Introduzione a

Ingresso e Uscita

Scrittura di un programma C++

Compilazione

Esecuzione

Sistema operativo

- Si possono sviluppare i propri programmi anche con sistemi operativi diversi da Linux durante il corso?
  - Sì, tenendo conto dei due punti seguenti
- Nessun supporto da parte del docente
- La prova di programmazione dovrà essere svolta su una delle macchine delle laboratorio, utilizzando il sistema operativo GNU/Linux installato sulla macchina

File contenenti programmi

- Nel caso più semplice, un programma C/C++ non è altro che un documento di testo, scritto appunto in linguaggio C/C++
- Tra i suffissi tipici dei file contenenti programmi:
  - Linguaggio C → **.c**
  - Linguaggio C++ → **.cc**

Editor di testo

- Di cosa abbiamo bisogno per scrivere programmi in C/C++
- Ci basta un editor di testo (gedit, kedit, kate, emacs, vi, ...)
- Qualsiasi editor va bene
- Non useremo IDE (torneremo su questo argomento a fine corso)

Ingresso e uscita

- Input/Output
  - Ingresso di informazioni (da elaborare) all'interno di un processo
  - Uscita di informazioni (elaborate) da un processo
- Esempio: stampa di informazioni sullo schermo, lettura di valori da tastiera

iostream

- Un programma che deve effettuare *input/output* 'classico' deve contenere le direttive

  #include <iostream>
  using namespace std;

  - Tali direttive devono precedere il primo punto in cui viene effettuato l'input/output

Anatomia programma C++

```
direttive

main()

{

  istruzioni

}
```

Nota sulle parentesi graffe

- Sulle tastiere italiane:

{ Alt + 123 sul tastierino numerico

} Alt + 125 sul tastierino numerico

{ Alt Gr + Shift + è

} Alt Gr + Shift + +

Stampa su terminale

- Sintassi più semplice per stampare una stringa su terminale:
  - *cout<<stringa ;*
  - ove stringa è una sequenza di caratteri delimitata da doppi apici "
  - "*esempio di stringa*"
  - Daremo informazioni più precise ed entreremo in maggiori dettagli nella prossima esercitazione

Primo esercizio 1/4

- Scrivere un programma che stampi *Ciao mondo* sul terminale e memorizzarlo in un file dal suffisso **.cc**
- Aspettare prima di eseguirlo!

Primo esercizio 2/4

```
#include <iostream>

using namespace std;

main()

{

  cout<<"Ciao mondo!" ;

}
```

Esecuzione programma 1/2

- L'elaboratore è in grado di eseguire il programma che abbiamo appena scritto?

Esecuzione programma 2/2

- No

Linguaggi

- I microprocessori presenti negli elaboratori hanno un proprio linguaggio di programmazione, e capiscono solo quello
  - Il cosiddetto **linguaggio macchina**
  - E' un linguaggio molto elementare, cosiddetto di basso livello, in cui non è facile scrivere programmi
- Sono stati quindi definiti molti linguaggi, di alto livello, che rendono molto più agevole la programmazione

Traduzione

- Però, per poter essere eseguito, ossia per diventare ciò che si definisce un **eseguibile**, un programma scritto in un linguaggio ad alto livello deve essere **tradotto** in linguaggio macchina
  - Questa operazione è effettuata da strumenti chiamati tipicamente **compilatori**
- Il C/C++ è un linguaggio ad alto livello
- Quindi, per poter eseguire programmi scritti in linguaggio C/C++ bisogna prima tradurli in linguaggio macchina

File sorgente

- File sorgente (unità di traduzione): file di testo che contiene il programma scritto nel linguaggio di partenza
  - O solo parte del programma in programmi più evoluti
- Quello che abbiamo creato è quindi un file sorgente di un programma in C++

Compilazione

- Per ottenere un programma eseguibile a partire dal nostro sorgente possiamo utilizzare un compilatore per il linguaggio C++
  - Schema:
    - Sorgente->Compilazione->Eseguibile

Compilatore gcc 1/2

- gcc: GNU Compiler Collection
- *g++*: front end al gcc per compilare sorgenti C++
- Tutte le informazioni sul compilatore:
  - *http://www.gnu.org/software/gcc/*
- *man g++*
- Progetto GNU:
  - *http://www.gnu.org/*

