UTEC
UNIVERSIDAD DE INGENIERÍA Y TECNOLOGÍA

# Programación Orientada a Objetos 1

**Práctica Calificada 1**
**Pregrado**
**2021-I**
**Profesor: Rubén Rivas**
**Lab. 1.06**

## Indicaciones específicas:

- Esta evaluación contiene 7 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 su correspondiente par de archivos, cabecera y fuentes, con el número de la pregunta. Por ejemplo:
  1. `p1.cpp`, `p1.h`
  2. `p2.cpp`, `p2.h`
  3. `p3.cpp`, `p2.h`
- Deberás subir estos archivos, individuales o comprimidos en un archivo **ZIP**, directamente a www.gradescope.com.

## Competencias y criterios de desempeño:

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

  Aplica conocimientos de computación apropiados para la solución de problemas definidos y sus requerimientos en la disciplina del programa. (nivel 2)

  Diseña, implementa y evalúa soluciones a problemas complejos de computación. (nivel 2)

  Crea, selecciona, adapta y aplica técnicas, recursos y herramientas modernas para la práctica de la computación y comprende sus limitaciones. (nivel 2)

- Para los alumnos de las carreras de Ingeniería

  Capacidad para aplicar conocimientos de matemática. (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 | 7 | |
| 2 | 7 | |
| 3 | 6 | |
| Total: | 20 | |

1. (7 points) **Emparejando dígitos**
Escribir un programa que permita leer un texto que incluirá solamente caracteres numéricos ('0','1','2','3','4','5','6','7','8','9'), el programa debe ubicar el dígito dentro del texto de mayor valor numérico luego de ello debe cálcular cuanto se debe incrementar cada dígito para que todos tengan el mismo valor del dígito mayor, el programa deberá mostrar la suma de todos esos incrementos.

Listing 1: Input Format

```
12351230
```

- El ingreso de los valores no requiere utilizar etiquetas (std::cout)

Listing 2: Output Format

```
23
```

Algunos ejemplos de diálogo de este programa serían:

Ejemplo 1

```
103840182920284941
```

Output

```
96
```

Ejemplo 2

```
2363517534326251610
```

Output

```
96
```

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`). |

2. (7 points) **El producto más grande**
Escribir un programa y una función que reciba 2 parámetros: el primero, un texto que solo incluirá caracteres numéricos y el segundo, un valor numérico entero que represente una cantidad **d** de dígitos adyacentes (uno al lado de otro). El programa deberá buscar y retornar la secuencia de **d** dígitos adyacentes de modo que al multiplicar todos los dígitos de la secuencia se obtenga el mayor valor posible.

Listing 3: Input Format

```
17939491
3
```

- No se requiere validar los valores, se debe asumir que se ingresa correctamente los digitos
- El ingreso de los valores no requiere utilizar etiquetas (std::cout)

Listing 4: Output Format

```
949
```

Algunos ejemplos de diálogo de este programa serían:

Ejemplo 1

```
12319291028849102823401728430498129204584731912819292829
29
5
```

Output

```
28849
```

Ejemplo 2

```
44736293746521740584647584527904848383942193847305858334845
4
5848384583211838173849573848
7
```

Output

```
8495738
```

La rúbrica para esta pregunta es:

| Criterio | Logrado | Parcialmente Logrado | No Logrado |
|---|---|---|---|
| Código | Ha implementado funciones 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`). |

3. (6 points) **Reemplazar un caracter**
Escribir una función recursiva cuyo nombre sea **reemplazar_caracter** que reciba 3 parámetros el primero del tipo std::string, el segundo de tipo char que defina el **carácter actual** dentro del texto que será reemplazado y el tercero también de tipo char que defina el **nuevo carácter**, la función debe de permitir reemplazar el **carácter actual** por el **nuevo carácter**. La función retornará el texto modificado usando el tipo de retorno de la función.

NOTA: El texto original no debe ser modificado.

Listing 5: Input Format

```
texto de prueba
e
-
```

- El ingreso de los valores no requiere utilizar etiquetas (std::cout)

Listing 6: Output Format

```
texto de prueba
t-xto d- pru-ba
```

Algunos ejemplos de diálogo de este programa serían:

Ejemplo 1

```
reemplazar un caracter
a
#
```

Output

```
reemplazar un caracter
reempl#z#r un c#r#cter
```

Ejemplo 2

```
15,29,12,14,2,3,11
,
|
```

Output

```
15,29,12,14,2,3,11
15|29|12|14|2|3|11
```

La rúbrica para esta pregunta es:

| Criterio | Logrado | Parcialmente Logrado | No Logrado |
|---|---|---|---|
| Código | Ha implementado funciones recursivas 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 (`1.0pts`) | Existen algunos errores sintácticos o de compilación. (`0.5pts`). | 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`). |