UNIVERSITA' DEGLI STUDI ROMA TRE

DIPARTIMENTO DI FISICA "E. AMALDI"

Laboratorio di Programmazione e Calcolo

6 crediti

a cura di

Severino Bussino

Anno Accademico 2021-22

Docenti del Corso

Severino Bussino

Dipartimento di Matematica e Fisica
Stanza 67 - tel. (06 5733) 7285

sbussino@uniroma3.it

Alessandro Di Cicco

Dipartimento di Matematica e Fisica

alessandro.dicicco@roma3.infn.it

Daniele Paoloni

Dipartimento di Matematica e Fisica

daniele.paoloni@uniroma3.it

Pagina Web del Corso

http://webusers.fis.uniroma3.it/~bussino/Lab_Prog_Calc.html

Le diapositive delle lezioni verranno aggiornate durante lo svolgimento del Corso

Programma del Corso (1)

Elementi di programmazione ad oggetti (linguaggio **C++**)
in ambiente Linux

---

circa 20 ore di lezione frontale e 30 ore di laboratorio

Materiale Didattico per il Corso (1)

Voi stessi durante le esercitazioni in Laboratorio

"... Sofia ricordava bene quelle volte in cui sua madre o i professori avevano cercato di insegnarle qualcosa per cui lei non nutriva il benché minimo interesse. Invece, quando si era impegnata in prima persona, allora sì che aveva veramente «imparato»! A volte succedeva che di colpo capisse qualcosa, ed era proprio questo che veniva chiamato «sapere»."

(Jostein Gaarder, "Il mondo di Sofia")

Materiale Didattico per il Corso (2)

Trasparenze delle Lezioni

(disponibili sul sito web)

Lippman, Lajoie, Moo C++ primer

Addison Wesley 2012 (5ª ed.)

Anche in italiano (3ª ed. - senza Moo tra gli autori)

Barone, Marinari, Organtini, Ricci-Tersenghi

Programmazione Scientifica

Pearson Education 2006

Valutazione del Corso

- Il voto del Corso si puo' ottenere con una valutazione **"in itinere"** (come descritto in seguito) oppure superando un esame a febbraio o a giugno (per chi non ha raggiunto la sufficienza, per chi non e' soddisfatto del voto ottenuto, per chi e' stato costretto a numerose assenze, ecc.)

- Chi vuole puo' superare l'intero corso (6 crediti) con un unico esame a partire da febbraio (una prova pratica ed una prova orale)

0) Struttura del Corso

Introduzione alla programmazione ad oggetti

Elementi di programmazione in linguaggio **C++**

Alcuni esempi

1. ATLAS (LHC al Cern)
   1.000.000 canali di lettura
   bunch-crossing ogni 25 ns
   40 Tbyte/sec 100 Mb/sec

   Cluster di circa 1000 CPU (PC)

   Software specifico per il trigger di III livello

Alcuni esempi (segue)

2. ARGO-YBJ (Raggi Cosmici –Tibet)

   circa 1 Tbyte/mese di dati da memorizzare

   dati disponibili per l'analisi

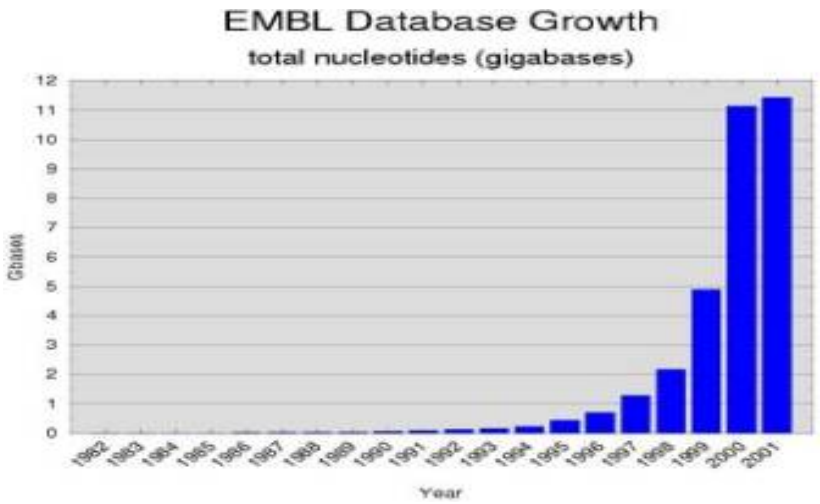
   Cluster di PC per la gestione dello spazio disco

