Міністерство освіти і науки України

Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського"

Факультет інформатики та обчислювальної техніки

Кафедра ІПІ

Звіт

з лабораторної роботи № 3 з дисципліни

«Основи програмування 2. Модульне програмування»

„ Класи і об’єкти”

Варіант 8

Виконав(ла) *Дацьо Іван Іванович*
(шифр, прізвище, ім'я, по батькові)

Перевірив *Вєчерковська Анастасія Сергіївна*
(прізвище, ім'я, по батькові)

Київ 2022

Лабораторна робота 1

Текстові файли

Мета – вивчити механізм створення і використання класів.

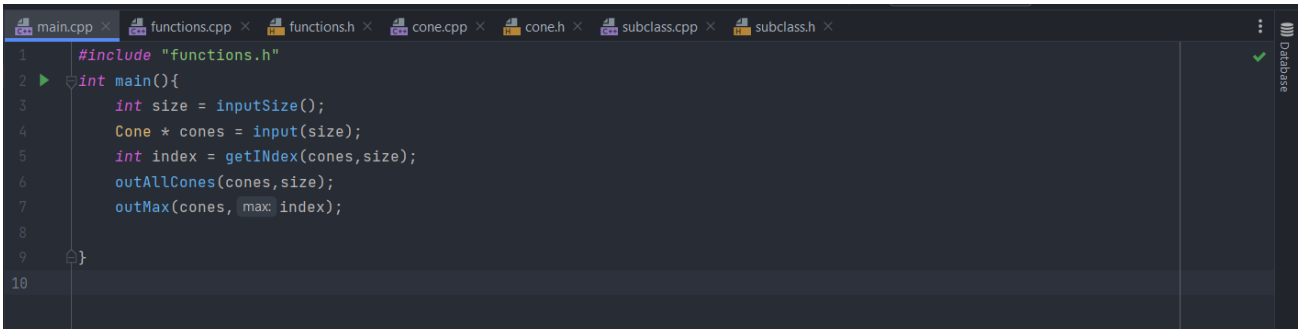
Варіант 8

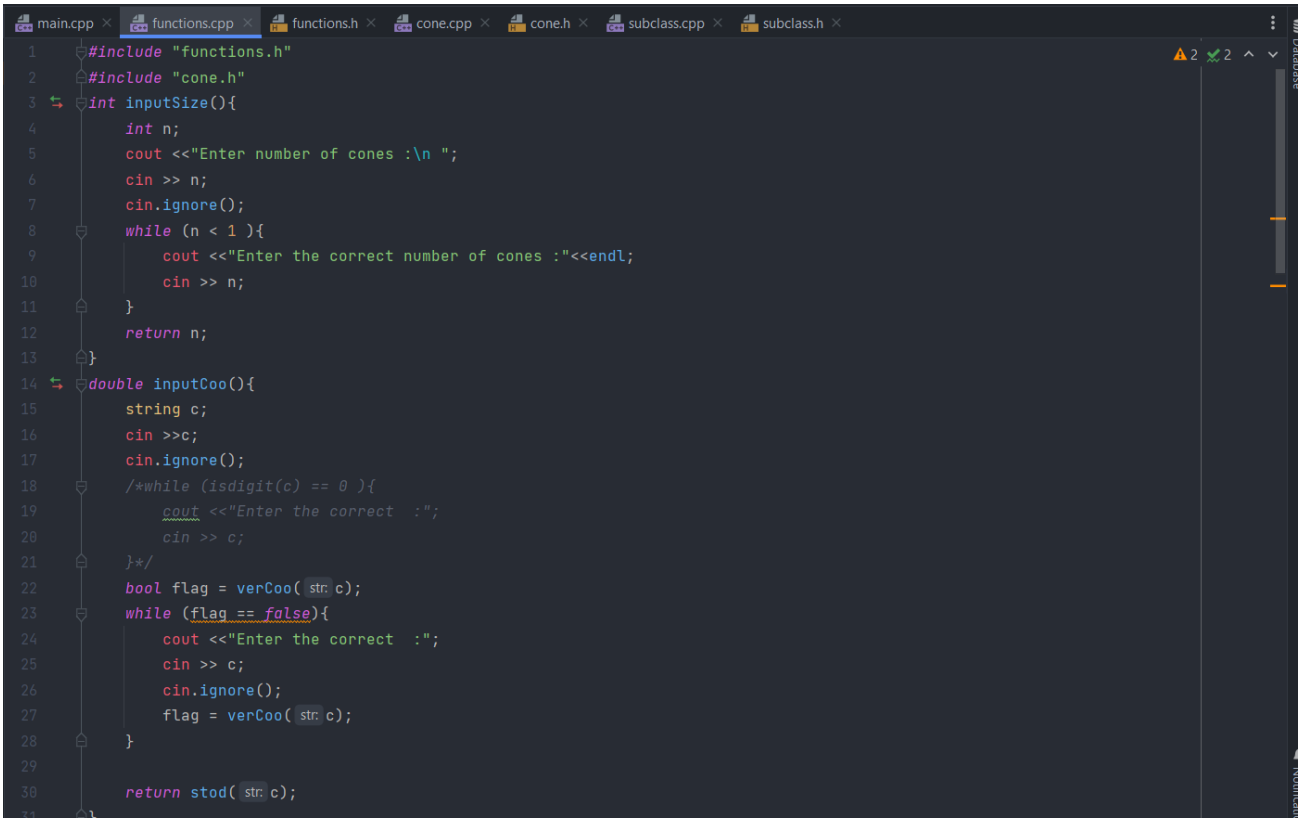
Задача

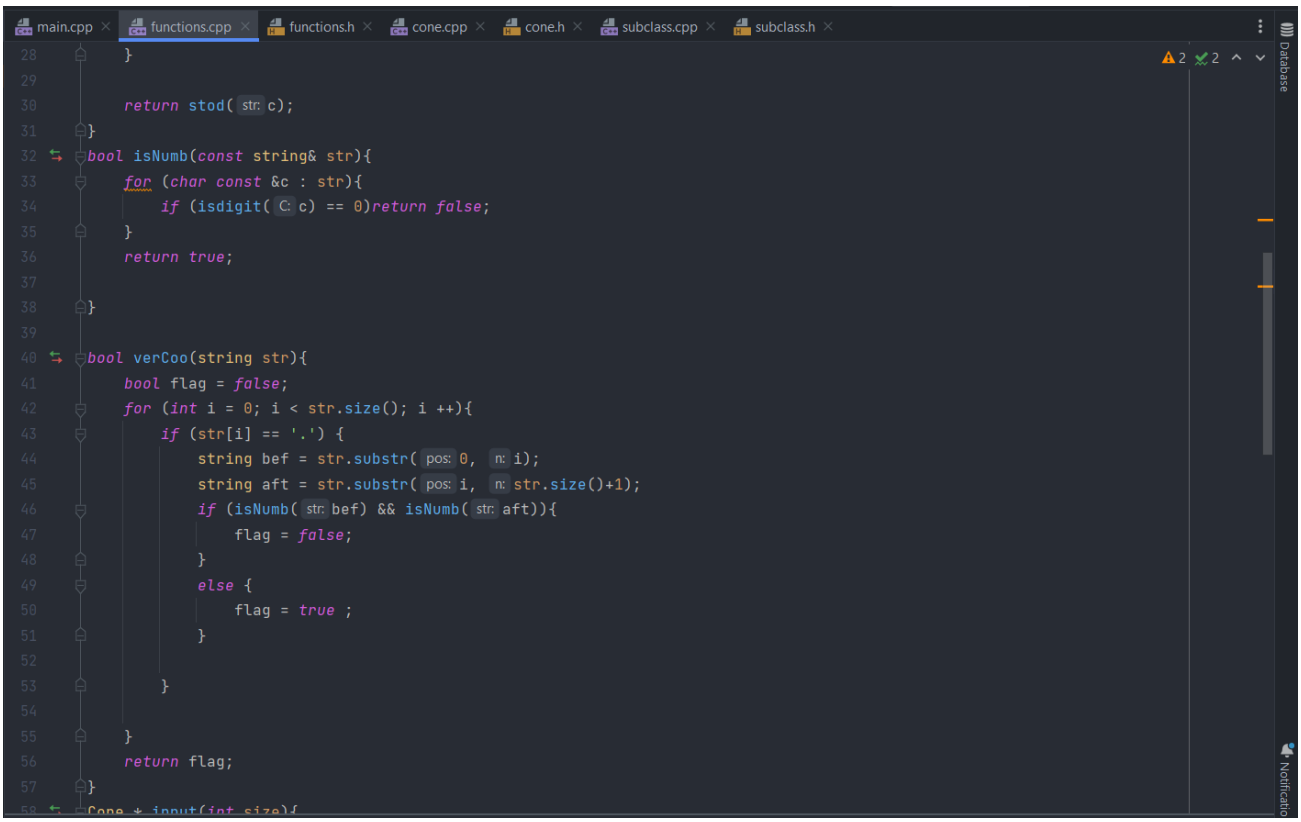
8. Розробити клас "конус", який заданий координатами центру основи, координатами вершини та радіусом основи. Створити масив об'єктів даного класу. Визначити конус з найбільшою твірною.

