolatore. Per fare ciò è necessario scrivere dei programmi.

Linguaggio di programmazione: strumento per scrivere programmi. È un linguaggio che permette la comunicazione tra il programmatore (umano) e il calcolatore. Il programmatore può così istruire il computer a risolvere dei problemi.

Programma: sequenza di istruzioni eseguibile dalla CPU.





Interpreti e compilatori effettuano la traduzione da un linguaggio di programmazione al linguaggio macchina, comprensibile dalla CPU.

Interprete: il codice sorgente è tradotto riga per riga durante l'esecuzione.

Compilatore: il codice sorgente è tradotto globalmente e trasformato in un file eseguibile.

Compilatori

- ✓ Esecuzione più veloce
- ✓ Errori nel codice vengono trovati prima dell'esecuzione
- ✗ La compilazione richiede tempo (es. vedere gli effetti di una piccola modifica)
- ✗ Poco portabile

Esempi: C, C++, BASIC, Visual Basic, Pascal, ...

Interpreti

- ✗ Esecuzione più lenta
- ✗ Un errore banale nel codice potrebbe essere scoperto solo dopo ore di esecuzione
- ✓ Vedere subito “cosa fa” il programma
- ✓ Facilmente utilizzabile su sistemi operativi diversi

Esempi: Python, Javascript, MATLAB, Perl, PHP, ...

```
riccardo@riccardo-X550LA: ~/Scrivania
riccardo@riccardo-X550LA:~$ ls
Documenti  examples.desktop  lms  Musica  root  Scrivania  Video
Dropbox   Immagini          Modelli  Pubblici  Scaricati  texmf  wordpress
riccardo@riccardo-X550LA:~$ cd Scrivania/
riccardo@riccardo-X550LA:~/Scrivania$ scp rilollin@lxplus126.cern.ch:/tmp/riloll
in/output.root .
Password:
output.root                                100% 7108KB  3.5MB/s  00:02
riccardo@riccardo-X550LA:~/Scrivania$
```

Sia che si usi Linux, sia che si usi Windows, è importante imparare a lavorare da linea di comando.

È un modo per dare istruzioni al sistema operativo alternativo (e spesso molto più potente) al point-and-click del mouse.

Perché imparare a usare l'interfaccia da riga di comando:

- si può fare tutto ciò che si fa da interfaccia grafica e molto di più;
- è veloce;
- è più sicura;
- si possono creare script;
- potrete atteggiarvi a grandissimi nerd.

Comandi per file e directory

- `pwd` mostra in quale directory si è attualmente posizionati
- `cd` permette di spostarsi in un'altra directory
 - `cd ~` per spostarsi nella propria home directory
 - `cd ..` per spostarsi indietro di una directory
- `ls` mostra i file presenti nella directory corrente
 - `ls -l` mostra anche altre informazioni sui file
- `cp` per copiare un file
 - esempio: `cp file.pdf file2.pdf` crea una copia del file `file.pdf` e la chiama `file2.pdf`
- `mv` sposta un file in un'altra directory oppure lo rinomina
 - esempio: `mv file.pdf ~/Documenti` sposta il file `file.pdf` nella cartella `Documenti`
 - esempio: `mv file.pdf ciao.pdf` rinomina il `file.pdf` in `ciao.pdf`
- `rm` rimuove un file
- `<TAB>` = autocompletamento

Un primo programma in C++

Hello, world! è un semplicissimo programma che stampa a schermo la frase “Hello, world!”.

```
#include <cstdio>

int main() {
    printf("Hello, world!\n");
    return 0;
}
```

codice sorgente

```
Hello, world!
```

risultato in fase di esecuzione

- Il codice sorgente va salvato in un file `.cpp` (es. `helloworld.cpp`).
- Da terminale si compila digitando `g++ -o ciao helloworld.cpp`.
- Il programma “ciao” così ottenuto può essere eseguito con `./ciao`.

Come nei linguaggi naturali, anche nei linguaggi di programmazione c'è un lessico (vocabolario), una sintassi (regole di composizione delle frasi) e una semantica (significato delle frasi).

```
#include <stdio>

int main() {
    printf("Hello, world!\n");
    return 0;
}
```

Ogni istruzione deve terminare con un punto e virgola ;

Blocchi di codice sono racchiusi da parentesi graffe { ... }

Le librerie sono richiamate con `#include` e sono dei file che contengono una serie di funzioni già pronte

```
#include <stdio>

int main() {
    printf("Hello, world!\n");
    return 0;
}
```

`printf()` è una funzione definita nella libreria `stdio` (C Standard Input and Output library)

Per andare a capo dentro una *stringa* si usa `\n`

La funzione `main()` è una funzione speciale che definisce il punto di inizio dell'esecuzione del programma

```
#include <stdio>
int main() {
    printf("Hello, world!\n");
    return 0;
}
```

La funzione `main()` è definita come `int`, cioè il suo "risultato" è un numero intero. Per questo motivo deve terminare con `return 0`

Il return code può essere usato da altre applicazioni per capire se il programma è terminato correttamente. `return 0` implica che non ci sono stati errori nell'esecuzione

```
// il mio primo programma in c++
#include <cstdio>

int main() {
    printf("Hello, world!\n"); // ciao
    /* posso anche
    commentare su più
    righe.*/
    return 0;
}
```

È possibile inserire dei **commenti** all'interno del codice. I commenti vengono completamente **ignorati dal compilatore**, ma servono al programmatore per avere più chiaro il significato di alcune porzioni di codice. Si può commentare con `//` (commento a riga singola) oppure con `/* ... */` (commento su più righe).

Variabili e tipi di dati

Variabile: porzione di memoria in cui può essere contenuto un valore che può essere modificato nel corso di esecuzione del programma.

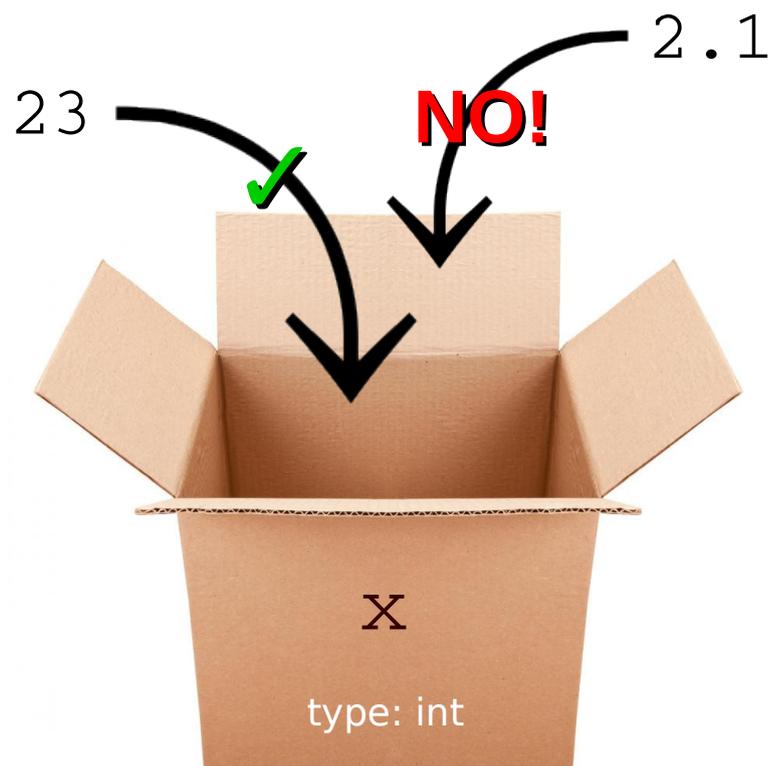
Le variabili sono caratterizzate da un nome e da un tipo di dato.

Nome: sequenza di uno o più caratteri alfanumerici (più l'underscore `_`) e deve iniziare con una lettera o con l'underscore.

- *X*, *gianfranco*, *Gianfranco*, *_boh*, *vettore3* sono tutti nomi accettabili (attenzione: il C++ è case-sensitive, distingue tra maiuscole e minuscole!).
- *3vettore*, *x\$d*, *c<*, *32!d* non sono nomi accettabili (contengono caratteri speciali o iniziano con un numero).

Tipo di dato: indica quale tipo di valori può assumere la variabile (numero intero, numero decimale, carattere, stringa, numero booleano, ...).

Variabili e tipi di dati



```
#include <stdio>

int main() {
    int x;
    x = 23;
    return 0;
}
```

`int x;` è la **dichiarazione** della variabile. Riserva una cella di memoria per una nuova variabile chiamata `x`.

`x = 23;` **assegna** alla variabile `x` (ovvero inserisce nella cella di memoria riservata) il valore 23.

Avremmo potuto anche scrivere semplicemente `int x = 23;` (variabile **inizializzata**).

Tipi di dato fondamentali del C++

- `int` : numeri interi
- `float` : numeri a virgola mobile
- `double` : numeri a virgola mobile (doppia precisione)
- `char` : singolo carattere
- `bool` : variabile booleana (vero/falso, 1/0)
- `void` : nessun valore (utile per funzioni che non devono ritornare un valore)
- ...

Operatori del C++

- **Operatore di assegnamento (=)**: assegna un valore a una variabile.

Esempio: `x = 5;` .

L'assegnamento avviene sempre da destra a sinistra: `x = y;` assegna ad `x` il valore contenuto in `y`, mentre `y` resta inalterata.

```
#include <cstdio>

int main() {
    int x, y;
    x = 5;
    y = 3;

    printf("x = %d, y = %d \n", x, y);
    return 0;
}
```

x = 5, y = 3

```
#include <cstdio>

int main() {
    int x, y;
    x = 5;
    y = 3;
    x = y;
    printf("x = %d, y = %d \n", x, y);
    return 0;
}
```

x = 3, y = 3

(`%d` indica che lì va inserito il valore di una variabile di tipo intero, specificata dopo)

Operatori del C++

- **Operatori aritmetici (+, -, *, /, %)**

Esempi (int x = 5, int y = 3):

- **Somma**

```
z = x+y;  
printf("z = %d", z);
```

```
z = 8
```

- **Sottrazione**

```
z = x-y;  
printf("z = %d", z);
```

```
z = 2
```

- **Moltiplicazione**

```
z = x*y;  
printf("z = %d", z);
```

```
z = 15
```

- **Divisione**

```
z = x/y;  
printf("z = %d", z);
```

```
z = 1
```

- **Modulo (resto della divisione)**

```
z = x%y;  
printf("z = %d", z);
```

```
z = 2
```

*****, **/** e **%** hanno la precedenza su **+** e **-**.
Si possono usare le parentesi tonde per cambiare l'ordine in cui sono eseguite le operazioni.

Esercizio 1

- Definire due variabili float **b** e **h** e assegnargli un qualche valore;
- Stampare a schermo il risultato di somma, differenza, prodotto e divisione dei due numeri.

Suggerimento: per stampare un float usare %f dentro a printf()

Esercizio 2

- Modificare il seguente codice in modo da leggere due variabili **int x, y** scelte dall'utente;

```
#include <stdio>

int main() {
    int x, y;
    printf("Inserire la prima variabile: ");
    scanf("%d",&x);
    // .....
    return 0;
}
```

- assegnare alla prima variabile x il risultato del prodotto tra x e y;
- stampare a schermo x.

Funzioni

Se un blocco di codice è ripetuto più volte o costituisce un algoritmo ben definito, può essere utile definire una funzione. Come le variabili, le funzioni devono avere un tipo.

Esempio: quadrato di un numero reale.

```
#include <stdio>

float Quadrato(float x);

int main() {
    float a = 3.72;
    float a2 = Quadrato(a);
    printf("%f al quadrato fa %f \n", a, a2);
    return 0;
}

float Quadrato(float x) {
    return x*x;
}
```

Definizione del prototipo
della funzione

Prototipo:
TipoRestituito NomeFunzione(
TipoArgomento1 Argomento1, ...);

Dichiarazione della funzione

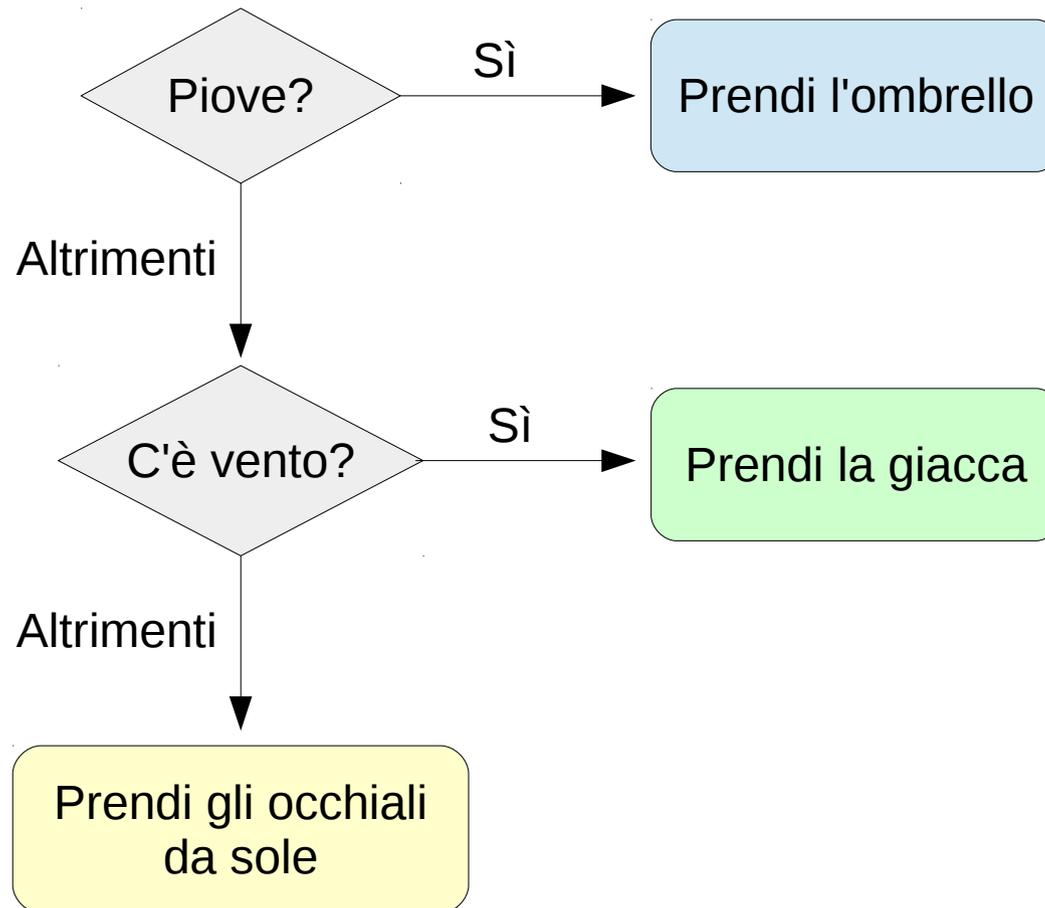
3.720000 al quadrato fa 13.838400

Strutture condizionali (if-else)

Finora abbiamo sempre visto istruzioni eseguite in sequenza. È utile avere uno strumento per modificare il flusso del programma.

Le struttura `if-else` permette di specificare che un dato blocco di istruzioni deve essere eseguito solo se si verificano certe condizioni.

Esempio:



Strutture condizionali (if-else)

```
int main() {  
    bool rain, wind;  
    rain = false;  
    wind = true;  
  
    if (rain == true) {  
        PrendiOmbrello();  
    }  
    else if (wind == true) {  
        PrendiGiacca();  
    }  
    else {  
        PrendiOcchialiSole();  
    }  
    // fine della struttura if-else  
  
    return 0;  
}
```

Strutture condizionali (if-else)

```
int main() {  
    bool rain, wind;  
    rain = true;  
    wind = false;  
  
    if (rain == true) {  
        PrendiOmbrello();  
    }  
    else if (wind == true) {  
        PrendiGiacca();  
    }  
    else {  
        PrendiOcchialiSole();  
    }  
  
    return 0;  
}
```

Attenzione a non confondere `==` con `=`.
`=` è l'operatore di assegnamento (`x = y` → "x diventa uguale a y", mentre `x == y` → "x e y sono uguali?").

in questo esempio le parentesi graffe non sarebbero necessarie, in quanto c'è una sola istruzione per ogni blocco. Se c'è più di una istruzione, diventano obbligatorie.

Operatori logici

È possibile utilizzare gli operatori logici come AND (&&), OR (||) e NOT (!).

a	b	a && b	a b
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	1

0 → FALSE
1 → TRUE

Esempio:

```
if (a && !b) { ...
```

Se a e non-b sono entrambe vere ...

Operatori logici e di confronto

AND logico	&&
OR logico	
NOT logico	!
Uguale	==
Diverso	!=
Maggiore	>
Minore	<
Maggiore o uguale	>=
Minore o uguale	<=

Esercizio 3 - Maggiore o minore

- Scrivere un programma che legga da tastiera tre numeri e determini quale dei tre numeri sia il maggiore

Esercizio 4 - Pari o dispari

- Leggere una variabile intera scelta dall'utente;
- Se la variabile è pari, stampare a schermo “è un numero pari”, altrimenti stampare a schermo “è un numero dispari”.

Suggerimento: ricordate che esiste l'operatore modulo % (resto della divisione intera).

Esercizio 5 - Programma che riconosce se un anno è bisestile

- Leggere una variabile intera scelta dall'utente (anno)
- Stampare a schermo “è un anno bisestile” se l'anno inserito è bisestile, “non è un anno bisestile” altrimenti.

Suggerimento:

Nel calendario gregoriano, quindi, sono bisestili:

- gli anni non secolari il cui numero è divisibile per 4;
- gli anni secolari il cui numero è divisibile per 400.

cit. Wikipedia

Soluzione esercizio 3

```
#include <stdio>

int main() {
    int a, b, c;
    int massimo;

    printf("Primo numero: ");
    scanf("%d", &a);
    printf("Secondo numero: ");
    scanf("%d", &b);
    printf("Terzo numero: ");
    scanf("%d", &c);

    if (a >= b) massimo = a;
    else massimo = b;

    if (c >= massimo) massimo = c;

    printf("%d è il massimo dei tre numeri \n", massimo);

    return 0;
}
```

Strutture iterative (ciclo for)

Con le strutture iterative (cicli) è possibile eseguire ripetutamente un blocco di istruzioni.

Esempio:

```
#include <stdio>

int main() {
    for (int i=0; i<5; i++){
        int iquad = i*i;
        printf("%d \n", iquad);
    }
    return 0;
}
```

Valore iniziale

Valore finale

Incremento

`i++` significa `i=i+1`

Attenzione ai punti e virgola!

`(int i=0; i<5; i++)`

La variabile `i` esiste solo all'interno del ciclo, poi viene distrutta.

“Per `i` che va da 0 a 4, incrementando `i` di una unità ogni volta, ...”

```
0
1
4
9
16
```

Esercizio 6 - Elevamento a potenza

- Leggere un numero float x e un numero int n scelti dall'utente;
- Calcolare x^n e stampare il risultato a schermo.

Esercizio 7 - Fibonacci

- Leggere un numero intero n scelto dall'utente
- Stampare a schermo i primi n numeri nella **successione di Fibonacci**.

Suggerimento: nella sequenza di Fibonacci, ciascun termine è la somma dei due precedenti. L' n -esimo numero di Fibonacci $F(n)$ è dato da: $F(n) = F(n-1) + F(n-2)$. È necessario definire i due termini iniziali.

Dunque, i primi termini della successione sono: 1, 1, 2, 3, 5, 8, 13, 21, ...

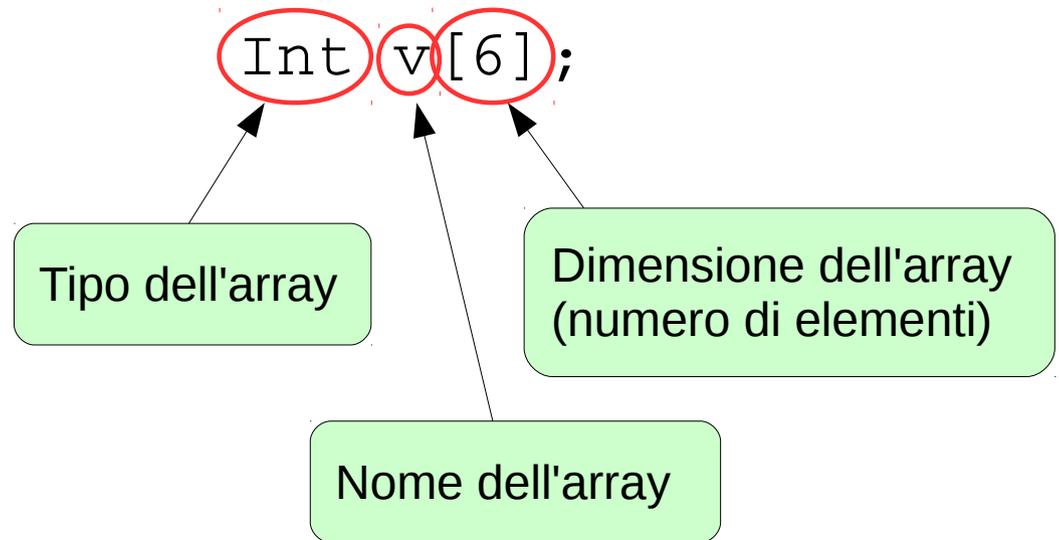
Array

Gli array sono insiemi di variabili dello stesso tipo, analoghi a vettori o matrici.

Esempio:

```
int v[6];  
  
v[0] = 5;  
v[1] = 3;  
v[2] = 3;  
v[3] = 10;  
v[4] = 0;  
v[5] = 122;
```

Se la dimensione è n , il primo elemento è 0 e l'ultimo elemento è $n-1$.



Attenzione alle parentesi quadre!

Array

Esempio: calcolare la media dei voti di 10 studenti.

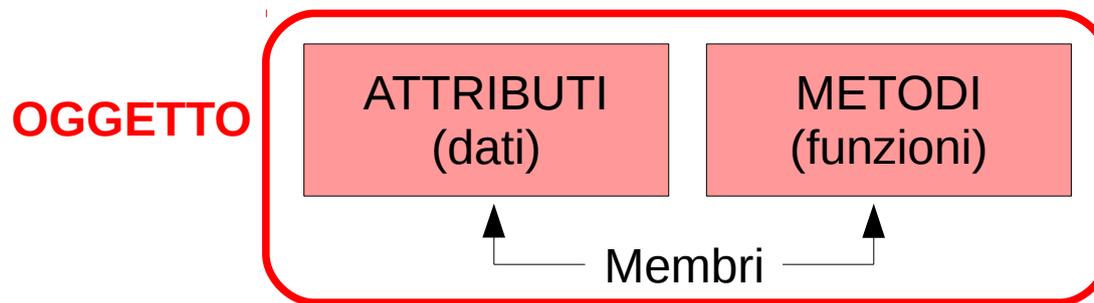
```
#include <stdio>

int main() {
    int voto[10] = {27, 28, 28, 22, 30, 18, 25, 28, 30, 27};
    float media = 0;
    for (int istud=0; istud<10; istud++){
        media = media + voto[istud];
    }
    media = media/10;
    printf("La media dei voti è %f \n", media);
    return 0;
}
```

Notare le parentesi graffe { ... } nella dichiarazione e inizializzazione dell'array.

Programmazione orientata ad oggetti

Oggetto: insieme di dati, con le funzioni che servono a manipolare quegli stessi dati.



Stato di un oggetto: valore di tutti i suoi parametri in un certo istante.

Classe: tipo di dato astratto che permette la creazione di oggetti secondo le caratteristiche della classe stessa.

Gli oggetti sono *istanze* di (= “appartengono a”) una classe.

In sostanza gli oggetti sono come delle variabili e le classi sono i corrispondenti tipi.

Programmazione orientata ad oggetti

Esempio: classe **Rettangolo**

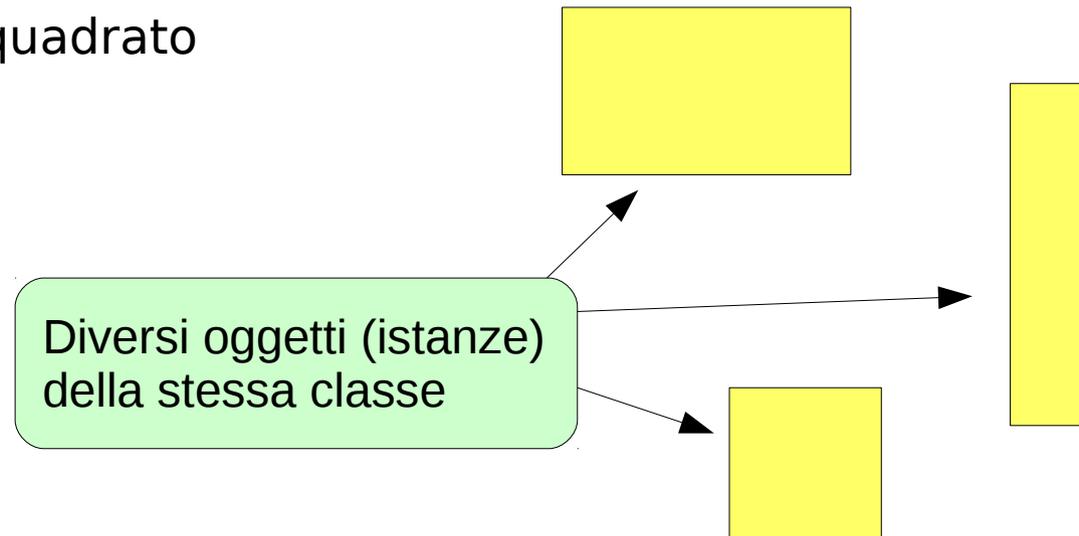
Attributi:

- base
- altezza

Gli attributi definiscono completamente un particolare rettangolo (oggetto).

Metodi:

- impostare valori di base e altezza
- calcolare l'area
- sapere se è un quadrato



Programmazione orientata ad oggetti

```
#include <cstdio>

class Rettangolo {
    private:
        float base, altezza;
    public:
        void ImpostaValori(float, float);
        float Area();
        bool Quadrato();
};

void Rettangolo::ImpostaValori(float b, float h) {
    base = b;
    altezza = h;
}

float Rettangolo::Area() {
    return base*altezza;
}

bool Rettangolo::Quadrato() {
    if (base == altezza) return true;
    else return false;
}

...
```

I membri definiti come private non sono visibili all'esterno della classe, quelli public sì

Programmazione orientata ad oggetti

Rettangolo è la classe,
 r1 è un oggetto della
 classe Rettangolo

```
...  
  
int main() {  
    Rettangolo r1;  
    r1.ImpostaValori(4.,3.);  
  
    float A = r1.Area();  
    printf("L'area del rettangolo è %f.\n", A);  
  
    bool q = r1.Quadrato();  
    if (q) printf("è un quadrato.\n");  
    else printf("non è un quadrato.\n");  
  
    return 0;  
}
```

Si può accedere a ogni
 membro pubblico dell'oggetto
 inserendo un punto (.) tra il
 nome dell'oggetto e il nome
 del membro

Programmazione orientata ad oggetti

Vantaggi della programmazione orientata ad oggetti:

- ✓ Divisione del programma in unità auto-consistenti
- ✓ Livello di astrazione più alto: programmazione più vicina all'utente che alla macchina.
- ✓ Semplificazione nella manutenzione e nel riuso del codice.

Esempio: un videogioco



Puntatori

Un puntatore è una variabile il cui contenuto è l'indirizzo di memoria di un'altra variabile.

```
int x = 5; // definisce una variabile il cui contenuto è 5
int * p; // definisce un puntatore a una variabile int
p = &x; // assegna al puntatore p l'indirizzo di memoria della variabile x

printf("%d \n",x); // stampa 5
printf("%d \n",p); // stampa 38987504, l'indirizzo di memoria di x
printf("%d \n",*p); // stampa 5, il contenuto della variabile puntata da p
```

ROOT

ROOT è un **framework orientato ad oggetti** usato per l'**analisi dati** in fisica delle particelle.

ROOT fornisce:

- molte **classi** utili per creare istogrammi, riempirli, fare un fit dei dati e molto altro;
- un'**interfaccia grafica** che permette di visualizzare gli oggetti creati;
- **CINT**, un interprete di C++.



ROOT

Data Analysis Framework

Per aprire l'interprete, digitare `root` in un terminale.

```
// Dichiarazione dell'istogramma
TH1F * h = new TH1F("hvoti", "voti degli studenti", 6, 24.5, 30.5);

h->Fill(26); // incrementa di 1 il bin 26
h->Fill(27);
h->Fill(28,4); // incrementa di 3 il bin 28
h->Fill(30);
h->Fill(27);

h->Draw(); // disegna l'istogramma
```

Creare un'istogramma

1) Metodo statico

TH1F h(argomenti...)

