# Код

/\*

\* Для некоторой простоты я предполагал, что конус круговой, расположен вертикально и вершина выше чем основание. Плюс радиус основания и высота предполагаются положительными, но никакой проверки не делаается.

\*/

#include <iostream>

using namespace std;

class Point {

private:

double x;

double y;

double z;

public:

Point();

Point(double, double, double);

~Point();

double getX() { return x; }

double getY() { return y; }

double getZ() { return z; }

void setX(double x\_) { x = x\_; }

void setY(double y\_) { y = y\_; }

void setZ(double z\_) { z = z\_; }

void printPoint();

void inputPoint();

};

class Cone {

private:

Point center;

double radius;

double height;

public:

Cone();

Cone(double, double, double, double, double);

~Cone();

double getRadius() { return radius; }

double getHeight() { return height; }

double getCenterX() { return center.getX(); }

double getCenterY() { return center.getY(); }

double getCenterZ() { return center.getZ(); }

void printCone();

void inputCone();

bool isIn(Point);

};

bool isIn(Point p, Cone c) {

if ((p.getZ() < c.getCenterZ()) || (p.getZ() > c.getCenterZ() + c.getHeight())) {

return false;

}

if ((c.getRadius() \* (p.getZ() - c.getCenterZ()) + c.getHeight() \* sqrt(pow((p.getX() - c.getCenterX()), 2) + pow((p.getY() - c.getCenterY()), 2)) - c.getRadius() \* c.getHeight()) > 0) {

return false;

}

return true;

}

int main()

{

Point p;

Cone c;

p.inputPoint();

c.inputCone();

// cout << c.isIn(p) << endl;

string result = (isIn(p, c)) ? "Yes" : "No";

cout << endl << "Q: Is the point inside of the cone?\nA: " << result << endl;

p.printPoint();

c.printCone();

return 0;

}

Point::Point() { // default constructor

x = 0.0;

y = 0.0;

z = 0.0;

}

Point::Point(double a, double b, double c) { // constructor with parameters

x = a;

y = b;

z = c;

}

Point::~Point() { // destructor. It's empty becasue we didn't allocate any memory with new

}

void Point::printPoint() {

cout << endl << "Point coordinates is (" << x << ", " << y << ", " << z << ")" << endl;

}

void Point::inputPoint() {

cout << "Enter point coordinates" << endl;

cin >> x >> y >> z;

}

Cone::Cone() { // default cnstructor

Point center(0, 0, 0);

radius = 1.0;

height = 1.0;

}

Cone::Cone(double x\_, double y\_, double z\_, double radius\_, double height\_) {

center.setX(x\_);

center.setY(y\_);

center.setZ(z\_);

radius = radius\_;

height = height\_;

}

Cone::~Cone() { // destructor. Same as Point class, didn't allocate any memory, so there is nothing to delete

}

void Cone::printCone() {

cout << endl << " Cone" << endl;

cout << "Center of the base: (" << getCenterX() << ", " << getCenterY() << ", " << getCenterZ() << ")" << endl;

cout << "Radius of the base: " << radius << endl;

cout << "Height: " << height << endl;

}

void Cone::inputCone() {

double x\_, y\_, z\_;

cout << "Enter cone parameters (coordinates of center of base, raduis, height):" << endl;

cin >> x\_ >> y\_ >> z\_ >> radius >> height;

center.setX(x\_);

center.setY(y\_);

center.setZ(z\_);

}

bool Cone::isIn(Point p) { // some geometry-based checks, it should work

if ((p.getZ() < center.getZ()) || (p.getZ() > center.getZ() + height)) {

return false;

}

if ((radius \* (p.getZ() - center.getZ()) + height \* sqrt(pow((p.getX() - center.getX()), 2) + pow((p.getY() - center.getY()), 2)) - radius \* height) > 0) {

return false;

}

return true;

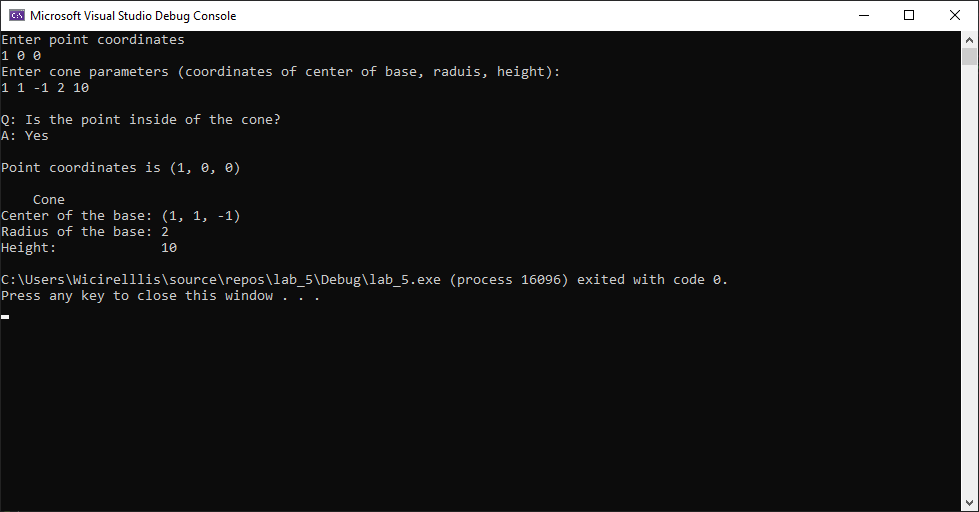
}

# 2.

Проверим работу программы на каком-нибудь входе. Пусть ввод слеующий:

1 0 0 (точка)

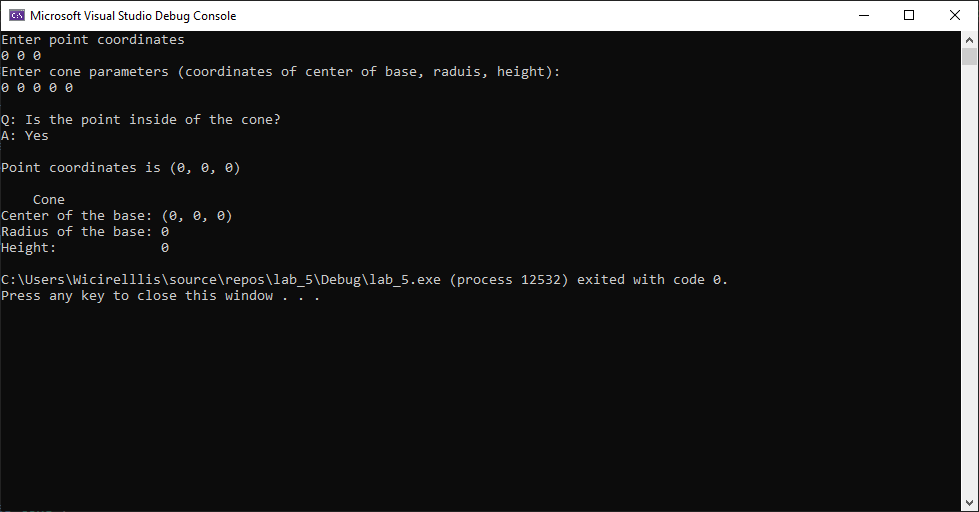
1 1 -1 2 10 (конус, первые три числа – это координаты центра основания, потом радиус основания, потом высота)

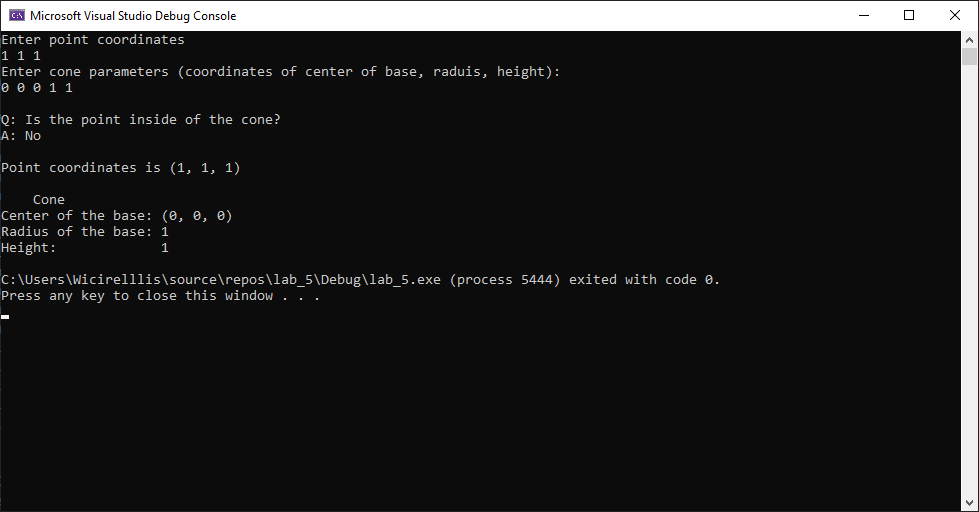


Работает (по крайней мере на данном входе) верно.

# 3.

Несколько скринов вывода





# 4. Вопросы

1) Что такое класс? Что класс может в себе хранить?

Класс – это фактически структура с методами (=функциями). Т.е. в классе есть данные (переменные базвых типов, структуры, другие классы), и функции. Хранить классс может все что угодно.

2) Какие спецификаторы доступа существуют и в чем их особенности?

Private, public, protected.

Private – используется по-умолчанию, доступ есть только у членов класса. Например прайват переменную можно прочитать из любого метода класса, но снаружи нельзя. Прайват определяется для класса, а не инстанса класса.

Public – доступно всем.

Protected – подобно прайват, доступ есть из самого класса и наследованных от него.

3) Что такое метод? Для чего нужны конструктор и деструктор? Как они задаются?

Метод – это просто функция привязанная к классу.

Конструктор – это метод, который вызывается при создании экземпляра класса. Называется также как и класс. Может иметь параметры. Можно перегрузить конструктор, задав несколько вариантов, в зависимости от количества и типа параметров.

Деструктор вызывается при удалении инстанса класса. Нужен, например, чтобы освободить за собой память. Называется как класс с тильдой (~) в начале. Деструктор может быть только один.

Конструктор и деструктор не возвращают значение, поэтому их объявление выглядит как-то так

class MyClass {

public:

int add()

MyClass();

MyClass(int, int);

~MyClass();

};

В этом примере add – это просто метод класса, после него идет конструктор без параметров, потом конструктор с двумя параметрами, потом деструктор.