Saad Ahmad

01-134222-130

**Lab 3**

**Classes & Objects I**

# Objectives

**Following programming skills should be acquired in this lab:**

* To understand the basic principles of data hiding and **encapsulation**.
* To differentiate between a **class** and an **object**.
* To understand and use keyword **public** and **private**.
* To understand the use of **scope resolution operator**.

# Encapsulation

Encapsulation also lead to data abstraction or hiding. As using encapsulation also hides the data. In C++ encapsulation can be implemented using Class and [access modifiers](https://www.geeksforgeeks.org/access-modifiers-in-c/) (public, private keywords). Without encapsulation, which involves the use of one or more classes, it is difficult to define the object oriented programming.

# Basic Terminology

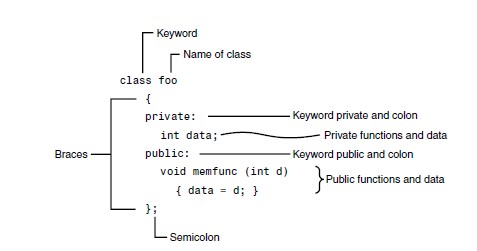
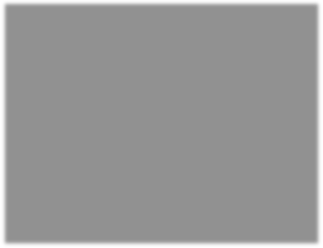
The following is a list of terms that you need to understand in object oriented programming:

|  |  |
| --- | --- |
| **Term** | **Description** |
| class | Is a group of data and methods (functions). A class is very much like a structure, it is just a type used to create a variable which can be manipulated through method in a program. |
| object | Is an instance of a class, which is similar to a variable, defined as an instance of a type. An object is what you actually use in a program since it contains values and can be changed. |
| method | Is a function contained within the class. You will find the functions used within a class often referred to as methods in programming literature. |
| message | Is similar to a function call. In object oriented programming, we send messages instead of calling functions. In programming terms, event or action of the object is normally used to describe a consequence of sending message. |

## Classes and Objects

A ***class*** is an expanded concept of a structure: instead of holding only data, it can hold both data and functions. An *object* is an instantiation of a class. In terms of variables, a class would be the type, and an object would be the variable.

Classes are generally declared using the keyword class, with the following format:



*class class\_name*

*{*

*access\_specifier\_1: member1;*

*access\_specifier\_2:*

*member2;*

*} ;*

An ***access specifier*** is one of the following three keywords: private, public or protected. These specifiers modify the access rights that the members following them acquire:

* ***private*** members of a class are accessible only from within other members of the same class or from their *friends*.
* ***public*** members are accessible from anywhere where the object is visible.

***By default, all members of a class declared with the class keyword have private access for all its members***. Therefore, any member that is declared before one other class specifier automatically has private access.

## Example 3.1

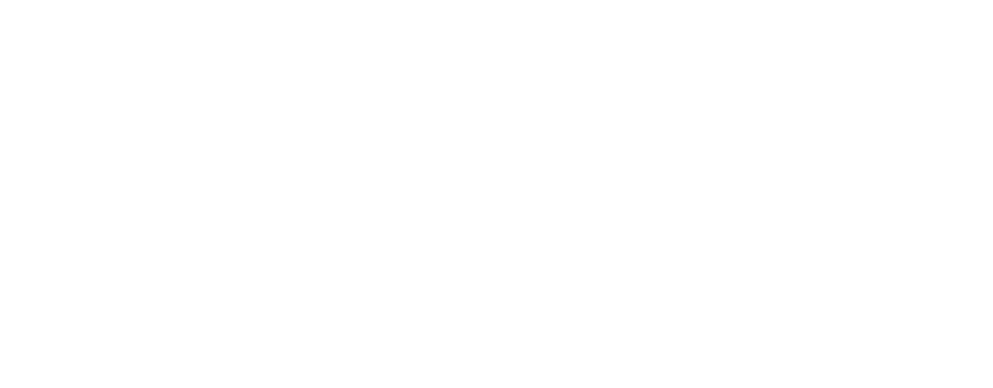
|  |  |
| --- | --- |
| #include <iostream>  using namespace std;    class **Rectangle**  {  private:  int x, y;  public:  void setValues(int a,int b )  {  x = a;  y = b;  }  int area()  {  return (x\*y);  }  };  int **main** ()  {  Rectangle rect; rect.setValues (3,4);  cout << "Area: " << rect.area(); return 0;  } | **Output**:  Area: 12 |

Another way to define the above class

|  |  |
| --- | --- |
| #include <iostream>  using namespace std;    class **Rectangle**  {  int x, y;  public:  void setValues(int a,int b );  int **area**();  };    void Rectangle::**setValues** (int a, int b)  { x = a; y = b; }    int Rectangle::**area**()  { return (x\*y); } | **Output**:  Area: 12 |