   Software dedicato per l'accesso ai dati e l'analisi

3. GRID (Calcolo su rete geografica)

Alcuni esempi (segue)

4. Progetto genoma e studi di Biologia molecolare (*www.embnet.org*)



Alcuni esempi (segue)

Altre applicazioni:

1. Telefonini cellulari
2. Bancomat
3. Reti di Trasporto
   (ferrovie, metropolitane, …)

Scopo del corso (I)

- Familiarizzazione con l'uso del personal computer in ambiente Unix (useremo il sistema **Linux**)

- Acquisizione dei principi di base per la programmazione in linguaggio C ++

- Acquisizione dei principali elementi sintattici per la programmazione in linguaggio C (C++)

Scopo del corso (II)

"Idee" e non aspetti tecnici

= "Architetti" e non
"Esperti del Linguaggio C/C++"

Insegnare a progettare un approccio al Calcolo

Programma delle Esercitazioni di Laboratorio

Le Esercitazioni inizieranno il giorno 28 settembre per il gr. 1
il giorno 29 settembre per il gr. 2
il giorno 30 settembre per il gr. 3

Laboratorio di Calcolo

Via della Vasca Navale, 84

- Introduzione (1 esercitazione)
- Programmazione: linguaggio *C* e *C++* (8 esercitazioni)
- (Simulazione della) Esercitazione individuale (a dicembre)

- 1 Esercitazione individuale (a gennaio)

Organizzazione delle Esercitazioni di Laboratorio (1)

- ~~Gruppi di 2 persone (la stessa coppia per tutto il corso)~~
  Individuali

- Tre sessioni di laboratorio
  - Martedi' 14.30 – 17.30 Gruppo 1
  - Mercoledi' 14.30 – 17.30 Gruppo 2
  - Giovedi' 14.30 – 17.30 Gruppo 3

- Scegliere il giorno del laboratorio e mantenere la scelta per tutto il corso (**eccezioni** sono possibili per la coppia, compatibilmente con lo spazio disponibile in laboratorio, avvisando con anticipo)

Organizzazione delle Esercitazioni di Laboratorio (2)

- Variazioni al Calendario usuale

  il 9-10-11 novembre non ci saranno le Esercitazioni
  (settimana degli Esoneri)

  il 7 e 9 dicembre non ci saranno le Esercitazioni
  (l'8 dicembre e' festa)

- 1 esercitazione individuale (senza valutazione - simulazione)

  14 (gruppo 1) - 15 (gruppo 1) - 16 dicembre (gruppi 3)

- 1 esercitazione individuale (con valutazione)

  11 gennaio - 12 gennaio - 13 gennaio

Orario e Programma dettagliato delle lezioni e delle esercitazioni sulla pagina web del corso

http://webusers.fis.uniroma3.it/~bussino/Lab_Prog_Calc.html

Ultima lezione in Aula :

(presumibilmente) venerdi' 3 dicembre

Valutazione del Corso

Il voto in trentesimi viene attribuito principalmente in base al lavoro svolto durante le esercitazioni:

- Le **prime 3** (?) esercitazioni **non** vengono valutate.
- Le altre **5 esercitazioni (?)** valgono **4** punti e si considerano solo le 4 migliori (**totale 16 punti**). Il voto è individuale anche se il programma viene scritto in gruppi di due.
- L'esercitazione individuale **vale 10 punti**

  11 gennaio - 12 gennaio - 13 gennaio

- I rimanenti **4 punti** vengono assegnati in base ad un **test** sugli argomenti trattati nelle lezioni.
  Il **test** si svolgerà presumibilmente il **12 novembre** in Aula M2 ore 10:00 - 12:00
  (data, ora e aula **da confermare**)

- La partecipazione alle esercitazioni è **obbligatoria**.
- Gli **studenti lavoratori** che avessero problemi di orario sono pregati di farlo presente il prima possibile, per concordare una soluzione.
- Gli esercitatori valutano i programmi nelle condizioni in cui si trovano dopo le 3 ore di esercitazione.
- Gli studenti che **non** avranno ottenuto la sufficienza con questo tipo di valutazione potranno sostenere l'**esame in modo tradizionale**, con una prova pratica (realizzazione di un programma) ed una prova orale.

