

Submitted by : Spogmai jan

Roll no : 2430 - 0081

Submitted to : sir yasir niazi

Subject :oop

Assignment no :01

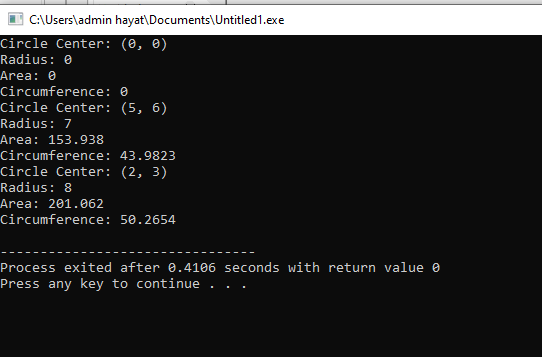
***TASK 1***:

Make a Circle class. It has three attributes radius, the x and the y coordinates of its center as double  
data type.  
a. Make a no argument constructor to initialize it’s attributes to 0, and a three argument  
constructor to initialize with the fixed values given by user.  
b. Make void setValues(float, float, float) functions to set x, y and radius.  
c. Make double area() function, and a double circumference() function to return area  
and circumference.  
d. Make void print() function to display x , y coordinates and radius of a circle.  
e. Call these functions in main() to check their working.

**CODE:**

|  |
| --- |
| #include <iostream>  #define PI 3.14159  using namespace std;  class Circle {  private:  double x, y, radius;  public:  // No-argument constructor  Circle() {  x = 0;  y = 0;  radius = 0;  }  // Three-argument constructor  Circle(double x\_val, double y\_val, double r) {  x = x\_val;  y = y\_val;  radius = r;  }  // Function to set values  void setValues(double x\_val, double y\_val, double r) {  x = x\_val;  y = y\_val;  radius = r;  }  // Function to calculate area  double area() {  return PI \* radius \* radius;  }  // Function to calculate circumference  double circumference() {  return 2 \* PI \* radius;  }  // Function to print details  void print() {  cout << "Circle Center: (" << x << ", " << y << ")" << endl;  cout << "Radius: " << radius << endl;  }  };  int main() {  // Creating a Circle object using no-argument constructor  Circle c1;  c1.print();  cout << "Area: " << c1.area() << endl;  cout << "Circumference: " << c1.circumference() << endl;  // Setting values using setValues function  c1.setValues(5, 6, 7);  c1.print();  cout << "Area: " << c1.area() << endl;  cout << "Circumference: " << c1.circumference() << endl;  // Creating a Circle object using three-argument constructor  Circle c2(2, 3, 8);  c2.print();  cout << "Area: " << c2.area() << endl;  cout << "Circumference: " << c2.circumference() << endl;  return 0;  } |

**Output:**



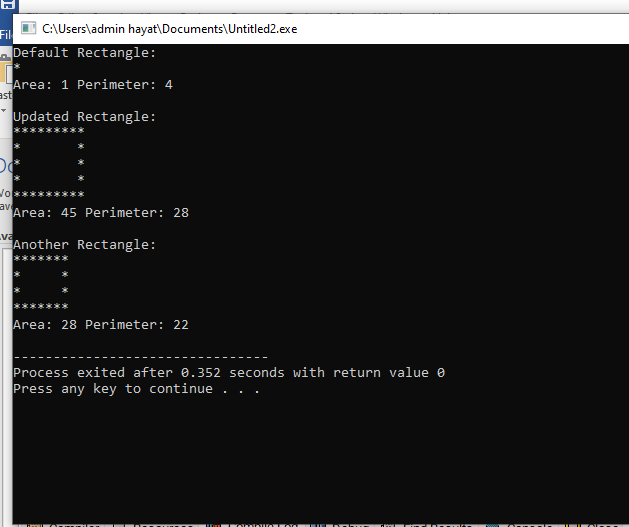
**Task 02:**

Create a class Rectangle with attributes int length and width  
a. Make a no argument constructor to initialize attributes to 1, also make a two argument  
constructor.  
b. Make two member functions to calculate and returns the perimeter and the area of the  
rectangle.  
c. Make void set( int l, int w) function for setting the length and width attributes.  
d. Make a draw() function that draws a rectangle using a character \* on console.  
\*\*\*\*\*\*\*\*\*  
\* \*  
\* \*  
\*\*\*\*\*\*\*\*\*