The most important new thing in this code is the operator of scope (::, two colons) included in the definition of setValues() and area(). It is used to define a member of a class from outside the class definition itself.





cout

<<

“, costs $” << cost << endl;

}

}

; // end of class

int

**main**

()

{

part part1;

part1.setPart(6244, 373, 217.55); //call member function

part1.showPart();

//call member

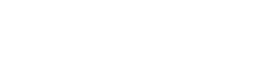
function

return

0

;

}



modelnumber;

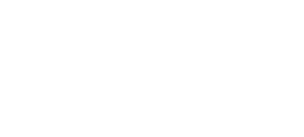
<<

<<

partnumber;



//display data



void

**showPart**

()

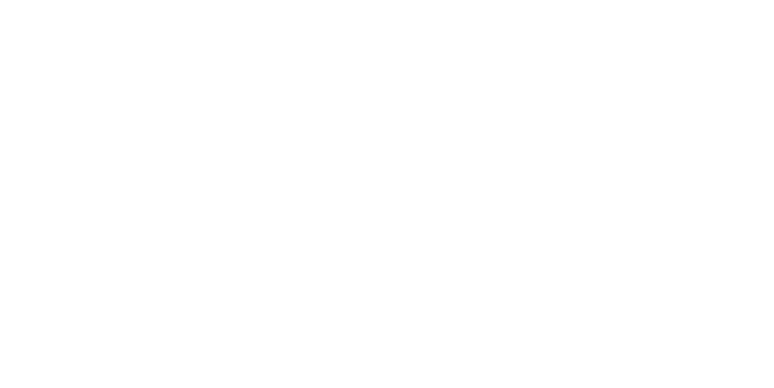
{

cout << “Model “

cout << “, part “



//set data



//ID number of widget

//ID number of widget part

//cost of part

public:

void

**setPart**

(

int mn, int pn, float c

)

{

modelnumber = mn;

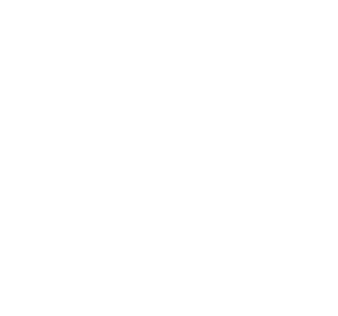
partnumber = pn;

cost = c;

}



//define class



#include <

iostream>

using namespace std;

class

**part**

{

private:

int modelnumber;

int partnumber;

float cost;

**Lab 3 Practice Exercises**

### Exercise 3.1

Write a class to calculate the area and circumference of a circle.

|  |
| --- |
| **Class Circle** |
| **//Area:Πr**  **2**    **Private:**    **float radius;**    **Public:**    **ComputeArea();**    **ComputeCircumference(); //Circumfernce:2Πr**  **void PrintArea();**  **void PrintCircumferecne();**  **void SetRadius(float a);** |

Code:

#include <iostream>

using namespace std;

class Circle {

float radius;

public:

float ComputeArea();

float ComputeCircumference();

void printArea();

void printCircumference();

void setRadius(float a);

};

void Circle::setRadius(float a) {

radius = a;

}

float Circle::ComputeArea() {

float a = 3.14 \* pow(radius, 2);

return a;

}

float Circle::ComputeCircumference() {

float a = 2 \* 3.14 \* radius;

return a;

}

void Circle::printArea() {

cout << "The are of the circle is :" << ComputeArea() << endl;

}

void Circle::printCircumference() {

cout << "The circumference of the circle is : " << ComputeCircumference() << endl;

}

int main() {

Circle c1;

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

float radius;

cin >> radius;

c1.setRadius(radius);

c1.ComputeArea();

