Programmazione 1

UniVR - Dipartimento di Informatica

Fabio Irimie

1° Semestre 2023/2024

Indice

1 Introduzione

L'informatica è la scienza che si occupa della rappresentazione (estrarre le informazioni rilevanti: quali, cosa, quante) ed elaborazione (come si arriva passo dopo passo a risolvere un problema) dell'informazione. È una scienza con un approccio rigoroso e formale.

1.1 Algoritmo

Dopo aver raccolto le informazioni del problema che si vuole risolvere si procede con la costruzione di un algoritmo che risolva il problema. Questo algoritmo viene scritto utilizzando un linguaggio di programmazione (C, Java, ...).

Definizione 1.1
Dato un esecutore e un problema l'algoritmo è una sequenza finita e ordinata di istruzioni elementari, comprensibili dall'esecutore, che risolvono il problema dato.

1.1.1 Proprietà

- **Sequenza**: l'ordine è importante,
- **Finita**: deve terminare,
- **Deterministico**: deve essere preciso,
- **Corretto**: deve risolvere il problema,
- **Completo**: deve considerare tutti gli aspetti del problema,
- **Generale**: deve risolvere tutti i casi del problema,
- **Efficienza**
  - **Tempo**: deve terminare in un tempo ragionevole,
  - **Spazio**: deve utilizzare una quantità di memoria ragionevole.

1.2 Linguaggio pseudo-formale

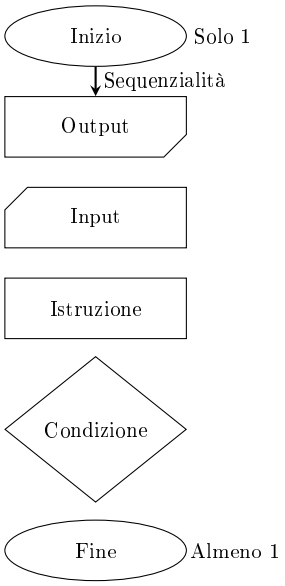
Il linguaggio si divide in:

- **Vocabolario**: insieme di simboli che possono essere usati,
- **Sintassi**: regole per la costruzione di frasi,
- **Semantica**: significato delle frasi corrette.

2 Diagrammi di flusso

I seguenti simboli vengono utilizzati per la costruzione di un diagramma di flusso che rappresenta un algoritmo:

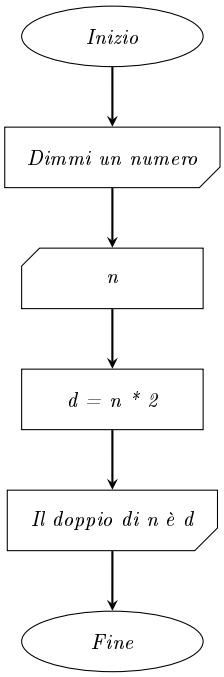
- **Inizio/fine**: rappresenta l'inizio e la fine dell'algoritmo,
- **Istruzione**: rappresenta un'istruzione,
- **Condizione**: rappresenta una condizione,
- **Sottoprogramma**: rappresenta un sottoprogramma.



