Unità F1

Primi programmi

Obiettivi

- Conoscere il significato di **dichiarazione** e **definizione** di variabili
- Conoscere i tipi di dato numerici
- Essere in grado di realizzare semplici algoritmi in pseudolinguaggio e **programmi C/C++** a struttura sequenziale
- Essere in grado di inserire **commenti**, dichiarare **variabili**, usare **costanti**, chiamare **funzioni di libreria**, **comunicare** e **ricevere** dati
- Conoscere le fasi della programmazione: dalla stesura del codice sorgente, all'esecuzione, alla correzione degli errori

Il linguaggio C

- Nel 1972 Dennis Ritchie nei Bell Laboratories progettò il **linguaggio C**
- Il linguaggio possiede tutti i costrutti di controllo dei linguaggi ad alto livello, ma consente anche operazioni tipiche dei linguaggi macchina.
- Il **codice è efficiente** (ottime prestazioni in termini di velocità di esecuzione)
- È un linguaggio essenziale, che può essere facilmente esteso senza limiti grazie a **librerie esterne**.
- È probabilmente il **linguaggio più diffuso** al mondo per quantità di codice scritto.

Il linguaggio C++

- Nel 1979 nei Bell Laboratories Bjarne Stroustrup progettò un'**estensione** del linguaggio, che venne chiamato "C con Classi" e che nel 1983 fu ribattezzato **C++** (**C plus plus**).
- Mantiene intatte tutte le caratteristiche del C: i compilatori C++ accettano correttamente tutti i programmi C
- C++ estende il linguaggio C introducendo strumenti per la **programmazione orientata agli oggetti**.
- Il C/C++ è un linguaggio **standard** definito ANSI/ISO (l'ANSI è l'Istituto nazionale americano per gli standard mentre l'ISO è l'Organizzazione per gli standard internazionali) e la sua ultima revisione risale al 1999.

Pseudolinguaggio

Pseudolinguaggio

```
Algoritmo Pitagora
/* Calcolo dell'ipotenusa */

catetol Di Tipo Reale
cateto2 Di Tipo Reale
ipotenusa di tipo Reale
catetol <- 3
cateto2 <- 4
ipotenusa <- RadiceQuadrata(catetol*catetol + cateto2*catetol)
scrivi (ipotenusa)

FineAlgoritmo Pitagora
```

- La coppia di parole chiave **Algoritmo** e **FineAlgoritmo** definiscono il punto iniziale e finale dell'esecuzione.
- Ogni algoritmo espresso in pseudolinguaggio sarà strutturato seguendo la seguente sintassi:

```
Algoritmo <Nome_Algoritmo>
<Corpo dell'algoritmo>
<Nome_Algoritmo>
FineAlgoritmo
```


Linguaggio C

Programma

```
/* Calcolo dell'ipotenusa */
#include <stdio.h>
#include <math.h>

main()
{
   float catetol;
   float cateto2;
   double ipotenusa;

   catetol = 3;
   cateto2 = 4;
   ipotenusa = sqrt(catetol*catetol+cateto2*cateto2);
   printf("%f", ipotenusa);
}
```

Commenti

- All'inizio del programma abbiamo un **commento** che ne indica lo scopo:
  ```
  /* Calcolo dell'ipotenusa */
  ```
- I commenti possono essere inseriti in qualsiasi punto del programma, iniziano con **/*** e terminano con ***/**. Tutto ciò che sta tra questi estremi, anche se occupa più linee, è ignorato dal compilatore.
- In alternativa il commento può essere preceduto da **//**
  ```
  // Calcolo dell'ipotenusa
  ```
  e terminare a fine riga.
- Anche nel nostro pseudolinguaggio si possono utilizzare queste stesse modalità per i commenti.

