Saad Ahmad

01-134222-130

Lab 4

Classes and Objects II

**Objectives:**

* To understand and define default and overloaded constructor
* To understand array of objects
* To understand pointer to object
* To understand the properties of the class

**Constructor**

Objects generally need to initialize variables or assign dynamic memory during their process of creation to become operative and to avoid returning unexpected values during their execution.

A class can include a special function called constructor, which is automatically called whenever a new object of this class is created. This constructor function must have the same name as the class, and cannot have any return type; not even void.

## Default Constructor

If you do not declare any constructors in a class definition, the compiler assumes the class to have a default constructor with no arguments. Therefore, after declaring a class like this one:

|  |
| --- |
| class Example  {  public:  int a,b,c;  void multiply (int n, int m) { a=n; b=m; c=a\*b; }  }; |

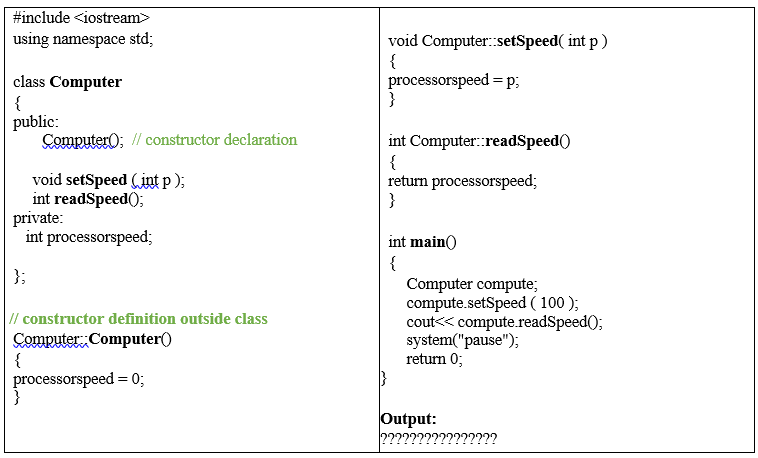
The compiler assumes that Example has a default constructor, so you can declare objects of this class by simply declaring them without any arguments:

Example ex;

If programmer wants to write default constructor, you can follow the following examples.

## Default Constructor

## Example 1:



Output:

100

## Example 2:

**Predict output????????**

## 

Output:

C1=0

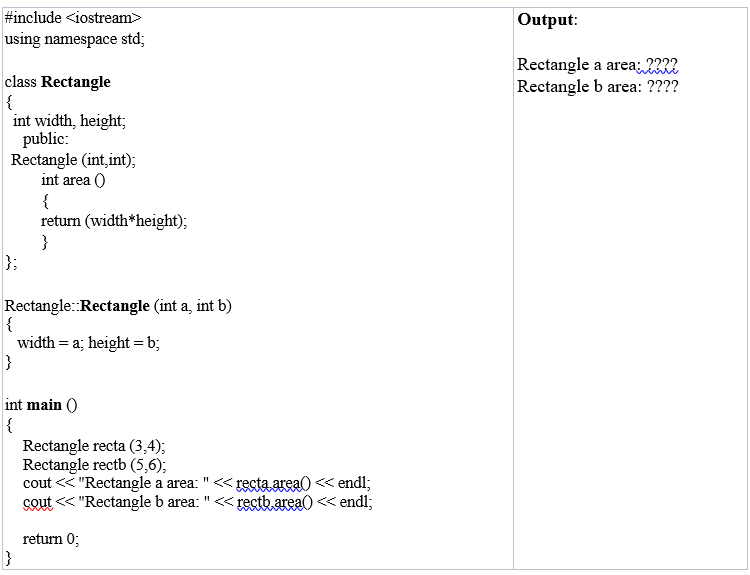
C2=0

C1=1

C2=2

## Parameterized Constructor

**Example 3**

****

**Output:**

**Rectangle a area: 12**

**Rectangle b area: 30**

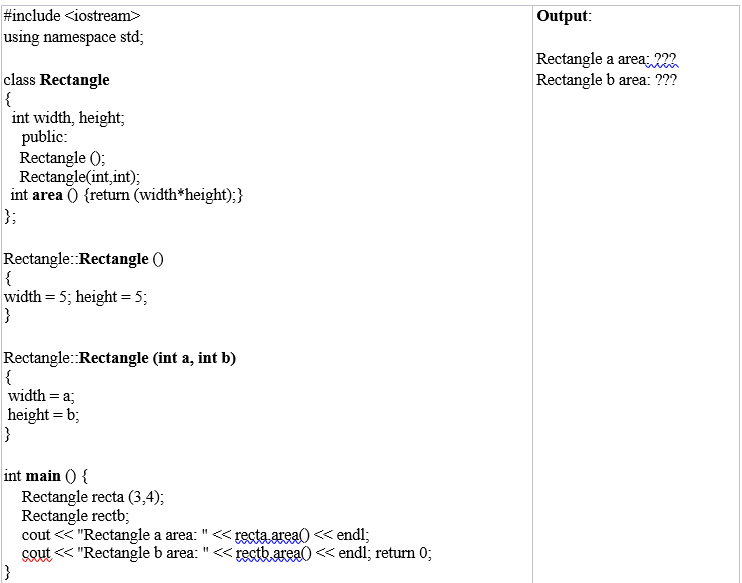


1. **Overloaded Constructors**

Like any other function, a constructor can also be overloaded with more than one function that have the same name but different types or number of parameters. Remember that for overloaded functions the compiler will call the one whose parameters match the arguments used in the function call. In the case of constructors, which are automatically called when an object is created, the one executed is the one that matches the arguments passed on the object declaration.

## Example 4:

As soon as you declare your own constructor for a class, the compiler no longer provides an implicit default constructor. So you have to declare all objects of that class according to the constructor prototypes you defined for the class:



# Output:

Rectangle a area : 12

Rectangle b area : 25

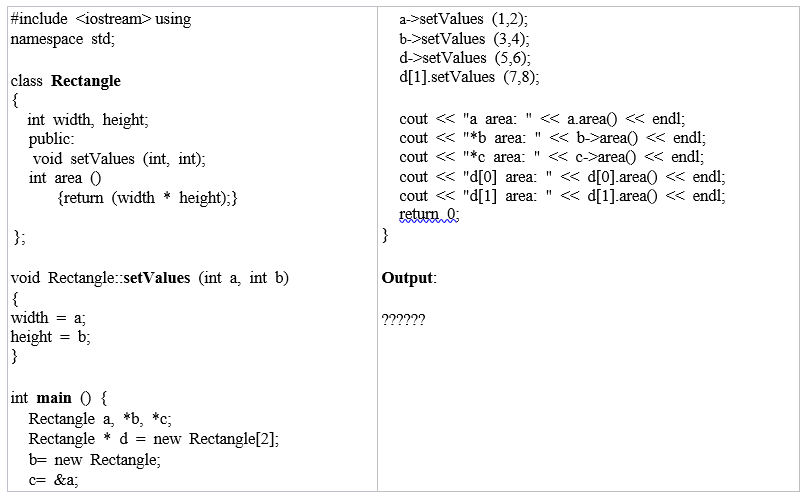
1. **Copy Constructor (Will cover in next lab)**

## Pointer to Class Object

It is perfectly valid to create pointers that point to classes. We simply have to consider that once declared, a class becomes a valid type, so we can use the class name as the type for the pointer. For example: **Rectangle \* prect**; is a pointer to an object of class Rectangle.

As it happened with data structures, in order to refer directly to a member of an object pointed by a pointer we can use the arrow operator (->) of indirection.

### Example 4.8

****

Output:

First fix the error : a.setValues(1,2);

a area: 2

b area: 12

c area: 2

d[0] area: 30

d[1] area: 56

Following is a summary on how can you read some pointer and class operators (\*, &, ., ->, [ ]) that appear in the previous example:

|  |  |
| --- | --- |
| **expression** | **can be read as** |
| \*x | pointed by x |
| &x | address of x |
| x.y | member y of object x |
| x->y | member y of object pointed by x |
| (\*x).y | member y of object pointed by x (equivalent to the previous one) |
| x[0] | first object pointed by x |
| x[1] | second object pointed by x |
| x[n] | (n+1)th object pointed by x |

**Lab Journal Exercises:**

**Exercise 1:**

Create an employee class. The member data should comprise an int for storing the employee number and a float for storing the employee’s compensation. Member functions should allow the user to enter this data and display it. Write **main()** function that allows the user to enter data for three employees and display it (Use Array of objects).