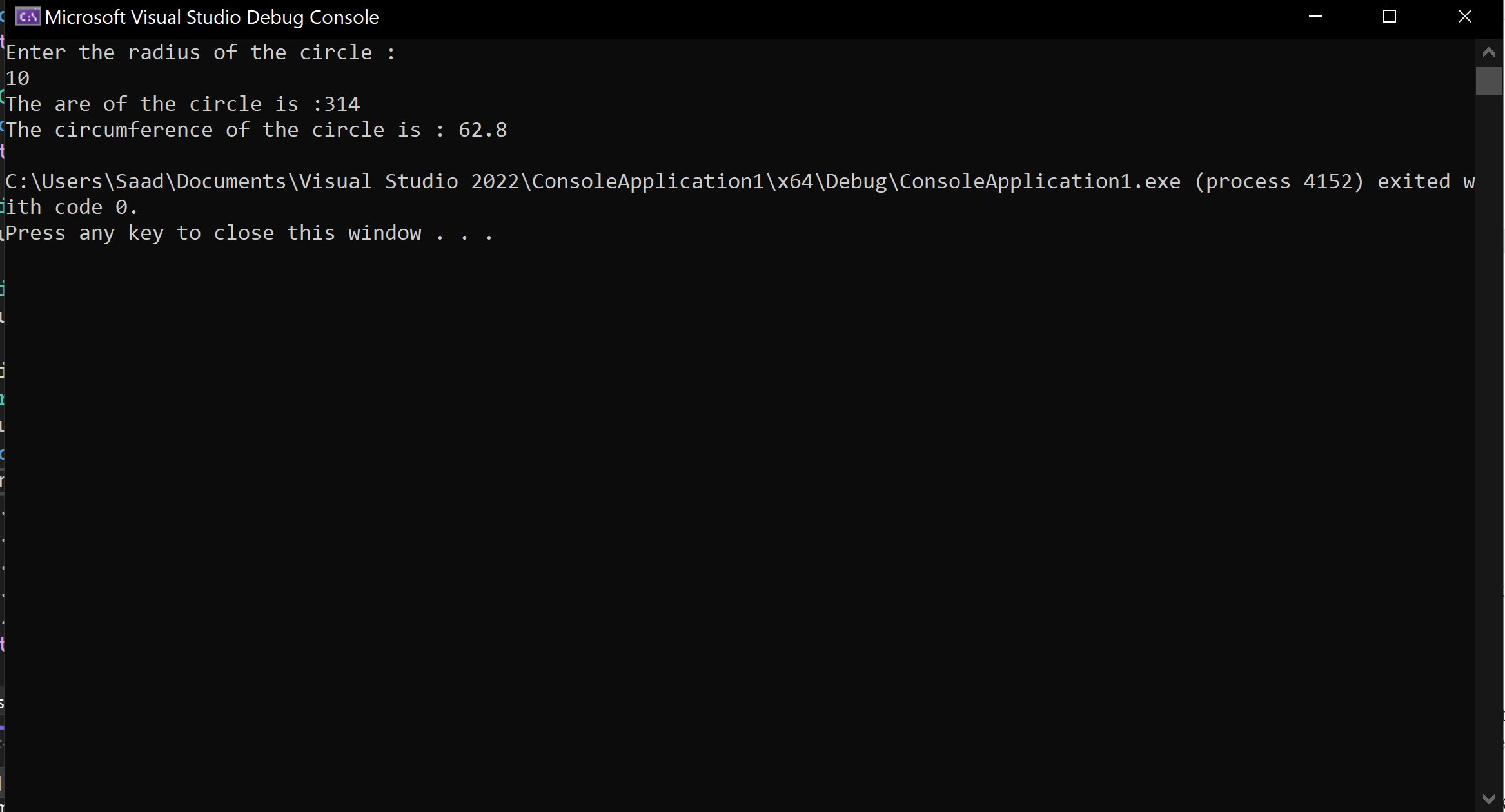
c1.ComputeCircumference();

c1.printArea();

c1.printCircumference();

return 0;

}



### Exercise 3.2

Write a C++ class named **date**. Its member data should consist of three ints: month, day, and year. It should also have two member functions: getdate(), which allows the user to enter a date in 12/31/02 format, and **showdate(),** which displays the date.

**Hint:** use strtok\_s function to tokenize the date which will be in string format, convert it into int to get individual values as month, day and year.

**Code:**

#include <iostream>

using namespace std;

class Date {

int month;

int day;

int year;

public:

void getDate();

void showDate();

};

void Date::getDate() {

char date[10];

cout << "Enter the date in (yy/dd/mm) form :" << endl;

cin >> date;

char\* token;

char\* next\_token;

token = strtok\_s(date, "/", &next\_token);

year = atoi(token);

token = strtok\_s(NULL, "/", &next\_token);

day = atoi(token);

token = strtok\_s(NULL, "/", &next\_token);

month = atoi(token);

}

void Date::showDate() {

cout << day << "/" << month << "/" << year << endl;

