**Demarchi Alessandro 5193343**

**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.

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.

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.

**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?

**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

**1 Documentazione / Commenti**

1.a La documentazione Doxygen è stata pensata correttamente, ogni file presenta una adeguata descrizione per ogni funzione, e ognuna di esse è inserita in una apposita sezione (constructors/destructor, basic handling…). Con una eccezione, il file main.cpp, nel quale non vi è alcun commento dedicato a doxygen: infatti, sarebbe corretto, a mio parere, inserire una breve descrizione anche di questo file, che è quello che permette di comprendere il funzionamento di tutto il progetto attraverso il testing. Sempre in questo file i commenti sono pochi, infatti si limitano a descrivere i tre casi di creazione dei poligoni. Invece, a livello progettuale, non capisco l’idea di inserire la funzione Update(), infatti i valori restituiti da GetSide(),Area() e Perimeter() sono generati alla chiamata di queste funzioni e non sono memorizzati in variabili all’interno della classe, perciò non devono essere “aggiornati”.

1.b La maggior parte delle specifiche sono state rispettate: la classe IsoTriangle presenta i parametri base e height nella sezione private, e la funzione GetSide() nella sezione public in maniera tale che possa essere chiamata all’occorrenza ( e non essendo un attributo come base e height ma una funzione non c’è pericolo che il parametro in se venga modificato). La funzione Draw() funziona correttamente e sfrutta a pieno il polimorfismo: è stata definita virtual =0 nella classe base astratta polygon, in maniera che venga obbligatoriamente ridefinita in ogni rispettiva classe derivata. Come da specifiche la funziona stampa una frase che identifica il disegno del poligono, l’area e il perimetro (attraverso la funzione Dump() ). Una specifica che non è stata rispettata è stata quella della quantità, non c’è un vero limite al numero di poligoni creabili, né un messaggio di errore che avverte il raggiungimento di esso. Non sono d’accordo poi sulla scelta dello scrivere in italiano il file main.cpp e una frase nella funzione Draw() in IsoTriangle.cpp, dato che il resto del progetto è scritto in inglese.

1.c Il file README non è stato inserito. Nonostante ciò, l’utilizzo del progetto software è intuitivo (anche per la sua semplicità, se il progetto software fosse stato più complesso e articolato la presenza di questo file sarebbe stata utile per un apprendimento veloce dell’utilizzo di esso).

**2 Compilazione e prima sessione di test**

2.a È possibile compilare correttamente il progetto, sono però presenti 4 warnings che avvertono di 4 getchar() il cui valore viene ignorato.

2.b In un normale utilizzo l’interfaccia funziona correttamente: tutti i tipi di poligoni vengono creati e memorizzati, per poi poter essere “disegnati”. Inoltre, l’interfaccia si presenta, come già detto prima, intuitiva e di facile utilizzo. Ci sono però dei problemi. Sono state eseguite delle prove sul normale funzionamento che il software è supposto avere (creazione e disegno di poligoni), sull’ acquisizione di dati da considerare come non validi e sulla quantità di poligoni creabili.

I test di funzionamento sono stati:

-Creazione di ognuna delle classi derivate di polygon

-Disegno di ognuna delle classi derivate di polygon

-Test dell’acquisizione della scelta: tentare di inserire valori al di fuori dell’intervallo [1 ; 5], valori decimali e caratteri

-Test di inserimento delle dimensioni dei poligoni: tentare di inserire valori al di fuori dell’intervallo [1 ; 5], valori decimali e caratteri

-Inserimento di poligoni oltre al limite massimo (cambiandolo dal codice)

Il progetto sfotware ha superato i test adempiendo al suo compito principale: creare e “disegnare” poligoni, ma con alcuni problemi:

1) Nel momento in cui si crea un poligono e si inseriscono dei valori negativi, o dei caratteri, come dimensioni, vengono giustamente stampate delle righe nell’interfaccia che segnalano l’errore, ma il poligono viene comunque inizializzato con 0 al posto del valore non valido. Ovviamente dipende dall’utilizzo che si fa di questo codice ma, a mio parere, la soluzione più logica sarebbe quella di eliminare il poligono che si sta creando una volta acquisito in ingresso un valore non consono per una dimensione.

2)Invece, nella fase di scelta, se si inseriscono valori con cifre decimali in input, il valore della cifra intera viene trasmesso all’input della scanf() attuale, mentre il valore della cifra decimale viene trasmesso come input alla prossima scanf(). Questo è dovuto al fatto che all’interno della scanf() è stato inserito un “%i”, se fosse stato utilizzato “%f” i valori decimali sarebbero rientrati nel case default.

3)Non esiste un limite massimo di poligoni creabili, né tantomeno un messaggio di raggiungimento di esso.

**3 Seconda sessione di test**

3.a La procedura di identificazione dei bug deve contenere la chiamata ad ogni funzione caratteristica della classe, provando ad inserire input non validi all’interno di esse, come numeri negativi per dimensioni o elementi diversi per l’operatore ==.

3.b Le tre classi rectangle, rhombus e IsoTriangle non hanno riscontrato bug o problemi di alcun tipo.