**Code:**

#include <iostream>

using namespace std;

class Employee {

int empnum;

float ecomp;

public :

Employee() {

empnum = 1;

ecomp = 100;

}

void setValues(int a, int b) {

empnum = a;

ecomp = b;

}

void display() {

cout << "Employee number : " << empnum << endl;

cout << "Employee compensation : " << ecomp << endl;

}

};

int main() {

const int size = 3;

int empnum, ecomp;

Employee e[size];

cout << "Enter Y to add employee details, Enter any character to not add employee details :" << endl;

char choice;

cin >> choice;

if (choice == 'Y' || choice == 'y') {

for (int i = 0; i < size; i++) {

cout << "Enter the employee number of employee " << i + 1 << " : " << endl;

cin >> empnum;

cout << "Enter the compensation of employee " << i + 1 << " : " << endl;

cin >> ecomp;

e[i].setValues(empnum, ecomp);

}

for (int i = 0; i < size; i++) {

e[i].display();

}

}

else {

for (int i = 0; i < size; i++) {

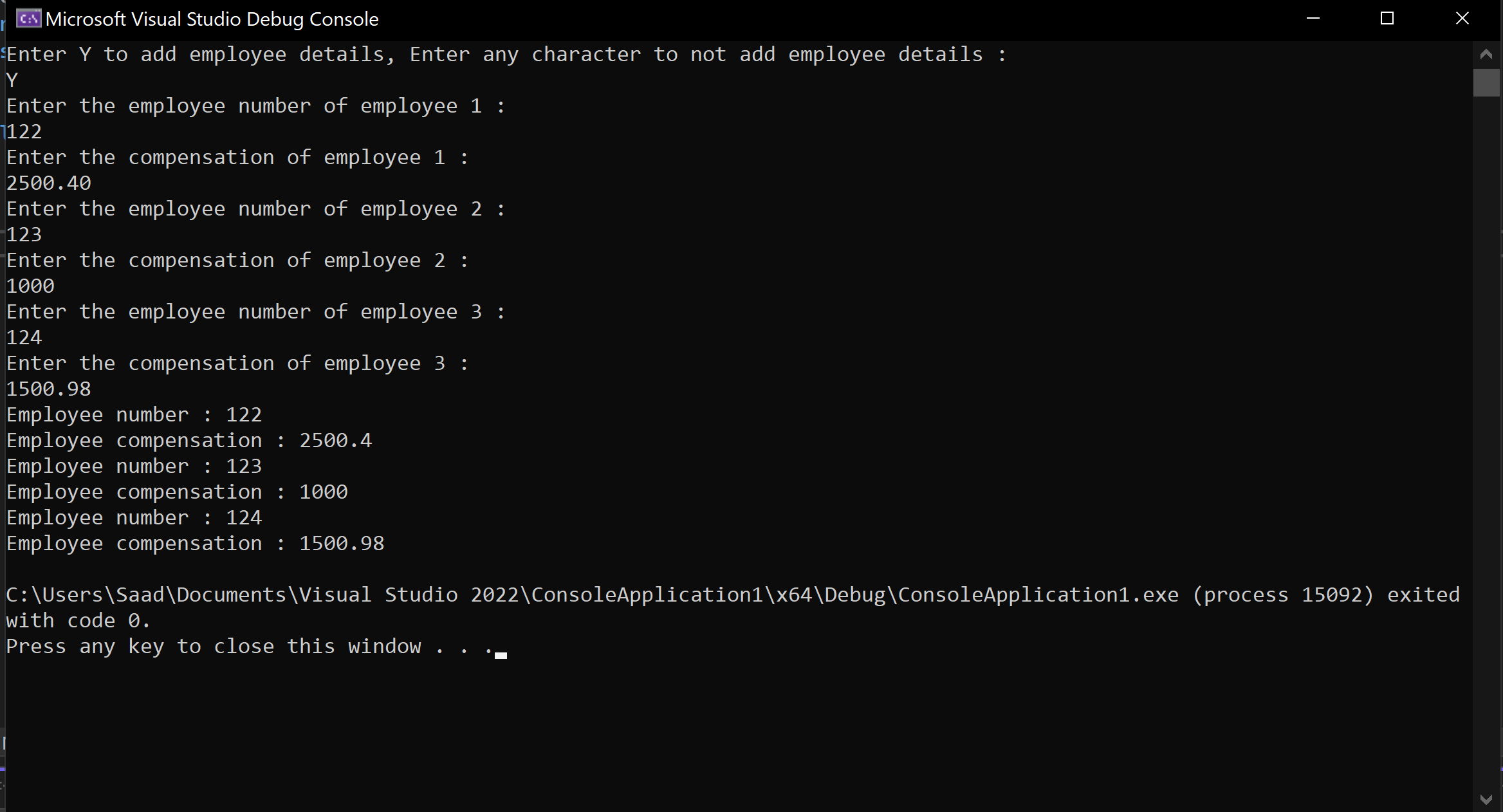
e[i].display();

