Programmazione 1

Per alcuni le cose che ho scritto in questo file potranno essere banali, ma io venivo da un liceo scientifico e avendo fatto poca programmazione volevo essere il più chiaro possibile e non saltare nessun collegamento logico :).

1. Scrittura di un programma

Si usa un ambiente di sviluppo Linux (**Ubuntu**). Il programma è scritto nel linguaggio **C++**. Il file sorgente si scrive con l'editor **Emacs.**

Un file presenta estensione .cc (prova.cc).

1.1. Compilazione di un programma

Usando il terminale di Linux (che si apre con la combinazione CTRL+ALT+T), si crea un file vuoto (touch file.cc), si apre il file nell'editor emacs (emacs file.cc), si scrive il programma e si salva.

La compilazione avviene sempre da terminale. Il comando è il seguente: g++ file.cc. In questo modo si crea un file eseguibile a.out (che corrisponde al .exe di Windows). Per eseguire questo file dal terminale usare il seguente comando: ./a.out (dove il punto indica la cartella corrente).

Per ottenere un file eseguibile con un nome personalizzato usare il seguente comando: g++ file.cc -o file.out. Che può essere così eseguito: ./file.out.

Se non si vuole creare un file eseguibile direttamente, ma prima un file oggetto si usano i seguenti comandi:
g++ -c file.cc (si crea un file oggetto file.o)
g++ file.o (si crea il file eseguibile a.out) oppure g++ file.o -o file.out (si crea l'eseguibile file.out)

Per compilare più file basta scrivere i file uno dopo l'altro:
g++ file.cc file1.cc file2.cc … fileN.cc (generano un unico file a.out).

2. Terminale Linux

Per acceder al terminale su Ubuntu usare la combinazione di tasti: CTRL+ALT+T.

Struttura del terminale: nomeutente/percorso dove si trova ora il terminale/carattere
Il carattere può essere $ (l'utente non ha privilegi) oppure # (l'utente ha i privilegi di default) Col simbolo ~ si indica che ora il terminale si trova nella home.

2.1 Comandi Linux indispensabili

- *cd "directory"*: entra nella cartella di nome "directory". Il comando *cd* .. permette di tornare nella cartella precedente, mentre *cd* invio permette di tornare alla cartella home.
- *ls "directory"*: mostra cosa c'è nella cartella
- *mkdir "directory"*: crea la cartella di nome "directory"
- *rm "file"*: rimuove il file di nome file (operazione irreversibile). Alcune opzioni di rm:
  - *rm -i "file"*: chiede all'utente se è sicuro di rimuovere il file (Y or N).
  - *rm – r "file"*: rimuove le cartelle e i loro contenuti ricorsivamente. Tutto quello dentro la cartella di partenza viene eliminato.
- *rmdir "directory"*: rimuove la cartella (solo se è vuota)
- *cat "file"*: mostra il contenuto di un file
- *clear*: pulisce la schermata

- *man “comando”*: mostra il manuale del comando scritto (ad esempio *man rm*, mostra cosa fa il comando rm e tutte le sue flag, cioè le sue opzioni che ne modificano il comportamento).
- *cp “file1” “file2”*: copia il contenuto di un file in un altro.
- *mv “file1” file2”*: cambia il nome del file1 rinominandolo file2” oppure se al posto di file2 c’è una cartella (*mv “file1” “directory”*) sposta il file1 nella cartella.
- *touch “file.txt”*: crea un file vuoto
- *exit*: chiude il terminale
- *pwd*: mostra il percorso della cartella nella quale si trova il terminale

Con TAB c’è l’autocompilazione.

3. Introduzione al C++

Template base di C++
---
using namespace std; permette di non scrivere *std::cout << "Ciao";* ma di scrivere solo *cout << "Ciao" ;*

#include <iostream> include la libreria iostream che permette di usare cout e cin (input output)

```
int main ()
{

return 0;
}
```
ritorna 0 perché main è una funzione e in quanto tale deve ritornare un valore (intero perché come si legge nella terza riga: int main())

Indentare bene (anziché indentare fare due spazi)

3.1 Sintassi del C++

Le parole sono costituite da tutte le lettere (maiuscole e minuscole; tra le lettere si considera anche questo carattere: _), tutti i numeri e alcuni caratteri speciali (! : ; , ‘ “ / {} () <> ? [] || % & + - * /). Non si usano gli accenti.

Non più di 80 caratteri per riga (questione di leggibilità del codice)

Commenti:
//commento su una riga

```
/* commento
    su più
    righe   */
```

Le entità di un programma C++ devono avere dei nomi che ne consentano l’individuazione (i cosiddetti identificatori). C++ è case sensitive: *Fact* è diverso da *fact*. Ci sono alcune parole chiave che non possono essere usate come identificatori (while, and, …).

3.2 Espressioni letterali

Le espressioni letterali denotano valori costanti.