**Code :**

|  |
| --- |
| #include <iostream>  using namespace std;  class Rectangle {  private:  int length, width;  public:  // No-argument constructor initializing attributes to 1  Rectangle() {  length = 1;  width = 1;  }  // Two-argument constructor  Rectangle(int l, int w) {  length = l;  width = w;  }  // Function to set values  void set(int l, int w) {  length = l;  width = w;  }  // Function to calculate area  int area() {  return length \* width;  }  // Function to calculate perimeter  int perimeter() {  return 2 \* (length + width);  }  // Function to draw rectangle  void draw() {  for (int i = 0; i < width; i++) {  for (int j = 0; j < length; j++) {  if (i == 0 || i == width - 1 || j == 0 || j == length - 1)  cout << "\*";  else  cout << " ";  }  cout << endl;  }  }  };  int main() {  // Creating a Rectangle object using no-argument constructor  Rectangle r1;  cout << "Default Rectangle:" << endl;  r1.draw();  cout << "Area: " << r1.area() << " Perimeter: " << r1.perimeter() << endl;  // Setting new values  r1.set(9, 5);  cout << "\nUpdated Rectangle:" << endl;  r1.draw();  cout << "Area: " << r1.area() << " Perimeter: " << r1.perimeter() << endl;  // Creating a Rectangle object using two-argument constructor  Rectangle r2(7, 4);  cout << "\nAnother Rectangle:" << endl;  r2.draw();  cout << "Area: " << r2.area() << " Perimeter: " << r2.perimeter() << endl;  return 0;  } |

**Output:**



**Task 03:**

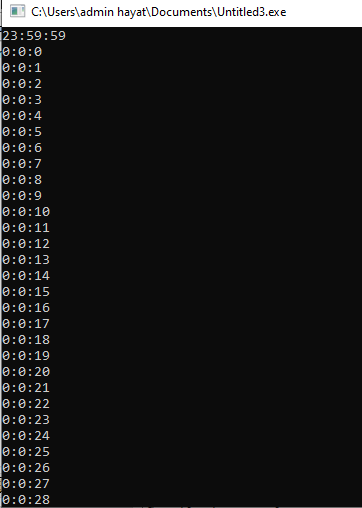
Create a class called time  
a. that has separate int member data for hours, minutes, and seconds.

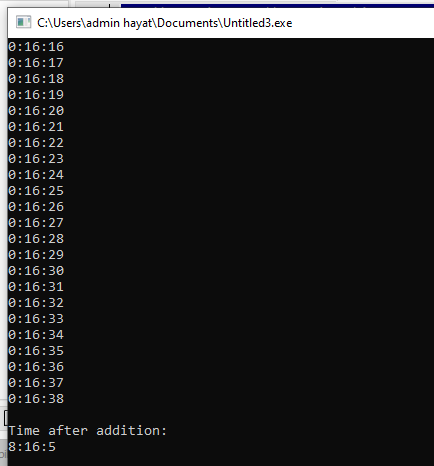
b. One constructor should initialize this data to 0, and another constructor should initialize it to  
fixed values.  
c. Make void print() to display time in 23:59:59 format.  
d. Make void setTime(int,int,int) to set hour, minute, second.  
e. Make a tick() member function that increments the time stored in a time object by one  
second.  
f. Make an add member function that should add two objects of type time passed as arguments.  
Be sure to test the following cases:  
i. Incrementing into the next minute.  
ii. Incrementing into the next hour.  
iii. Incrementing into the next day (i.e., 23:59:59 to 00:00:00).  
g. Make 1000 times loop in a main function. Call tick and print functions in that loop for an object.  
Also make two objects and add them to a third object and print their values

