**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 .ccp 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

Documentazione generata.

Polygon :

* In operator==() init() e polygon() il riferimento è segnato come “p” mentre nel commento è “r” o viceversa
* Area() Draw() e perimeter() non hanno un commento che generalmente spiega come dovrà essere implementata successivamente ma dice solamente che questo “sarà fatto”. I getters non presentano questo problema

Rectangle :

* In Draw() e perimeter() non ci sono specifiche su cosa fanno le funzioni

Rhombus :

* In operator=() e operator==() il riferimento è segnato come “p” mentre nel commento è “r” o viceversa

RightTriangle:

* In RightTriangle() Init() e operator==() il riferimento è segnato come “p” mentre nel commento è “r” o viceversa
* In operator<< non sono specificati i parametri os e r

1.b

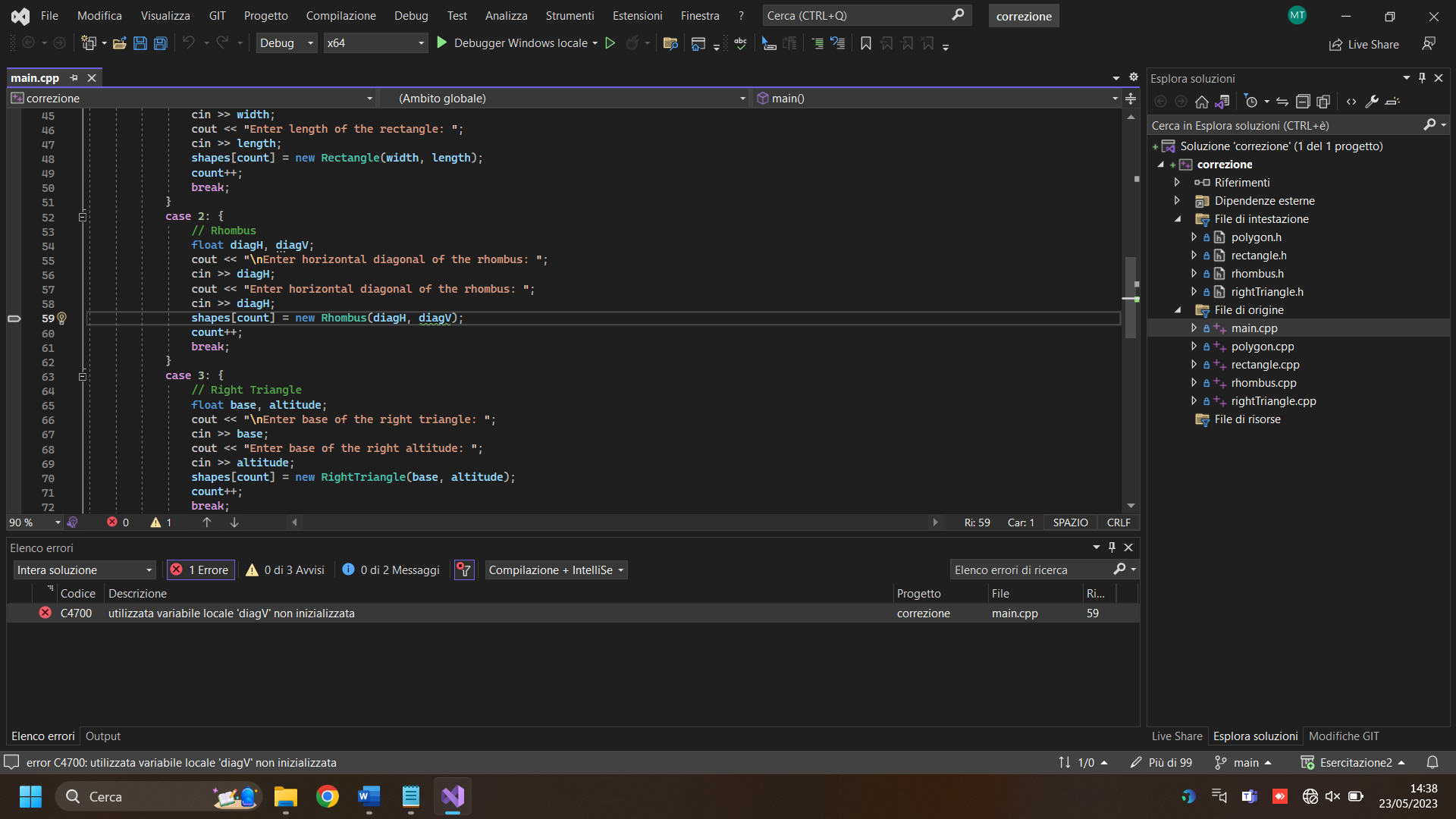
Il progetto rispetta le specifiche indicate dalla consegna