3 Esercizi con diagrammi di flusso

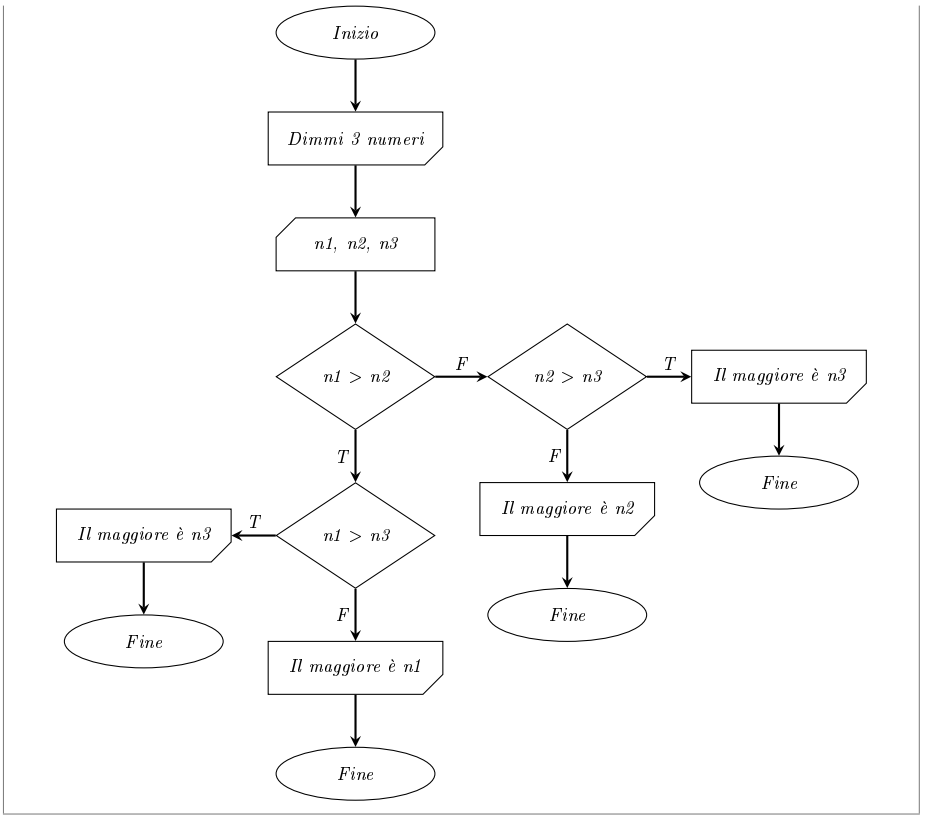
3.0.1 Dato un numero dare in output il doppio

Esercizio 3.1
Stampa il doppio di un numero:



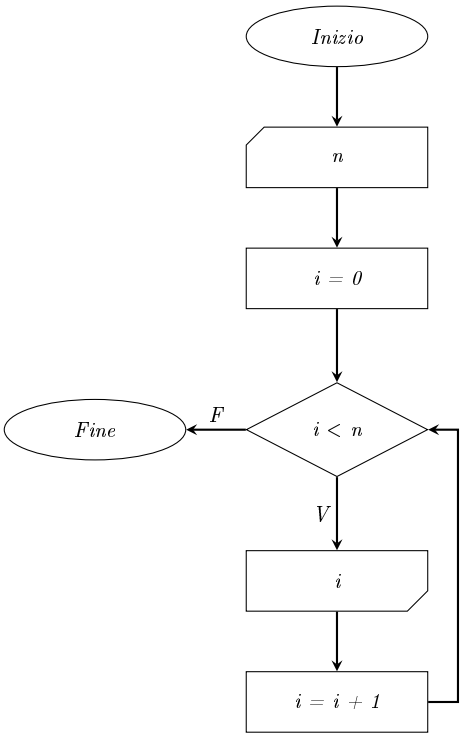
3.0.2 Dati 2 numeri dire il maggiore

Esercizio 3.2
Stampa il maggiore tra 2 numeri:



3.0.3 Stampa tutti i numeri naturali fino a n

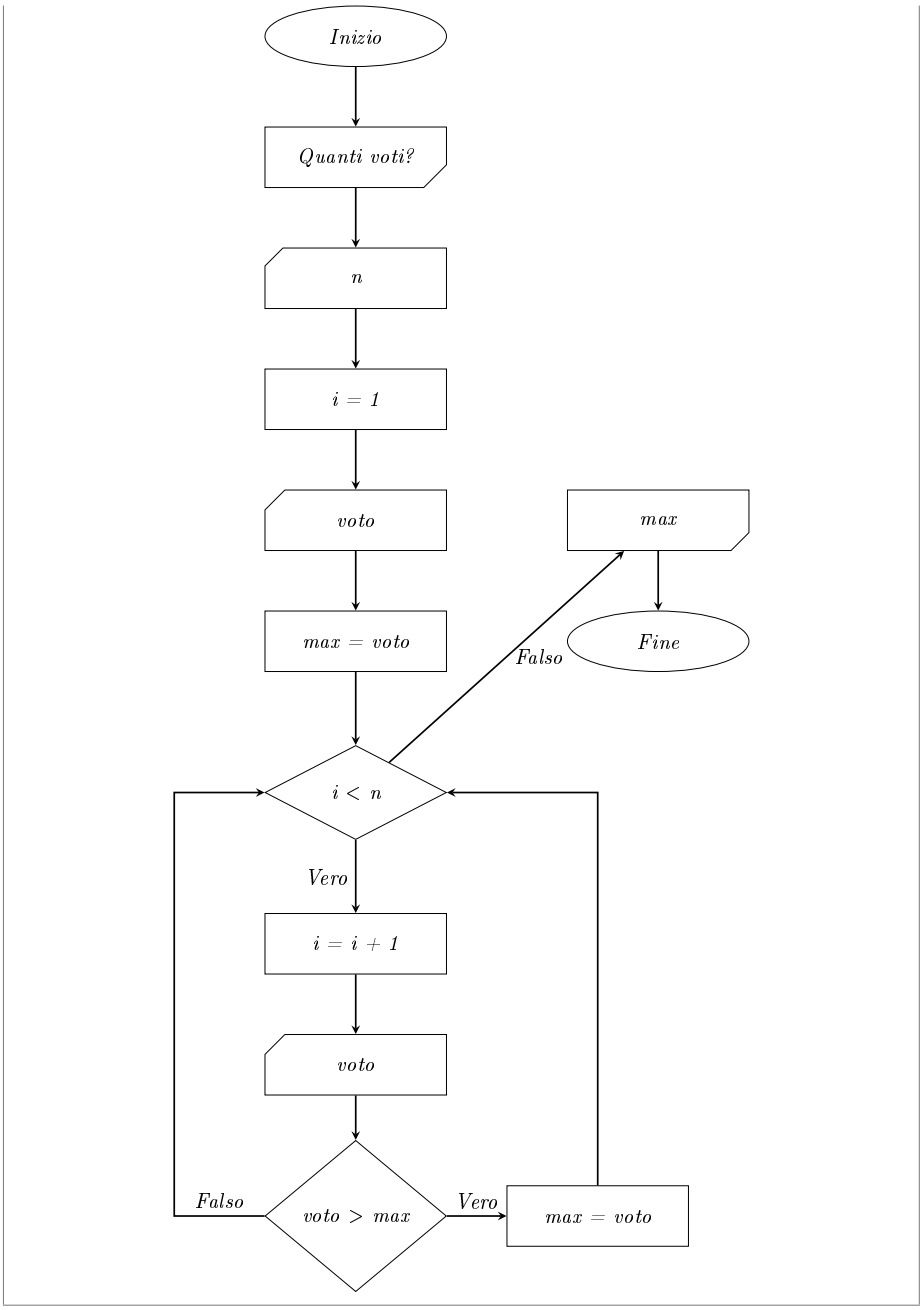
Esercizio 3.3
$\forall i \in [o, n]$ *stampa a video i:*



Questo è un ciclo a condizione iniziale, cioè appena la condizione diventa falsa si esce dal ciclo.

3.0.4 Trovare il più grande tra n numeri

Esercizio 3.4
Stampa il maggiore tra n numeri:

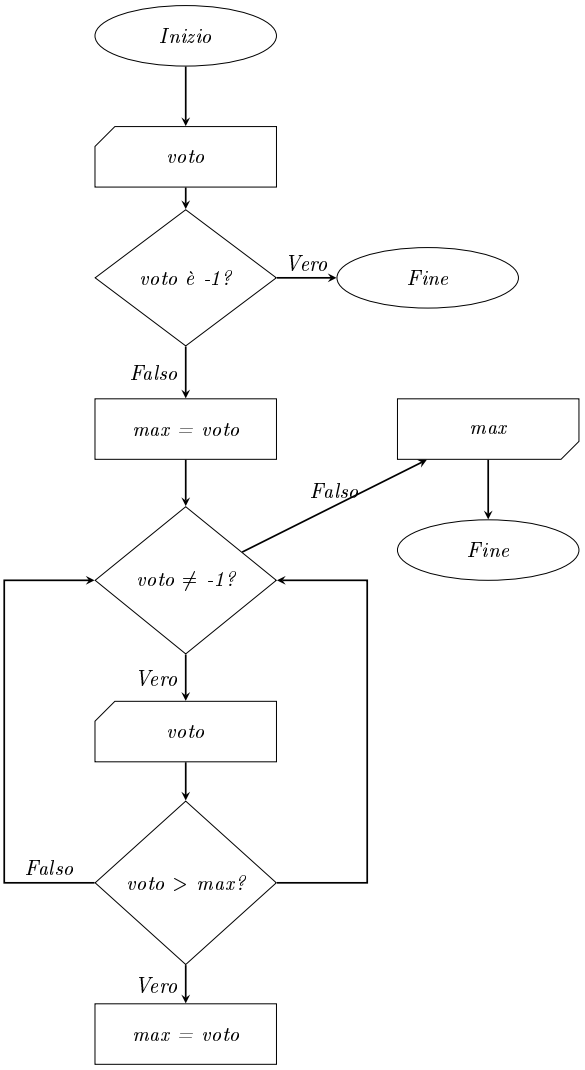


3.1 Ciclo guidato da sentinella

È un ciclo che termina quando viene inserito un valore particolare.

