Programación Orientada a Objetos 1

Práctica Calificada 1

Pregrado

2020-I

Profesora: María Hilda Bermejo

Lab. 2.02

Indicaciones específicas:

- Esta evaluación contiene 11 páginas (incluyendo esta página) con 3 preguntas. El total de puntos son 20.
- El tiempo límite para la evaluación es 100 minutos.
- Cada pregunta deberá ser respondida en un solo archivo con el número de la pregunta y tu código de estudiante. Por ejemplo:
  1. `p1_2020010202.cpp`
  2. `p2_2020010202.cpp`
  3. `p3_2020010202.cpp`
- Estos archivos deben estar en una carpeta llamada PC1. Una vez que termines de responder a las preguntas. Comprime la carpeta PC1, lo cual generará el archivo PC1.Zip
- Deberás subir el archivo **PC1.zip** directamente a www.gradescope.com, uno en cada ejercicio.

Competencias:

- Para los alumnos de la carrera de Ciencia de la Computación

  Aplicar conocimientos de computación y de matemáticas apropiadas para la disciplina. (Evaluar)

  Analizar problemas e identificar y definir los requerimientos computacionales apropiados para su solución.(Usar)

  Utilizar técnicas y herramientas actuales necesarias para la práctica de la computación. (Usar)

- Para los alumnos de las carreras de Ingeniería

  Capacidad de aplicar conocimientos de matemáticas (nivel 3)

  Capacidad de aplicar conocimientos de ingeniería(nivel 2)

  Capacidad para diseñar un sistema, un componente o un proceso para satisfacer las necesidades deseadas dentro de restricciones realistas (nivel 2)

Calificación:

Tabla de puntos (sólo para uso del professor)

Question	Points	Score
1	6	
2	7	
3	7	
Total:	20	

1. (6 points) **Dígitos iguales**
   Desarrolle un programa que permita imprimir todos los números que tengan dīgitos iguales, desde el 11 hasta un número que se ingrese como límite.

   - El programa debe verificar que el número que se ingrese como límite sea mayor o igual a 11. Si el usuario ingresa un número que no es mayor o igual a 11, debe volver a pedir el dato.
   - Note que el límite puede ser un número de varios dígitos.

   Algunos ejemplos de diálogo de este programa serían:

Listing 1: Ejemplo 1

```
Numero > 10  : 8
Numero > 10  : -3
Numero > 10  : 10
Numero > 10  : 999

11
22
33
44
55
66
77
88
99
111
222
333
444
555
666
777
888
999
```

Listing 2: Ejemplo 1

```
Numero > 10  : 11

11
```

Listing 3: Ejemplo 1

```
Numero > 10  : 9000
```

```
11
22
33
44
55
66
77
88
99
111
222
333
444
555
666
777
888
999
1111
2222
3333
4444
5555
6666
7777
8888
```

Listing 4: Ejemplo 1

```
Numero > 10   : 123456789

11
22
33
44
55
66
77
88
99
111
222
333
444
555
```

```
666
777
888
999
1111
2222
3333
4444
5555
6666
7777
8888
9999
11111
22222
33333
44444
55555
66666
77777
88888
99999
111111
222222
333333
444444
555555
666666
777777
888888
999999
1111111
2222222
3333333
4444444
5555555
6666666
7777777
8888888
9999999
11111111
22222222
33333333
44444444
55555555
```

```
66666666
77777777
88888888
99999999
111111111
```

La rúbrica para esta pregunta es:

Criterio	Logrado	Parcialmente Logrado	No Logrado
Algoritmo y Código	El algoritmo y código es preciso y finito y hace exactamente lo que el enunciado requiere. (`3pts`)	Es preciso, finito y hace la mitad o más de lo que el enunciado requiere. (`2pts`)	Hace menos de la mitad de lo que el enunciado requiere (`0pts`).
Sintaxis	No existen errores sintácticos o de compilación (`2pts`)	Existen algunos errores sintácticos o de compilación. (`1pts`).	El código no compila (`0pts`).
Optimizacion	El código es óptimo y eficiente (`1pts`)	El codigo es optimizable en algunas partes (`0.5pts`).	El codigo es redundante y/o no es optimo (`0pts`).

2. (7 points) **Polígonos**

   Desarrolle un programa que permita calcular el perímetro de un polígono, si se sabe el número de vértices y las coordenadas de cada uno de los vértices.