1.c

Il file README compare nella documentazione come” Related Pages” e, oltre a spiegare in modo chiaro il funzionamento del programma e l’uso dell’interfaccia, fornisce un tutorial su come compilare tramite il prompt dei comandi

2.a

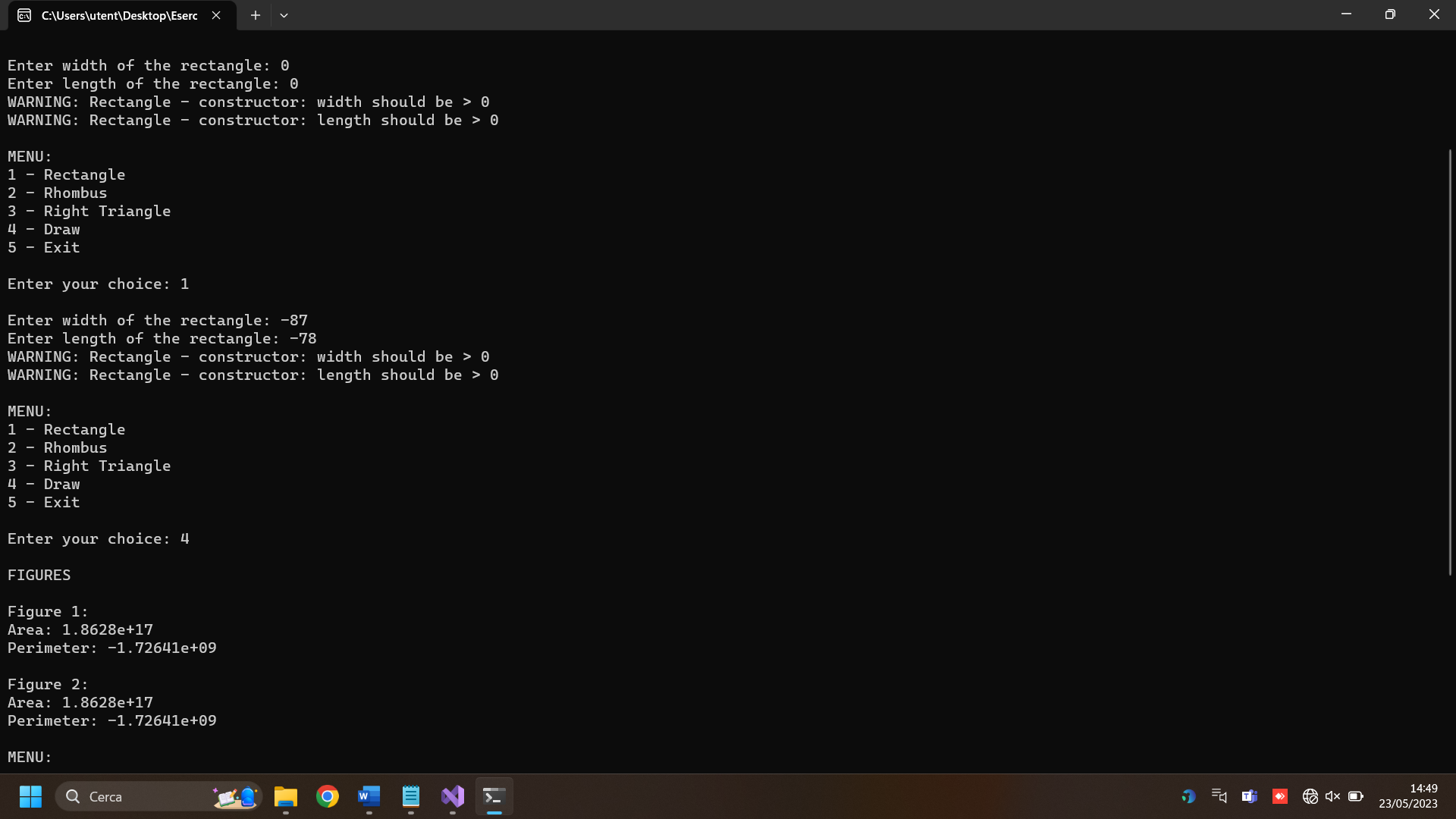
La variabile diagV non viene inizializzata (typo)