0) Struttura del Corso

1) Trattamento dell'informazione
Elementi di Architettura di un Computer

2) Sistemi operativi

1) Trattamento dell'informazione

Elementi di Architettura di un Computer

Breve introduzione autoconsistente

Verra' trattata in una delle prossime lezioni

0) Struttura del Corso

1) Trattamento dell'informazione

Elementi di Architettura di un Computer

Verra' trattata in una delle prossime lezioni

2) Sistemi operativi

Sistemi operativi

- Storicamente ogni costruttore di calcolatori ha scritto un proprio sistema operativo (*firmware*) che veniva distribuito insieme alla macchina.

- Con l'uniformarsi delle architetture e con l'avvento dei Personal Computer, la varietà di sistemi operativi è andata riducendosi.

- Attualmente Windows è il sistema largamente più diffuso, mentre Unix è quello più utilizzato negli ambienti scientifici.

Windows

- È stato introdotto a fini commerciali ed è caratterizzato da una particolare semplificazione dell'interfaccia utente.

- Per la qualità e quantità di software sviluppato in questo ambiente è senza dubbio il sistema più idoneo per l'utilizzo di applicazioni quali word processing, uso di database, indirizzamento di periferiche standard, scrittura di lucidi per presentazioni e lezioni …

- Non è ottimizzato per l'uso in rete e non supporta più di un utente: WindowsNT e Windows2000 sono nati per queste nuove funzioni (e sono poi evoluti in Windows XP, Windows 7, Windows 8 e 10)

Unix

- Sviluppato in ambienti scientifici è ottimizzato per usi non comuni e per l'utilizzo in rete dei calcolatori oltre che per l'uso di uno stesso calcolatore da parte di più utenti.

- Esistono dei sistemi Unix *freeware* quali Linux (che a sua volta esiste in varie distribuzioni: RedHat, Slackware, Suse...) o FreeBSD dei quali sono disponibili anche i codici sorgente (*open source*).

- Le varie distribuzioni forniscono delle interfacce utente semplificate, ma non sono standardizzate, anche se GNOME e KDE si stanno affermando come interfacce standard

Il *file-system* di Windows

- Il *file-system* è il sistema che gestisce non solo la collocazione dei files sul disco rigido ma più in generale tutti i dischi e periferiche di un computer.

- Sotto Windows ogni dispositivo hardware è associato ad una diversa unità logica ed è rappresentato nel *pannello di controllo* con un'icona: in particolare due dischi appaiono come due icone distinte.

- La maggior parte di voi ha familiarita' con questo sistema operativo, che non verra' usato nell'ambito di questo corso.

Il file-system di Unix

- Il file-system viene visto come un albero di directory.
- Tutti i dispositivi sono visti nello stesso modo dal sistema operativo, ovvero come file.
- Ogni file può essere montato (comando *mount*) sul file-system e quindi associato ad un particolare punto dell'albero logico.
- La gestione dei file sotto Unix sara' oggetto della prima esercitazione.

Linux

Esistono varie distribuzioni (anche in versione **Live**)
(RedHat, **Centos**, AlmaLinux, SuSe, Ubuntu, Fedora, etc.)

Pagina Web della distribuzione Fedora:

http://fedoraproject.org

Pagina Web della distribuzione AlmaLinux :

http://www.almalinux.org https://almalinux.org

Pagina Web della distribuzione Ubuntu :

http://www.ubuntu.com

Pagina Web della distribuzione CentOS :

http://www.centos.org

Come usare Linux su Windows

- Live Distribution

- Macchina Virtuale (Virtual Box o VMPlayer)

- Code::Block

http://www.codeblocks.org/

codeblocks-17.12mingw-setup.exe e' la versione con g++

http://sourceforge.net/projects/codeblocks/files/Binaries/17.12/Windows/codeblocks-17.12mingw-setup.exe