Una **costante carattere** viene rappresentata racchiudendo il corrispondente carattere fra apici: per esempio la costante ’a’ rappresenta il carattere ’a’.
I caratteri di controllo vengono rappresentati da combinazioni speciali dette sequenze di escape che iniziano con una barra invertita (backslash ’\’).
Alcuni esempi: nuova riga (vai a capo) (\n), tabulazione orizzontale (\t), tabulazione verticale (\v), spazio indietro (\b), ritorno carrello (\r).

Una **costante stringa** è una lista di caratteri compresa fra una coppia di virgolette, ad esempio: "Hello!”. Possono includere anche spaziature, caratteri non alfanumerici e sequenze di escape, ad esempio: "Hello, world!\n".

Carattere tra singolo apice, stringa tra doppio apice

Un **numero intero** viene rappresentato da una sequenza di cifre, che vengono interpretate in base decimale (default), in base ottale (se inizia con uno zero) o in base esadecimale se inizia con ’0x’ (o ’0X’)
Un **numero reale** (in virgola mobile) viene rappresentato facendo uso del punto decimale e della lettera ’e’ (o ’E’) per separare la parte in virgola fissa dall’esponente, ad esempio: -1235.6e-2 rappresenta -12,356. Si usa il punto per rappresentare la virgola (3.14 è 3,14).

Queste cose sono importanti da sapere per quanto riguarda la stampa a video di parole, frasi o numeri. Se voglio stampare a video un numero non lo metto tra virgolette.

3.3 Operatori e separatori

Gli **operatori** denotano certe operazioni nel calcolo delle espressioni. Alcuni di essi sono: + - * / % ^ & | = != >= > <= < == !.

Un operatore si dice: prefisso (se precede gli argomenti, cioè i suoi operandi, +34), suffisso (se li segue 34+) o infisso (se si interpone 3+4). Usare le parentesi per evitare ambiguità con gli operatori.

I principali **separatori** del C++ sono: () , ; : {}. I separatori sono simboli di interpunzione che indicano il termine di un’istruzione, separano elementi di liste, raggruppano istruzioni o espressioni ecc.

4. Variabili
Per memorizzare un valore in un'area di memoria si utilizzano entità chiamate **variabili** o **costanti**. Le variabili permettono la modifica del loro valore durante l'esecuzione del programma. Il valore di una costante rimane lo stesso per tutto il programma.

L' area di memoria corrispondente è identificata da un nome, che ne individua l'indirizzo in memoria.

Le variabili sono caratterizzate da una quadrupla:
- il nome (identificatore)
- il tipo (int, char, float, bool e molti altri)
- la locazione di memoria (l-value o indirizzo)
- il valore (r-value)

Definizione di variabili (serve a predisporre l'area di memoria per memorizzare la variabile)
Formato: *tipo identificatore;* Ad esempio *int i;*

Se (e solo se!) una variabile è definita esternamente ad ogni funzione, main() inclusa, (variabile globale) viene automaticamente inizializzata al valore 0.

Definizione con inizializzazione.
Formato *tipo identificatore=valore;*
L'espressione è un carattere o un numero, cioè costanti, oppure altre variabili precedentemente definite.

Ad esempio *char c='a';*
*int i=2*3;*
int a=i/2;

Dichiarazione
Formato: *extern tipo identificatore;* Ad esempio *extern int x;*

Dichiarazione e definizione sono diversi.
Definizione: quando il compilatore incontra una definizione di una variabile, esso predispone l'allocazione di un'area di memoria in grado di contenere la variabile del tipo scelto.
Dichiarazione: specifica solo il tipo della variabile e presuppone dunque che la variabile venga definita in un'altra parte del programma.

Con la dichiarazione si inserisce nella tabella di simboli del programma l'informazione che verrà aggiunta una variabile. Non si riserva la memoria. Infatti dopo la dichiarazione la variabile deve essere definita.

5. Costanti
Definite in questo formato. Il loro valore non varia durante il programma.
Formato: *const tipo identificatore = exp;*
exp deve essere una espressione il cui valore deve poter essere calcolato in fase di compilazione

Ad esempio:
const float pi=3.14;
const int k=10;
const int j=k-1;

6. Concetto di stream.
Uno stream è un flusso di caratteri (in ingresso o in uscita), tramite il quale il programma comunica con l'esterno.
Lo stream è una struttura logica costituita da una sequenza di caratteri, in numero teoricamente infinito, terminante con un apposito carattere che ne identifica la fine.
Gli stream vengono associati (con opportuni comandi) ai dispositivi fisici collegati al computer (tastiera, video) o a file residenti sulla memoria di massa.

In C++ esistono i seguenti stream predefiniti:
- **cin** (stream standard di ingresso). Input (tastiera, mouse, …).
- **cout** (stream standard di uscita). Output (stampante, monitor, …).
- **cerr** (stream standard di errore). Vanno scritti i messaggi di errore.

Le funzioni che operano su questi stream sono in una libreria di ingresso/uscita e per usarle occorre la direttiva: #include <iostream>.

6.1 Stream di uscita (output): COUT.
Lo stream di uscita standard è **cout**, quello su cui il programma scrive i dati (è tipicamente associato allo schermo).