**code :**

|  |
| --- |
| #include <iostream>  using namespace std;  class Time {  private:  int hours, minutes, seconds;  public:  // Constructor initializing time to 0  Time() {  hours = 0;  minutes = 0;  seconds = 0;  }  // Constructor initializing to fixed values  Time(int h, int m, int s) {  hours = h;  minutes = m;  seconds = s;  }  // Function to set time  void setTime(int h, int m, int s) {  hours = h;  minutes = m;  seconds = s;  }  // Function to print time in 23:59:59 format  void print() {  cout << hours << ":" << minutes << ":" << seconds << endl;  }  // Function to increment time by 1 second  void tick() {  seconds++;  if (seconds == 60) {  seconds = 0;  minutes++;  }  if (minutes == 60) {  minutes = 0;  hours++;  }  if (hours == 24) {  hours = 0;  }  }  // Function to add two Time objects  Time add(Time t) {  int totalSeconds = (hours \* 3600 + minutes \* 60 + seconds) +  (t.hours \* 3600 + t.minutes \* 60 + t.seconds);  int h = (totalSeconds / 3600) % 24;  int m = (totalSeconds / 60) % 60;  int s = totalSeconds % 60;  return Time(h, m, s);  }  };  int main() {  Time t1(23, 59, 58);  for (int i = 0; i < 1000; i++) {  t1.tick();  t1.print();  }  Time t2(5, 30, 15), t3(2, 45, 50);  Time t4 = t2.add(t3);  cout << "\nTime after addition:" << endl;  t4.print();  return 0;  } |

**Output:**





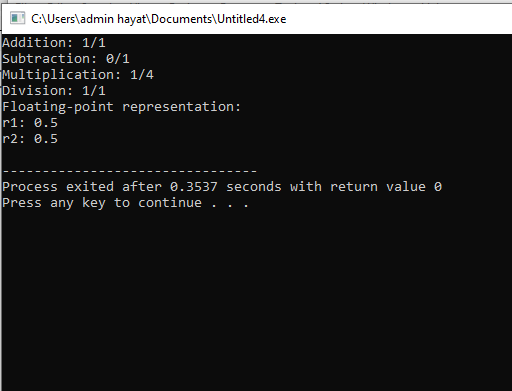
**Task 04:**

Create a class called Rational for performing arithmetic with fractions.  
a. Use integer variables to represent the private data of the class the numerator and the  
denominator.  
b. Provide a constructor that enables an object of this class to be initialized when it is declared.  
The constructor should contain default values in case no initializers are provided and should  
store the fraction in reduced form. For example, the fraction 2/4 would be stored in the object  
as 1 in the numerator and 2 in the denominator.  
c. Provide public member functions that perform each of the following tasks:  
i. Adding two Rational numbers. The result should be stored in reduced form.  
ii. Subtracting two Rational numbers. The result should be stored in reduced form.  
iii. Multiplying two Rational numbers. The result should be stored in reduced form.  
iv. Dividing two Rational numbers. The result should be stored in reduced form.  
v. Printing Rational numbers in the form a/b, where a is the numerator and b is the  
denominator.  
vi. Printing Rational numbers in floating-point format

**Code :**

|  |
| --- |
| #include <iostream>  using namespace std;  class Rational {  private:  int numerator, denominator;  // Function to reduce fraction  void reduce() {  int a = numerator, b = denominator;  while (b != 0) {  int temp = b;  b = a % b;  a = temp;  }  numerator /= a;  denominator /= a;  }  public:  // Constructor with default values  Rational(int num = 0, int den = 1) {  if (den == 0) den = 1; // Avoid division by zero  numerator = num;  denominator = den;  reduce();  }  // Add two Rational numbers  Rational add(Rational r) {  return Rational(numerator \* r.denominator + r.numerator \* denominator, denominator \* r.denominator);  }  // Subtract two Rational numbers  Rational subtract(Rational r) {  return Rational(numerator \* r.denominator - r.numerator \* denominator, denominator \* r.denominator);  }  // Multiply two Rational numbers  Rational multiply(Rational r) {  return Rational(numerator \* r.numerator, denominator \* r.denominator);  }  // Divide two Rational numbers  Rational divide(Rational r) {  return Rational(numerator \* r.denominator, denominator \* r.numerator);  }  // Print in fraction form  void printFraction() {  cout << numerator << "/" << denominator << endl;  }  // Print in floating-point format  void printFloat() {  cout << (double)numerator / denominator << endl;  }  };  int main() {  Rational r1(2, 4), r2(3, 6);  Rational r3 = r1.add(r2);  Rational r4 = r1.subtract(r2);  Rational r5 = r1.multiply(r2);  Rational r6 = r1.divide(r2);  cout << "Addition: "; r3.printFraction();  cout << "Subtraction: "; r4.printFraction();  cout << "Multiplication: "; r5.printFraction();  cout << "Division: "; r6.printFraction();    cout << "Floating-point representation: " << endl;  cout << "r1: "; r1.printFloat();  cout << "r2: "; r2.printFloat();  return 0;  } |

