Laboratorio di Informatica

Corso di laurea triennale in Fisica

Carlo Mereghetti

In questa lezione, inizieremo a scrivere software *modulare*. In sostanza, i nostri programmi non saranno più costituiti da un unico `main` ma conterranno in aggiunta una collezione (*libreria*) di funzioni che verranno opportunamente richiamate nel `main`. Ognuna di queste funzioni risolve un sotto-problema del problema generale, in accordo con l'approccio TOP-DOWN alla progettazione del software. Quindi, i nostri sorgenti `.C` saranno ora grosso modo composti dalle seguenti sezioni:

```
#include <...>              // INTESTAZIONE
#include <...>
...
using namespace std;

<eventuali dichiarazioni di variabili e struct>

tipo1 funzione1(...);
tipo2 funzione3(...);
tipo3 funzione3(...);

int main() {                // MAIN
        ...
        funzione1(...);
        ...
        funzione2(...);
        ...
        funzione3(...);
        ...
        return 0;
}


tipo1 funzione1(...) {      // FUNZIONI
        ...
}

tipo2 funzione2(...) {
        ...
}

tipo3 funzione3(...) {
        ...
}
```

Per il momento, rispetteremo la seguente convenzione nella scrittura dei sorgenti: la libreria delle funzioni dopo il `main` e l'elenco dei *prototipi delle funzioni* appena prima del main. Inoltre, per ora, scriveremo le librerie di funzioni nello stesso file in cui è presente il `main`. Vedremo in seguito come costruire librerie di funzioni indipendenti da qualsiasi `main` ed esportabili tra programmi.

Nota che le funzioni, oltre che nel `main`, possono anche essere richiamate entro altre funzioni.

Tutti i file (sorgenti ed eseguibili) prodotti andranno salvati sotto la directory `Lab5`.

Punti nel Piano

Acquisite dall'utente 10 punti nel piano cartesiano, le cui ascisse verranno memorizzate nell'array `X` mentre le corrispondenti ordinate verranno memorizzate nell'array `Y`. Dunque, l'insieme dei punti sarà rappresentato dalle coppie di coordinate:

```
(X[0],Y[0]), (X[1],Y[1]), ..., (X[8],Y[8]), (X[9],Y[9]).
```

Pensate al tipo più opportuno con cui definire gli array `X` e `Y`. Ricordiamo che esistono modi infinitamente migliori per rappresentare punti in un piano cartesiano, ma per il momento utilizzeremo questo.

Chiedete ora all'utente di specificare un cerchio nel piano acquisendone le coordinate del centro e il raggio. Definite la `struct cerchio` per memorizzare tale cerchio in una variabile e controllate che l'utente non inserisca un raggio negativo.

Il programma deve:

- Stampare i punti (cioè la coppia `(ascissa,ordinata)`) che giacciono nel cerchio (circonferenza esclusa).
- Stampare la massima distanza tra coppie di punti che giacciono nel cerchio.
- Stampare le coppie di punti nel cerchio che stanno alla massima distanza calcolata al punto precedente.

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

- `float getNonNeg()`: continua a chiedere un numero all'utente fintantochè tale numero è minore o uguale a `0` e ritorna il primo numero non negativo inserito;
- `float dist( float x, float y, float z, float k )`: ritorna la distanza tra i punti `(x,y)` e `(z,k)`;
- `boolean nelCerchio( float x, float y, cerchio c )`: ritorna `true` de il punto `(x,y)` giace nel cerchio `c`, altrimenti restituisce `false`;

Ricordiamo che tali funzioni vanno scritte dopo il `main`, ricordando però di riportare prima del `main` il loro prototipo.

ATTENZIONE!!!

Poiché il prototipo della funzione `nelCerchio` richiede di specificare il tipo `cerchio` del terzo parametro, è necessario che tale tipo appaia *prima* del prototipo. Per tale motivo, è necessario scrivere la definizione della `struct cerchio` all'esterno del `main` prima dei prototipi.