Práctica 3
Informática 2
Facultad de Ingeniería
Universidad de Antioquia

1 Introducción

En esta sesión de laboratorio los estudiantes profundizarán en tres temas: i) la manipulación de cadenas de caracteres, ii) la gestión de archivos y iii) el control de excepciones. La gestión de archivos permite que los programas almacenen y recuperen datos de manera persistente, es decir, que la información no se pierda al cerrar el programa. Esta habilidad es esencial para cualquier aplicación que necesite guardar configuraciones, registros de usuario o procesar grandes volúmenes de datos. La práctica se enfocará en las operaciones básicas de lectura y escritura de archivos de texto y binarios. Por otro lado, el manejo de excepciones proporciona un mecanismo para gestionar errores inesperados que pueden ocurrir durante la ejecución de un programa. En lugar de que el programa se cierre abruptamente, es deseable interceptar y manejar estos errores de manera controlada. Esto es crucial para crear aplicaciones robustas y confiables, asegurando que el programa pueda recuperarse de situaciones imprevistas, como la falta de un archivo o una operación que represente una indeterminación. La práctica consta de dos partes: en la primera parte el estudiante implementará 2 métodos de codificación; en la segunda parte usará estos métodos en una aplicación definida.

2 Objetivos

- Aprender a manipular cadenas de caracteres en C++.
- Aprender a gestionar archivos en C++.
- Familiarizarse con el manejo de excepciones.

3 Ejemplos

1. El profesor mostrará cómo crear un archivo y escribir en él.
2. El profesor mostrará cómo leer datos de un archivo carácter a carácter y línea por línea
3. El profesor realizará un ejemplo de cómo convertir caracteres leídos de un archivo a números.

4 Ejercicios

Tenga en cuenta que esta parte de la práctica debe ser realizada de las siguientes dos formas:

- Usando arreglos de `char`.
- Usando la clase `string`.

1. Escriba un programa para codificar archivos de texto. El programa debe recibir un número entero n que será la semilla de codificación y un número para seleccionar el método de codificación. La salida del programa debe ser un archivo codificado. El docente podrá evaluar la funcionalidad del

código probando con ambos métodos y diferentes valores de semilla. Usted debe también garantizar que las posibles excepciones en tiempo de ejecución sean tratadas correctamente sin interrumpir de manera anómala el flujo de su programa.

(a) **Primer método:** Se leen los caracteres del archivo y, utilizando el código ASCII, se pasa a una cadena de bits. Cada carácter corresponde a una palabra de 8 bits según su valor en el código ASCII. La cadena de bits resultante se separa en bloques de n bits. Estos grupos de n bits deben ser modificados según las siguientes reglas:

- En el primer bloque de n bits se intercambian todos los 0 por 1 y todos los 1 por 0 (es decir, se invierte cada bit).
- Los siguientes grupos de bits se cambian según lo que encuentre en el grupo anterior de la siguiente manera:
  - Si hay igual cantidad de 1s y 0s, se invierte cada bit del grupo.
  - Si hay mayor cantidad de 0s, se invierte cada 2 bits.
  - Si hay mayor cantidad de 1s, se invierte cada 3 bits.

Ejemplo: Si se ingresa $n = 4$ y el archivo contiene la cadena `AbCd`.

El archivo en binario sería:
`01000001011000100100001101100100`
El archivo codificado sería:
`10110100001111010001011010011011`

(b) **Segundo método:** En principio debe realizarse lo mismo que en el método anterior. Después de separar los dígitos binarios en grupos de n bits, cada bit dentro de un grupo se desplaza una posición a la izquierda, de manera que el primer bit del grupo codificado corresponde al último bit del grupo sin codificar, el segundo bit codificado corresponde al primero sin codificar y así sucesivamente hasta que el último corresponde al penúltimo sin codificar.
Ejemplo: Si se ingresa 4 y el archivo contiene `AbCd`.

El archivo en binario sería:
`01000001011000100100001101100100`
El archivo codificado sería:
`00101000001100010010100100110010`

El programa debe recibir los nombres del archivo de entrada y el archivo de salida por consola. Para esto puede consultar el uso de los argumentos de entrada de la función `main()`, *argc* y *argv*.

2. Escriba un programa para decodificar el archivo generado por el programa anterior. De la misma manera, debe recibir la semilla, el método de codificación y los nombres de los archivos de entrada y salida por consola.

5 Aplicación

Modelar un sistema de registro de usuarios de un sistema bancario que tenga las siguientes funcionalidades:

1. Acceder al sistema como usuario administrador. La validación de acceso se debe hacer abriendo un archivo `sudo.txt`. Dicho archivo debe estar codificado utilizando uno de los métodos del punto anterior. Una vez se valide el ingreso como administrador, se podrá realizar el registro de usuarios del sistema bancario. Cada usuario debe registrar cédula, clave y saldo (COP).

2. Los usuarios que no son administradores pueden realizar las siguientes operaciones:

   (a) Consultar saldo.

   (b) Retirar dinero especificando la cantidad deseada.

Cada ingreso al sistema por parte de los clientes para retirar dinero o consultar el saldo tiene un costo de 1000 COP. **Todas las transacciones deben registrarse en un archivo de texto y codificarse con uno de los métodos desarrollados en el punto anterior.**