3.1.1 Trova il voto più alto tra un insieme di voti di cardinalità a priori non specificata (terminare quando viene inserito -1)

Esercizio 3.5
Trova il voto più alto tra un insieme di voti di cardinalità a priori non specificata,terminando quando viene inserito -1



4 Linguaggio di programmazione C

È un linguaggio di programmazione:

- **General purpose**: può essere utilizzato per risolvere qualsiasi problema.
- **Dichiarativo**: si dichiarano le variabili e si specifica cosa fare con esse.
- **Imperativo**: si specifica come risolvere il problema.
- **Case sensitive**: distingue tra maiuscole e minuscole.
- **Compilato**: il codice sorgente viene tradotto in codice macchina.

4.1 Catena di programmazione

1. **Editing**: si scrive il programma con un editor
2. **Pre-processing**(Pre-processore): inclusione di librerie.
3. **Compilazione**(Compilatore): traduzione il programma C in codice oggetto comprensibile dal calcolatore. Vengono segnalati errori (statici) e se non ci sono errori viene generato un file eseguibile.
4. **Linking**: collegamento il codice oggetto di programma con le librerie.
5. **Loading**: il programma va caricato nella RAM prima dell'esecuzione
6. **Esecuzione**: il programma viene eseguito.

Un esempio di codice in C è il seguente:

```c
#include <stdio.h>

int main() {
      printf("Hello World!");
      return 0;
   }
```

4.2 Dichiarazione di variabili

Le variabili sono di 2 tipi:

- **Built-in**: sono già presenti nel linguaggio.
- **User-defined**: definite dall'utente.

Tipo: L'insieme dei valori ammissibili più le operazioni ammesse sull'insieme

- Intero:
  - `int` 16bit in complemento a 2
  - `long` 32bit in complemento a 2
- Reale:
  - `float` 32bit in virgola mobile
  - `double` 64bit in virgola mobile
- Carattere: `char` 8bit ASCII

Il codice è diviso in:

- **Parte dichiarativa**: dichiarazione delle variabili
- **Parte esecutiva**: istruzioni che vengono eseguite

Il seguente codice è un esempio che mostra la separazione delle 2 parti:

```c
#include <stdio.h>

int main() {
    //Parte dichiarativa
    int lato;
    int perimetro, area;
    char t;
    float media;

    //Parte esecutiva
    lato = 4;
    perimetro = lato * 4;

    return 0;
}
```

4.2.1 Operatori

In C le operazioni disponibili sono:

- + somma
- - sottrazione
- * moltiplicazione
- / divisione
- % resto della divisione intera

Un esempio di codice che usa gli operatori è il seguente:

```c
#include <stdio.h>

int main() {
    int a, b;
    float media;
    float bonus;

    a = 25;
    b = 26;
    media = (a + b) / 2; // 25.0
    return 0;
}
```

Con a e b interi la divisione è fatta per intero, quindi il risultato è 25 che poi viene assegnato alla variabile float *media*, quindi il risultato è 25.0. Per avere il risultato corretto bisogna dichiarare a e b come float. Un'altra opzione sarebbe di rendere float il 2 del divisore, in questo modo il risultato sarebbe: $(a + b) = 2.0 = 25.5$

Definizioni utili 4.1

```
t = 'a'; // Assegnato il carattere a alla variabile t
t = a; // Assegnato il valore della variabile a alla variabile t
```

4.3 Output (Stampa a video standard)

Per stampare a video si utilizza la funzione `printf()`:

```
printf("Hello World!\n"); // \n va a capo
printf("Hello \t world"); // \t tabulazione (4 spazi)
```

Per stampare il valore delle variabili:

```
printf("Il valore di a e' %d\n", a); // %d indica un intero
printf("Il valore di a e' %d e di b e' %d\n", a, b);

// %c indica un carattere
// %f indica un float
// %.2f indica un float con 2 cifre decimali (solo in output)
// %g indica un double
```

4.4 Input da tastiera

Per leggere da tastiera si utilizza la funzione `scanf()`:

```
scanf("%d", &a); // &a indica l'indirizzo di memoria di a
scanf("%d %d", &a, &b);
```