Linguaggio C++

Linguaggi di terza generazione

- Insieme di primitive ad alto livello, ognuna traducibile in una sequenza di primitive in linguaggio macchina
  - Es.: pesolordo ← pesocarico + pesoveicolo
  - Due load, una add, una store
- Programma traduttore (**compilatore**): traduce il programma in linguaggio macchina
- **Interprete**: traduce ogni primitiva ed esegue subito le primitive corrispondenti del l.m., senza memorizzare la traduzione
- Programmi scritti in un ling. di terza generaz. sono trasportabili da una macchina all'altra, basta cambiare compilatore

Paradigmi di programmazione - 1

- Lo sviluppo dei linguaggi di terza generazione ha seguito percorsi diversi → quattro approcci alternativi al processo di programmazione:
  - Imperativo
  - Dichiarativo
  - Funzionale
  - Orientato agli oggetti

Paradigmi di programmazione - 2

- **Imperativo** (procedurale):
  - programma = sequenza di comandi che descrive un algoritmo pensato dal programmatore
  - Es.: pseudocodice, Assembler, Pascal, C
- **Dichiarativo:**
  - Basta descrivere il problema e non il suo algoritmo
  - Algoritmo generale per risolvere i problemi
  - Ambiti ristretti o linguaggi basati su logica matematica (programmazione logica, Prolog)
- **Funzionale:**
  - Funzioni, con ingresso e uscita
  - Connesse in modo da costruire funzioni complesse a partire da funzioni elementari
  - LISP (LISt Processing)

Paradigmi di programmazione - 3

Paradigma orientato agli oggetti:

- Dati associati a procedure per gestirli (oggetti)
- Es.: elenco di nomi
  - Procedure per inserire un nuovo nome, eliminarne uno, vedere se l'elenco e' vuoto, ordinare l'elenco, ...
  - Un programma accede all'elenco e usa le sue procedure per gestirlo
- interfaccia grafica
  - Icone: oggetti, con procedure che vengono attivate da eventi (clic del mouse, ...)
- Ogni oggetto e' un'entita' autonoma
- Costruzione modulare del software
- Ogni procedura: programma imperativo
- C++ e JAVA

Il linguaggio C++

- Il C++ e` un linguaggio di programmazione (object-oriented, cioe' orientato agli oggetti)
- E' stato progettato all'inizio degli anni 1990 da Bjarne Stroustrup come una evoluzione del linguaggio di programmazione C che incorporova l'approccio di programmazione orientato agli oggetti
- Attualmente, il C++ e` il linguaggio di programmazione piu' utilizzato al mondo assieme al suo predecessore C ed al linguaggio Java

I programmi C++

- Ogni compilatore C++ traduce il testo del programma sorgente nelle **istruzioni del linguaggio macchina di un particolare processore**
- Se il programma C++ sorgente e' salvato nel file **`prog.cpp`** (**`.cpp`** e' l'estensione richiesta per i programmi C++), il compilatore viene invocato tramite il comando **`g++ prog.cpp`** da digitare nella shell di Linux quando la directory corrente contiene il file **`prog.cpp`**
- Se non si incontrano errori, la compilazione produce il file **`a.out`** che contiene il corrispondente programma eseguibile
- Invocando invece il comando **`g++ prog.cpp -o prog.exe`** la compilazione produce il file eseguibile di nome **`prog.exe`**

- Per eseguire il programma eseguibile di nome **`prog.exe`** basta digitare nella shell di Linux il comando **`./prog.exe`** (oppure, se Linux e' ben configurato, semplicemente **`prog.exe`**)

`ciao.cpp`

```
 1 // Programma ciao
 2 // Stampa sul video una frase
 3 /* autore Marco Rossi */
 6 #include<iostream>
 7 using namespace std;
 8
 9 main() {
10   cout << "Ciao ragazzi!!" << endl;
11 }
```

ciao.cpp

- C++ e' "**case-sensitive**": come in Unix/Linux, l'uso di lettere minuscole o maiuscole produce effetti diversi
- Possiamo scegliere il nome del file contenente il programma sorgente: ad esempio **ciao.cpp**

ciao.cpp

```
1 // Programma ciao
2 // Stampa sul video una frase
3 /* autore Marco Rossi*/
```

Linee 1-3 sono di **commento**

Due tipi di commento:

`//` su singola riga

`/* ... */` anche su piu' righe

ciao.cpp

```
6 #include<iostream>
7 using namespace std;
```

Si tratta di **dichiarazioni** necessarie per usare le librerie del linguaggio, cioe' alcuni programmi basilari forniti assieme al compilatore. Non preoccupiamoci di cio', ricordiamoci pero' di metterle **sempre** nei nostri programmini.

ciao.cpp

```
 9 main() {
10   cout << "Ciao ragazzi!!" << endl;
11 }
```

Le istruzioni del programma costituiscono il corpo della funzione principale **main(),** che e' delimitato dalle parentesi graffe. L'esecuzione del programma inizia dalla **prima** istruzione del main().

ciao.cpp

```
 9 main() {
10   cout << "Ciao ragazzi!!" << endl;
11 }
```

In questo caso, il programma e' costituito da un'unica istruzione che **stampa una stringa a video** e termina la linea stampando il carattere di fine linea **`endl`**. Una stringa e' una sequenza di caratteri racchiusa tra virgolette, come ad esempio **`"Ciao ragazzi!!"`**. Dopo l'esecuzione dell'unica istruzione il programma termina.

