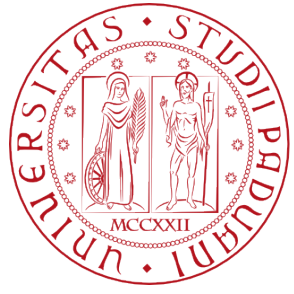
Variabili, I/O e Condizionali

Programmazione I

Laurea Triennale in Informatica
11 Dicembre 2017



Università degli Studi di Padova

Pensare da Informatici

Un Informatico:

- come un matematico, usa un linguaggio formale per descrivere le cose
- come un ingegnere, progetta sistemi complessi
- come uno scienziato, osserva il comportamento dei sistemi, formula ipotesi, e ne verifica i risultati

La competenza più importante è il *Problem Solving:*

- è la capacità di formulare problemi,
- pensare in modo creativo alle possibili soluzioni,
- ed esprimerle in maniera chiara ed accurata

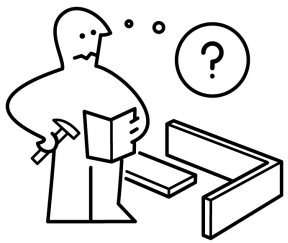
Che cos'è un programma?

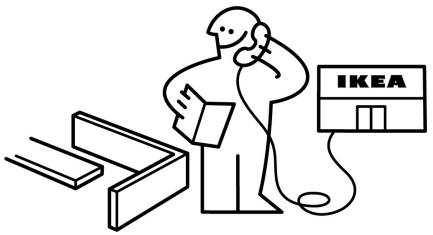
Programma:

- è una sequenza di istruzioni
- che esegue una computazione

Esempi di programmi:

- risoluzione di equazioni o altri problemi matematici
- ricette di cucina
- ma anche . . .





Tipi di istruzioni

Le *istruzioni* sono di cinque tipi base:

1. **Input:** lettura di dati dalla tastiera, file o altri dispositivi
2. **Output:** scrittura di dati sullo schermo, su file o altri dispositivi
3. **Matematiche:** istruzioni che eseguono semplici operazioni aritmetiche e logiche
4. **Esecuzione condizionale:** controlla una condizione ed esegui le istruzioni appropriate
5. **Ripetizione:** esegui più volte alcune operazioni, spesso con alcune variazioni

Linguaggi di alto e basso livello

Linguaggi di basso livello

Sono "comprensibili" dal calcolatore, come il linguaggio macchina:

- istruzioni primitive semplici (e.g. max 2 operandi)
- attenzione all'efficienza (costi, complessità, velocità)
- difficile e noioso da utilizzare per un programmatore

Linguaggi di alto livello

Sono quelli utilizzati dall'utente per scrivere programmi:

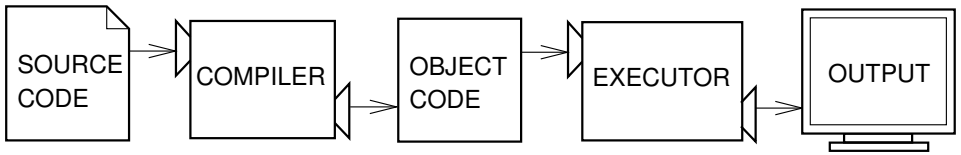
- istruzioni complesse e flessibili
- comprensibile e più facile da usare per un programmatore
- non è comprensibile direttamente dal calcolatore

Per eseguire programmi scritti in linguaggi di alto livello, è necessaria una traduzione in linguaggio macchina

Ci sono due modi per effettuare la traduzione:

- **Interpretazione:** Il codice sorgente del programma è tradotto una istruzione alla volta

- **Compilazione:** Il codice sorgente del programma viene tradotto completamente prima di essere eseguito



Il linguaggio C++

- Sviluppato da Bjarne Stroustrup nel 1983 come un miglioramento del linguaggio C
- L'*implementazione* canonica del C++ prevede che la traduzione avvenga tramite *compilazione*
- Mette a disposizione molti costrutti di programmazione avanzati (che noi non vedremo)
  - programmazione a oggetti
  - template
  - ...
- È efficiente in termini di memoria e veloce nell'esecuzione

Struttura minima di un programma C++


```
// Inclusione delle librerie ausiliarie
#include<iostream>

using namespace std;

// Corpo principale del programma
int main() {
    /*
     * Dichiarazione delle variabili
     */

    /*
     * Istruzioni
     */
}
```

Programmi e variabili

- I programmi sono parametrici:
  - producono un risultato che dipende da un insieme di dati di partenza
  - descrivono la soluzione non di un singolo problema, ma di una classe di problemi strutturalmente equivalenti

- Le istruzioni del programma fanno riferimento a variabili, il cui valore non è fissato a priori ma cambia a seconda della situazione elaborativa in cui l'esecutore si trova
  - variabili per contenere i dati di partenza
  - variabili per memorizzare i risultati intermedi
  - variabili per salvare il risultato finale