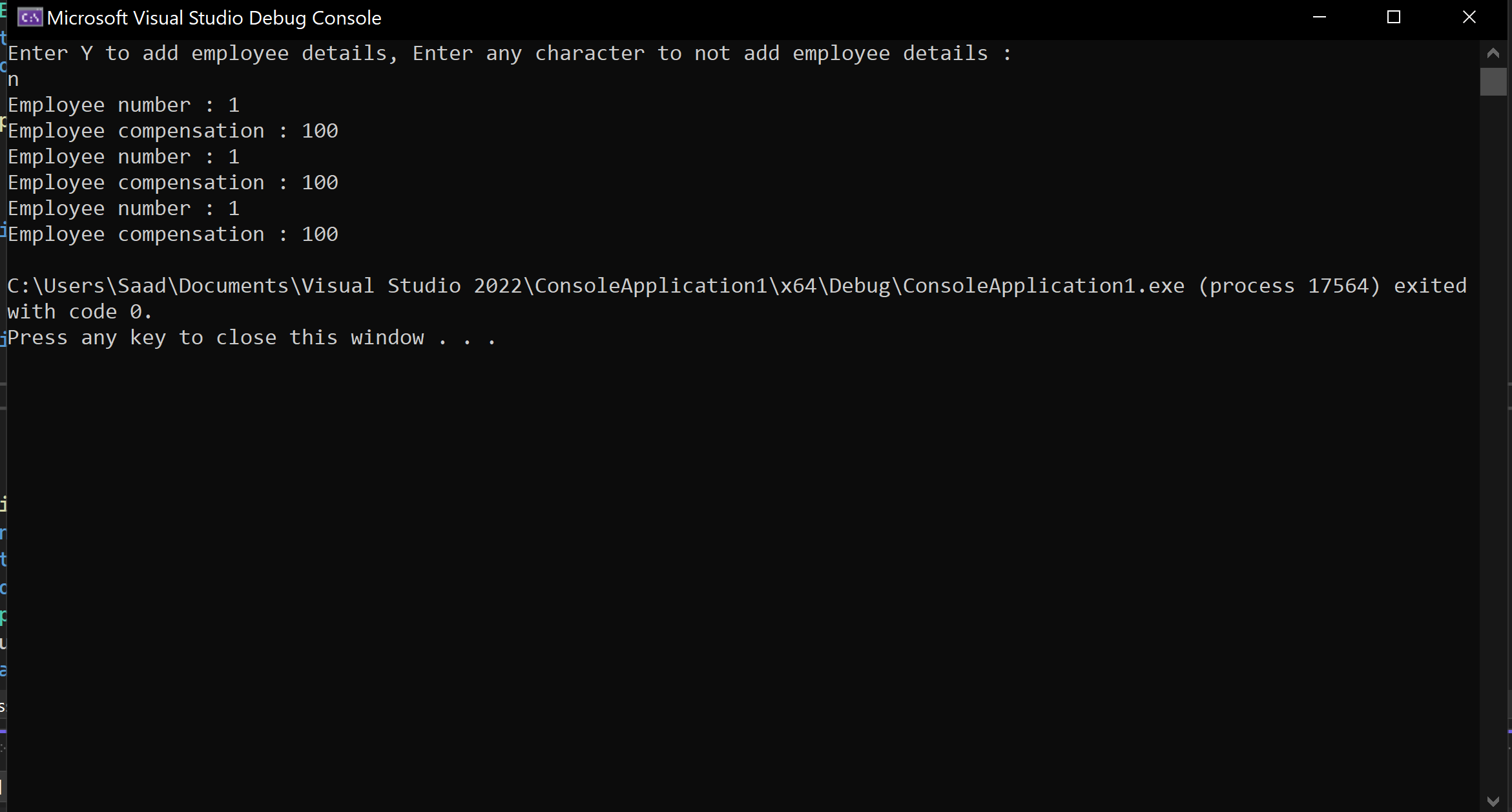
}

}

}

Output:





**Exercise 2:**

Create a C++ class named **Sphere,** which has one data members, radius of the sphere. Write set() and get() methods to store and return the radius of the Sphere class. It also has a function named Area() which calculates the area of the Sphere. Input the radius from the user and calculate the area of the Sphere.

Code:

#include <iostream>