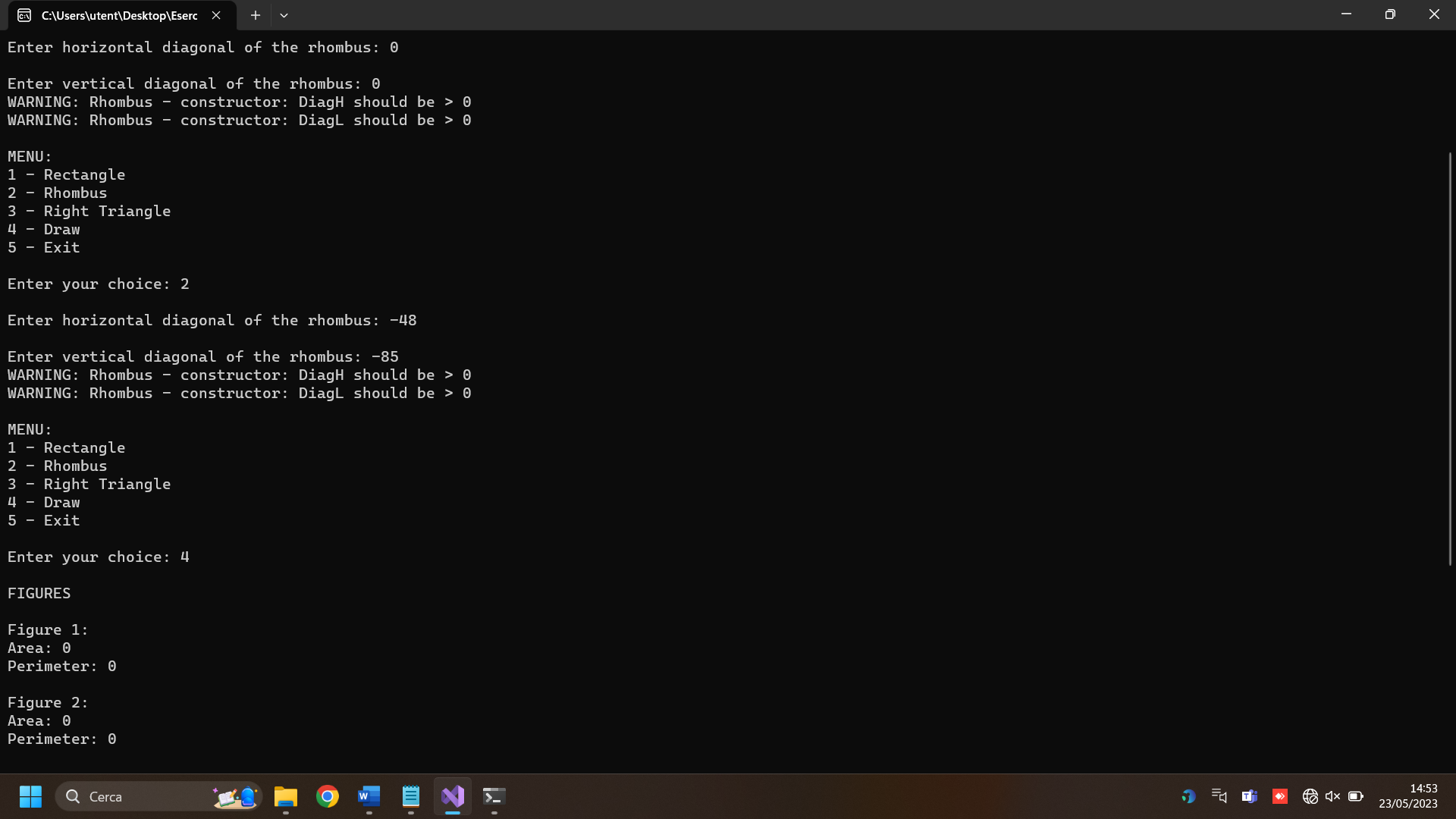
2.b

Quando si inseriscono dati errati, più precisamente valori negativi o nulli per lati, diagonali, basi, altezze viene segnalato come avviso e non errore. L’oggetto viene creato ma non distrutto immediatamente (o direttamente non creato)

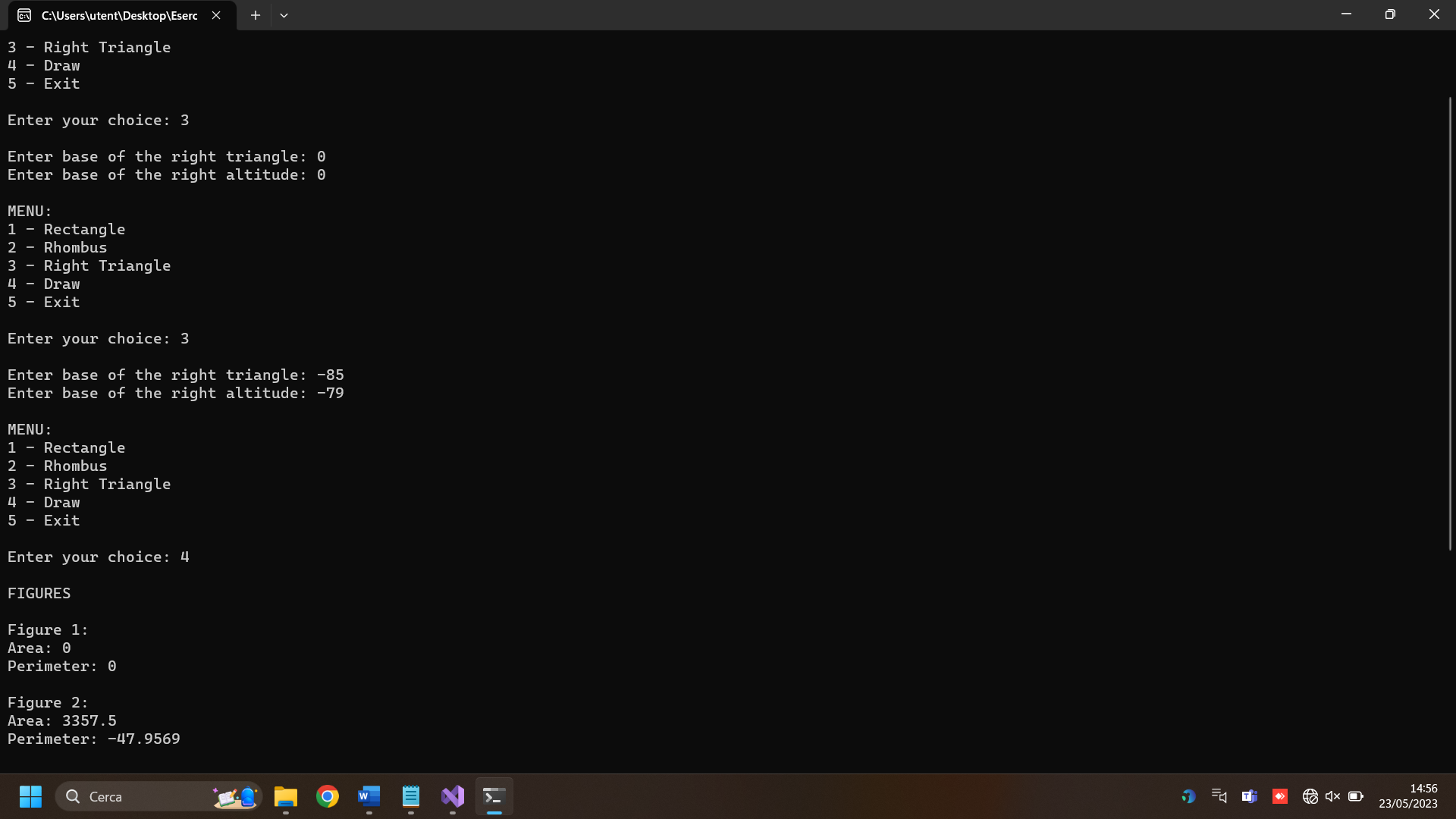
* Rettangolo non inizializzato (con più valori esce lo stesso risultato)



* Rombo settato con area=0 e perimetro=0



* Triangolo rettangolo non inizializzato e non compare il messaggio di warning (area comunque corretta)



Per testare ho inserito per ognuno dei tre poligoni tre tipi di valori: entrambi positivi (terne pitagoriche 3-4-5) entrambi zero e entrambi negativi (casuali). Inoltre inserendo un char al posto di un numero intero, il programma va incontro ad un errore e il main stampa il menù in un loop infinito

3.a

Procedura di test:

/// \file Test.cpp

/// \author Tomaghelli Moreno

#include"rectangle.h"

#include"rhombus.h"

#include"rightTriangle.h"

/// numero di casi limite da testare

/// costruttore di default 1 caso

/// costruttore con parametri 3 casi: zero-zero, numeri negativi, numeri positivi interi, numeri con la virgola, numeri limite con la virgola

#define N\_CASES 6

int main() {

/// creazione poligoni, 3 per ogni caso (un rettangolo, un rombo, un triangolo rettangolo)

Polygon\* universo[3 \* N\_CASES] = {0};

/// costruttore di default

universo[0] = new Rectangle;

universo[0]->Dump();

universo[1] = new Rhombus;

universo[1]->Dump();

universo[2] = new RightTriangle;

universo[2]->Dump();

/// costruttore di copia e a parametri positivi interi

Rectangle a(4,5);

a.Dump();

Rectangle a2(a);

a2.Dump();

Rhombus b(4,3);

b.Dump();

Rhombus b2(b);

b2.Dump();

RightTriangle c(3,4);

c.Dump();

RightTriangle c2(c);

c2.Dump();

/// costruttore a parametri

for (int i = 3; i < 3 \* N\_CASES; i=i++) {

float n[N\_CASES - 1] = { 3,0,-3,0.3,333333333333333333333.3333333 };

float m[N\_CASES - 1] = { 4,0,-4,0.4,444444444444444444444.4444444 };

/// rettangolo i

universo[i] = new Rectangle(n[i / 3], m[i / 3]);

universo[i]->Dump();

i++;

/// rombo i+1

universo[i] = new Rhombus(n[i / 3], m[i / 3]);

universo[i]->Dump();

i++;

/// triangolo rettangolo i+2

universo[i] = new RightTriangle(n[i / 3], m[i / 3]);

universo[i]->Dump();

}

return 0;

}

3.b

La procedura di test ha dato risultati positivi nei casi di numeri > 0 interi, con virgola e approciando il limite massimo. I problemi sui numeri >=0 sono quelli sopracitati