*Приложение*

ТЕХНИЧЕСКИ УНИВЕРСИТЕТ – ГАБРОВО

Катедра „Компютърни системи и технологии“

**КУРСОВА РАБОТА**

Тема: Пресмятане лица на определен брой ромбове

Разработил: Станислав Пламенов Манолов

ф. номер: 22372111 курс: I-ви курс специалност СКИ

Проверил:............................

Подпис: .............

Дата: ...............

Теоретична част

1.Описание на заданието:

Да се създаде програма, която пресмята лицето на определен брой ромбове и кой от тях е с най-голяма площ. Да се декларира структура описваща ромб чрез страна и ъгъл между страните, да се дефинира масив от ромбове и масив от площите на тези ромбове. Търсенето на площ на ромб, както и на най-голяма стойност да се реализира с отделни функции. Да се сортира масива по площ на фигурите. Данните да се изведат на екрана в подходящ формат.

Линк към програмата в Github:

Програмния код:

#include <iostream>

#include <stdio.h>

#include <cmath>

#include <algorithm>

int RombS(int a, int h);

using namespace std;

int main()

{

int a, h, s;

int arrayOfRombs[20];

int currentArrayIndex = 0;

char choice;

do

{

printf("Enter value for A:\n");

scanf\_s("%d", &a);

printf("Enter value for H:\n");

scanf\_s("%d", &h);

s = RombS(a, h);

if (currentArrayIndex < 20)

{

arrayOfRombs[currentArrayIndex] = s;

currentArrayIndex++;

}

else

{

printf("Array of Rombs is full!\n");

break;

}

printf("S =%d\n", s);

printf("Want to enter more values? Y/N\n");

cin >> choice;

}

while (choice != 'n' && choice != 'N');

printf("Array of Rombs: ");

std::sort(arrayOfRombs, arrayOfRombs + currentArrayIndex, std::greater<int>());

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

{

printf("%d ", arrayOfRombs[i]);

}

}

int RombS(int a, int h)

{

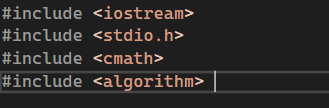
int s = a \* h;

return s;

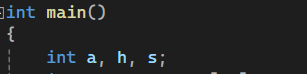
}

2.Описание на кода

* Първото и най-важно нещо, което трябва да си добавим са следните библиотеки:



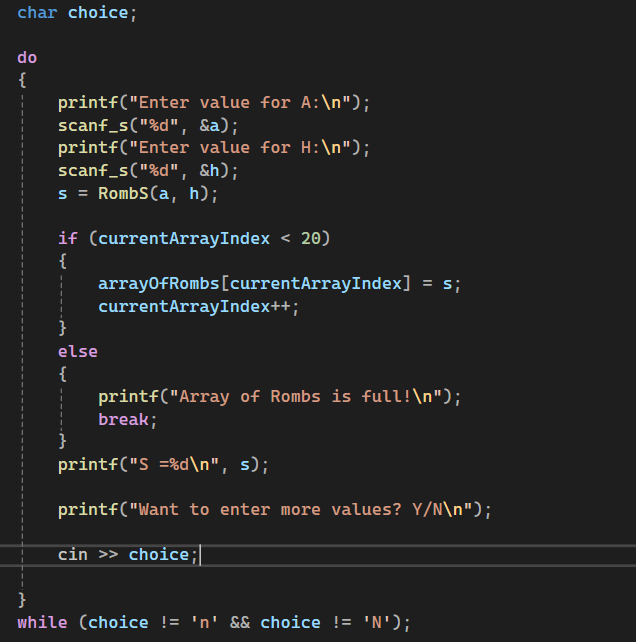
* След добавянето на библиотеките, които са ни нужни, дефинираме във функцията main три променливи, които ще ни отговарят за страна „а“ и за височина „h“ и още една за лицето.



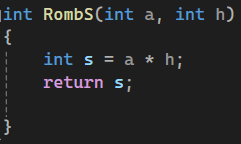
* Декларираме си масив от тип int, в който ще се запазват пресметнатите лица от функцията ни, както и променлива също от тип int, който да ни следи до кой индекс сме стигнали в масива.

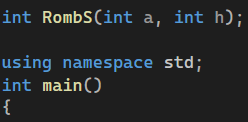


* Създаваме do-while цикъл, в който да се въвеждат стойности за “a” и “h” от потребителя и ще се извика функцията за пресмятането на лице на ромб. Тези действия ще се изпълняват, колкото пъти иска потребителя или докато се напълни масива.



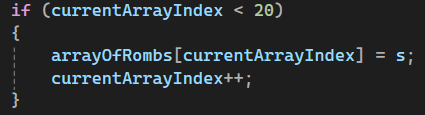
* За да работи do-while цикъла, който пита потребителя дали иска да продължи да въвежда страна и височина на ромба, си дефинирахме една променлива, която докато е равна на символа ‘y’ или ’Y’ ще продължава да извършва действията, докато потребителя не въведе символите ‘n’ или ‘N’ от конзолата. Или се напълни масива със стойности.
* Следващата стъпка е да си създадем функцията, която ще изчислява лицето и да я извикаме в main функцията.



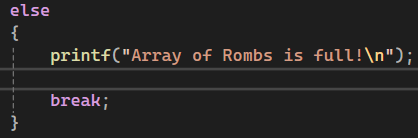
* Също така я деклариме над main функцията. 
* Извикваме я в do-while цикъла по следния начин:



* За да запаметим пресметнатите лица в масив, създаваме оператор if-else, който да проверява дали променливата, която следи индексите на масива, е винаги по малка от елементите на масива:



* И ако променливата е по-малка от елементите на масива, тя да присвои стойноста на променливата с пресметнатото лице и да я запише в масива. Накрая да отиде към следващия елемент (да се инкрементира с едно).
* В противен случай ако променливата мине стойността 20 да изпише че масива е пълен и да се прекрати програмата.



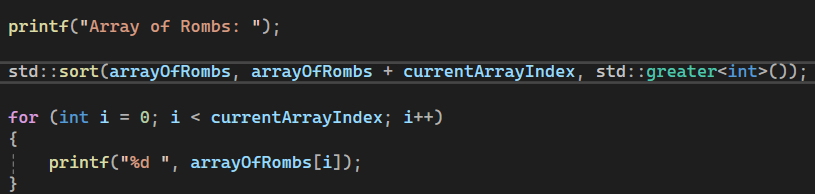
* И за финал извеждаме масива в сортиран вид. От най-малкото до най-голямото. Това става чрез библиотеката:

, която си добавихме в началото. Това става по следния начин:



* Във функцията sort задаваме елементите които ще ни трябват, в случая е масива ни и променливата, която проверява за индексите. С std::greater<int>(); сортираме масива от най-голямото към най-малкото и също му задаваме и тип Int.

Следния код ще изглежда така:



3.Блокова схема на програмата