Variabili: nomi, tipi e valori

Una variabile è un nome che si riferisce ad un valore di un certo tipo

- Il nome di una variabile può essere lungo a piacere
- Un nome può essere composto sia da lettere che numeri
  - ma deve iniziare con una lettera
  - non può contenere spazi
- Una variabile deve avere un tipo:
  - `int` (numero intero) — `2, 3, -5`
  - `float` (numero in virgola mobile) — `1.0, -7.22`
  - `char` (carattere) — `'a', 'b', 'c'`
  - `string` (stringa di caratteri) — `"Ciao", "Pippo!"`
  - `bool` (booleano) — `true, false`

Dichiarare variabili

Prima di utilizzare una variabile bisogna dichiarare il nome e il tipo

- Dichiarazione semplice:
  ```
  int x;
  ```
- Dichiarazione multipla:
  ```
  char primaLettera, secondaLettera;
  ```
- Dichiarazione con inizializzazione:
  ```
  float pi = 3.14;
  ```
- Dichiarazione multipla con inizializzazione:
  ```
  int a = 1, b = 3, x;
  ```

Assegnare variabili

L'operazione principale che riguarda le variabili è l'assegnamento

```
nome = valore;
nome = espressione;
```

- A sinistra c'è un `nome` di variabile (dichiarata in precedenza)
- A destra un `valore` o un'`espressione`
  - dello stesso tipo della variabile!
- L'operazione cambia il valore associato alla variabile
- Il nuovo valore sostituisce quello vecchio

Usare le variabili: espressioni

Un'espressione è una combinazione di valori, variabili, ed operatori

- Operatori aritmetici:
  - `3 + 2`
  - `x - 5`
  - `b * h`
- Divisione:
  - intera tra `int`: `14 / 3` dà come risultato `4`
  - razionale tra `float`: `14.0 / 3` dà come risultato `4.667`
- Modulo (resto della divisione tra `int`):
  - `n % 2`
- Raggruppati usando le parentesi:
  - `(x + y) * 6`

Input/Output

`#include<iostream>` ci mette a disposizione due variabili speciali per scrivere valori sullo schermo e leggerli dalla tastiera

- `std::cout` (Console OUTput)

```
std::cout << "Hello World!";
std::cout << x;
std::cout << "La somma e':" << x + y << std::endl;
```

- `std::cin` (Console INput)

```
std::cin >> x;
std::cin >> nome;
```

- Le conversioni tra i tipi vengono gestite in modo automatico
- Con `using namespace std;` all'inizio del programma non occorre scrivere `std::`

Espressioni Booleane

- Sono espressioni di tipo `bool`
- Possono assumere i valori `true` e `false`
- Costruite a partire dagli operatori relazionali

  `x == y` il valore di `x` è uguale al valore di `y`
  `x != y` il valore di `x` è diverso dal valore di `y`
  `x < y` il valore di `x` è minore del valore di `y`
  `x <= y` il valore di `x` è minore o uguale al valore di `y`
  `x > y` il valore di `x` è maggiore del valore di `y`
  `x >= y` il valore di `x` è maggiore o uguale al valore di `y`
- combinati con gli operatori logici (e le parentesi):

  `e1 && e2` vera se sia `e1` che `e2` sono vere
  `e1 || e2` vera se `e1` è vera oppure `e2` è vera
  `! e1` vera se `e1` è falsa, e falsa se `e1` è vera

Esecuzione condizionale

Per scrivere programmi che siano utili, abbiamo quasi sempre bisogno di:

- controllare condizioni e
- cambiare il comportamento del programma di conseguenza

Le istruzioni di esecuzione condizionale ci danno questa possibilità:

```
if(x > 0) {
    cout << "x e' positivo" << endl;
}
```

- L'espressione booleana dopo `if` è chiamata condizione
- Se è vera, le istruzioni tra parentesi graffe vengono eseguite
- Se è falsa non succede niente

Esecuzione alternativa

Una forma più complessa di `if` è l'esecuzione alternativa:

```
if(x > 0) {
    cout << "x e' positivo" << endl;
} else {
    cout << "x e' zero o negativo" << endl;
}
```

- Se la condizione è vera, allora le istruzioni poste nel ramo `if` vengono eseguite
- Se è falsa si eseguono le istruzioni del ramo `else`

Condizionali annidati


```
if(x > 0) {
    cout << "x e' positivo" << endl;
} else {
    if(x == 0) {
        cout << "x e' zero" << endl;
    } else {
        cout << "x e' negativo" << endl;
    }
}
```

- In questo esempio abbiamo un primo `if` con due rami:
  - Il primo ramo contiene una istruzione di stampa
  - Il secondo ramo (`else`) contiene un altro condizionale, con due rami a sua volta

Esempio: risolvere un'equazione

Scrivere un programma che prende come parametri i coefficienti di un'equazione di primo grado ($ax + b = 0$) e scrive sullo schermo: `indeterminata`, oppure `impossibile`, oppure la soluzione dell'equazione.

Esempio: risolvere un'equazione

Scrivere un programma che prende come parametri i coefficienti di un'equazione di primo grado ($ax + b = 0$) e scrive sullo schermo: `indeterminata`, oppure `impossibile`, oppure la soluzione dell'equazione.

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

int main() {
    float a, b;

    cin >> a;
    cin >> b;
    if(a == 0) {
        if(b == 0) {
            cout << "indeterminata" << endl;
        } else {
            cout << "impossibile" << endl;
        }
    } else {
        cout << -b/a << endl;
    }
}
```