   - El número de vértices, es un número mayor o igual a 3 y menor o igual a 10. El programa debe verificar el ingreso del dato, y volverá a pedir otro valor si no se encuentra en ese rango.
   - El programa irá solicitando el valor de cada vértice e irá calculando el valor de cada lado. Note que para calcular el valor el último lado, debe considerar la última y la primera coordena ingresada.
   - Para hallar la distancia entre dos coordenadas puede utilizar la fórmula del teorema de Pitágoras:

- Distancia = $\sqrt{(x2 - x1)^2 + (y2 - y1)^2}$

   Adicionalmente el programa indicará el nombre del polīgono según la siguiente tabla:

Número de lados	Polígono
3	Triángulo
4	Cuadrilátero
5	Pentágono
6	Hexágono
7	Heptágono
8	Octógono
9	Nonágono
10	Decágono

   Algunos ejemplos de diálogo de este programa serían:

Listing 5: Ejemplo 1

```
Numero de vertices : 4
Vertice num 1
x = 3
y = 2
Vertice num 2
x = 6
y = 2
Vertice num 3
x = 6
y = 5
Vertice num 4
x = 3
y = 5
```

```
Es un Cuadrilatero
Su perimetro es : 12
```

Algunos ejemplos de diálogo de este programa serían:

Listing 6: Ejemplo 1

```
Numero de vertices : 2
Numero de vertices : 14
Numero de vertices : 6
Vertice num 1
x = 2
y = 4
Vertice num 2
x = 4
y = 4
Vertice num 3
x = 5
y = 3
Vertice num 4
x = 4
y = 1
Vertice num 5
x = 2
y = 1
Vertice num 6
x = 2
y = 3
Es un Hexagono
Su perimetro es : 10.6503
```

La rúbrica para esta pregunta es:

Criterio	Logrado	Parcialmente Logrado	No Logrado
Algoritmo y Código	El algoritmo y código es preciso y finito y hace exactamente lo que el enunciado requiere. (4pts)	Es preciso, finito y hace la mitad o más de lo que el enunciado requiere. (2pts)	Hace menos de la mitad de lo que el enunciado requiere (0pts).
Sintaxis	No existen errores sintácticos o de compilación (2pts)	Existen algunos errores sintácticos o de compilación. (1pts).	El código no compila (0pts).
Optimizacion	El código es óptimo y eficiente (1pts)	El codigo es optimizable en algunas partes (0.5pts).	El codigo es redundante y/o no es optimo (0pts).

3. (7 points) **Días**
Desarrolle un programa que permita leer como dato una fecha: dia, mes y año, para que luego el programa indique:

   - La cantidad de días que han transcurrido desde el 1ero de enero de ese año hasta la fecha ingresada. Note que el día de la fecha actual, aun no ha transcurrido.
   - La cantidad de días que faltan para que llegue año nuevo.

   Recuerde que la cantidad de días que tiene cada mes es como se indica en la tabla:

Número de mes	Número de días
1,3,5,7,8,10,12	31
4,6,9,11	30
2	28 si no es año bisiesto y 29 si es bisiesto

   Para efectos de este programa, considere que un año es bisciesto si es múltiplo de 4.

   A continuación se le presenta el código correspondiente a la función main(), y se pide que Ud, escriba el programa completo, implementando la **función contarDias**. Que está enviado en sus dos últimos parámetros, **las direcciones de memoria** de las variables: **diasTranscurridos** y **diasParaAnioNuevo**, en donde la función dejará los datos correspondientes.
   Adicionalmente a esta función, Ud. puede diseñar e implementar las funciones que considere necesarias.

Listing 7: Ejemplo 1

```
#include <iostream>
using namespace std;

int main()
{ unsigned int dia=0, mes=0, anio=0;
  unsigned int diasTranscurridos=0, diasParaAnioNuevo=0;

 cout << "Dia:"; cin >> dia;
 cout << "Mes:"; cin >> mes;
 cout << "Annio:"; cin >> anio;

 contarDias(dia, mes, anio, &diasTranscurridos,
            &diasParaAnioNuevo);
 cout << "Han␣transcurrido:" << diasTranscurridos << "\n";
 cout << "Faltan␣para␣anio␣Nuevo:" << diasParaAnioNuevo;
 return 0;
}
```

Algunos ejemplos de diálogo de este programa serían:

Listing 8: Ejemplo 1

```
Dia   : 14
Mes   : 10
Annio : 2020
Han transcurrido  : 287
Faltan para anio Nuevo : 78
```

Listing 9: Ejemplo 1

```
Dia   : 30
Mes   : 6
Annio : 2020
Han transcurrido  : 181
Faltan para anio Nuevo : 184
```

Listing 10: Ejemplo 1

```
Dia   : 27
Mes   : 7
Annio : 2009
Han transcurrido  : 207
Faltan para anio Nuevo : 157
```

Listing 11: Ejemplo 1

```
Dia   : 1
Mes   : 1
Annio : 2020
Han transcurrido  : 0
Faltan para anio Nuevo : 365
```

Listing 12: Ejemplo 1

```
Dia   : 31
Mes   : 12
Annio : 2019
Han transcurrido  : 364
Faltan para anio Nuevo : 0
```

La rúbrica para esta pregunta es:

Criterio	Logrado	Parcialmente Logrado	No Logrado
Código	Ha implementado funciones y punteros en forma correcta y lógica (`4pts`)	Existen algunos errores menores en la implementación (`2pts`)	El diseño y la implementacion del código no son correctos (`0pts`).
Sintaxis	No existen errores sintácticos o de compilación (`2pts`)	Existen algunos errores sintácticos o de compilación. (`1pts`).	El código no compila (`0pts`).
Optimizacion	El código es óptimo y eficiente (`1pts`)	El código es optimizable en algunas partes (`0.5pts`).	El código es redundante y/o no es óptimo (`0pts`).