**Matteo Caroleo**

**S5216938**

**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 Doxygen viene generata correttamente per entrambe le classi, manca però il Doxyfile quindi vengono perse alcune specifiche della documentazione (non essenziali per l’utilizzo).

La documentazione della classe IsoTrapezoid risulta ben ordinata nel documento, la descrizione dei metodi è sintetica ma efficace. La descrizione della funzione reset risulta poco utile. Vale lo stesso per la classe IsoTriangle e Polygon.

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

-) Nel progetto dovevano comparire 3 sottoclassi della classe Polygon: Rhombus, Rectangle e una sottoclasse data singolarmente per gruppo. Di esse ne compaiono solo 2: IsoTriangle & IsoTrapezoid, ciò denota un errore nella comprensione della consegna dell’esercitazione in quanto doveva essere presente solo una delle due, oltre alle classi rhombus & rectangle.

Nella classe IsoTrapezoid è mancante la funzione Side(), Area() & Perimeter().

Compaiono oltre 2 cartelle : html & latex, che sono superflue.

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.

-) Il file README è mancante

---------------------------------------------------------------------------------------------------------------------------

**2 Compilazione e prima sessione di test**

2.a Verificare se è possibile compilare il progetto

-) Il progetto non viene compilato.

Nella classe IsoTrapezoid, nella funzione float GetSide(), manca il valore di ritorno. Inoltre non è stata definita la funzione Side(), quindi in tutte le istanze in cui è chiamata non riesce a compilare.

Nel main, viene incluso il file rhombus.h che non è presente nel progetto.

Nel file IsoTrapezoid.h le guardie sono sbagliate: gli #ifndef controllano un file sbagliato

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?

-) Per testare il programma è stato creato un main differente + un benchmark.

----------------------------------------------------------------------------------------------------------------

**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

La procedura di test consiste nel testare il corretto funzionamento di tutti i metodi della classe IsoTriangle. Non ho potuto testare le altre classi in quanto non compilabili.   
  
Il test prevede la chiamata di diversi metodi, per testare poi il corretto funzionamento viene chiamata la funzione Dump().

In particolare sono stati chiamati i seguenti metodi:

* Costruttore di default
* Costruttore con passaggio parametri positivi
* Costruttore con passaggio a parametri negativi
* Costruttore di copia dell’oggetto
* Operatore di assegnazione
* Operatore di comparazione
* Catena di operatori della classe
* Funzione Init
* Funzione Init di copia
* Funzione Reset
* Funzioni setters (con e senza parametri negativi)
* Funzioni getters
* Funzioni Draw

Inoltre sono state chiamate funzioni relative alla classe poligono per verificare il corretto funzionamento sia della classe che del polimorfismo. Sono stati in particolare provate le funzioni dump, GetArea e GetPerimeter. Infine è stato provato il distruttore.

**Un altro test che è stato compiuto è stato verificare l’efficacia della scelta di calcolare area e perimetro ogni volta che viene aggiornato qualsiasi parametro dell’oggetto.**

Nel codice esaminato, area e perimetro vengono chiamate ogni volta che l’oggetto viene instanziato e in tutti i setter. Viene quindi creato da me un altro IsoTriangle che calcola area e perimetro solo quando vengono richieste e viene confrontato con la precedente classe mettendo a paragone il cpu time di un programma che le utilizza entrambe.

Il test viene eseguito su un altro file .cpp chiamato “benchmark”, che importa sia la classe originale che quella modificata da me.  
Quest’ultima, ora chiamata “new\_isotriangle”, modifica la funziona Init(), SetBase() & SetHeight(), commentando le chiamate alle funzioni area e perimetro. Un’altra modifica è stata rimuovere la stampa del costruttore/distruttore, per avere una console più leggibile.

Il benchmark crea 2 array di 10.000 oggetti diversi: IsoTriangle (originale) & new\_isotriangle. Vengono poi iterati dei cicli for che modificano base e altezza, rispetto a quelli di default, e chiamano la funzione GetArea() & GetPerimeter(). Lo scopo era di simulare un utilizzo realistico di queste quattro funzioni.

3.b Riportare i risultati del test

Della classe IsoTriangle, **tutti i risultati del test sono stati superati con successo,** ossia:

Il costruttore di default viene chiamato correttamente, sono stati inseriti i parametri di default cioè tutti 0.

Il costruttore con parametri controlla correttamente se sono negativi e nel caso inserisce i valori di default

Il costruttore di copia funziona correttamente, viene testato anche l’operatore ‘==’ & ‘=’ , che funzionano correttamente.

I setters funzionano correttamente e controllano che il parametro non sia negativo.

I getters, la dump e la funzione draw funzionano correttamente.

**Per il test sulla performance, iterando il test diverse volte si ottiene un valor medio in percentuale di miglioramento del 50% se non calcolasse ogni volta area e perimetro.**Con il nuovo metodo proposto si hanno però dei momenti nell’esecuzione in cui i parametri di perimetro, area, base e altezza non sono coerenti. Se però si accede al parametro Area con i giusti metodi, è impossibile avere (quando richiesti) un’area non in funzione dei parametri