- Cygwin

https://www.cygwin.com/

0) Struttura del Corso

1) Trattamento dell'informazione

Elementi di Architettura di un Computer

Verra' trattata in una delle prossime lezioni

2) Sistemi operativi

3) Introduzione alla Programmazione ad oggetti (OO)

3) Introduzione alla Programmazione ad oggetti (OO)

Alcuni esempi

1. ATLAS (LHC al Cern)
   1.000.000 canali di lettura
   bunch-crossing ogni 25 ns
   40 Tbyte/sec 100 Mb/sec

   Cluster di circa 1000 CPU (PC)

   Software specifico per il trigger di III livello

Alcuni esempi (segue)

2. ARGO-YBJ (Raggi Cosmici –Tibet)

   circa 1 Tbyte/mese di dati da memorizzare

   dati disponibili per l'analisi

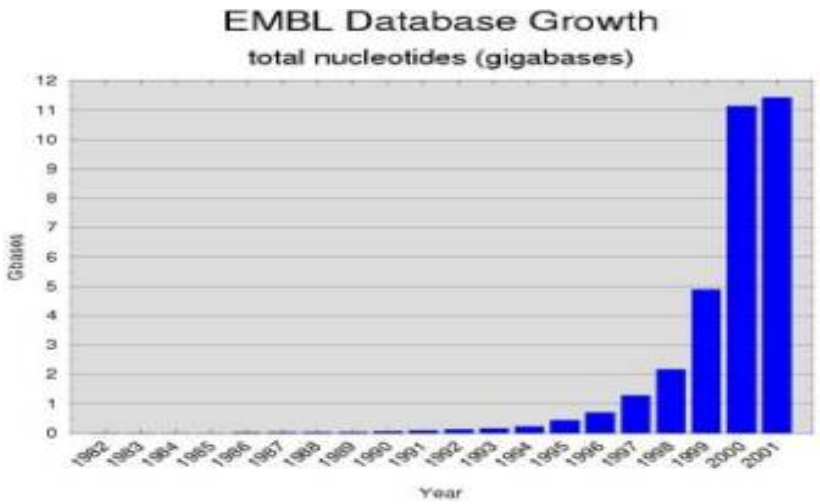
   Cluster di PC per la gestione dello spazio disco

   Software dedicato per l'accesso ai dati e l'analisi

3. GRID (Calcolo su rete geografica)

Alcuni esempi (segue)

4. Progetto genoma e studi di Biologia molecolare (*www.embnet.org*)



Alcuni esempi (segue)

Altre applicazioni:

1. Telefonini cellulari
2. Bancomat
3. Reti di Trasporto
   (ferrovie, metropolitane, …)

Quali richieste alle strategie di programmazione? (1)

1. Robustezza
   protetto nell'accesso ai dati

2. Possibilita' di ri-utilizzo del codice
   1. risorse umane ed economiche
   2. maggiore affidabilita'

Quali richieste alle strategie di programmazione? (2)

3. Portabilita'
    1. sistemi operativi diversi
    2. diverse versioni di uno stesso sistema

    A motivo di architetture complesse
    e di sviluppo nel tempo

4. Flessibilita' e Organizzazione del Codice (Semplicita')

    gestione ed evoluzione del codice

Caratteristiche Fondamentali della Programmazione ad Oggetti (OO) (1)

1. Incapsulamento

   I dati dell'oggetto sono nascosti ad altri oggetti ed è possibile accedervi solo attraverso modalita' ben definite

   → Robustezza Flessibilita'

1. Robustezza
2. Possibilita' di ri-utilizzo del codice
3. Portabilita'
4. Flessibilita' e Organizzazione del Codice (Semplicita')

Caratteristiche Fondamentali della Programmazione ad Oggetti (OO) (2)

2. Ereditarieta’

Gli oggetti complessi possono essere costruiti a partire da oggetti più semplici. Gli oggetti complessi derivano tutto o parte del loro comportamento dagli oggetti a partire dai quali sono stati generati

→ Ri-utilizzo del codice

Organizzazione del codice

1. Robustezza
2. Possibilita’ di ri-utilizzo del codice
3. Portabilita’
4. Flessibilita' e Organizzazione del Codice (Semplicita')

Caratteristiche Fondamentali della Programmazione ad Oggetti (OO) (3)