Il programma

- L'**esecuzione** di ogni programma parte sempre da **main()** seguito da una parentesi graffa aperta, **{**, che definisce l'inizio delle istruzioni del programma
- e termina con una parentesi graffa chiusa, **}**, che definisce la fine del programma.
- All'interno delle parentesi graffe troviamo una serie di istruzioni che verranno eseguite in **sequenza**;
- ogni istruzione termina sempre con un punto e virgola.

```
main()
{
  <Corpo del programma>
}
```


Variabili (dichiarazione)

- Una variabile è un **nome** che rappresenta una **locazione di memoria**, mentre il dato memorizzato in quella locazione è il **valore** della variabile.
- Prima di utilizzare una variabile è necessaria una **fase dichiarativa** in cui viene associato al **nome** il **tipo** dei valori che la variabile può contenere.
- Sintassi della dichiarazione in pseudolinguaggio:
  ```
  <nome_variabile> di tipo <tipo>
  ```
  Esempio
  ```
  cateto1 Di Tipo Reale
  ```
- Sintassi della dichiarazione in C:
  ```
  <tipo> <nome_variabile>;
  ```
  Esempio
  ```
  float cateto1;
  ```


Case sensitive

- Il linguaggio C è **case-sensitive**: le lettere maiuscole sono diverse da quelle minuscole.
- La variabile identificata da **cateto1** non ha niente a che vedere con **Cateto1** o **CATETO1**, così come **main** è diverso da **Main**.

Variabili (definizione)

- Le dichiarazioni provocano la **definizione delle variabili**: la creazione di uno spazio di memoria centrale a loro riservato.
- Il **tipo** di una variabile determina lo spazio di memoria che viene riservato (la dimensione dipende dall'implementazione del linguaggio).

Tipi di dato numerici

TABELLA 1 Tipi di variabile

Tipo	Range
int	Se l'implementazione del linguaggio gli riserva 4 byte (32 bit) sono ammessi numeri interi da –2147483648 a +2147483647.
float	Se l'implementazione riserva 4 byte sono ammessi numeri decimali positivi e negativi da 3.4E–38 a 3.4E+38, approssimativamente tra 10 elevato alla –38 e 10 alla +38. La lettera E significa 10 elevato al numero che la segue, 3.4E – 38 significa 3,4 moltiplicato 10 elevato alla –38.
double	Lo standard impone che per il tipo double venga riservato uno spazio di memoria maggiore uguale al tipo float; spesso le implementazioni gli riservano il doppio dello spazio. Se l'implementazione riserva 8 byte sono ammessi numeri decimali positivi e negativi da 1.7E–308 a 1.7E+308.

Dichiarazione di più variabili

- Le variabili dello stesso tipo possono essere dichiarate facendo seguire all'indicatore del tipo i loro nomi separati da una virgola.

```
float cateto1, cateto2;
```

- Anche nel nostro pseudolinguaggio utilizzeremo le stesse convenzioni per i nomi e per la dichiarazione delle variabili.

```
variabile, variabile,.. Di Tipo tipo
```


I nomi (identificatori) della variabili

- I **nomi** o **identificatori** non possono essere parole chiave del linguaggio, per esempio non possiamo definire una variabile **int**, **float**, **double**.
- Iniziano sempre con una lettera o con un carattere di sottolineatura _ e sono composti da lettere, cifre e _.
- Il numero dei caratteri non è limitato ma solo i primi *n* caratteri del nome sono significativi, dove *n* dipende dall'implementazione:
  - almeno 247 caratteri devono essere significativi secondo lo standard.

Assegnamento

- La sintassi dell'assegnamento in linguaggio C è:

```
<variabile> = <espressione>;
```

- Il simbolo **=** identifica l'operatore di assegnamento, corrisponde a ← nello pseudolinguaggio.
- Un'espressione può essere costituita da
  - un **valore costante**

```
cateto1 = 3;
```

  - valori costanti e **operatori matematici**

```
ipotenusa = 3*3+4*4;
```

  - una **variabile**

```
cateto1 = cateto2;
```

  - **più variabili**

```
ipotenusa = cateto1*cateto1+cateto2*cateto2;
```

  - **funzioni**

```
ipotenusa = sqrt(cateto1*cateto1+cateto2*cateto2);
```


Funzioni predefinite

- Una **funzione** è costituita da un insieme di istruzioni che svolgono un determinato compito.
- Possiamo pensarla come un programma che riceve valori in entrata, effettua elaborazioni e restituisce un valore in uscita.
- Il linguaggio C affida a **funzioni predefinite** molti compiti operativi.
- Per esempio la funzione **sqrt(valore)** restituisce la radice quadrata del valore presente tra le parentesi tonde.

```
ipotenusa = sqrt(cateto1*cateto1+cateto2*cateto2);
```

- La **libreria del linguaggio** che contiene la funzione predefinita **sqrt()**, come molte altre **funzioni matematiche**, è la libreria standard **math.h**.
- Per poter utilizzare le funzioni definite al suo interno, il programma deve essere preceduto dall'inclusione del **file d'intestazione math.h**:

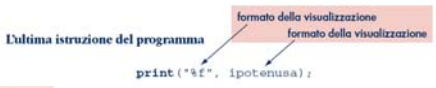
```
#include <math.h>
```


La libreria math

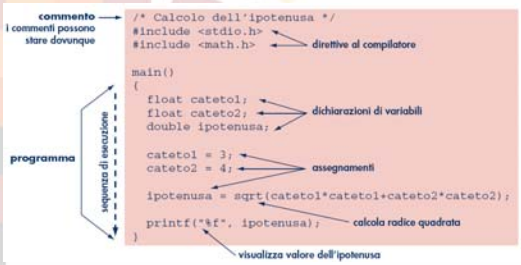
TABELLA 2 Funzioni di math.h

Funzione di math.h	Funzione matematica
sin(x)	seno di x
cos(x)	coseno di x
tan(x)	tangente di x
sinh(x)	seno iperbolico di x
cosh(x)	coseno iperbolico di x
log(x)	logaritmo in base E di x
log10(x)	logaritmo in base 10 di x
sqrt(x)	radice quadrata di x
pow(x, y)	x elevato alla potenza y

Output



Il programma completo

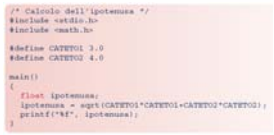


Costanti

- Le costatnti si definiscono mediante la direttiva del compilatore **#define** seguita dal nome della costante e dal suo valore.

```
#define CATETO1 3
```

- Sulle costanti non potrà essere effettuata l'operazione di assegnamento.



Convenzioni sui nomi (identificatori)

- Il nome di una costante segue le stesse regole degli altri identificatori.
- Per distinguere nel programma le costanti dalle variabili, spesso si preferisce utilizzare solo i caratteri maiuscoli per le costanti e i caratteri minuscoli, se necessario intercalati da caratteri maiuscoli, per le variabili.
- In base a questa convenzione, **LUNGHEZZA** e **PARTENZA** sono costanti, **temperatura** e **votoStudente** variabili.

Scrittura

- **La sintassi dell'operazione di scrittura in Pseudolinguaggio:**

```
Scrivi (<espressione>)
```

- Il linguaggio C **fornisce la funzione di output printf**, che offre molte **possibilità di scrittura formattata**.
- Tutto ciò che appare tra i doppi apici, esclusi i caratteri che seguono immediatamente il carattere **%** (che indicano il formato), viene visualizzato, per cui:

```
printf("Ipotenusa cm. %f", Ipotenusa);
```

- Visualizza

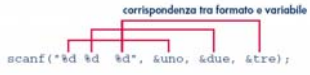
```
Ipotenusa cm. 5.000000
```


Lettura

- **La sintassi dell'operazione di lettura in Pseudolinguaggio:**

```
Leggi (<nome_variabile>)
```

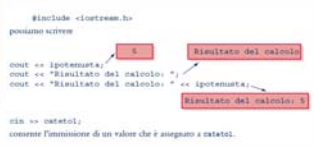
- In modo analogo il linguaggio C fornisce la funzione di input **scanf**.