Installazione g++

- Soluzione migliore per capire come installare il compilatore
  - Cercare tra le risorse relative alla propria distribuzione
    - Forum, manuali, ...
- Scorciatoia: cercare con google

Esempio:

installazione gcc g++ <nome_distribuzione>

Compilatore gcc 2/2

- Sintassi più semplice, da una *shell*, per generare un programma eseguibile da un file sorgente:
- *g++ nome_sorgente.cc*
  - Assegna un nome predefinito al programma eseguibile, tipicamente *./a.out*
- *g++ -o nome_eseguibile nome_sorgente.cc*
  - Permette di scegliere il nome del programma eseguibile

Errore tipico

g++ -o nome_sorgente.cc

- Manca il nome dell'eseguibile dopo l'opzione -o

- O non usate affatto l'opzione -o, oppure

  g++ -o nome_eseguibile nome_sorgente.cc

Proviamo ...

- ... a compilare ed eseguire il nostro programma ...

Messaggi di errore 1/2

- Può darsi che la compilazione non sia andata a buon fine
  - In questo caso il compilatore ha sicuramente scritto dei messaggi

Messaggi di errore 2/2

- Se ci sono problemi, il compilatore può comunicare
  - Warning (avvisi): c'è qualcosa di 'sospetto' nel codice, ma si può comunque generare un eseguibile
  - Error: ci sono errori che impediscono la conclusione della compilazione
- **LEGGETELI per capire cosa c'è che non va nel programma !!!**

Invocazione programma 1/3

- Supponiamo che il file eseguibile si trovi nella cartella corrente. In questo caso, per far partire il programma può bastare scrivere il nome del file eseguibile e premere *invio*
  - Come abbiamo visto il nome predefinito del file eseguibile è *a.out*
- In base a quello che abbiamo appreso sui percorsi assoluti e relativi ci pare che scrivere il solo nome del programma corrisponda ad usare un percorso relativo
- Le cose però non stanno così per quanto riguarda l'esecuzione dei file ...

Invocazione programma 2/3

- Se si immette il solo nome del file, la *shell* cerca in verità il file eseguibile in una serie di cartelle predefinite
- Se siamo fortunati, tra le cartelle predefinite della *shell* c'è anche la cartella corrente
- Se invece siamo sfortunati, la cartella corrente non è nell'elenco, e la *shell* ci dice che non trova il programma

Invocazione programma 3/3

- Nel secondo caso abbiamo due possibilità:
  - Usare un percorso assoluto
    Esempio: **`/home/paolo/a.out`**
  - Usare un percorso relativo dicendo però esplicitamente alla *shell* che il file va cercato **qui**. Per farlo utilizziamo il nome speciale **`.`**
    Esempio: **`./a.out`**
  - Con entrambe le soluzioni la *shell* non cerca nelle proprie cartelle predefinite, ma bensì esattamente dove le indichiamo noi

Se tutto ha funzionato ...

- Forse il *prompt* riappare appiccicato al nostro messaggio ...
- Non siamo andati a capo!
- Bisognerebbe poter stampare il carattere ***a capo*** (***newline***)

Sequenze di controllo

- I caratteri non visualizzabili (caratteri speciali) possono essere rappresentati mediante sequenze di controllo (*escape sequence*)

- \n *newline*
- \t tabulazione
- \\ barra inversa
- \' apice
- \” virgolette

Primo esercizio 3/4

- Modificare il programma affinché vada anche a capo

Primo esercizio 4/4

```
#include <iostream>

using namespace std;

main()

{

  cout<<"Ciao mondo!\n" ;

}
```

Accodamento operatori

- Gli *operatori di uscita* << possono essere accodati l'uno all'altro

  Esempio: *cout<<"Ciao "<<"mondo\n";*

- Gli argomenti verranno stampati l'uno di seguito all'altro

- Non solo le stringhe possono essere passate come argomento ...

Manipolatori

- Ulteriori *oggetti* che possono essere passati all'operatore di uscita
- Modificano in qualche modo la formattazione dell'ingresso/uscita
- Esempio:
  - *endl:* equivalente alla sequenza di controllo *n*

Esercizio 2 1/2

- Usare il manipolatore *endl* per andare a capo nel precedente programma

Esercizio 2 2/2

```
#include <iostream>

using namespace std;

main()

{

  cout<<"Ciao mondo!"<<endl ;

}
```