**Bricarello Luca 5248168**

**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.

Per effettuare l’analisi controllo le sezioni “Class List” e “File List” della documentazione generata

-) **Class list**:

1. **Polygon**

La documentazione della classe **Polygon** è gestita bene, per le funzioni virtualizzate **Area**() e **Perimeter**() dice però che sono state implementate soltanto da **RightTrap** e **NON** dalle Classi **Rectangle** e **Rhombus**

1. **Rectangle**

Anche questa è strutturata bene, però le funzioni **Area**() e **Perimeter**() dice che sono implementate da **Polygon**, controllando il codice le 2 funzioni **ritornano** il valore senza andare a modificare il valore della variabile **area** e **perimeter** della classe **Polygon**.

1. **Rhombus**

Nella documentazione della classe **Rhombus** si ripresenta *l’errore del punto precedente* riguardo le funzioni virtualizzate **Area**() e **Perimeter**().

**Tutte** le altre funzioni membro della classe **Rhombus** **NON** sono state documentate.

1. **RightTrap** – **righttrapezoid**

Sono presenti **2 sezioni** per la documentazione della classe chiamate nel codice **RightTrap** e **righttrapezoid,** la seconda è **vuota** e viene generata dal file **righttrapezoid.h**, probabilmente l’errore si verifica perché la classe non è stata chiamata allo stesso modo del file.

Per quanto riguarda la sezione **RightTrap**: La lista delle funzioni membro è un po’ incasinata presenta documentazione che dovrebbe stare nella descrizione specifica della funzione, per quanto riguarda la documentazione specifica di ogni classe membro sembra essere strutturata e spiegata abbastanza bene.

-) **File list**:

Nella sezione file list mancano molti file:

* **manca** il file **main.cpp** e **quindi la sua documentazione**
* file **polygon.cpp**
* file **rhombus.cpp**
* file **righttrapezoid.cpp**
* il link al file **righttrapezoid.h** non funziona

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.

Specifiche **Rectangle**

* **Area**() e **Perimeter**() funzionano come dovrebbero funzionare **GetArea**() e **GetPerimeter**(): non modificano il valore delle variabili **area** e **perimeter**

Specifiche **Rhombus**

* Non presenta la funzionalità **Draw**()
* **Area**() e **Perimeter**() funzionano come dovrebbero funzionare **GetArea**() e **GetPerimeter**(): non modificano il valore delle variabili **area** e **perimeter**

Specifiche **RightTrap**

* Le specifiche sono state rispettate, la funzione che dovrebbe chiamarsi da specifiche **GetSide**() è stata chiamata **ObliqueSide**()
* **Area**() e **Perimeter**() funzionano come dovrebbero funzionare **GetArea**() e **GetPerimeter**(): non modificano il valore delle variabili **area** e **perimeter**

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 NO**N è stato **caricato**.

**2 Compilazione e prima sessione di test**

2.a Verificare se è possibile compilare il progetto

Il progetto **compila**

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?

L’interfaccia utente funziona e permette di scegliere una delle 5 opzioni.

* **Primo test**

Provo ad aggiungere un oggetto **Rectangle** di cui mi vengono chieste le dimensioni dopo averle inserite mi viene correttamente ripresentato il menu, una volta scelta l’opzione **Draw** il programma genera una **eccezione** “violazione di accesso durante la lettura del percorso xxx”, probabilmente si tratta di un errore di **overflow** quando si cicla sull’array.

Dopo aver controllato il codice noto che l’eccezione è dovuta ad un errore nel codice: quando si sceglie l’opzione 4 viene svolto il seguente codice:

for (int i = 0; i < 10; i++)

{

polygons[i]->Draw();

}

Se vengono inseriti meno di 10 oggetti il for cicla su elementi non inizializzati dell’array generando quindi l’eccezione.

Se vengono inseriti più di 10 oggetti nell’array dall’undicesimo oggetto in poi non viene più effettuato il **Draw**().