}

int main() {

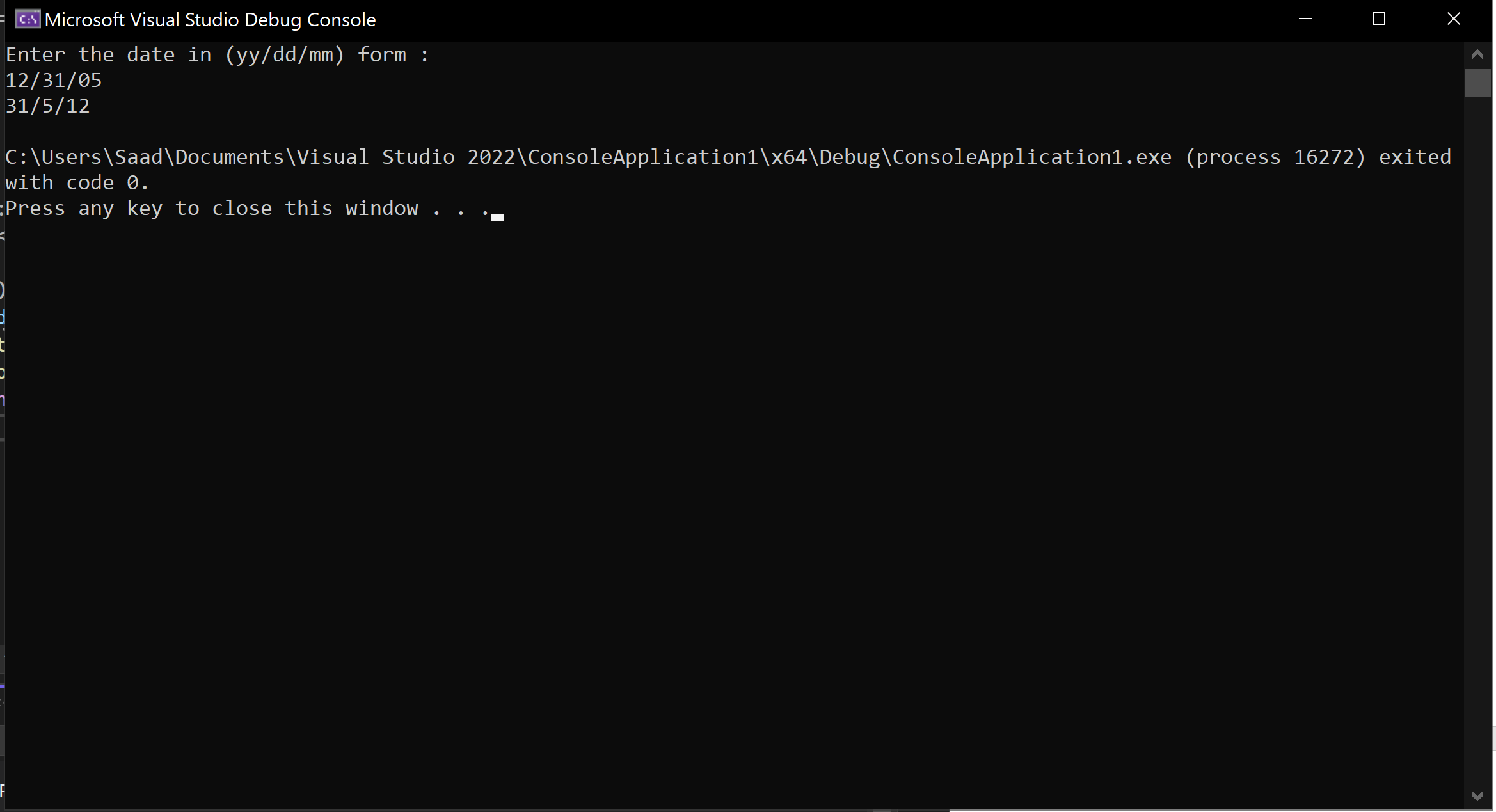
Date d1;

d1.getDate();

d1.showDate();

return 0;

}

**Output:**

### Exercise 3.3

Create a class called ship that incorporates a ship’s name, ship’s number and location in longitudes and latitudes. A member function of the ship class should get a position i.e location from the user and store it in the object; another function should report/display the serial number and position.

Write a main() program that creates three ships, asks the user to input the position of each, and then displays each ship’s number, name and position.

**Note:** location and position is same.

Code:

#include <iostream>

using namespace std;

class Ship {

string name;

int num;

float longitude;

float latitude;

public:

void setData(string a, int id, float longi, float lat) {

name = a;

num = id;

longitude = longi;

latitude = lat;

}

void displayData() {

cout << "Ship name = " << name << endl;

cout << "Ship number = " << num << endl;

cout << "Ship's postition : (" << latitude << "," << longitude << ")" << endl;

cout << endl;

}

};

int main() {

Ship s[3];

string a;

int id;

float longit;

float lat;

for (int i = 0; i < 3; i++) {

cout << "Enter the name of ship " << i + 1 << ":" << endl;

cin >> a;

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

cin >> id;

cout << "Enter the latitude : " << endl;

cin >> lat;

cout << "Enter the longitude : " << endl;

cin >> longit;

s[i].setData(a, id, longit, lat);

}

cout << endl;

for (int i = 0; i < 3; i++) {

s[i].displayData();

}

}

Output:

