



Corso di Laboratorio 2 – Primo semestre

Silvia Arcelli, Luca Pasquini

29 Settembre 2014

Struttura del Corso- I Semestre

- Un Modulo di programmazione C++ (S.Arcelli):
 - Lezioni Frontali/Esercitazioni (22 h)
 - Laboratorio (12 h, 4 turni da 3 h)
- Un modulo di programmazione LabVIEW (L.Pasquini)
 - Lezioni Frontali/Esercitazioni (14 h)
 - Laboratorio (8 h, 2 turni da 4 h)

Struttura del Corso- I Semestre

Calendario delle lezioni:

- **Modulo di programmazione C++** (S.Arcelli):
 - Fino alla prima metà di novembre
- **Modulo di programmazione LabVIEW** (L.Pasquini)
 - Seconda metà di novembre, prima metà di dicembre

Struttura del Corso- I Semestre

Programma Modulo Programmazione C++/ROOT

- **Linguaggio C++** : richiami alla di sintassi di base C++; programmazione a oggetti – il concetto e la struttura di una classe, metodi e attributi, incapsulamento e meccanismo di protezione dei membri di una classe. Reimpiego di codice: aggregazione ed ereditarietà. Polimorfismo. I Template - La C++ Standard Template Library. Esempi di utilizzo di Container e di Algoritmi STL.
- **Metodi Monte Carlo**: Generatori di numeri pseudocasuali e semplici applicazioni per la generazione di distribuzioni fisiche (uniforme, esponenziale, gaussiana...) con metodo del rigetto e della trasformazione inversa
- **Il framework ROOT** per l'analisi dati: l'interfaccia utente (linea di comando, GUI, macros), istogrammi, funzioni e fit, uso dei Trees

Struttura del Corso- I Semestre

Esercitazioni in Laboratorio – C++:

- Realizzazione di un programma di generazione di particelle elementari in eventi fisici, utilizzando la programmazione ad oggetti, le tecniche di base di simulazione Monte Carlo, e le funzioni di analisi dati di ROOT
- Lo studente dovrà completare il programma, che sarà discusso in sede d'esame

Struttura del Corso- I Semestre

Quali requisiti per il computer che utilizzerete? Modulo C++:

- un portatile di fascia media ($\geq 2\text{GB RAM}$)
- Requisiti di Sistema Operativo: Vanno bene Windows , Mac-OX , ogni declinazione di Unix/Linux (Ubuntu, Suse,...) (tenete presente che il sistema operativo di riferimento in ambito fisico, per la sua diffusione, è Linux)
- Compilatori integrati negli IDE (Visual Studio, Xcode), GCC di GNU/Linux
- Tuttavia, per l'installazione ottimale di ROOT è vivamente consigliato, se avete un sistema operativo Windows, di installarsi una macchina virtuale con sistema operativo di tipo Linux
 - Virtual Box
 - Sistemi operativi host: Ubuntu, Scientific Linux,....

Struttura del Corso- I Semestre

Esercitazioni in Laboratorio – MODULO C++ / ROOT

- Disponibilità in Laboratorio: Una ventina di PC Windows con una macchina virtuale con Scientific Linux e Root installati
- Sulla base dell'esperienza degli scorsi anni, la maggioranza degli studenti lavora sul proprio portatile
- “Censimento” di chi intende utilizzare i PC in Laboratorio

Struttura del Corso- I Semestre

Programma Modulo LabVIEW

- Dispositivi per l'acquisizione e generazione di segnali di tensione: architettura hardware e modalità di comunicazione con il computer. Connessione fisica di sensori/attuatori al dispositivo. Campionamento di segnali: aliasing e teorema di Nyquist. Libreria software DAQ (DATA acQUISITION) in LabVIEW. Generazione di segnali. Tempistiche di acquisizione e generazione: single-point, bufferizzata finita, bufferizzata continua. Il trigger: acquisizione e generazione con trigger analogico e digitale. Scrittura e lettura di linee digitali. Cenni alla gestione dei contatori.

Struttura del Corso- I Semestre

Esercitazioni in Laboratorio – LabVIEW:

- I prova: Misure su vari circuiti resistivi per la verifica delle relazioni tensione-corrente (legge di Ohm) su resistenze in serie e in parallelo
- Il prova: Misure di transitori di carica e scarica di condensatori. Generazione di onde e acquisizione con trigger.
- Lo studente dovrà scrivere il codice LabVIEW per lo svolgimento delle esperienze. All'esame orale, dovrà discutere una relazione da lui preparata sulle esperienze svolte, **comprensiva del diagramma a blocchi di suddetto codice.**

Struttura del Corso- I Semestre

Esercitazioni in Laboratorio – MODULO LABVIEW

- Nel Laboratorio di Fisica 2 (Berti-Pichat) sono disponibili 18 PC portatili e 18 Schede ELVIS II. LabVIEW 2011 e driver di acquisizione dati pre-installati.
- E' possibile lavorare sul proprio portatile, a patto di aver già installato correttamente i driver ELVIS
- Si può installare sul PC del laboratorio un programma preparato a casa: per evitare problemi di compatibilità, **salvare il codice per la versione LabVIEW 2011**

Struttura del Corso- I Semestre

Quali requisiti per il computer che utilizzerete? Modulo LabVIEW

- un portatile di fascia media (≥ 2 GB RAM)
- LabVIEW, versione dal 2011 in avanti (per installazione vedi sotto)
- Requisiti di Sistema Operativo: per l'installazione gratuita di LabVIEW e dei driver di acquisizione dati (che dovrete aver già fatto l'anno scorso) è necessario sistema operativo Windows. In caso si utilizzi Mac o Linux, è necessario installare una macchina virtuale Windows.
- Le istruzioni per l'installazione sono reperibili su campus. Docente: Luca Pasquini; corso: Laboratorio di Fisica 1

Struttura del Corso- I Semestre

IN SINTESI:

- se avete Windows, installate una macchina virtuale Linux;
- se avete Linux/Mac, installate una macchina virtuale Windows;

Struttura del Corso- I Semestre

Turni di Laboratorio, modulo di programmazione C++:

I prova: 4, 5, 6 Novembre 2014 ore 9-12

II prova: 11, 12, 13 Novembre 2014 ore 9-12

III prova: 18, 19, 20 Novembre 2014 ore 9-12

IV prova: 25, 26, 27 Novembre 2014 ore 9-12

Turni di Laboratorio, modulo LabVIEW:

I prova: 9, 10, 11 Dicembre 2014 ore 9-13

II prova: 16, 17, 18 Dicembre 2014 ore 9-13

Struttura del Corso- I Semestre

La frequenza dei laboratori è **obbligatoria**:

- (4/4) per il modulo di C++/Root
- (2/2) per il modulo di LabVIEW
- E' previsto un **turno di recupero** per entrambi i moduli ,che si terrà ai primi di gennaio (durante la settimana di lezione prima dell' inizio della sessione straordinaria di esami)

Struttura del Corso- I Semestre

Modalità d'esame:

Verifica dell'apprendimento degli argomenti di base del corso (C++, STL, ROOT, LabVIEW) con un test di 10 domande a risposta multipla. Inoltre,

- Modulo Programmazione C++:
 - Discussione del programma svolto in laboratorio
 - Discussione di un programma a scelta libera elaborato dallo studente contenente gli argomenti trattati al corso (programmazione ad oggetti, uso di ROOT)
- Modulo LabVIEW:
 - Discussione della relazione sulle esperienze di laboratorio, incluso codice LabVIEW sviluppato dallo studente, durante la quale verrà verificata la conoscenza delle tematiche esposte a lezione.

Struttura del Corso- I Semestre

- Sarà possibile sostenere l'esame già a partire dalla sessione straordinaria di gennaio/febbraio 2015 :
- L'esame per la prima parte di corso dovrà essere sostenuto **prima** di dare l'esame relativa alla seconda parte (con il prof. Boscherini)
- Verrà dato un **voto unico** per i due moduli del primo semestre di insegnamento, e questo voto sarà trasmesso al prof. Boscherini. Il voto finale sarà registrato solo **dopo** aver sostenuto l'esame per la parte di corso al secondo semestre con il prof. Boscherini.

Struttura del Corso- I Semestre

Materiale didattico, modulo di programmazione C++/ROOT:

- Slides disponibili al link:
<http://www.bo.infn.it/~arcelli/LezioniLabII.html>
- Altro materiale legato ai contenuti del corso sempre alla stessa pagina
- Testi consigliati: ad esempio “C++ Corso di programmazione” di Lippman, o “Navigating C++” di Anderson ma anche altri vanno benissimo
- Moltissime risorse disponibili online, ad esempio:
- <http://www.cplusplus.com> per il C++/STL
- <http://root.cern.ch> per il pacchetto ROOT

Struttura del Corso- I Semestre

Materiale didattico, modulo di programmazione LabVIEW:

- Slides disponibili su campus (scaricarle poco prima dell'inizio del ciclo di lezioni, saranno aggiornate) cercando per docente=Luca Pasquini e materia d'esame=Laboratorio di Fisica 2.
- Risorse on-line (tutorials, programmi di esempio) sul sito National Instruments (www.ni.com)
- Programmi di esempio nell'ambiente LabVIEW

Struttura del Corso- I Semestre

Contatti, modulo di programmazione C++/ROOT:

- Silvia.arcelli@bo.infn.it, Francesco.noferini@bo.infn.it
- Ricevimento venerdì ore 14, previo appuntamento via e-mail

Contatti, modulo di programmazione LabVIEW:

- luca.pasquini@unibo.it
- Ricevimento lunedì ore 14, previo appuntamento via e-mail

Programmazione C++

- Programmazione in Fisica :
 - Fisica Teorica: Calcoli perturbativi in QFT, Manipolazione Tensoriale in GR, Calcoli ab-initio nella fisica dei solidi
 - Fisica sperimentale: Acquisizione, gestione dati e loro analisi (Enorme mole di dati (GB/s) dai rivelatori a LHC); operazione e controllo di strumentazione complessa e integrata (beamline di luce di sincrotrone ad ELETTRA); simulazioni di Monte Carlo
- La natura della ricerca richiede nella maggior parte dei casi l'utilizzo di strumenti software non commerciali, ma "custom made"
- Non necessariamente un fisico deve diventare un **programmatore esperto**, ma - in generale- i fisici sono quanto meno dei **programmatori competenti** (sanno interpretare, progettare, scrivere del codice per realizzare i loro obiettivi di ricerca)
- All'esterno del percorso accademico, saper programmare è una competenza fondamentale nel mondo del Lavoro.

Programmazione C++-il linguaggio

- nel 1980 il ricercatore informatico danese **Bjarne Stroustrup**, produsse una versione modificata del C, che chiamò: "**C con le classi**". Il nuovo linguaggio univa la potenza e l'efficienza del C con la novità concettuale della programmazione a oggetti, allora ancora in stato embrionale(ereditarietà, ...)
- Il nome C++ fu introdotto per la prima volta nel 1983, per suggerire la sua natura di avanzamento dal C, nel quale ++ è l'operatore di incremento.
- In seguito alla diffusione sempre più ampia del linguaggio nel 1990 si formò un comitato per la standardizzazione del C++, che avvenne alla fine del **1998**. In questi ultimi anni il C++ si è ulteriormente evoluto(ultimo standard nel **2011**)
- Estremamente diffuso anche nell'ambito della ricerca , che ora vede progetti di sempre più grandi dimensioni e a cui partecipano molti ricercatori (ambiente di sviluppo "complesso")

Sistema Operativo-Linux

Per le loro caratteristiche, i sistema operativi di tipo Unix sono i più diffusi in ambito universitario e della ricerca: In particolare, sistemi operativi *open source* di tipo GNU/Linux:

- Il progetto GNU fu lanciato nel 1984 da Richard. Cosa è GNU/Linux?

Gnu **N**on è **U**nix!

"GNU, che sta per "Gnu's Not Unix" (Gnu Non è Unix), è il nome del sistema software completo e Unix-compatibile che sto scrivendo per distribuirlo liberamente a chiunque lo possa utilizzare."

[Richard Stallman, Dal manifesto GNU, <http://www.gnu.org/gnu/manifesto.html>]

Sistema Operativo-Linux

- Per GNU/Linux e' disponibile un compilatore C/C++ nella suite **gcc**, che gestisce diversi linguaggi di programmazione: ,
- comandi **gcc** e **g++** (gcc=GNU Compiler Collection <http://www.gnu.org/onlinedocs/gcc>). Quale versione di gcc sto usando?

gcc -v

gcc version 3.4.6 20060404(Red Hat 3.4 6-11)

Il Compilatore gcc

Sia gcc che g++ processano file di input attraverso i seguenti passi:

- 1) **preprocessing**
 - rimozione dei commenti
 - interpretazioni di direttive per il preprocessore denotate da "#" come: #include, #define ,...
- 2) **compilation**
 - traduzione del codice sorgente in codice assembly
- 3) **assembly**
 - creazione del codice oggetto
- 4) **linking**
 - combinazione delle funzioni definite in altri file sorgenti o definite in librerie con la funzione main() per creare il file eseguibile.

gcc-File Extensions

Estensioni dei files:

Alcuni suffissi di moduli presenti nel processo di compilazione (per i sorgenti, guardare nel manuale da riga di comando `-man gcc-` quali sono i suffissi permessi):

- .c** modulo sorgente C; da preprocessare, compilare e assemblare
- .cc, .cpp, .cxx, .c++, .C** modulo sorgente C++; da preprocessare, compilare e assemblare
- .h, .hh, .H** modulo per il preprocessore; di solito non nominato nella riga di comando
- .s** modulo assembly
- .o** modulo oggetto; da passare linker che produce l'eseguibile

Inoltre:

- .a** librerie statiche
- .so** librerie dinamiche

gcc-Compilazione

- Con g++ (gcc) tutte queste fasi sono esplicate con un **singolo comando**:

g++ test.cpp

Compila, linka e produce l'eseguibile a.out

- **Passaggi intermedi di compilazione:**

Per compilare senza effettuare il link :

g++ -c test.cpp  **test.o** , file oggetto

Per effettuare il link riferirsi al file oggetto:

g++ -o test test.o

gcc-Compilazione

- Per conferire **un nome specifico all'eseguibile**;

g++ -o test test.cpp

- per mandare in esecuzione il programma è sufficiente invocarne il nome da linea di comando:

./test

N.B. Occorre anteporre il path della directory corrente, **./**, oppure assicurarsi che **./** sia presente nella variabile d'ambiente **\$PATH**,

> echo \$PATH

> PATH=./:\$PATH

Sviluppo di codice- Modularità

Modularità nello sviluppare programmi:

- Un programma modulare è un programma diviso in componenti piu' piccole con funzioni specifiche. La programmazione modulare è:
 - piu' facile da comprendere per l'utente
 - facilita lo sviluppo in un ambiente con molti programmatori
 - è più semplice da debuggare
- La programmazione ad oggetti ben si adatta a questo approccio

Buone prassi di “ingegneria del software”:

- **Separare** il codice in file di **intestazione** (.h) e file di **implementazione** (.cpp)
- Organizzare il codice in moduli concettualmente indipendenti

File di Intestazione

- I file di intestazione (header file) sono file al cui interno vengono posti **prototipi di funzioni**, definizione di tipi, costanti simboliche
- Sono pensati come strumento atto a separare **l'interfaccia** di un insieme di funzioni dal codice che ne fornisce **l'implementazione**.
- Il loro uso è associato alla direttiva al preprocessore `#include`, il cui effetto è quello di modificare il programma sorgente sostituendo la linea di codice dove appare questa direttiva con il contenuto del file specificato nella direttiva.
- `#include <nomefile.h>` (ricerca in `/usr/include` e `/usr/local/include`)
- `#include "nomefile.h"` (ricerca nella directory corrente)

Per aggiungere un percorso alla ricerca degli header files, in compilazione usare l'opzione **-I*dir***

File di Intestazione

- Separare il codice in file di intestazione e di implementazione è conveniente perchè ogni modifica **nell'implementazione** del codice contenuto in un modulo, se non cambia l'interfaccia, non necessita la ricompilazione degli altri moduli che compongono il programma ed utilizzano quel codice.
- Per evitare inclusioni multiple di uno stesso file, abituarsi ad usare le direttive **#ifndef** -**#define** in filename.h:

```
#ifndef FILENAME_H
#define FILENAME_H
//...codice del file di intestazione....//
#endif
```

Compilazione di Moduli

Compilazione di un programma modulare:

main.cpp,

myfunc.h ,

myfunc.cpp .

```
#include<iostream>
#include"myfunc.h"
using namespace std;
int main() {
    int a=6, b=3;
    cout<<"a/b="<<ratio(a,b)<
    <endl;
    return 0; }
```

```
#ifndef MYFUNC_H
#define MYFUNC_H
int ratio(int a, int b);
int product(int a, int b);
#endif
```

```
int ratio(int a, int b)
{    return a/b; }
int product(int a, int b)
{    return a*b; }
```

- Per compilare: **g++ -Wall -o test main.cpp myfunc.cpp**
- il file **myfunc.h** **non appare** nella riga di comando, incluso da gcc in fase di precompilazione.

Compilazione-Warnings e Errors

Il compilatore invia messaggi all'utente:

- **warning** : il compilatore rileva delle condizioni “sospette” che potrebbero dare luogo a problemi (variabili non inizializzate,..)- la compilazione va comunque avanti
- **errors**: condizioni di errore di sintassi, per cui la compilazione si interrompe

Controllo del livello di warning :

Per inibire tutti i messaggi usare l' opzione **-w**

g++ -w -o test test.cpp

Per usare il massimo livello di warning usare l' opzione **-Wall**

g++ -Wall -o test test.cpp :

Compilazione “pignola” (conformità con standard ISO del linguaggio)

g++ -pedantic -o test test.cpp :

Suggerimento: usate il massimo livello di warning

Opzioni di Ottimizzazione del Codice

- Il compilatore gcc gestisce il livello di ottimizzazione del codice attraverso l'opzione -O:
 - -O1 Ottimizzazione minima
 - -O2 Ottimizzazione media
 - -O3 Ottimizzazione massima
 - -O0 Nessuna ottimizzazione

Livello	Tempo di esecuzione (s)
O0	36.9
O1	8.4
O2	8.4
O3	8.4

```
int main() {  
    int a=10, b=1, c;  
    for (int i=0; i<1e9; i++)  
        c=i+a*b-a/b;  
    return 0;  
}
```

Si consiglia di ottimizzare il codice (se non si deve debuggare).