| LAB 1 | Profesor: Rubén Calabuig |
|---|---|
| Trabajo Práctico Integrador Nº 01 | ESTRUCTURAS FUNCIONES Y LIBRERÍAS |

# FICHA TÉCNICA

**ÁREA DE INFLUENCIA: Programación**
**EDAD DE LOS ALUMNOS: Estudiantes de la Tecnicatura Superior en Programación**
**Año 1 – Cuatrimeste 1.**

---

# INTRODUCCIÓN

Tanto las estructuras como las funciones constituyen los pilares del paradigma de Programación Modular y Estructurada (PME).

Este paradigma, está ampliamente difundido en la prácticas de programación en los ámbitos profesionales, y si bien está siendo reemplazado paulatinamente por la Programación Orientada a Objetos (OOP), constituye un antecedente fundamental para el desarrollo de esta última.

Es por lo expuesto que ningún aspirante a ser un profesional de la programación puede dejar de conocer y dominar todos los conceptos y técnicas que la PME requiere.

---

# COMPETENCIAS

Al finalizar el trabajo práctico, deberás:

- Haber desarrollado las competencias que le permitan el análisis crítico en la solución de problemas.
- Haber adquirido conceptos, procedimientos y modelos para el desarrollo de algoritmos.
- Haber desarrollado confianza, disciplina y perseverancia en la programación de algoritmos.
- Poseer la habilidad práctica que requiere la programación.
- Tener nociones generales sobre el paradigma de programación modular y estructurada.
- Comprender y aplicar el concepto de portabilidad.
- Comprender y utilizar diferentes entornos de programación.
- Codificar algoritmos en lenguaje C/C++.
- Utilizar el vocabulario específico del área.

---

# TAREA

Al finalizar el trabajo práctico, deberás entregar un archivo de librería con funciones de cadena codificadas en lenguaje C++, y un archivo menú, codificado en el mismo lenguaje, donde demuestres con ejemplos los usos de cada una de las funciones incluidas en la librería mencionada.

---

# PRODUCTO

- Un archivo de librería tipo “.h” con todas las funciones solicitadas, codificadas en lenguaje C++.
- Un archivo de tipo “.cpp” donde se incluya la librería de tipo “.h” y pueda ser probada por medio de un menú.

---

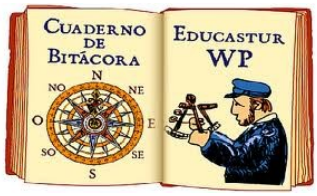
# CONSIGNAS DE DESARROLLO

- Lee atentamente todas las consignas antes de iniciar la tarea.
- Para llevar a cabo la tarea, te sugiero que dividas el proceso en etapas:

## • Etapa 1: Organización

**¡Organízate! Sin orden no podrás alcanzar los objetivos.**
**Recuerda que tienes poco tiempo para entregar tu producción.**

- Lleva un cuaderno de bitácora donde registres todos los datos que te resulten relevantes para el desarrollo de la tarea.




- Crea una carpeta en tu disco, donde puedas almacenar todo el material que consideres necesario para tu tarea.

---

## • Etapa 2: Codificación

1. Crea un proyecto en Code::Blocks, llamado “cadenas.cbp”.
2. Dentro del proyecto creado, abre un nuevo archivo de tipo “.h” y llámalo “cadenas.h”.
3. Desarrolla dentro del archivo “cadenas.h”, las funciones que se lista a continuación, codificándolas en lenguaje C++.
   ***Nota:*** *utiliza siempre funciones estándar para permitir que tu proyecto sea portable.*

3.1 **strLen**
- 3.1.1 **Acción:** cuenta la cantidad de caracteres que integran una cadena.
- 3.1.2 **Parámetros:** cadena de caracteres.
- 3.1.3 **Devuelve:** un entero que representa la cantidad de caracteres con excepción del ‘\0’.