Il codice corretto è il seguente:

for (int i = 0; i < control; i++)

{

polygons[i]->Draw();

}

Dove il for cicla fino al valore della variabile **control** (già inserita nel programma, serve per contare il numero di elementi inseriti nella lista) che avrà il valore del numero di oggetti fino ad allora inseriti nella lista.

Effetto la modifica e continuo il test:

* **Secondo Test**

Stesso test precedente una volta scelta l’opzione 4 non vado in eccezione ma non viene stampato nulla a schermo.

L’errore si trova a riga 18, la variabile che conta il numero di oggetti presenti in memoria viene reinizializzato a 0 ogni volta che si riparte dall’inizio del while, bloccando il ciclo dell’opzione 4 **Draw**.

Correggo spostando l’inizializzazione prima del while e passo al test successivo

* **Terzo test**

Stesso Test precedente ora funziona

* **Quarto test**

Inserisco un **Rectangle** ed un **Rhombus**, eseguo la **Draw**: **Eccezione** generata dovuta all’assenza della funzione **Draw**() in **Rhombus**, inserisco una **Draw**() provvisoria e continuo.

* **Quinto test**

Stesso test del punto 4 non funziona: il problema precedente era dovuto al fatto che il codice per l’opzione 2 (Creazione di **Rhombus**) **NON è stato scritto**, inserisco un codice provvisorio e continuo il test.

* **Sesto test**

Inserisco un **Rectangle**, un **Rhombus** ed un **Trapezoid** ed eseguo la **Draw** funziona tutto correttamente.

* **Settimo test**

Inserisco una lettera al posto di un numero, il programma continua a stampare di continuo il menu.

* **Ottavo test**

Effettuo un **test di overflow** sulla lista.

Non è stato effettuato un controllo che blocchi la creazione di oggetti quando la lista è piena, il programma genera eccezione quando viene chiuso, probabilmente perché non ha modo di accedere agli oggetti nella lista per distruggerli.

Controllando l’opzione 5 (**Exit**) noto che non è stato fatto nulla per distruggere gli oggetti creati dinamicamente.

* **CONCLUSIONI**

Il programma presenta diversi errori molto gravi e risulta essere NON utilizzabile per il suo scopo allo stato originale.

**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

Riporto il programma allo stato originale ed eseguo i test

Per le classi **Rectangle**, **Rhombus** e **RightTrap** è stata creata la seguente procedura di test:

- Test Costruttori

- Test Setters

- Test Getters

- Test Draw

- Test Dump

- Test operator =

- Test operator ==

I risultati del test sono stati controllati attraverso l’utilizzo di **breakpoint**; perciò, il codice stampa a schermo solo quando le funzioni chiamate hanno questo compito.

Il programma con la procedura di test è stato caricato assieme a questo file.

3.b Riportare i risultati del test

Di seguito sono riportati i risultati del test:

Errori **Rectangle**

- Non presenta la funzione **Dump**()

- Ho accesso alle funzioni **Area**() e **Perimeter**() dal main dovrebbero essere **private**

Errori **Rhombus**

- Manca la funzione **Draw**()

- Ho accesso alle funzioni **Area**() e **Perimeter**() dal main dovrebbero essere **private**

Errori **RightTrap**

- La classe presenta l’implementazione delle funzioni **GetArea**() e **GetPerimeter**(), mentre invece la classe avrebbe dovuto sfruttare l’implementazione delle stesse funzioni ereditate da **Polygon**.

- il costruttore di inizializzazione implementato, se vengono inseriti dei valori non ammissibili al posto di mettere tali valori a 0 lascia quelli già presenti in memoria.

- l’implementazione della funzione **ObliqueSide**() che da specifiche dovrebbe essere chiamata **GetSide**() è errata, nel **teorema di Pitagora** viene fatto il prodotto invece che la somma