**Pastorino Riccardo – Mat: S4871945**

**1 Documentazione / Commenti**

1.a Generare la documentazione usando Doxygen e verificare se la documentazione prodotta permette di orientarsi nella struttura del progetto software. Indicare i punti che potrebbero essere migliorati.

La documentazione permette facile interpretazione ed orientamento tra le classi.

Si segnalano i seguenti accorgimenti:

* Si nota che, nella voce “classes” di Doxygen, la classe right\_trapezoid risulta duplicata (Il duplicato non rispetta la gerarchia delle classi). Si noti che, nel file.h, la classe derivata è dichiarata come "Right\_trapezoid", tuttavia, nel commento “ /// @class right\_trapezoid ”, la classe è implementata come "right\_trapezoid" con una "r" minuscola come prima lettera.

* Nella “file list” non risultano i file polygon.cpp, rectangle.cpp e rhombus.cpp, in quanto non sono stati documentati con apposita “brief description”;

Non è possibile risalire alla documentazione del file main.cpp. seppure siano presenti i commenti delle variabili di inizializzazione.

* In assenza del file README, non è possibile disporre di una descrizione preliminare del progetto.

1.b Utilizzando la documentazione e i commenti inseriti nei file ‘.cpp’ e ‘.h’ verificare se il progetto software aderisce alle specifiche assegnate durante la prima settimana. Indicare le eventuali specifiche non rispettate.

Le specifiche date vengono rispettate. Si ritiene utile procedere con la compilazione del programma, per un’analisi più dettagliata.

1.c Verificare che il file README introduca correttamente lo scopo del progetto software e che dia sufficienti informazioni per un corretto uso dell’interfaccia a riga di comando. Indicare eventuali mancanze e/o possibili migliorie.

Al momento del clone non è stato pervenuto il file README.

**2 Compilazione e prima sessione di test**

2.a Verificare se è possibile compilare il progetto

2.b Test dell’interfaccia a riga di comando: l’interfaccia funziona correttamente? L’interfaccia è di facile utilizzo? Quali prove sono state eseguite per fare il test?

Sono emersi alcuni problemi in fase di compilazione (in parte dovuti ad alcune ambiguità con il compilatore g++). Di seguito si riportano tutti gli errori di codice riscontrati ed eventuali cause.

Compilazione del main.cpp:

1. “ ‘main’ must return ‘int’ ”:

La funzione main è definita con un return type ‘void’. In C++ sarebbe opportuno ricorrere ad un return di tipo ‘int’ (inserendo il valore di ritorno, ‘return 0’).

1. Errori di tipo “ use of undeclared identifier 'scanf\_s' ”:

La funzione scanf\_s non è una funzione C++ standard bensì è una funzione specifica di Microsoft (per un input più sicuro). Se si utilizza un compilatore non Microsoft, si può non avere ‘scanf\_s’ disponibile. In alternativa si potrebbe usare scanf, che è una funzione C++ standard.

Riguardo i files ‘.cpp’ e ‘.h’ delle classi derivate (file.cpp e file.h), sono apparsi 1 errore e alcuni warnings:

* Classe Rhombus:

Nel file.cpp e file.h è stato riscontrato un warning del tipo “using directive refers to implicitly-defined namespace 'std' ”;

* nel file “right\_trapezoid.cpp”:

in fase di compilazione si ha un errore del tipo “ use of undeclared identifier 'sqrt' ”. Per ovviare a tale errore, potrebbe risultare opportuno includere la libreria “math.h”.

A seguito dei problemi di compilazione sopra citati, sono state attuate apposite correzioni al fine di riuscire a compilare il programma. Risolti gli errori, il programma compilava correttamente.

A questo punto, per il testing dell’interfaccia, è stata adottata una procedura base: sono state create diverse figure geometriche (1 o più per ciascuna figura), inserendo le rispettive dimensioni di lati/diagonali. Il programma non ha presentato problemi, rispondendo correttamente alle richieste, con ulteriore possibilità di visualizzare graficamente le figure (non richiesto nelle specifiche).

All’uso, l’interfaccia risulta essere intuitiva, ordinata e facile all’uso.

**3 Seconda sessione di test**

3.a Definire una procedura di test che permetta di identificare possibili bug nel codice prodotto per le singole classi e i singoli metodi di ogni classe

3.b Riportare i risultati del test

Si vogliono elencare le procedure attuate:

Il primo approccio prevede il testing del menu utente - Sono stati dati in pasto valori negativi e/o caratteri al programma, al fin di testare la sua risposta dinnanzi ad una scelta non prevista.

In particolare, si voleva testare il valore di scelta, per muoversi all’interno del menu, e i valori associati alle dimensioni delle figure.

Nella scelta delle opzioni del menu utente, sono stati inseriti valori > 4 e valori negativi; il programma riporta (correttamente) che la scelta non è idonea. Se invece si digitano dei caratteri, il programma si comporta come se si avesse digitato l’exit code ‘0’.

Per quanto riguarda la scelta delle dimensioni dei poligoni:

* creazione del rettangolo:

per valori negativi e/o per caratteri - warning da parte del costruttore.

* creazione del rombo:

per valori negativi e/o caratteri – genera errore;

* creazione del trapezio rettangolo

per valori negativi e/o caratteri – genera errore;

Alcuni accorgimenti:

Stando a quanto riportato nel main.cpp, risulta un problema logico nel ciclo ‘while (true)’ - il programma non sarà in grado di stampare 100 figure - Il ciclo continuerà a essere eseguito indefinitamente fino a quando non verrà rilevata un'istruzione ‘return’. Tuttavia, non esiste alcuna condizione o meccanismo per uscire dal ciclo. Di conseguenza, si genererà un’eccezione e il programma continuerà a stampare oltre le 100 figure.

Eventuale soluzione: si pensava alla creazione di una variabile che, come un contatore, si aggiorna ogni volta che viene stampata una figura, in modo tale da arrestare il ciclo.

Inoltre, un trapezio rettangolo non può essere tale, se una base non è maggiore dell’altra; poteva essere utile includere anche questo controllo. Dall’interfaccia, risulta che si può inserire stesso valore per le 2 basi.