3.2 **strCpy**
- 3.2.1 **Acción:** copia una cadena en otra.
- 3.2.2 **Parámetros:** cadena destino, cadena fuente.
- 3.2.3 **Devuelve:** nada.

3.3 **strCat**

3.3.1 **Acción:** concatena dos cadenas dejando el resultado en la cadena inicial.
3.3.2 **Parámetros:** cadena inicial, cadena final.
3.3.3 **Devuelve:** nada
***Nota:*** *la cadena inicial debe ser tan larga como la suma de la longitud de ambas cadenas.*

3.4 **strCmp**

3.4.1 **Acción:** compara dos cadenas.
3.4.2 **Parámetros:** cadena 1, cadena 2.
3.4.3 **Devuelve:**
Un 0 si son iguales.
Un número positivo si la primera es mayor que la segunda.
Un número negativo si la segunda es mayor que la primera

3.5 **strFind**

3.5.1 **Acción:** busca un carácter dentro de una cadena.
3.5.2 **Parámetros:** cadena, carácter.
3.5.3 **Devuelve:**
La posición donde se produce la primera ocurrencia.
Un –1 si no lo encuentra.

3.6 **strCnt**

3.6.1 **Acción:** cuenta cuántas de veces que aparece un carácter dentro de una cadena.
3.6.2 **Parámetros:** cadena, carácter.
3.6.3 **Devuelve:** la cantidad de veces que aparece el carácter en la cadena.

3.7 **strInv**

3.7.1 **Acción:** invierte los caracteres de una cadena.
3.7.2 **Parámetros:** cadena
3.7.3 **Devuelve:** nada.

3.8 **strRpl**

3.8.1 **Acción:** reemplaza el carácter de una posición dada de una cadena por otro.
3.8.2 **Parámetros:** cadena, nuevo carácter, posición
3.8.3 **Devuelve:** nada.

3.9 **strTrunc**

3.9.1 **Acción:** trunca una cadena en una posición dada.
3.9.2 **Parámetros:** cadena, posición.
3.9.3 **Devuelve:** nada.

3.10 **strSub**

3.10.1 **Acción:** busca una subcadena dentro de una cadena.
3.10.2 **Parámetros: cadena, subcadena.**
3.10.3 **Devuelve:**
La posición inicial de la subcadena.
Un –1 si no se encuentra.

3.11 **strToUpper**

3.11.1 **Acción:** convierte una cadena a mayúsculas.
3.11.2 **Parámetros:** cadena.
3.11.3 **Devuelve:** nada.

3.12 **strToLower**

3.12.1 **Acción:** convierte una cadena a minúsculas.

3.12.2 **Parámetros:** cadena.

3.12.3 **Devuelve:** nada.

3.13 **strToInt**

3.13.1 **Acción:** convierte una cadena de caracteres a un número entero.

3.13.2 **Parámetros:** cadena

3.13.3 **Devuelve:** el número entero.

**Nota:** los elementos de la cadena deben ser caracteres cuyo código ASCII se encuentre en el rango 48..57.

3.14 **intToStr**

3.14.1 **Acción:** convierte un número entero a cadena.

3.14.2 **Parámetros:** número, cadena

3.14.3 **Devuelve:** nada.

4. Crea un nuevo archivo llamado “menu.cpp” dentro del proyecto “cadenas.cbp”, donde se incluya la biblioteca “cadenas.h”
   ***Nota:*** *utiliza la directiva del preprocesador “#include”.*

5. Dentro del archivo “menu.cpp”, desarrolla un menú, donde cada opción muestre un ejemplo de aplicación de una de las funciones incorporadas en la biblioteca “cadenas.h”.

   ***Nota 1:*** *El menú debe ser cíclico, y sólo debe permitirse el abandono del mismo a través de la opción “**salir**”.*

   ***Nota 2:*** *Todas la opciones deben estar validas, y en los casos de ingresos no válidos, deben informarse, pero bajo ningún concepto debe cancelarse el programa.*

---

| LAB 1 | Profesor: Rubén Calabuig |
|---|---|
| Trabajo Práctico Integrador Nº 01 | ESTRUCTURAS FUNCIONES Y LIBRERÍAS |

# CONSIGNAS DE PRESENTACIÓN

La presentación del trabajo práctico debe realizarse cumpliendo con las siguientes premisas:

1. El trabajo debe ser presentado bajo la forma de proyecto integrado. Bajo ningún caso se aceptará la entrega de archivos sueltos.

2. Ambos archivos integrantes codificados del proyecto (cdenas.h y menu.cpp) deben estar debidamente documentados. Con tal fin, se incluye a continuación, los modelos correspondientes a los encabezados de archivos y de funciones respectivamente.

Modelo de encabezado de archivo:

```
////////////////////////////////////////////////////////////////////////////
// ARCHIVO             : nombre del archivo incluyendo la extension
// AUTOR               : nombre del autor
// FECHA DE CREACION   : dd/mm/aaaa.
// ULTIMA ACTUALIZACION: dd/mm/aaaa.
// LICENCIA            : GPL (General Public License) - Version 3.
//==========================================================================
// SISTEMA OPERATIVO   : Linux (Ubuntu) / Windows XP / Windows 7.
// IDE                 : Code::Blocks - 8.02 / 10.05
// COMPILADOR          : GNU GCC Compiler (Linux) / MinGW (Windows).
// LICENCIA            : GPL (General Public License) - Version 3.
//==========================================================================
// DESCRIPCION:
//             aquí va una descripción general de la libreria
//
////////////////////////////////////////////////////////////////////////////
```

Modelo de encabezado de función:

```
//*************************************************************************
//                          DEFINICION DE LAS FUNCIONES
//=========================================================================
// FUNCION   : tipo nombre(lista de parametros)
// ACCION    : explicar brevemente que es lo que hace la funcion y como.
// PARAMETROS: lista de parametros (uno por linea donde se indique: tipo, nombre,
//             que representa, y valores posibles si existieran limitaciones).
// DEVUELVE  : tipo --> explicacion si representa algo.
//-------------------------------------------------------------------------
```

---

# CONSIGNAS DE ENTREGA

La entrega del trabajo práctico debe realizarse teniendo en cuenta a las siguientes consignas:

1. Sólo deben entregarse los archivos tipo “.cbp”, “.cpp” y “.h”
   ***Nota:*** *bajo ningún concepto se corregirán entregas que no cumplan con esta consigna.*

2. Los tres archivos mencionados en el punto anterior, deben comprimirse en un archivo de tipo “.zip”.
   ***Nota:*** *no se aceptarán otros formatos de compresión que no sean “.zip”.*

3. El **nombre del archivo** comprimido debe presentar la siguiente estructura:

   **LAB1_TPI01_<apellidos y nombres del alumno>.zip.**

4. **Medio de entrega:**
   El archivo resultante de esta etapa, deberá ser entregado por cualquier medio electrónico en forma personal.
   ***Nota:*** *no se aceptarán envíos por correo electrónico, ni ninguna otra herramienta de Internet.*

---

# RECURSOS

Para el desarrollo del siguiente práctico, el alumno debe contar con los siguientes recursos:

- Una computadora con sistema operativo MS-Windows (XP o Seven) o GNU/Linux.
- IDE Code::Blocks con compilador MinGw o Gcc.
- Una aplicación compresora de archivos como el 7zip.
- Algún dispositivo de almacenamiento portable (ej.: pendrive) para el transporte de los archivos.

---

# EVALUACIÓN

Para la evaluación se tendrán en cuenta los siguientes puntos:

1. Cumplimiento de:
   1. Consignas de programación (debe respetarse claramente el paradigma PME.
   2. Consignas de presentación.
   3. Consignas de entrega.
   4. Consignas de documentación.
2. El proyecto debe poder compilarse tanto bajo MS-Windows como en GNU/Linux.
3. El proyecto funcione correctamente.

---