Sintassi dei programmi

- Le parentesi **graffe** devono sempre essere **bilanciate**
- Si possono scrivere **piu' dichiarazioni o istruzioni sulla stessa riga di testo**, ma quasi sempre per leggibilita' si preferisce una sola dichiarazione o istruzione per riga
- **Il modo in cui un programma** e' scritto in termini di righe e spazi **non incide** sulla sua correttezza sintattica ne' tantomeno sulla sua esecuzione. Il programma che viene compilato non deve avere un aspetto particolare per quanto riguarda **righe, spazi, tabulazioni ed indentazioni**. Solitamente, **e' buona norma scrivere le istruzioni ordinatamente** una sotto l'altra usando le **indentazioni** per aumentare la leggibilita' del codice

Identificatori

- I "**nomi**" presenti in un programma, ovvero quelli "inventati" dal programmatore, vengono chiamati **identificatori**.
- In generale, una stringa in un certo alfabeto ALF di simboli e' una sequenza finita di simboli dell'alfabeto ALF
- Un identificatore in C++ e' una **stringa costituita da lettere (minuscole e maiuscole) e numeri, deve sempre iniziare con una lettera, e gli spazi non sono consentiti**
- Esempi di identificatori validi: **`pippo, Pippo, x, temp, temp2, CONST, numeroGrande`**
- Esempi di identificatori non validi: **`2temp, 27, pippo pluto`**

Keyword

- Le **parole chiave** (**keyword**) sono identificatori riservati del C++ e non possono essere usati come identificatori in un programma
- Alcuni keyword importanti sono:
  - **`bool, break, case, char, double, else, false, float, for, if, int, long, new, return, static, switch, this, true, void, while`**

Tipi

- C++ e' un linguaggio **tipizzato**, come la maggior parte dei moderni linguaggi di programmazione
- In parole semplici, potremmo dire che l'idea di base della tipizzazione consiste nel fatto che non si devono e non si possono confondere numeri con stringhe o con valori booleani, perche' sono di tipi diversi (o, si dice, hanno tipi diversi)
- Piu' precisamente, essere tipizzato significa che **tutte** le **variabili** ed **espressioni** hanno un tipo
- C++ possiede **cinque tipi primitivi di base**: **`int, float, double, bool, char`**
  - **`int`**: numeri interi (positivi e negativi) rappresentati in complemento a 2 su 4 byte. Quindi sono i numeri interi nell'intervallo [-2^{31}, $2^{31}-1$]
  - **`bool`**: valori booleani, con due soli valori: **`true`** e **`false`**
  - **`float`**: numeri in virgola mobile rappresentati in 4 byte
  - **`double`**: numeri in virgola mobile in precisione doppia rappresentati in 8 byte
  - **`char`**: caratteri rappresentati secondo la codifica ASCII in 1 byte

Variabili

- **Identificatori simbolici** (come in Assembler), permettono di riferirsi in modo simbolico al **contenuto della memoria**, evitando l'uso degli indirizzi assoluti in memoria
- Questi identificatori simbolici nella terminologia dei linguaggi di programmazione vengono detti **variabili**
- Quindi le variabili memorizzano i valori che vengono manipolati dal programma
- In un programma in esecuzione, ad ogni variabile e' associato un indirizzo assoluto fisico in memoria ove e' memorizzato il valore di quella variabile
- **Ogni variabile ha un tipo**. Questo significa che nel programma viene esplicitamente dichiarato che un identificatore di variabile conterra' valori di uno specifico tipo primitivo
- Un tipo determina **la memoria necessaria** per registrare i valori di variabili di quel tipo. Ad esempio, i tipi **`int`** e **`float`** richiedono 4 byte, mentre **`char`** e **`bool`** richiedono 1 byte

Dichiarazioni di variabili

- Nel nostro approccio semplificato, le variabili potranno essere dichiarate solamente all'interno del **`main()`**
- Schemi generali per dichiarare variabili:
  - **`tipo id;`**
  - **`Esempio: int x; variabile intera x`**
  - **`tipo id1, id2, id3;`**
  - **`Esempio: int x1, x2, x3; variabili intere x1, x2 e x3`**
  - **`tipo id = valore;`**
  - **`Esempio: float y=3.5; variabile reale y inizializzata a a 3.5`**
  - **`tipo id1 = valore1, id2 = valore2;`**
  - **`Esempio float y1=3.5, y2=6.9; variabile reali y1 inizializzata a 3.5 e y2 a 6.9`**
- Le prime due istruzioni dichiarano (sono istruzioni di dichiarazione) una o piu' variabili di qualche tipo. La terza e la quarta dichiarano e **simultaneamente inizializzano** (sono istruzioni di dichiarazione e inizializzazione) le variabili con un dato valore iniziale
- E' buona norma inizializzare le variabili prima di usarle in una espressione o in una istruzione.
- Una variabile dichiarata ma non contestualmente inizializzata conterra` un valore imprevedibile (e potra` portare ad errori logici nel programma).

Dichiarazione di variabili

Esempi

```
int temperatura;     // dichiarazione senza inizializzazione

float tassa = 14.5;  // dichiarazione e inizializzazione

int m,n;  // dichiarazione multipla senza inizializzazione

bool b = false; // dichiarazione e inizializzazione
```

Se nel main() trovo il comando

cout << tassa;
Allora stampa: 14.5

cout << temperatura;
Allora stampa un valore imprevedibile

Esempio

```
#include<iostream>
using namespace std;

main(){
  int k;
  float f;
  double d;
  char c;
  cout << k << " " << f << " " << d << " " << c <<  endl;
}

/* stampa valori imprevedibili.
   Sulla mia macchina ha stampato:

   4 1.49614e+20 1.95429e+159 ÿ
*/
```