Код:

C++







```
main.cpp | functions.cpp | functions.h | cone.cpp | cone.h | subclass.cpp | subclass.h
28      }
29
30      return stod( str: c);
31  }
32  bool isNumb(const string& str){
33      for (char const &c : str){
34          if (isdigit( c: c) == 0)return false;
35      }
36      return true;
37
38  }
39
40  bool verCoo(string str){
41      bool flag = false;
42      for (int i = 0; i < str.size(); i ++){
43          if (str[i] == '.') {
44              string bef = str.substr( pos: 0,  n: i);
45              string aft = str.substr( pos: i,  n: str.size()+1);
46              if (isNumb( str: bef) && isNumb( str: aft)){
47                  flag = false;
48              }
49              else {
50                  flag = true ;
51              }
52
53          }
54
55      }
56      return flag;
57  }
58  Cone * input(int size){
```

```
main.cpp | functions.cpp | functions.h | cone.cpp | cone.h | subclass.cpp | subclass.h
55      }
56      return flag;
57  }
58  Cone * input(int size){
59      Cone * cones = new Cone[size];
60      for (int i = 0; i < size; i++){
61          cout <<"Cone "<<i+1<<":"<<endl;
62          Point points[2];
63          for (int j = 0; j < 2; j ++){
64              if (j == 0){
65                  cout <<"Point O(center of foundation) :\n";
66              }
67              else{
68                  cout <<"Point A(the top of the cone) :\n";
69              }
70              cout <<"Enter x coordinate : \n";
71              double x = inputCoo();
72              cout <<"Enter y coordinate : \n";
73              double y = inputCoo();
74              cout <<"Enter z coordinate : \n";
75              double z = inputCoo();
76              points[j] = Point(x,y,z);
77          }
78          cout <<"Enter the radius of the base of the cone :\n";
79          double r = inputCoo();
80          cones[i] = Cone(points , r);
81      }
82      return cones;
83  }
84  int getINdex(Cone * cones , int size){
```

main.cpp | functions.cpp | functions.h | cone.cpp | cone.h | subclass.cpp | subclass.h

```
    }
    return cones;
}
int getINdex(Cone * cones , int size){
    int index;
    for (int i = 1; i < size; i++){
        if (cones[i].getGenerator() > cones[0].getGenerator()){
            index = i;
        }
    }
    return index;
}
void outAllCones(Cone* cones ,int size){
    cout <<"All cones :"<<endl;
    for (int i = 0; i < size; i++){
        cout <<"Cone "<<i + 1<<":"<<endl;
        cones[i].out();
    }
}
void outMax(Cone* cones , int max){
    cout <<"Max generator have this cone : ("<<max + 1<<") :"<<endl;
    cones[max].out();
}
```

