Primi Elementi di Programmazione in C++

Linguaggi di alto livello

- I vantaggi della programmazione in un linguaggio di alto livello sono

  - l'astrazione: indipendenza della descrizione dell'algoritmo dai dettagli della macchina che lo deve eseguire;

  - la portabilità del codice: sottoprodotto dell'astrazione che consente l'esecuzione dello stesso programma su una vasta gamma di calcolatori diversi, a condizione che esista un compilatore in grado di tradurlo nel linguaggio di ogni macchina.

Sviluppo del *software*

Due diverse filosofie di programmazione:

- si costruisce il diagramma di flusso delle informazioni e lo si traduce in una sequenza di istruzioni

- ci si concentra sulla identificazione degli oggetti che si vogliono manipolare, sulle loro proprietà e sulle relazioni tra di essi. Per ogni tipo di oggetto si scrive (o si trova già scritto) il codice che ne consente la manipolazione, il programma vero e proprio si limita ad utilizzare questi oggetti.

Oggetti

- Un oggetto è un'entità salvata in memoria sulla quale si possono effettuare operazioni.

- Ogni oggetto ha un'identità (un nome), uno stato (valore o valori salvati in memoria) e un insieme di operazioni o funzioni che gli possono essere applicate

Linguaggi di programmazione

- Pascal, Fortran e C sono alcuni dei linguaggi tradizionalmente usati in ambito scientifico.

- La programmazione orientata agli oggetti ha sviluppato dei linguaggi che sono più idonei a questa filosofia e che forniscono gli strumenti per gestire gli oggetti, primi fra tutti JAVA e C++.

Sviluppo di un programma OO

La realizzazione di un programma orientato agli oggetti richiede tre fasi distinte

Analisi

Disegno

Implementazione

Analisi

- Comprensione del problema
- Definizione di cosa debba fare il *software*
- Identificazione dei dati da fornire in ingresso e da ottenere in uscita
- Identificazione degli oggetti nel problema: quali sono le cose che possiamo o vogliamo manipolare?

Disegno

- Scelta di un algoritmo (sequenza di operazioni) che consenta di ottenere il risultato cercato
- Modularizzazione (spezzettamento) del problema in problemi elementari o comunque più semplici
- Ricerca di eventuale software già esistente (nel nostro caso classi di C++ predefinite)

Implementazione

- Scrittura del codice sorgente
- Produzione di un programma eseguibile
- Test del programma che deve
  1. Girare senza intoppi
  2. Fornire risultati plausibili

Problema

Determinare il vostro voto finale, in trentesimi, sapendo che questo è dato da

- 4 esercitazioni che hanno un peso pari a 4/30
- un'esercitazione individuale che ha un peso di 1/3
- un test finale che ha un peso di 4/30

assumendo che tutti i voti siano espressi in trentesimi.

Analisi

- Leggere accuratamente il problema, eventualmente parafrasarlo, accertarsi di averlo capito e di sapere cosa ci si aspetta dal programma
- Identificare gli *input* del problema: i voti ottenuti nelle 6 prove
- Identificare l'*output* del problema: il voto finale
- Cercare gli oggetti coinvolti: i voti (numeri), i pesi (numeri),la tastiera del computer per inserire i dati di input, lo schermo per visualizzare l'output

Disegno

- Utilizziamo un paradigma di basso livello denominato IPO: Input Process Output

Input: accesso ai dati iniziali, nel nostro caso i 6 voti

Processamento: manipolazione dei dati, nel nostro caso il calcolo del voto finale

Output: trasmissione del risultato, nel nostro caso la visualizzazione del voto finale

Input

Dare dei nomi ai vari oggetti

- "lab1","lab2", "lab3", "lab4" per le prime 4 esercitazioni
- "prova" per il programma individuale
- "test" per il test finale
- Il programma dovrà chiedere all'utente di inserire da tastiera dei valori da assegnare a questi oggetti e dovrà leggerli

Processamento

- Il voto finale sarà dato dall'espressione

(lab1+lab2+lab3+lab4)*4./30. + prova/3. + test*4./30.

Output

- Dare al risultato da inviare in uscita un nome, ad esempio "votoFinale"
- Scrivere sullo schermo il risultato con un opportuna etichetta che spieghi di cosa si tratta

Implementazione

- Utilizzare un *editor* di testo per creare un *file* contenente il codice sorgente, un programma in C++ nel nostro caso
- Invocare il precompilatore, il compilatore ed il *linker* per ottenere un (programma) eseguibile

Il programma

```
// questo programma calcola il voto finale

#include <iostream.h>

int main(  ) {

          float lab1,lab2,lab3,lab4;
          float prova,test;
          float votoFinale;
          cout << " inserire i voti"
                  << " lab1 lab2 lab3 lab4 prova e test"
                  << " in questo ordine"
                  << endl;
          cin >>lab1>>lab2>>lab3>>lab4>>prova>>test;
          votoFinale=(lab1+lab2+lab3+lab4)*4./30.+prova/3.+test*4./30.;
          cout << " voto finale " << votoFinale << endl;
        return 1;
}
```

Dichiarazione degli oggetti

- Ogni oggetto in un programma C++ deve essere introdotto con una dichiarazione prima di poter essere utilizzato
- Nel nostro esempio

```
float lab1,lab2,lab3,lab4;
float prova,test;
float votoFinale;
```

cin e cout sono dichiarati nel file iostream.h che viene incluso dall'instruzione del precompilatore

```
#include <iostream.h>
```

Inizializzazione

- Avremmo potuto dare agli oggetti float un valore iniziale, utilizzando una delle seguenti dichiarazioni:

```
float lab1=20.,lab2=25.,lab3=18.5;
```

o

```
float lab4(30.),lab1(21.0);
```

Operazioni

- Gli oggetti di tipo **float** capiscono le operazioni +, * (ma anche -, % etc...)
- L'oggetto **cin**, di tipo **istream**, identificato con la tastiera, capisce l'operazione >>
- L'oggetto **cout**, di tipo **ostream**, identificato con lo schermo, capisce l'operazione <<

Sintassi degli oggetti cout e cin

- `cout<<espressione1<<...<<espressione`*n* `;`

  dove ogni espressione può essere una stringa di caratteri racchiusa tra doppi apici o un'espressione che restituisca un valore numerico. Un'espressione speciale è *endl* (andata a capo)

- `cin>>oggetto1>>...>>oggetto`*n* `;`

  dove ogni oggetto è o un oggeto di tipo stringa o un oggetto di una classe numerica

Test dell'implementazione

```
inserire i voti lab1 lab2 lab3 lab4 prova e test in questo ordine
                0    0    0    0    0       0
votoFinale 0

inserire i voti lab1 lab2 lab3 lab4 prova e test in questo ordine
                30   30   30  30   30      30
votoFinale 30

inserire i voti lab1 lab2 lab3 lab4 prova e test in questo ordine
                20   20   20  20   20      20
votoFinale 20

inserire i voti lab1 lab2 lab3 lab4 prova e test in questo ordine
                30   30   30   30   20      30
votoFinale 26.6667
```

NB

- I test rivelano solo la presenza di errori, non la loro assenza !

Elementi del programma

- Il programma è realizzato con una sequenza di elementi base: parole chiave (*keywords*), identificatori, costanti e simboli speciali
  - Keywords: int, float, return etc…
  - Identificatori: main, cout, cin, lab1 etc…
  - Costanti: " voto finale " , 3. , 30. etc…
  - Simboli speciali: #, (,), +, *, <<, >> etc…

Il programma

```
// questo programma calcola il voto finale

#include <iostream.h>

int main(  ) {

        float lab1,lab2,lab3,lab4;
         float prova,test;
         float votoFinale;
         cout << " inserire i voti"
              << " lab1 lab2 lab3 lab4 prova e test"
              << " in questo ordine"
              << endl;
         cin >>lab1>>lab2>>lab3>>lab4>>prova>>test;
         votoFinale=(lab1+lab2+lab3+lab4)*4./30.+prova/3.+test* 4./30.;
         cout << " voto finale " << votoFinale << endl;
       return 1;
}
```

Identificatori

- Gli identificatori degli oggetti e delle funzioni sono una sequenza continua di caratteri (massimo 32) appartenenti alla lista

  'a'..'z', 'A'..'Z', '0'..'9', '_'

  con il vincolo che il primo carattere non sia un numero

- Quali di questi sono identificatori validi?

  abc, ABC, a_1, a1, m/h, 5a, voto Finale, votoFinale

- Risposta: abc,ABC,a_1,a1,votoFinale

Costanti

- Reali: 0.1 3. 3.0 1.234 -12.5 0.0 0. 1e10 5e-3
- Interi: -1 0 1 –44456 +877
- Caratteri: 'A' ' ' '\n'
- Stringhe di caratteri: "qualunque cosa tra doppi apici"

Simboli speciali

- Commenti
  - **//** all'inizio di una riga di commento
  - **/*** commento ***/**
- Direttive del precompilatore
  - Tutte le righe che iniziano con il simbolo **#**
- Operatori
  - **+,-,*,/,>>,<<** etc…
- Punteggiatura
  - Ogni istruzione C++ termina con ;
  - Ogni parte autoconsistente del programma è racchiusa tra parentesi graffe **{ }**