using namespace std;

class Sphere {

float radius;

public:

Sphere(float r) {

setRadius(r);

}

void setRadius(float a) {

radius = a;

}

float getRadius() {

return radius;

}

float Area() {

return (4 \* 3.14 \* (getRadius() \* getRadius()));

}

};

int main() {

float rad;

cout << "Enter the radius of the sphere : " << endl;

cin >> rad;

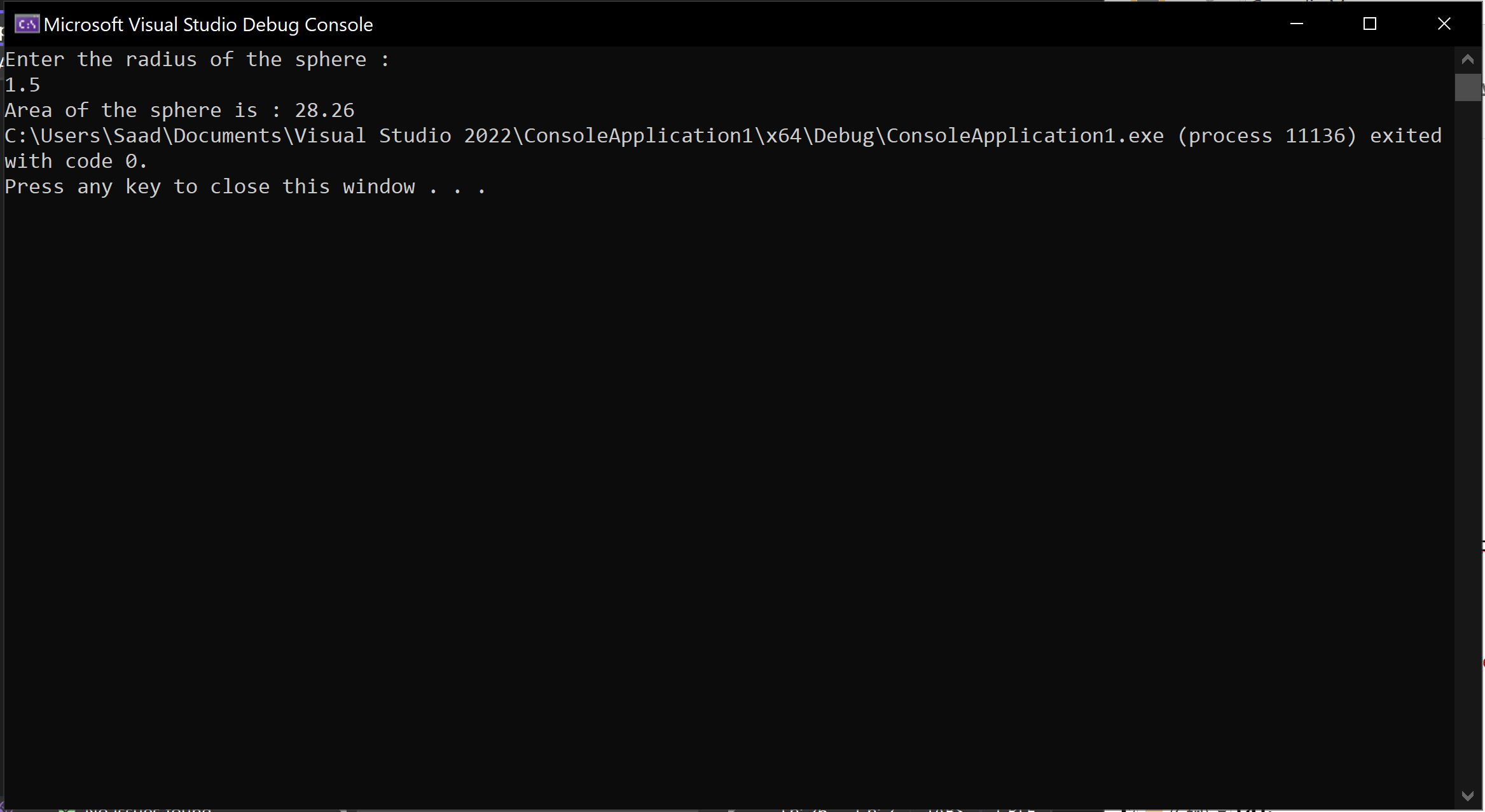
Sphere r(rad);