3. Polimorfismo

   Oggetti *simili* possono essere trattati, in alcuni casi, come se fossero dello stesso tipo, senza la necessità di implementare trattamenti specifici per distinguere tra le varie tipologie

Ereditarieta' piu' efficiente Flessibilita'

La Portabilita' e' una caratteristica del C++ standard (ANSi-C++)

1. Robustezza
2. Possibilita' di ri-utilizzo del codice
3. Portabilita'
4. Flessibilita' e Organizzazione del Codice (Semplicita')

(Altri) Paradigmi di Programmazione (1)

1. Programmazione Procedurale

   Si decide quale algoritmo puo' risolvere il problema dato

   Il linguaggio permette di implementare l'algoritmo scelto

2. Programmazione Modulare

   Si decide quali moduli utilizzare.

   Si separano i dati in modo da nasconderli entro

   ciascun modulo

3. Programmazione definita dall'utente

   Si decide quali tipi utilizzare.

   Si genera un insieme di operazioni per i vari tipi

(Altri) Paradigmi di Programmazione (2)

4. Programmazione Orientata agli Oggetti
   (*Object Oriented Programming*)

   Si decide quali classi utilizzare.

   Si genera un insieme di operazioni per le varie classi

   Si raggruppano le proprieta' comuni tramite
   il meccanismo dell'ereditarieta'

5. Programmazione Generica

   Si identificano una serie di algoritmi e si parametrizzano

   in modo che possano essere utilizzati per piu' classi o tipi definiti dall'utente.

(Altri) Paradigmi di Programmazione (3)

Il C++ permette di implementare una
Programmazione definita dall'utente con ereditarieta'
(= Programmazione Orientata agli Oggetti)
e la Programmazione Generica.

N.B.: attenzione al legame tra linguaggio e
paradigma di programmazione

In sintesi …

1. Esigenze di una moderna programmazione
   - Robustezza
   - Possibilita' di ri-utilizzo del codice
   - Portabilita'
   - Flessibilita' e Organizzazione del Codice (Semplicita')

2. Paradigmi di programmazione
   - Programmazione Procedurale
   - Programmazione Modulare
   - Programmazione definita dall'utente
   - **Programmazione Orientata agli Oggetti** (*Object Oriented Programming*)
   - **Programmazione Generica**

3. Caratteristiche della programmazione ad Oggetti
   - Incapsulamento
   - Ereditarieta'
   - Polimorfismo

Come si progetta un Programma ?

1. Incapsulamento
2. Ereditarieta'
3. Polimorfismo

- La scelta degli Oggetti
- Le relazioni tra gli oggetti
- I casi d'uso (*use case*)
- Le interazioni tra gli oggetti
- La responsabilita' delle classi

Incapsulamento ⟹ Classi

Progettazione di un Programma (1)

1. La scelta degli Oggetti

   Quali "Oggetti" entrano nel problema?

   Quali proprieta' hanno tali oggetti?

2. Le relazioni tra gli oggetti

   Quali sono le relazioni tra gli oggetti?

   Possono essere raggruppati?

   Quali proprieta' sono condivise da oggetti simili?

   Quali proprieta' permettono di distinguere oggetti simili tra loro?

   Quali oggetti sono superflui?

Progettazione di un Programma (2)

3. I casi d'uso (*use case*) e gli scenari

   *In quali situazioni **concrete** tali oggetti dovranno essere utilizzati?*

   Esempio dal Bancomat

   Lo scenario come ulteriore specificazione di un caso d'uso

4. Le interazioni tra gli oggetti

   Non e' sufficiente capire la relazione "geometrica" tra gli oggetti….

   … e' necessario anche valutare come gli oggetti interagiscono "dinamicamente" tra loro

Progettazione di un Programma (3)

5. La responsabilita' delle classi

   E' chiaro di chi e' la responsabilita' di un punto chiave del programma?

   Una classe ha troppe responsabilita'?

   Una classe e' irrilevante e puo' essere "assorbita" da un'altra?

Una sessione CRC

(Classi, Responsabilita', Collaborazione):

capire come funziona dinamicamente il programma che si vuole realizzare

Laboratorio di Programmazione e
Calcolo - AA 2021-22 - lezione 1