main.cpp | functions.cpp | functions.h | cone.cpp | cone.h | subclass.cpp | subclass.h

```
#pragma once
#include <string>
//#include <ctype.h>
#include "cone.h"
using namespace std;
int inputSize();
double inputCoo();
bool verCoo(string str);
bool isNumb(const string& str);
Cone * input(int size);
int getINdex(Cone * cones , int size);
void outAllCones(Cone* cones ,int size);
void outMax(Cone* cones , int max);
```

main.cpp | functions.cpp | functions.h | cone.cpp | cone.h | subclass.cpp | subclass.h

```cpp
#include "cone.h"
Cone::Cone(){};
Cone::Cone(Point Or, Point A, Radius r) {
    this->Or = Point(Or.getX() ,Or.getY(), Or.getZ());
    this->A = Point(A.getX(),A.getY(),A.getZ());
    this->r = Radius(r.getR());
    this->l = gtgenerator();


}
Cone::Cone(Point points[2],Radius r) {
    this->Or = Point(points[0].getX(),points[0].getY(),points[0].getZ());
    this->A = Point(points[1].getX(),points[1].getY(),points[1].getZ());
    this->r = Radius(r.getR());
    this->l = gtgenerator();


}
double Cone::gtgenerator() {
    double gen = pow( x: (Or.getX()-A.getX()), y: 2) + pow( x: (Or.getY()-A.getY()), y: 2)+ pow( x: Or.getZ() - A.getZ(), y: 2)+ pow( x: (r
    return sqrt( x: gen);
}
double Cone::getGenerator() {
    return l;
}
void Cone::out(){
    cout << "Point O :";
    Or.outPoint();
    cout <<"\t";
    cout << "Point A :";
```

main.cpp | functions.cpp | functions.h | cone.cpp | cone.h | subclass.cpp | subclass.h

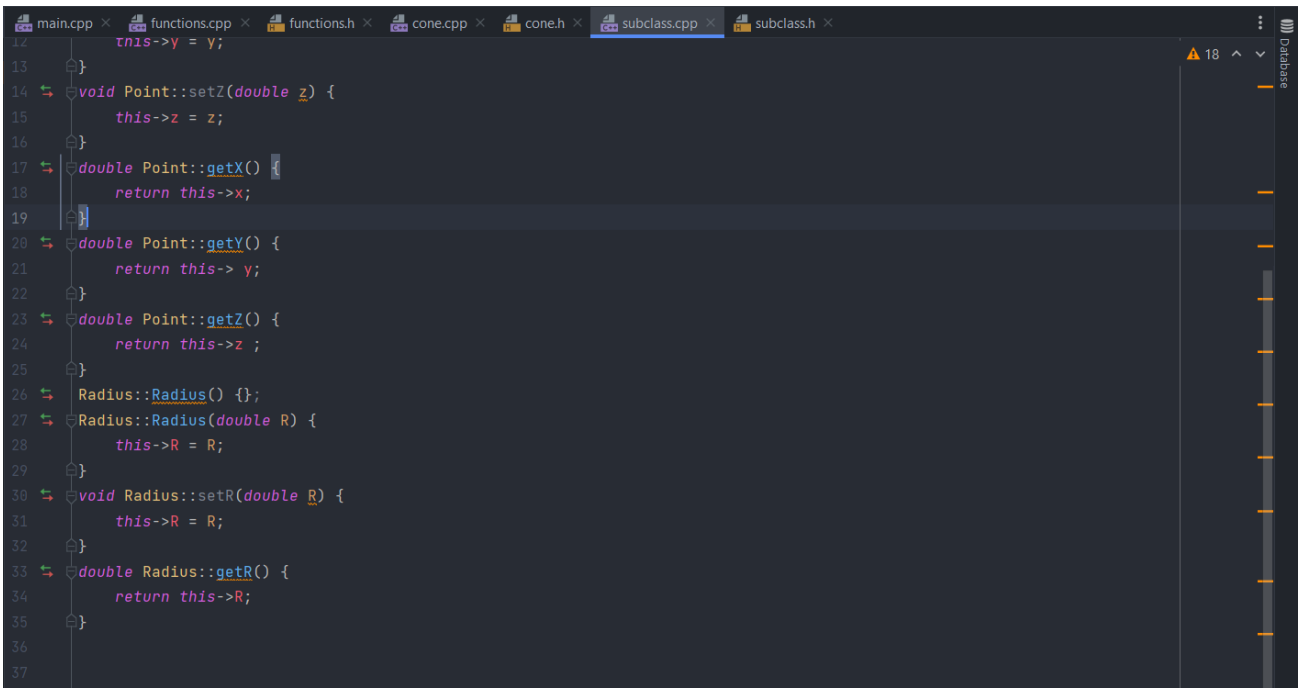
```cpp
}
double Cone::gtgenerator() {
    double gen = pow( x: (Or.getX()-A.getX()), y: 2) + pow( x: (Or.getY()-A.getY()), y: 2)+ pow( x: Or.getZ() - A.getZ(), y: 2)+ pow( x: (r
    return sqrt( x: gen);
}
double Cone::getGenerator() {
    return l;
}
void Cone::out(){
    cout << "Point O :";
    Or.outPoint();
    cout <<"\t";
    cout << "Point A :";
    A.outPoint();
    cout <<"\t";
    cout <<"Radius :";
    r.outRadius();
    cout <<"\t";
    cout <<"Generator :"<<getGenerator()<<endl;


}
```

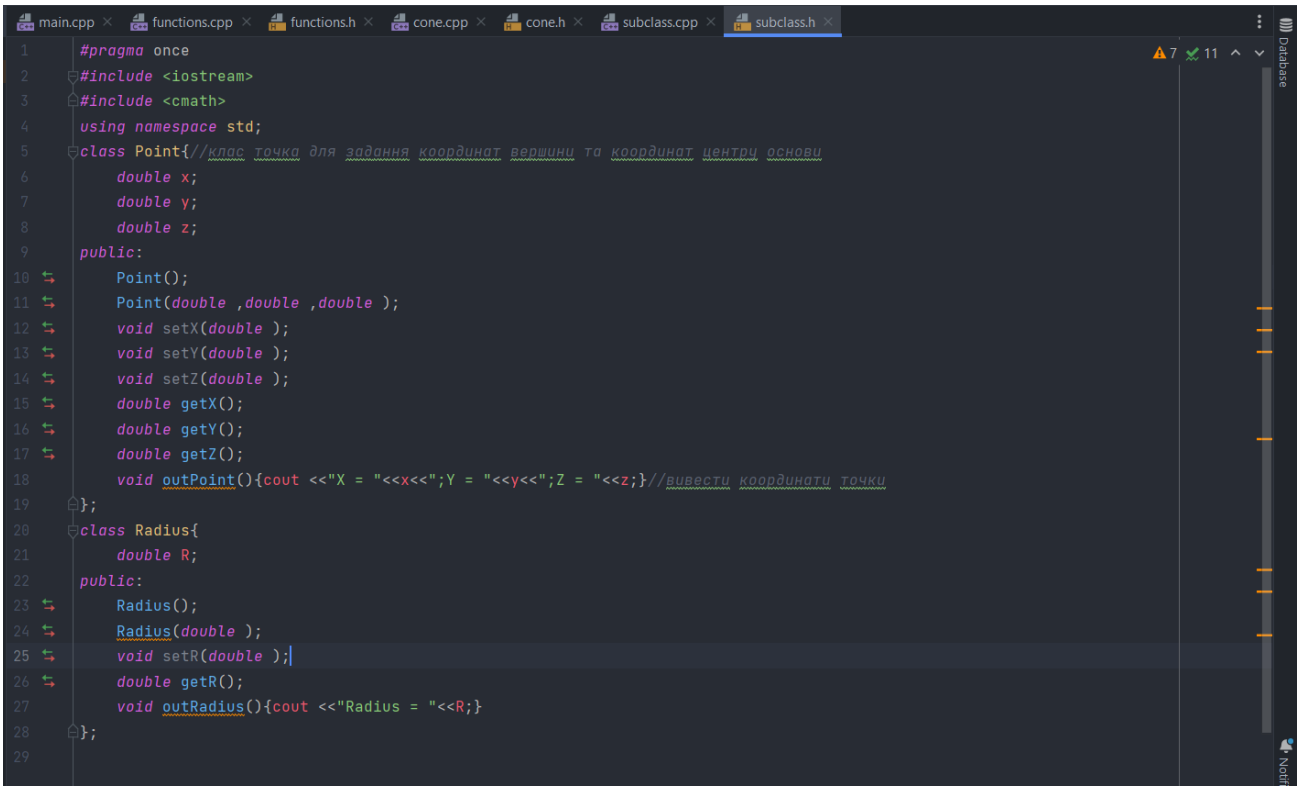
main.cpp | functions.cpp | functions.h | cone.cpp | cone.h | subclass.cpp | subclass.h

```cpp
#pragma once
#include "subclass.h"
class Cone{
    Point Or , A;//Or - центр основи ; A - вершина
    Radius r;// r - радіус основи
    double l ;//l - твірна
    double gtgenerator();

public:
    Cone();
    Cone(Point Or , Point A , Radius r);
    Cone(Point points[2], Radius r);
    double getGenerator();
    void out();

};

```

main.cpp | functions.cpp | functions.h | cone.cpp | cone.h | subclass.cpp | subclass.h

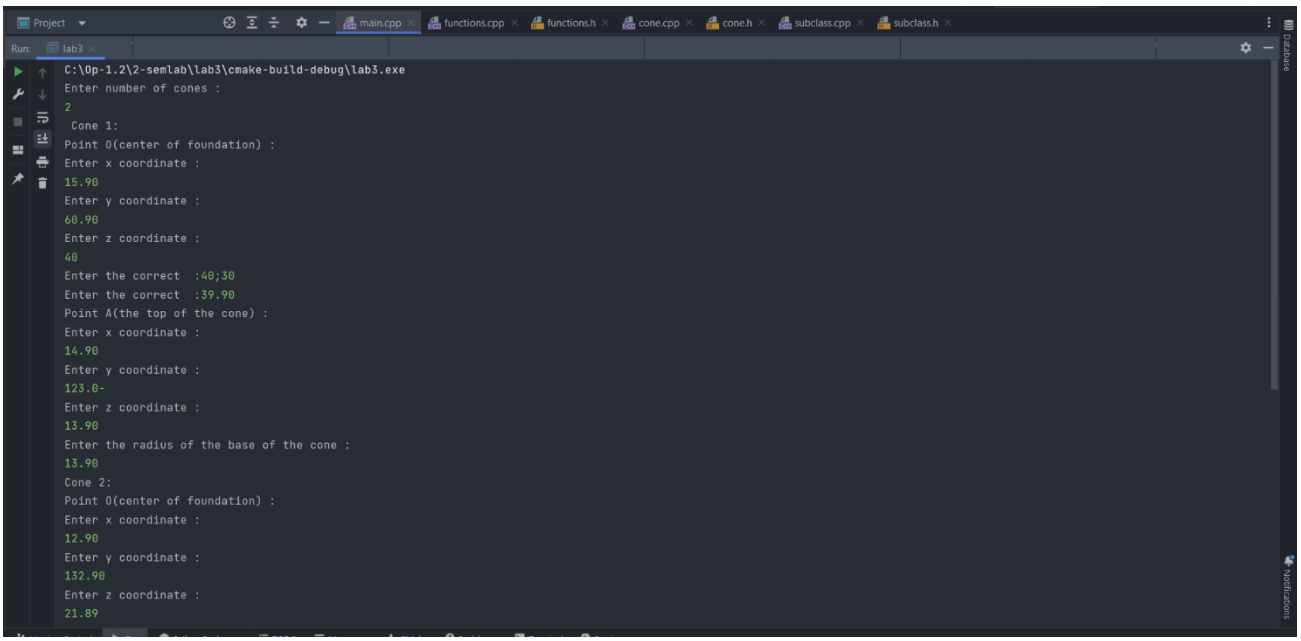
```cpp
#include "subclass.h"
Point::Point() {};
Point::Point(double x , double y, double z){
    this->x = x;
    this->y = y;
    this->z = z;
}
void Point::setX(double x) {
    this->x = x;
}
void Point::setY(double y) {
    this->y = y;
}
void Point::setZ(double z) {
    this->z = z;
}
double Point::getX() {
    return this->x;
}
double Point::getY() {
    return this-> y;
}
double Point::getZ() {
    return this->z ;
}
Radius::Radius() {};
Radius::Radius(double R) {
    this->R = R;
}
void Radius::setR(double R) {
```

