# Laboratorio di Informatica

Corso di laurea triennale in Fisica

Carlo Mereghetti

In questa lezione verranno approfonditi alcuni concetti e strumenti quali:

- lettera e scrittura da/su file,
- utilizzo di array dinamici,
- utilizzo di `struct` per modellare dati,
- modularizzazione del codice mediante function.

## I Numeri Complessi

Un *numero complesso* ha la forma

$$z = a + ib$$

dove $a, b$ sono numeri reali denominati, rispettivamente, parte *reale* e parte *immaginaria* di $z$ mentre $i = \sqrt{-1}$ è l'*unità immaginaria*. Vale ovviamente $i^2 = -1$. Dati due numeri complessi $z_1 = a + ib$ e $z_2 = c + id$, si definiscono le seguenti operazioni realizzabili mediante le usuali regole di somma e prodotto di binomi:

- **Somma:** $z_1 + z_2$ è il numero complesso le cui parti reale e immaginaria si ottengono da $(a + ib) + (c + id)$. Ad esempio, la somma di $(1 - i5) + (3.5 + i4)$ è $4.5 - i$.
- **Prodotto:** $z_1 \cdot z_2$ è il numero complesso le cui parti reale e immaginaria si ottengono da $(a + ib) \cdot (c + id)$. Ad esempio, il prodotto di $(1 - i5) \cdot (3.5 + i4)$ è $23.5 - i13.5$.

Dato il numero complesso $z = a + ib$, si definisce inoltre:

- **Complesso coniugato:** $z^*$ è il numero complesso $a - ib$.
- **Modulo:** $|z|$ è il reale positivo definito come $\sqrt{z \cdot z^*} = \sqrt{a^2 + b^2}$.

Un tipo per definire variabili contenenti numeri complessi potrebbe essere il seguente:

```
struct complesso {
     float re;
     float im;
};
```

Ad esempio, la dichiarazione della variabile

```
complesso z;
```

consente di memorizzare in `z` il numero complesso $4.5 - i3.2$ assegnando, rispettivamente, 4.5 al campo `re` e $-3.2$ al campo `im` di `z` (ricordate che ai campi di una variabile `struct` si accede mediante la sintassi `variabile.campo`).

ATTENZIONE!!! Nel codice del programma che viene chiesto di seguito, conviene mettere la definizione del tipo `struct complesso` all'esterno del `main`, in modo che ogni funzione o prototipo definiti nel programma possa dichiarare variabili e parametri di tipo `complesso`.

Tutti i file (sorgenti ed eseguibili) prodotti andranno salvati sotto la directory `Lab6`.

## Il Programma

Per la gestione dei *file*, ricordate la libreria `<fstream>` la quale mette a disposizione variabili di tipo `ifstream` per aprire file in lettura e `ofstream` per aprire file in scrittura. Gli *array dinamici* si dichiarano con la sintassi `tipo *nome = new tipo[dimensione]` e si utilizzano come array tradizionali; ad esempio `nome[3]` denota la componente di posto 3 nell'array `nome`.

Il file `complessi.in`, che potete copiare nella vostra cartella di lavoro, contiene un certo numero di numeri complessi. In particolare, parte reale e parte immaginaria di ogni numero complesso sono espresse per riga separate da uno spazio.

1. Caricate in un array dinamico di opportuna dimensione e tipo base `complesso` i numeri complessi nel file `complessi.in`. Per calcolare la dimensione di tale array, sarà sufficiente contare le righe del file `complessi.in` utilizzando in un ciclo la funzione `getline( in, s )` dove `in` è lo `stream` associato all'apertura del file `complessi.in` mentre `s` è una variabile di appoggio di tipo `string`. Una volta contate le righe di `complessi.in` *ricordate di ritornare all'inizio del file* per la lettura dei dati dei numeri complessi. Ciò si ottiene mediante la sequenza di comandi

   ```
   in.clear();
   in.seekg( 0 );
   ```

   oppure, più prosaicamente, chiudendo e riaprendo il file `complessi.in`. Ricordate anche che è buona norma controllare che il file `complessi.in` esista e, in caso contrario, interrompere il programma restituendo al `main` il codice `-1`.

2. Esaminando l'array di complessi appena caricato, scrivete sia a video che in un file dal nome `complessi.out` i seguenti dati opportunamente preceduti da una riga di spiegazione (didascalia):

   - Il coniugato di ogni numero complesso il cui modulo supera `3`.
   - La somma di tutti i numeri complessi con parte reale negativa.
   - Il prodotto dei primi tre numeri complessi.

   Nel file `complessi.out` i numeri complessi vanno salvati nello stesso formato in cui erano presenti nel file di input `complessi.in`, cioè, parte reale e immaginaria su una riga separate da spazio.

Per modularizzare la struttura del codice, prevedete la scrittura delle seguenti function:

- `complesso coniugato( complesso a )`: restituisce il complesso coniugato di `a`;
- `complesso somma( complesso a, complesso b )`: restituisce la somma di `a` e `b`;
- `complesso prodotto( complesso a, complesso b )`: restituisce il prodotto di `a` e `b`;
- `double modulo( complesso a )`: restituisce il modulo di `a`;
- `void stampa( complesso a )`: stampa a video `a` nella forma $3.5 - i4.6$.

Tali funzioni vanno scritte dopo il `main`, ricordando di riportare prima del `main` il loro *prototipo*.

La pagina seguente mostra come dovrebbe apparire il file `complessi.out` .....

Contenuto del file `complessi.out`

```
Coniugato dei complessi con modulo superiore a 3
-3.31   0.51
-4.44   -5.63
2.27    -2.08
4.31    -0.13
6.04    0.76
7.75    -0.6

Somma dei complessi con parte reale negativa
-20.21  4.37

Prodotto dei primi tre complessi
-8.82058        6.65561
```