Il programma completo

Input output in C++

- In C++ **cin** (console input) e **cout** (console output) rappresentano rispettivamente lo standard input e lo standard output.
- Oltre a **scanf** e **printf** definite in **stdio.h**, si hanno altre possibilità per leggere e scrivere grazie a **cin** e **cout** definite in **iostream.h**

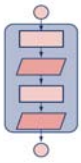


Il programma in C++

```
/* Calcolo dell'ipotenusa */
#include <iomanip.h>
#include <math.h>

main()
{
float cateto1, cateto2, ipotenusa;

   cout << "Calcolo ipotenusa\n\n";
   cout << "Primo cateto: ";
   cin >> cateto1;
   cout << "Secondo cateto: ";
   cin >> cateto2;

   ipotenusa = sqrt(cateto1*cateto1+cateto2*cateto2);

   cout << "Ipotenusa: " << ipotenusa << "\n";
   }
```


Sequenza

- Tutti gli algoritmi e i programmi equivalenti che abbiamo visto sono composti da una **sequenza** di istruzioni che si susseguono a partire dall'istruzione iniziale fino a quella finale
- In nessun modo è possibile alterare il flusso di esecuzione.
- Definiamo questa struttura **sequenza**.



Sequenza in pseudolinguaggio

- Nello **pseudolinguaggio** l'inizio e la fine di un blocco di istruzioni da eseguire in sequenza è definito da apposite parole riservate.
- Fino a questo punto abbiamo incontrato la parola riservata **Algoritmo**, che definisce l'inizio dell'algoritmo e **Fine Algoritmo** che ne determina la fine.
- All'interno abbiamo una sequenza di istruzioni, una per ogni riga.

Sequenza in C/C++

- Nel linguaggio **C** sono le **parentesi graffe** che delimitano una sequenza di istruzioni
- Le parentesi graffe stabiliscono anche l'inizio e la fine del programma.

Fasi della programmazione

Esempio: analisi, progettazione, codifica e test

- **Problema**: determinare la circonferenza e l'area di un cerchio conoscendo il suo raggio.
- **Analisi**
- Le formule risolutive sono:
  Circonferenza = 2*r * π
  Area = r2 * π
  dove *r* è il raggio del cerchio e π è la costante 3,14.
- L'analisi ci porta a definire l'input e l'output del programma.
- **Problema**: calcolo circonferenza e area di un cerchio.
- **Input**: valore del raggio.
- **Ouput**: valori della circonferenza e dell'area.

Progettazione e codifica

- Dopo aver definito le variabili e la sequenza di istruzioni da eseguire formuliamo la seguente codifica:

Pseudolinguaggio

```
/* Area e circonferenza di un cerchio */
Algoritmo Cerchio
raggio, circonferenza, area   di tipo Reale;
costante PIGRECO = 3.14;

Scrivi "Raggio: "
Leggi raggio

circonferenza ← 2*raggio* PIGRECO
area ← raggio*raggio* PIGRECO

Scrivi "Circonferenza: ", circonferenza
Scrivi "Area: ", area
FineAlgoritmo
```


Codifica in linguaggio C

Linguaggio C

```
/* Area e circonferenza di un cerchio */
#include <stdio.h>
#include <math.h>

#define PIGRECO 3.14

main()
{
  float raggio;
  double circonferenza, area;

  printf("Raggio: ");
  scanf("%f", &raggio);

  circonferenza = 2 * raggio * PIGRECO;
  area = pow(raggio, 2) * PIGRECO;

  printf("Circonferenza: %f\n", circonferenza);
  printf("Area: %f\n", area);
}
```


Codifica in linguaggio C++

Linguaggio C++

```
/* Area e circonferenza di un cerchio */
#include <iomanip.h>
#include <math.h>

#define PIGRECO 3.14

main()
{
  float raggio;
  double circonferenza, area;

  cout<<"Raggio: ";
  cin>>raggio;

  circonferenza = 2 * raggio * PIGRECO;
  area = pow(raggio, 2) * PIGRECO;

  cout<<"Circonferenza: "<<circonferenza<<" \n";
  cout<<"Area          : "<<area;
}
```


Test del programma

- Compiliamo e correggiamo eventuali errori sintattici indicati dal compilatore.
- Eseguiamo il programma con diversi valori in ingresso e verifichiamone il corretto funzionamento
- Confrontiamo i risultati con quelli ottenuti con la nostra calcolatrice.
- Se l'output restituito a video non è di nostro gradimento, miglioriamo la formattazione.