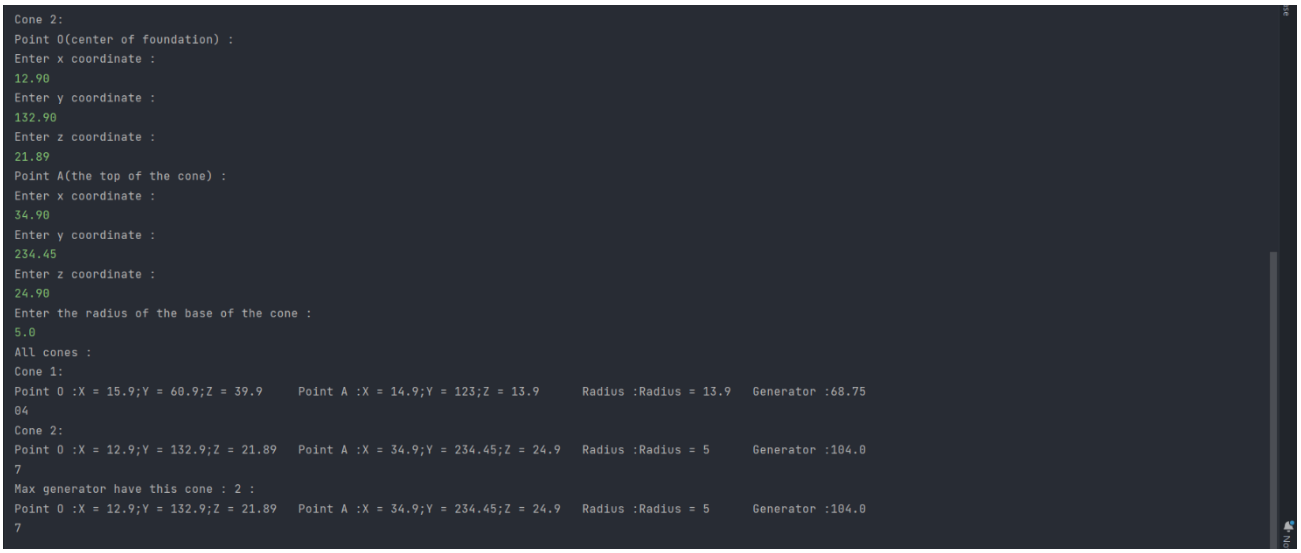




Тестування :

C++





Висновок : При виконанні лабораторної роботи було вивчено особливості роботи з класами та об’єктами та використано ці навички під час написання програм. В результаті роботи було отримано програму яка обчислює твірну конуса при відомім координатам вершини , координатам основи , та радіусом основи . Також при декількох конусах програма виводить конус з найбільшою твірною.