**output**

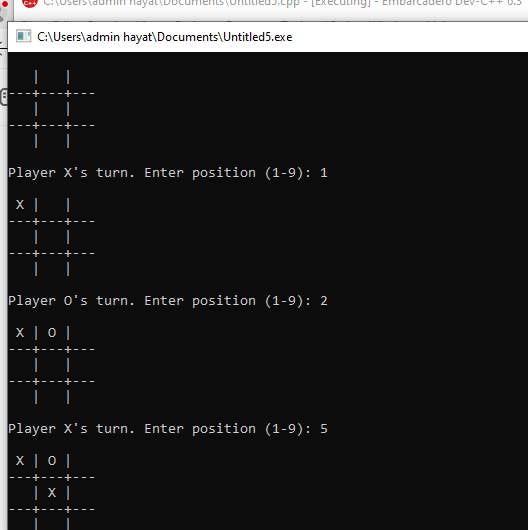


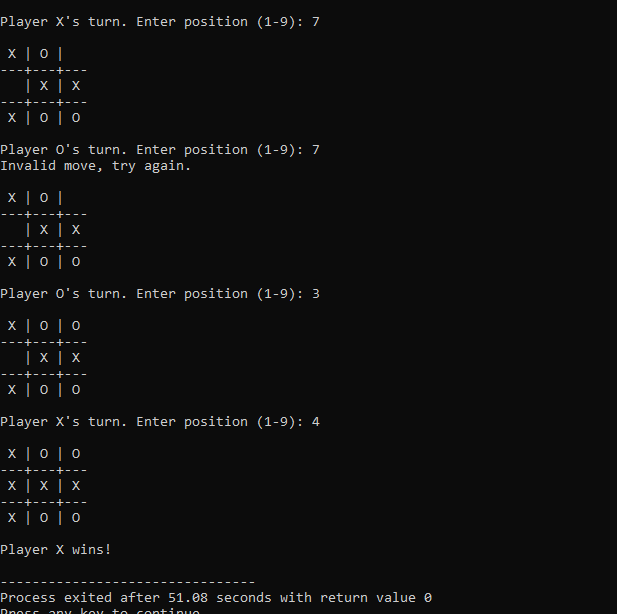
**Task 05 :**

**Code:**

|  |
| --- |
| #include <iostream>  using namespace std;  class TicTacToe {  private:  char board[9];  int moves;  public:  TicTacToe() {  for (int i = 0; i < 9; i++) board[i] = ' '; // Empty board  moves = 0;  }  void displayBoard() {  cout << "\n";  for (int i = 0; i < 9; i++) {  cout << " " << board[i] << " ";  if (i % 3 != 2) cout << "|";  if (i % 3 == 2 && i != 8) cout << "\n---+---+---\n";  }  cout << "\n\n";  }  bool makeMove(int pos, char player) {  if (pos < 1 || pos > 9 || board[pos - 1] != ' ') return false;  board[pos - 1] = player;  moves++;  return true;  }  bool checkWin() {  int winPatterns[8][3] = {  {0, 1, 2}, {3, 4, 5}, {6, 7, 8},  {0, 3, 6}, {1, 4, 7}, {2, 5, 8},  {0, 4, 8}, {2, 4, 6}  };  for (auto &p : winPatterns) {  if (board[p[0]] != ' ' && board[p[0]] == board[p[1]] && board[p[1]] == board[p[2]])  return true;  }  return false;  }  bool isDraw() {  return moves == 9 && !checkWin();  }  };  int main() {  TicTacToe game;  int move;  char player = 'X';  while (true) {  game.displayBoard();  cout << "Player " << player << "'s turn. Enter position (1-9): ";  cin >> move;  if (!game.makeMove(move, player)) {  cout << "Invalid move, try again.\n";  continue;  }  if (game.checkWin()) {  game.displayBoard();  cout << "Player " << player << " wins!\n";  break;  }  if (game.isDraw()) {  game.displayBoard();  cout << "It's a draw!\n";  break;  }  player = (player == 'X') ? 'O' : 'X';  }  return 0;  } |

**Output:**





**=============================**