Per scrivere dati si usa l'istruzione di scrittura:
stream << espressione;
Ad esempio:
cout << "La somma è " << somma << endl;
cout << 3+2;
endl corrisponde ad uno '*\n*' e permette di andare a capo.

L'istruzione di scrittura comporta: il calcolo del valore dell'espressione, la conversione in una sequenza di caratteri e il trasferimento della sequenza nello stream.

Si possono anche scrivere delle scritture multiple di questo tipo: *cout << x << y << endl;* che corrisponde a
cout << x;
cout << y;
cout << endl;

6.1 Stream di ingresso (input): CIN.
Lo stream di ingresso standard è **cin**, quello da cui il programma preleva i dati (è tipicamente associato alla tastiera).

Per prelevare dati si usa l'istruzione di lettura:
stream >> espressione; dove espressione deve essere un'espressione dotata di indirizzo (=variabile)
cin >> x;

L'istruzione di lettura comporta in ordine:
1. il prelievo dallo stream di una sequenza di caratteri
2. la conversione di tale sequenza in un valore che viene assegnato alla variabile

L'istruzione di ingresso ha una forma più generale che consente letture multiple, nel formato
cin >> var1 >> var2 >> var3;
var1, *var2* e *var3* sono tre variabili che possono anche essere di tipo diverso.

6.1.1 Lettura di un carattere da cin.
Consideriamo l'istruzione *cin >> x*, dove *x* è di tipo char
✶✶✶✶a✶✶-14.53✶✶728✶✶ … questo è l'input del nostro utente, cioè quello che lui ha scritto
⇑
(qui "✶" rappresenta uno spazio e "⇑" il cursore)

- Se il carattere puntato dal cursore è una spaziatura: il cursore si sposta in avanti per trovare un carattere che non sia una spaziatura.
- Se il carattere puntato dal cursore non è una spaziatura:
  1. il carattere viene prelevato
  2. il carattere viene assegnato alla variabile x (anche un numero di una cifra viene associato alla variabile x in quanto viene considerato non un int ma un carattere. Se viene scritto un numero con due cifre il programma associa alla variabile x solamente la prima cifra, considerandola un carattere)
  3. il puntatore si sposta alla casella successiva

6.1.2 Lettura di un numero da cin.
Consideriamo l'istruzione *cin >> x*, dove *x* è di tipo int
✶✶✶✶a✶✶-14.53✶✶728✶✶ … questo è l'input del nostro utente, cioè quello che lui ha scritto
⇑
(qui "✶" rappresenta uno spazio e "⇑" il cursore)

- Se il carattere puntato dal cursore è una spaziatura: il cursore si sposta in avanti per trovare un carattere che non sia una spaziatura.
- Se la sequenza di caratteri è interpretabile come un numero compatibile con il tipo di x
  1. la sequenza di caratteri viene prelevata
  2. la sequenza viene convertita nel suo corrispondente valore (es: il valore intero 728) che viene assegnato alla variabile x. Se l'utente inserisse -14.53, ad x verrebbe assegnato -14.
  3. il puntatore si sposta alla casella successiva
- Altrimenti, l'operazione di prelievo non avviene e lo stream si porta in uno stato di errore.

6.2 Alcune utili funzioni della libreria <iostream>

cin.eof(): ritorna un valore diverso da 0 se lo stream cin ha raggiunto la sua fine (End Of File).
Se dovesse ritornare 0 significherebbe che ci sono altri caratteri da leggere sullo stream.
Questo metodo verifica che l'ultimo carattere dello stream (il carattere end of file) sia stato letto.

Va usato sempre dopo almeno un'operazione di lettura e richiede un separatore dopo l'ultimo elemento letto.

cout << cin.eof(); per vedere cosa ritorna.

cin.fail(): ritorna un valore diverso da 0 se lo stream cin ha rilevato uno stato di errore (ad esempio l'inserimento di una stringa per int) o un end-of-file

cout << cin.fail(); Se usato dopo un input ritorna 1 o altri valori, c'è stato un errore nello stream di ingresso. Se ritorna 0 tutto nella norma

Non necessariamente usato dopo almeno un'operazione di lettura e non richiede un separatore dopo l'ultimo elemento letto.

cin.clear(): ripristina lo stato normale dallo stato di errore. Se dopo aver usato *cin.fail()* notiamo che c'è un errore, usando *cin.clear()* ripristineremo lo stato normale. Infatti se facciamo di nuovo *cin.fail()* otterremo il valore 0.

cin.ignore(): permette di scartare i caratteri rimasti nello stream. Viene utilizzato dopo che lo stream è andato in fase di errore. Ad esempio può essere utilizzato se al posto di un numero intero l'utente ha inserito un carattere. Se non si usa *cin.ignore()* non sarà possibile effettuare altre operazioni di input.

Esempio:

```
cin >> x; //x è un intero
          //se l'utente inserisce un carattere lo stream va in stato di errore e non è più possibile fare
            altre operazioni di input

…
cin.clear();
cin.ignore();

cin >> …      //questo cin è possibile solo grazie ai due costrutti usati in precedenza che liberano
              l'errore nello stream
```