**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.

Nella documentazione doxygen non è presente il file main (commenti mancanti) e i file .cpp di ogni classe eccetto per la classe rectangle.

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 main, dentro lo switch della creazione di un rombo non è presente la parte di codice che permette di creare un rombo.

Commenti “doxygen compatbili” mancanti nei file:

* Main.cpp
* Rhombus.cpp
* Rectangle.cpp (solo nelle funzioni create)
* Righttrapezoid.cpp (solo nelle funzioni create)

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.

README: il file non esiste nella repository di github.

**2 Compilazione e prima sessione di test**

2.a Verificare se è possibile compilare il progetto

Il progetto viene compilato correttamente

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?

Gli oggetti vengono creati correttamente ma quando si decide di disegnarli il programma genera un eccezione sulla chiamata della draw. La creazione di un rombo non è possibile. Alla selezione della exit non sembra che siano chiamati i distruttori delle classi.

L’interfaccia non è molto comprensibile (richiede tutti i parametri insieme non uno alla volta) ma è comunque utilizzabile correttamente.

Non esiste il controllo se l’utente inserisce valori negativi o uguali a zero.

Le prove svolte sono state quelle di lanciare il programma e fingere di essere uno user generico che doveva usare tutte le funzioni del programma.

**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

/// @brief that checks if th maain program is correcy and also fins the errors

/// @author Lorenzo Pitzalis

#include <iostream>

#include "polygon.h"

#include "righttrapezoid.h"

#include "rectangle.h"

#include "rhombus.h"

/// dichiaro i puntatori a classi

Polygon\* polygons[99];

/// variabili di debug

bool debug = false;

string var;

using namespace std;

int main() {

/// dichiaro contatore per il puntatore

int count = 0;

/// inizio caso della creazione di un rettangolo

/// caso parametri normali

Rectangle\* X1 = new Rectangle;

X1->SetDim(5, 4);

polygons[count] = X1;

count++;

/// breakpoint for debug

if (debug == false) {

cout << endl << flush;

cout << "Punto di interruzione: premi un tasto per continuare" << endl;

var = cin.get();

cout << endl << flush;

debug = true;

}

debug = false;

/// caso parametri negativi

Rectangle\* X2 = new Rectangle;

X2->SetDim(-5, -4);

polygons[count] = X2;

count++;

/// breakpoint for debug

if (debug == false) {

cout << endl << flush;

cout << "Punto di interruzione: premi un tasto per continuare" << endl;

var = cin.get();

cout << endl << flush;

debug = true;

}

debug = false;

/// caso parametri uguali a zero

Rectangle\* X3 = new Rectangle;

X3->SetDim(0, 0);

polygons[count] = X3;

count++;

/// breakpoint for debug

if (debug == false) {

cout << endl << flush;

cout << "Punto di interruzione: premi un tasto per continuare" << endl;

var = cin.get();

cout << endl << flush;

debug = true;

}

debug = false;

/// fine caso rettangolo

/// inizio caso della creazione di un rombo -> impossibile causa errore nella classe derivata

/\* /// caso parametri normali

Rhombus\* Y = new Rhombus;

Y->SetDim(8, 4);

polygons[count] = Y;

count++;

/// caso parametri negativi

Rhombus\* Y = new Rhombus;

Y->SetDim(-8, -4);

polygons[count] = Y;

count++;

/// caso parametri uguali a zero

Rhombus\* Y = new Rhombus;

Y->SetDim(0, 0);

polygons[count] = Y;

count++;\*/

/// fine caso rombo

/// inizio caso della creazione di un trapezio rettangolo

/// caso parametri normali

RightTrap\* Z1 = new RightTrap;

Z1->SetDim(5, 4, 7);

polygons[count] = Z1;

count++;

/// breakpoint for debug

if (debug == false) {

cout << endl << flush;

cout << "Punto di interruzione: premi un tasto per continuare" << endl;

var = cin.get();

cout << endl << flush;

debug = true;

}

debug = false;

/// caso parametri negativi

RightTrap\* Z2 = new RightTrap;

Z2->SetDim(-5, -4, -7);

polygons[count] = Z2;

count++;

/// breakpoint for debug

if (debug == false) {

cout << endl << flush;

cout << "Punto di interruzione: premi un tasto per continuare" << endl;

var = cin.get();

cout << endl << flush;

debug = true;

}

debug = false;

/// caso parametri uguali a zero

RightTrap\* Z3 = new RightTrap;

Z3->SetDim(0, 0, 0);

polygons[count] = Z3;

count++;

/// breakpoint for debug

if (debug == false) {

cout << endl << flush;

cout << "Punto di interruzione: premi un tasto per continuare" << endl;

var = cin.get();

cout << endl << flush;

debug = true;

}

debug = false;

/// caso basi uguali

RightTrap\* Z4 = new RightTrap;

Z4->SetDim(5, 7, 7);

polygons[count] = Z4;

count++;

/// fine caso trapezio rettangolo

/// breakpoint for debug

if (debug == false) {

cout << endl << flush;

cout << "Punto di interruzione: premi un tasto per continuare" << endl;

var = cin.get();

cout << endl << flush;

debug = true;

}

debug = false;

/// Inizio prova funzione Draw

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

polygons[i]->Draw();

/// Fine prova funzione Draw

return 0;

}

3.b Riportare i risultati del test

Il programma non consente di riinserire i parametri se errati (uguali a zero, basi uguali nel caso del trapezio o valori negativi) ma stampa solo un messaggio di errore. Inoltre memorizza comunque i dati nell’array di puntatori a classe base. La classe Rhombus sembra funzionare ma non è possibile testarla a causa dell’inesistenza della funzione draw nella classe derivata.

Per qualche motivo, quando viene chiamata la funzione draw per stampare gli oggetti creati, il programma genera un eccezione e va in crash.

Il programma una volta usciti, non chiama i distruttori e non svuota l’array di puntatori.