cout <<"Area of the sphere is : " << r.Area();

return 0;

}

Output:



**Exercise 3 (a):**

Create a class named **myString**,which consists of one string data member and three functions. Input the string from the user at run time then call the following functions on it.

* **void CountConsonants( )** //displays the total number of consonants in the string
* **void VowelCount( )** //displays the total vowels in the string
* **void Palindrome( )** //displays if the string is a palindrome or not

**Code:**

#include <iostream>

using namespace std;

class myString {

string str;

public:

myString(string a) {

str = a;

}

void CountConsonants() {

int count = 0;

for (int i = 0; i < str.length(); i++) {

if (str[i] != 'A' && str[i] != 'E' && str[i] != 'I' && str[i] != 'O' &&

str[i] != 'U' && str[i] != 'a' && str[i] != 'e' && str[i] != 'i' &&

str[i] != 'o' && str[i] != 'u' ) {

count++;

}

}

cout << "There are " << count << " consonants in this word " << endl;

}

void VowelCount() {

int count = 0;

for (int i = 0; i < str.length(); i++) {

if (str[i] == 'A' || str[i] == 'E' || str[i] == 'I' || str[i] == 'O' ||

str[i] == 'U' || str[i] == 'a' || str[i] == 'e' || str[i] == 'i' ||

str[i] == 'o' || str[i] == 'u') {

count++;

}

}

cout << "There are " << count << " vowels in this word " << endl;

}

void Palindrome() {

int count = 0;

int a = str.length() - 1;

for (int i = 0; i < str.length(); i++) {

if (str[i] == str[a]) {

count++;

}

a--;

}

if (count == str.length()) {

cout << "This word is a palindrome" << endl;

}

else {

cout << "This word is not a palindrome" << endl;

}

}

};

int main() {

string str;

cout << "Enter a string : " << endl;

cin >> str;

myString str1(str);

str1.CountConsonants();