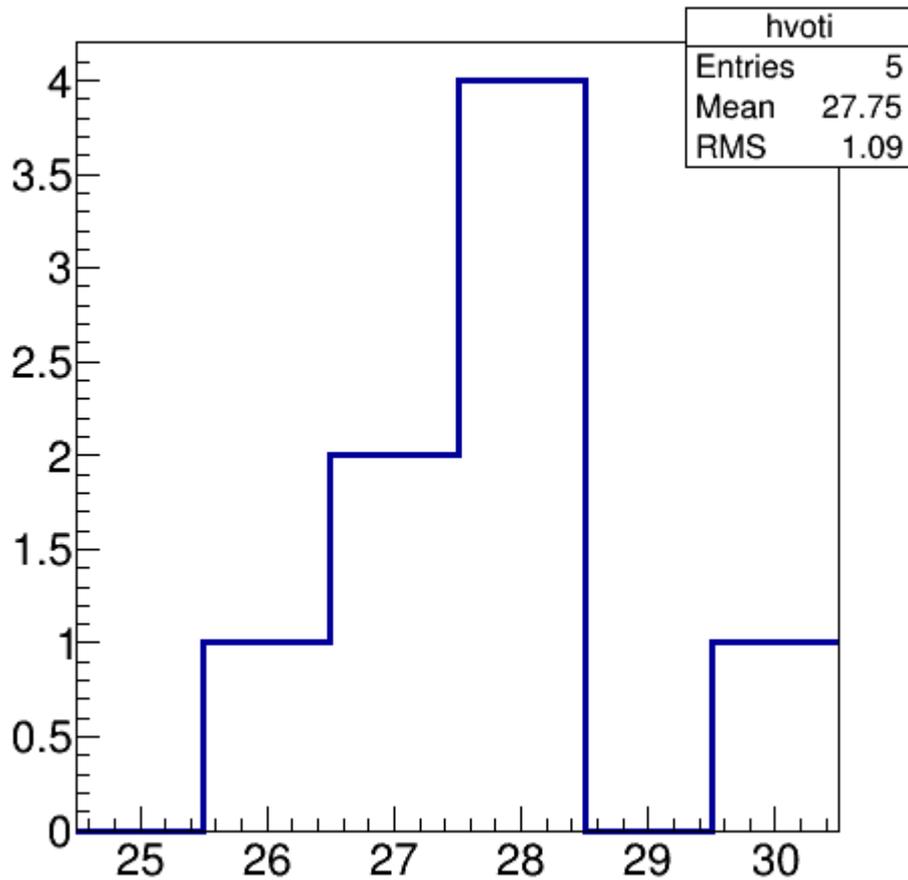
- crea l'oggetto istogramma di tipo TH1F
- statico: viene allocata automaticamente memoria per l'oggetto ma non può essere rimossa “al volo”, ci pensa il programma.
- `h.Fill(...)`

2) Metodo dinamico

TH1F * h = new TH1F(argomenti...)

- istogramma è un puntatore di tipo TH1F
- crea l'oggetto
- può essere creato(new)/rimosso(delete) “al volo”.
- `h->Fill(...)`

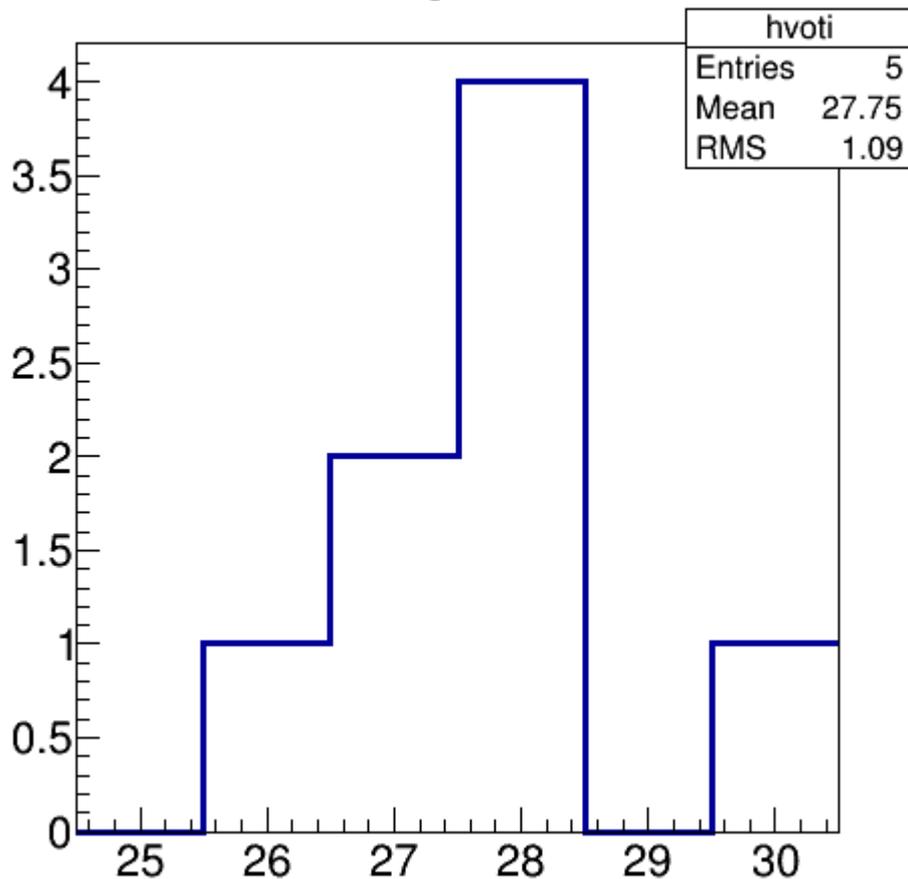
voti degli studenti



```
TH1F * h = new TH1F("hvoti", "voti degli  
studenti", 6, 24.5, 30.5);
```

- TH1F: classe istogramma di float 1-dimensionale. (altre opzioni: TH2F, TH3F, TH1I, ...).
- "hvoti": nome dell'istogramma.
- "voti degli studenti": titolo dell'istogramma.
- 6: numero di bin.
- 24.5, 30.5: x iniziale e x finale.

voti degli studenti



```
h->Fill(27);
h->Fill(28,4);
```

- `Fill()` è un metodo della classe `TH1`:
 - `Fill(x)`: riempie di una unità il bin corrispondente ad x .
 - `Fill(x,w)`: riempie di una quantità w il bin corrispondente ad x .
- Si può usare anche per istogrammi 2 e 3-dimensionali:
 - `Fill(x,y)`, `Fill(x,y,w)`, `Fill(x,y,z)`, `Fill(x,y,z,w)`.

```
h->Draw();
```

- `Draw()` disegna l'istogramma. Possono essere specificate molte opzioni:
 - `Draw("SAME")` sovrappone l'istogramma a un istogramma precedentemente disegnato.
 - `Draw("E")` disegna le barre d'errore.
 - ...

- **Tbrowser b** mostra l'interfaccia grafica di ROOT
- **.q** exit cint
 - .qqq exit cint - mandatory
 - .qqqqq exit process immediately
 - .qqqqqqq abort process
- Tab-completion dei comandi e dei nomi di file
- Tutti i tipi del C++ sono disponibili:
 - int → **Int_t**
 - float → **Float_t**
 - double → **Double_t**
 - ...
- I nomi delle classi iniziano con T:
 - **TH1F** → istogramma 1-dimensionale contenente float
 - **TF1** → funzione 1-dimensionale
 - **TFile** → file

Istogrammi

Aprirete root, create e riempite un istogramma:

```
TH1F * h1 = new TH1F("h1","titolo istogramma",100,-3.,3.);  
h1->FillRandom("gaus",10000);
```

Provate a smanettare con l'istogramma che avete creato:

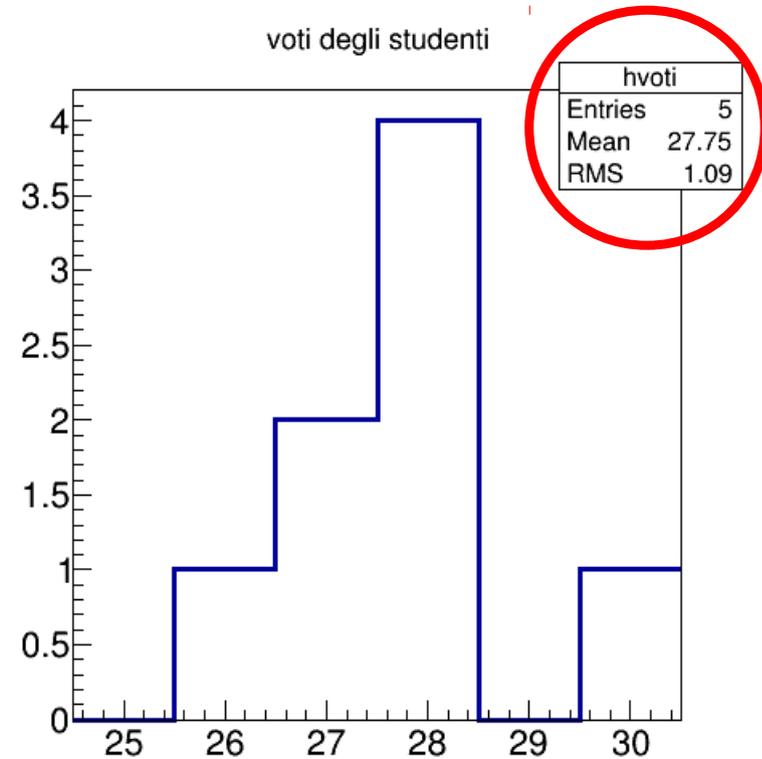
- Cambia colore della linea `h1->SetLineColor(kRed);`
- Dai un titolo `h1->SetTitle("titolo");`
- Titolo dell'asse x `h1->SetTitle("asse x");`
- Colore del marker `h1->SetMarkerColor(kGreen);`
- Dimensione del marker `h1->SetMarkerSize(2.);`
- Stile del marker `h1->SetMarkerStyle(kStar);`
- Istogramma con barre di errore `h1->Draw("e");`
- Sovrapponi un secondo istogramma `h2->Draw("same");`
- Riempi istogramma con gaussiana `h1->FillRandom("gaus");`

Molte informazioni su come funzionano gli istogrammi su ROOT le potete trovare su: <https://root.cern.ch/root/html534/guides/users-guide/Histograms.html>

Statistics Box

Mostra diverse informazioni sull'istogramma:

nome, media, rms, numero di entries, ...



- Default `gStyle->SetOptStat();`
- Togli la statbox `gStyle->SetOptStat(0);`
- Mostra tutto `gStyle->SetOptStat(1111111);`
- Mostra nome e numero di eventi `gStyle->SetOptStat(11);`

I programmi scritti per ROOT vengono solitamente chiamati **Macro**.

Differenze rispetto a un programma in C++: la **funzione principale** non è più `main()` ma deve avere lo **stesso nome del file di codice**.

File: Prova.C 



```
#include <cstdio>

float Quadrato(float x);

// La funzione principale si chiama come il file.
// Essendo di tipo void, non va messo "return 0".
void Prova() {
    float a = 3.72;
    float a2 = Quadrato(a);
    printf("%f al quadrato fa %f \n", a, a2);
}

float Quadrato(float x) {
    return x*x;
}
```

Con le Macro, si può utilizzare direttamente CINT per compilare.
Perciò, dopo aver digitato `root` da terminale:

```
CINT/ROOT C/C++ Interpreter version 5.18.00, July 2, 2010
Type ? for help. Commands must be C++ statements.
Enclose multiple statements between { }.
root [0] .L Prova.cpp+
root [1] Prova()
Il Quadrato di 3.720000 è 13.838400
root [2] .x Prova.cpp+
Il Quadrato di 3.720000 è 13.838400
root [3].q
```

`.L` carica la macro

Le funzioni nella macro
caricata sono disponibili
per essere usate

`.x` Carica la macro ed
esegue la funzione principale

Senza il `+` alla fine del nome della macro, la macro viene interpretata.

Esempio: fit gaussiano

```
{
  TH2F *hpxpy = new TH2F("hpxpy", "py vs px", 40, -4, 4, 40, -4, 4);
  TH1F *hpx = new TH1F("hpx", "px", 40, -4, 4);
  TH1F *hpy = new TH1F("hpy", "py", 40, -4, 4);

  Double_t px, py;

  for (Int_t i=0; i<50000; i++){
    gRandom->Rannor(px, py);
    hpxpy->Fill(px, py);
    hpx->Fill(px);
    hpy->Fill(py);
  }

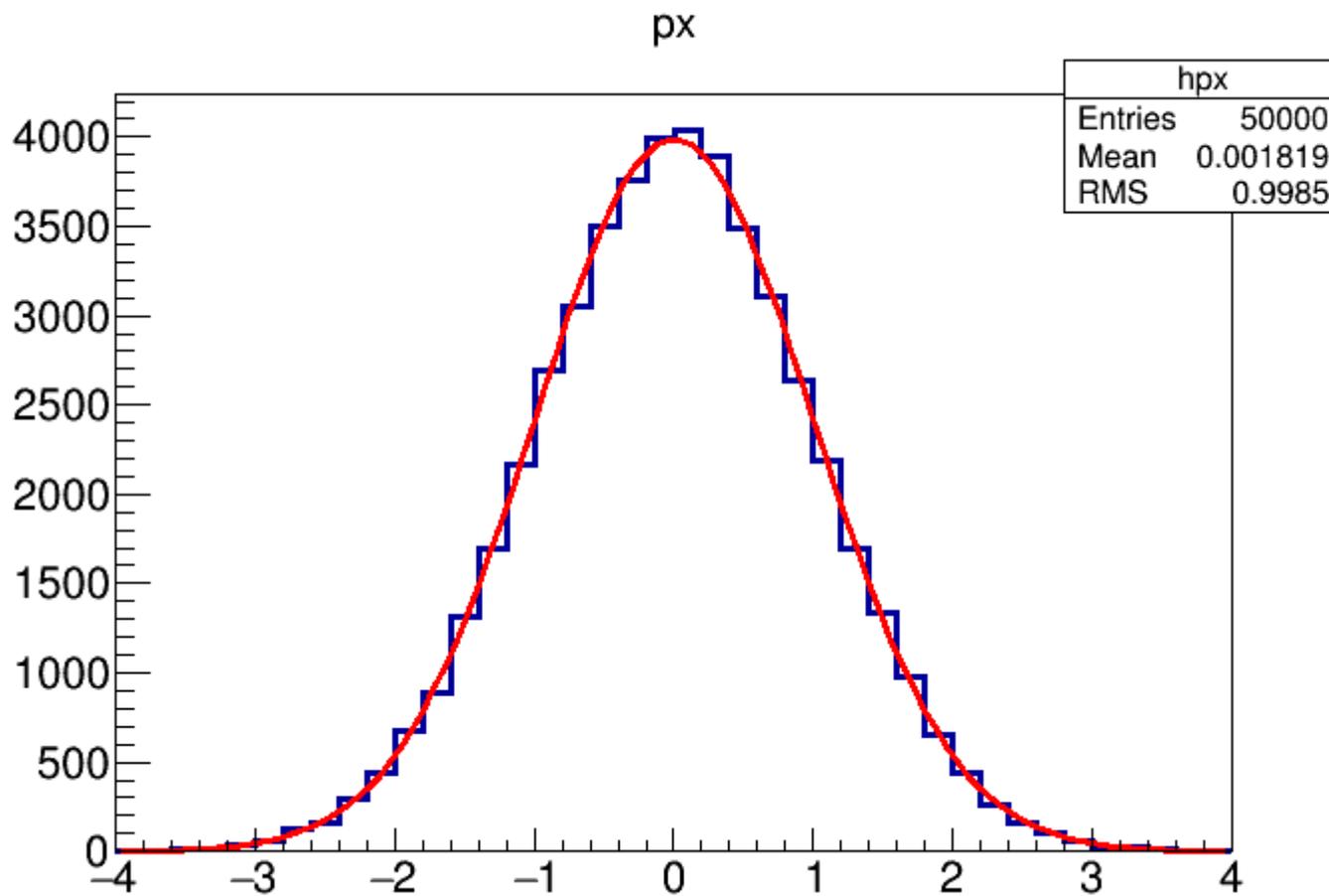
  hpxpy->Draw("col");
}
```

- 1) Apri il terminale
- 2) edita la macro
- 3) esegui la macro

Esempio: fit gaussiano

4) `hpx->Draw()`

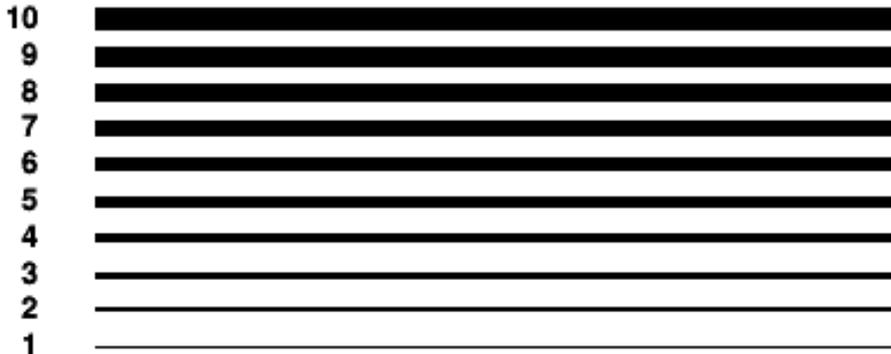
5) esegui un fit gaussiano tramite l'interfaccia "fit panel"



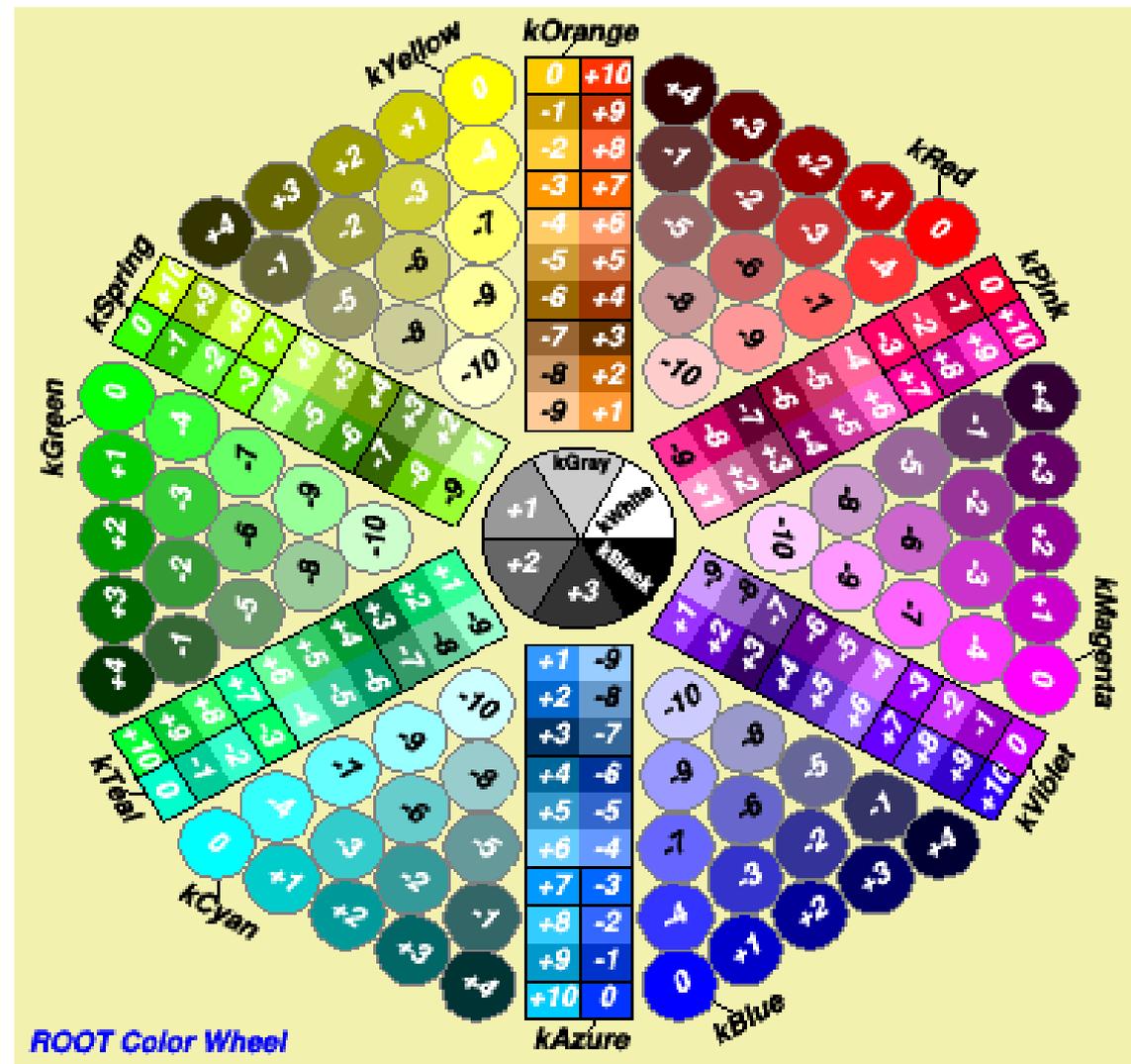
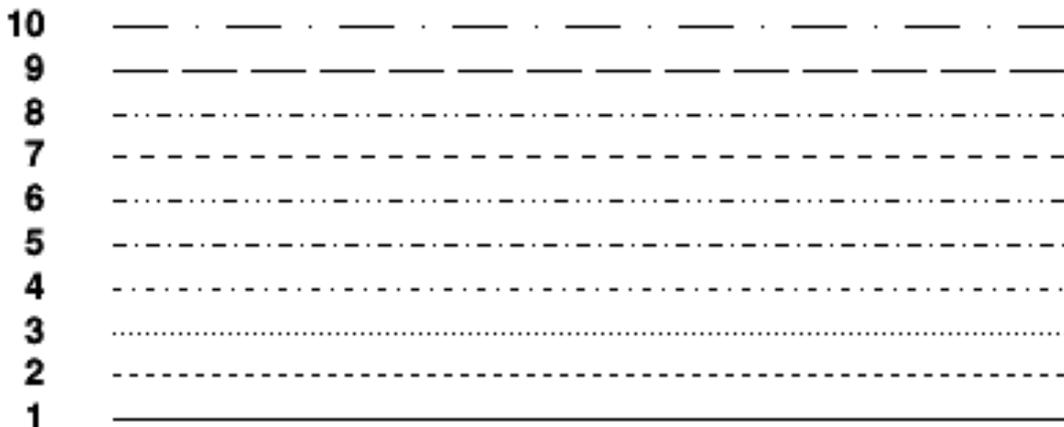
Un po' di cosmesi

- Line**

```
h->SetLineWidth(4);
```



```
h->SetLineStyle(3);
```



```
h->SetLineColor(kMagenta+2);
```

```
h->SetFillColor(kBlue);
```

Un po' di cosmesi

- **Marker**

```
h->SetMarkerSize(2);
```

```
h->SetMarkerColor(kBlue-1);
```

```
h->SetMarkerStyle(kStar);
```

Marker number	Marker shape	Marker name
1	dot	kDot
2	+	kPlus
3	*	kStar
4	o	kCircle
5	x	kMultiply
6	small dot	kFullDotSmall
7	medium dot	kFullDotMedium
8	large scalable dot	kFullDotLarge
9 -->19	large scalable dot	
20	full circle	kFullCircle
21	full square	kFullSquare
22	full triangle up	kFullTriangleUp
23	full triangle down	kFullTriangleDown
24	open circle	kOpenCircle
25	open square	kOpenSquare
26	open triangle up	kOpenTriangleUp
27	open diamond	kOpenDiamond
28	open cross	kOpenCross
29	full star	kFullStar
30	open star	kOpenStar
31	*	
32	open triangle down	kOpenTriangleDown
33	full diamond	kFullDiamond
34	full cross	kFullCross

Drawing options (Histogram)

Si possono concatenare più opzioni con o senza spazi

es: `h->Draw("same e1 p");` oppure `h->Draw("samee1p");`

Opzioni (solo alcuni esempi):

"E"	disegna le barre d'errore
"SAME"	sovrappone l'istogramma all'immagine precedente
"B"	bar chart
"E1"	disegna le barre d'errore con delle linee perpendicolari ai limiti
"E2"	disegna le barre d'errore con dei rettangoli
"X0"	quando usato con una delle "E", sopprime l'errore lungo X
"P"	disegna un marker in ogni bin

Esempio

Fit di un graph con una funzione esterna.

```
Double_t fitf(Double_t *v, Double_t *par){
    Double_t fitval = par[0]+par[1]*TMath::Sin(par[2]*v[0]);
    return fitval;
}

void myfit(){
    float x[4]={0,3,6,9};
    float y[4]={2.5,5.1,2.0,1.2};
    float errx[4]={1,1,1,1};
    float erry[4]={0.3,0.3,0.7,0.9};

    TGraphErrors *gr3 = new TGraphErrors(4,x,y,errx,erry);
    TF1 *func = new TF1("fit",fitf,0,25,3);
    func->SetParameters(3,2,0.6);
    func->SetParNames("par0","par1","par2");

    gr3->SetMarkerStyle(21);
    gr3->Draw("AP");
    gr3->Fit("fit");
}
```

`TGraph gr(n,x,y); //senza errori`

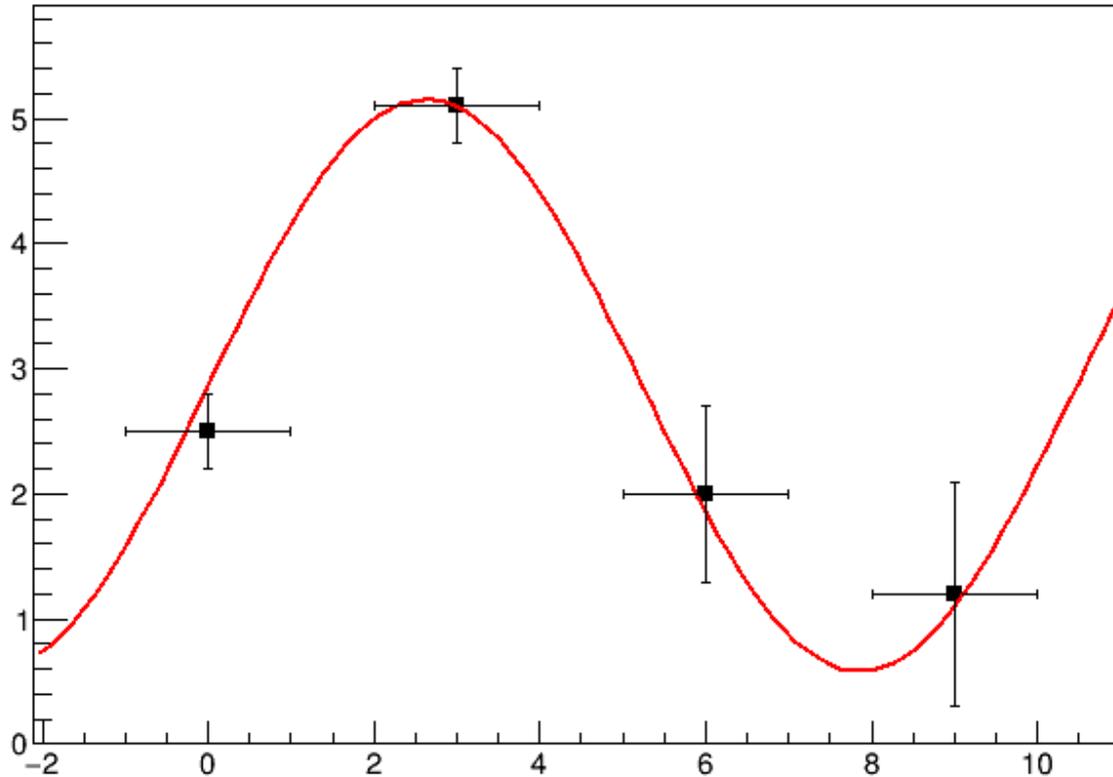
`TGraphErrors gr(n,x,y,errx,erry) //con gli errori`

dove x e y sono array di n elementi (così come errx e erry, gli errori associati) 52

Esempio

Fit di un graph con una funzione esterna.

Graph



Drawing options (Graph):

- "*" disegna un asterisco in ogni punto
- "P" disegna il marker in ogni punto
- "L" disegna una linea che unisce i punti

Per sovrapporre due TGraph (o due TgraphErrors), il secondo non deve avere l'opzione "A". 53

Esempio

Il seguente esempio mostra come leggere dati da un file, metterli in un istogramma e salvare l'istogramma in un file root.

```
void basic2() {
//   example of macro to create can ntuple reading data from an ascii file.
//   This macro is a variant of basic.C
//Author: Rene Brun

    TString dir = gSystem->UnixPathName(__FILE__);
    dir.ReplaceAll("basic2.C", "");
    dir.ReplaceAll("./", "/");

    TFile *f = new TFile("basic2.root", "RECREATE");
    TH1F *h1 = new TH1F("h1", "x distribution", 100, -4, 4);
    TTree *T = new TTree("ntuple", "data from ascii file");
    Long64_t nlines = T->ReadFile(Form("%sbasic.dat", dir.Data()), "x:y:z");
    printf(" found %lld points\n", nlines);
    T->Draw("x", "z>2");
    T->Write();
}
```

sorgente: <https://root.cern.ch/root/html/tutorials/tree/basic2.C.html>

dati: <http://www.cern.ch/iosys/basic.txt>

Draw a simple graph

<http://root.cern.ch/root/html/tutorials/graphs/graph.C.html>

Draw two graphs with error bars

<https://root.cern.ch/root/html/tutorials/graphs/gerrors2.C.html>

Draw 2D function

<http://root.cern.ch/root/html/tutorials/graphs/surfaces.C.html>

Fill a 1D histogram from a parametric function

<https://root.cern.ch/root/html/tutorials/hist/fillrandom.C.html>

Simple fitting example

<http://root.cern.ch/root/html/tutorials/fit/fit1.C.html>

Questo è solo un po' più complesso!

<https://root.cern.ch/root/html/tutorials/geom/station1.C.html>

Esercizio

- 1) Crea un graph con 5 punti
- 2) I punti da inserire sono (1.0, 2.1), (2.0, 2.9), (3.0, 4.05), (4.0, 5.2), (5.0, 5.95)
- 3) Metti gli errori di x a 0.0 e gli errori di y a 0.1
- 4) Disegna il graph (con assi e barre d'errore)
- 5) Crea una funzione 1dim $f(x)=mx+b$ e usala per fare un fit dei punti
- 6) Ottieni i parametri m e b dalla funzione e le loro incertezze

Graph