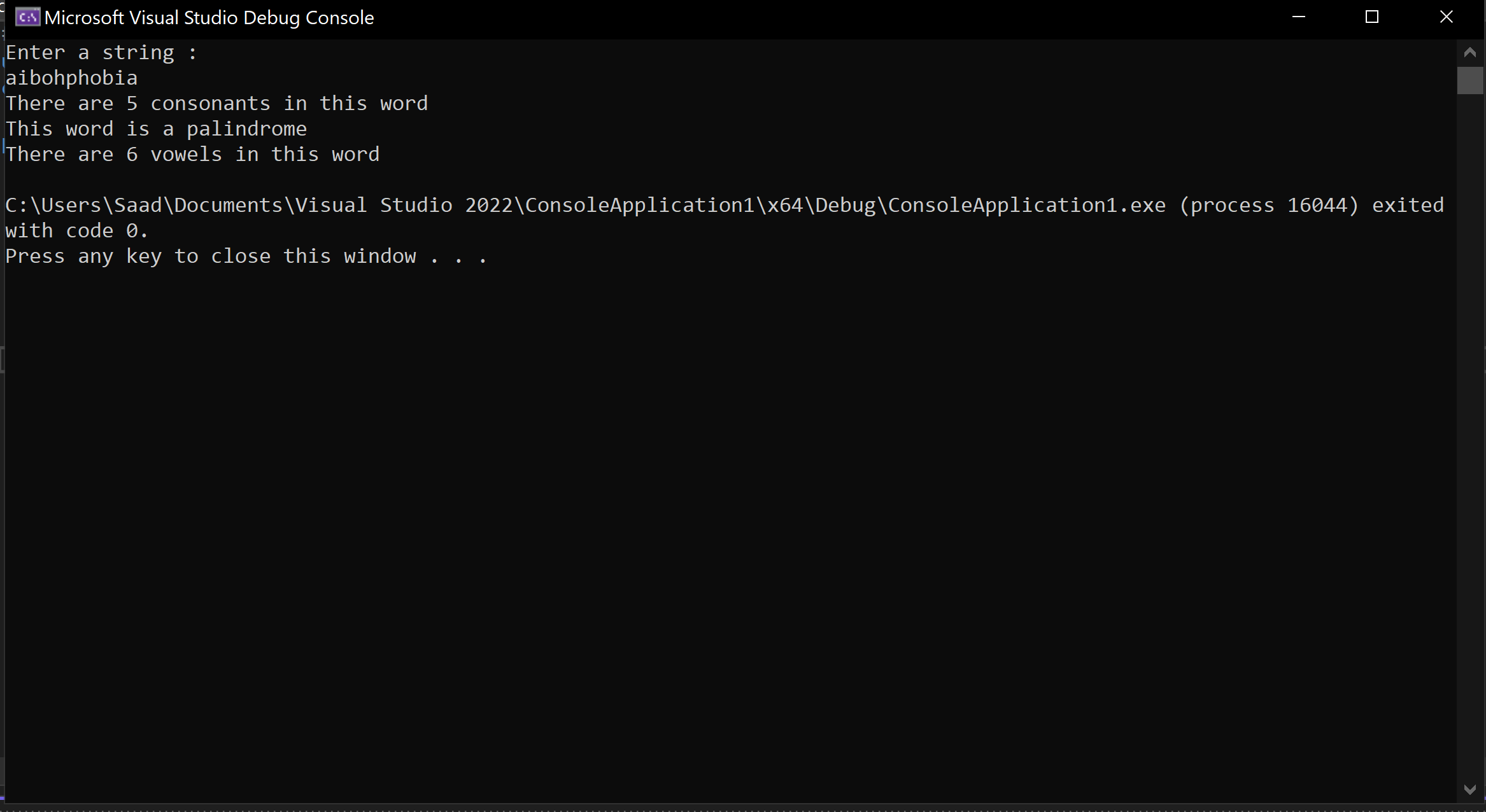
str1.Palindrome();

str1.VowelCount();

return 0;

}

Output:



**Exercise 3 (b):**  modify the above part 3(a) and add following functionality.

* **Use pointer of object to call the member function of the class.**

#include <iostream>

using namespace std;

class myString {

string str;

public:

myString(string a) {

str = a;

}

void CountConsonants() {

int count = 0;

for (int i = 0; i < str.length(); i++) {

if (str[i] != 'A' && str[i] != 'E' && str[i] != 'I' && str[i] != 'O' &&

str[i] != 'U' && str[i] != 'a' && str[i] != 'e' && str[i] != 'i' &&

str[i] != 'o' && str[i] != 'u' ) {

count++;

}

}

cout << "There are " << count << " consonants in this word " << endl;

}

void VowelCount() {

int count = 0;

for (int i = 0; i < str.length(); i++) {

if (str[i] == 'A' || str[i] == 'E' || str[i] == 'I' || str[i] == 'O' ||

str[i] == 'U' || str[i] == 'a' || str[i] == 'e' || str[i] == 'i' ||

str[i] == 'o' || str[i] == 'u') {

count++;

}

}

cout << "There are " << count << " vowels in this word " << endl;

}

void Palindrome() {

int count = 0;

int a = str.length() - 1;

for (int i = 0; i < str.length(); i++) {

if (str[i] == str[a]) {

count++;

}

a--;

}

if (count == str.length()) {

cout << "This word is a palindrome" << endl;

}

else {

cout << "This word is not a palindrome" << endl;

}

}

};

int main() {

string str;

cout << "Enter a string : " << endl;

cin >> str;

myString \*str1 = new myString(str);

str1->CountConsonants();

str1->Palindrome();

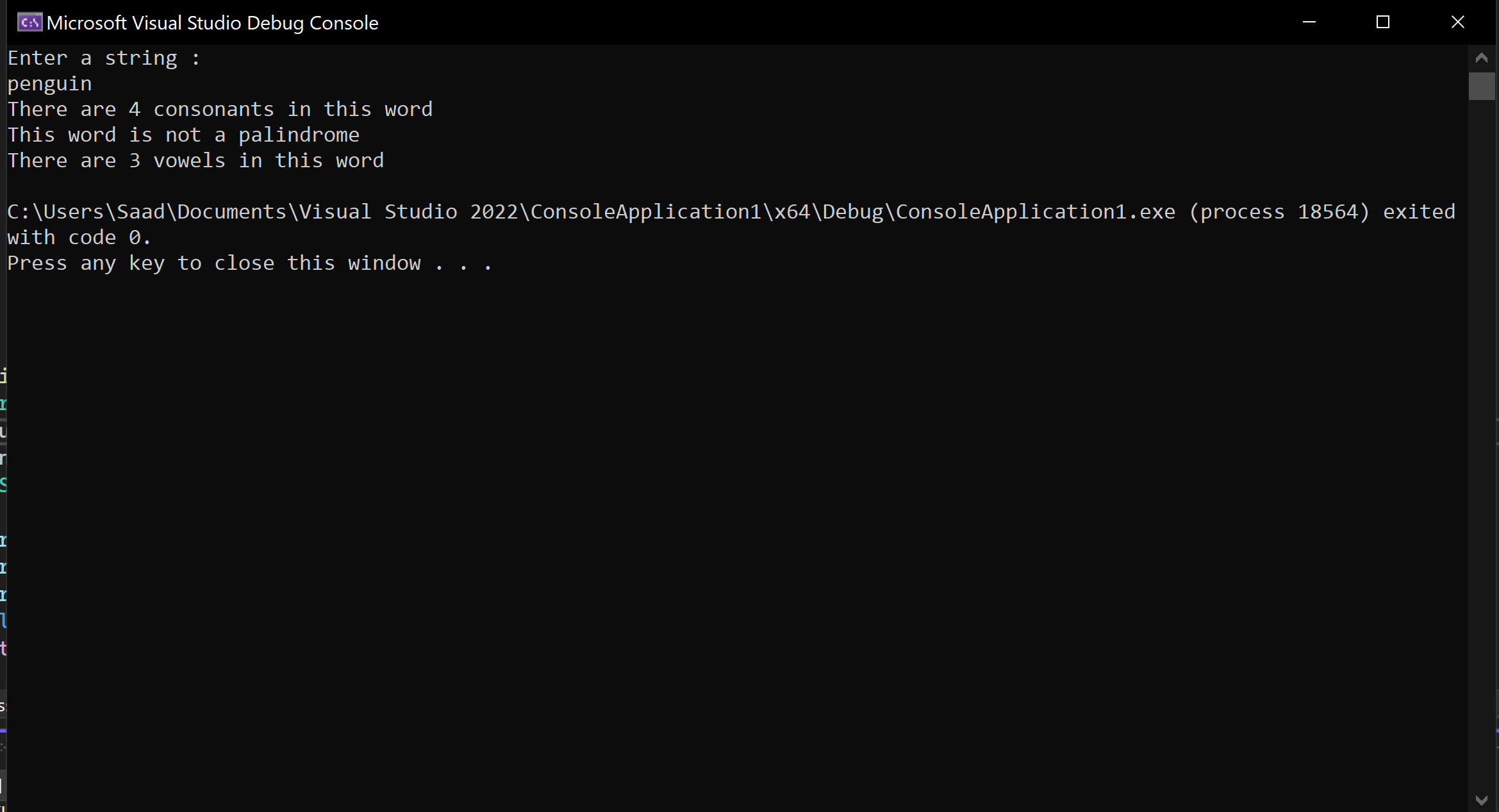
str1->VowelCount();

delete str1;

return 0;

}

**Output:**

****