IDENTIFICACIÓN:
DISCIPLINA: SOFTWARE
CURSO: 2º BTI
FECHA: 2-05-2024; 9-05-2024

CONTENIDOS:
Archivos en C++. Operaciones básicas. Ejemplos

CAPACIDADES:
- Aplica la programación estructurada en la solución de problemas utilizando un lenguaje de Programación de alto nivel.

INDICADORES
- Define correctamente el tipo registro.
- Declara correctamente las variables para interactuar con un archivo
- Resuelve correctamente el problema usando archivos.
- Participa activamente en la clase realizando todas las actividades propuestas.

ACTIVIDAD 1
Leer el siguiente texto:
La **librería** *fstream* en C++ es fundamental para trabajar con archivos y flujos de entrada/salida. Explicaremos los conceptos clave y se darán ejemplos para cada uno de los aspectos básicos que debemos tener en cuenta para trabajar con streams o flujos.

1. *fstream*:
    - La clase *fstream* permite la manipulación de archivos en C++. Proporciona una interfaz para abrir, leer, escribir y cerrar archivos.
    - Los objetos de esta clase mantienen un **búfer de flujo interno** (llamado *filebuf*) que realiza operaciones de entrada/salida en el archivo asociado.
    - Puedes asociar un archivo con un objeto *fstream* al construirlo o llamando a la función *open*.
2. **Funciones clave**:
    - *open*: Abre un archivo con un modo específico (lectura, escritura, etc.).
    - *close*: Cierra el archivo asociado al objeto *fstream*.
    - *read*: Lee datos desde el archivo.
    - *write*: Escribe datos en el archivo.
3. **Flags**:
    - *ios::in*: Abre el archivo en modo de lectura.
    - *ios::app*: Abre el archivo en modo de escritura, pero agrega datos al final sin sobrescribir el contenido existente.
    - *ios::binary*: Abre el archivo en modo binario (útil para archivos no textuales).

4. **Ejemplo**:

```
#include <iostream>
#include <fstream>


int main() {
    // Ejemplo de escritura en un archivo
    std::ofstream archivo_salida("mi_archivo.txt", std::ios::out);
    if (archivo_salida.is_open()) {
        archivo_salida << "Hola, mundo!";
        archivo_salida.close();
    }

    // Ejemplo de lectura desde un archivo
    std::ifstream archivo_entrada("mi_archivo.txt", std::ios::in);
    if (archivo_entrada.is_open()) {
        std::string contenido;
        getline(archivo_entrada, contenido);
        std::cout << "Contenido del archivo: " << contenido << std::endl;
        archivo_entrada.close();
    }

    return 0;
}
```

5. En este ejemplo:

   - Abrimos un archivo llamado *"mi_archivo.txt"* en modo de escritura (*ios::out*).
   - Escribimos *"Hola, mundo!"* en el archivo.
   - Luego, abrimos el mismo archivo en modo de lectura (*ios::in*).
   - Leemos la línea completa y mostramos su contenido por pantalla.

ACTIVIDAD 2 EXPLICACIÓN DE LECTURA Y ESCRITURA DE ARCHIVOS CON REGISTROS EN C++ UTILIZANDO fstream

Para leer y escribir una estructura en un archivo binario utilizando *fstream* en C++, sigue estos pasos:

1. **Escribir una estructura en un archivo binario**:

   - Abre un archivo en modo de escritura binaria (*ios::out* | *ios::binary*).

- Escribe la estructura en el archivo utilizando la función *write*.
- Cierra el archivo después de escribir los datos.

Ejemplo:

```cpp
#include <iostream>
#include <fstream>

struct MiEstructura {
    int id;
    double valor;
    char nombre[20];
};

int main() {
    MiEstructura datos;
    datos.id = 1;
    datos.valor = 3.14;
    strcpy(datos.nombre, "Ejemplo");

    std::ofstream archivo_salida("mi_archivo.bin", std::ios::out | std::ios::binary);
    if (archivo_salida.is_open()) {
        archivo_salida.write(reinterpret_cast<char*>(&datos), sizeof(datos));
        archivo_salida.close();
    }

    return 0;
}
```

2. **Leer una estructura desde un archivo binario**:

- Abre un archivo en modo de lectura binaria (*ios::in | ios::binary*).
- Lee la estructura desde el archivo utilizando la función *read*.
- Cierra el archivo después de leer los datos.

Ejemplo:

```cpp
#include <iostream>
#include <fstream>

struct MiEstructura {
    int id;
    double valor;
    char nombre[20];
};

int main() {
    MiEstructura datos;

    std::ifstream archivo_entrada("mi_archivo.bin", std::ios::in | std::ios::binary);
    if (archivo_entrada.is_open()) {
        archivo_entrada.read(reinterpret_cast<char*>(&datos), sizeof(datos));
        archivo_entrada.close();

        std::cout << "ID: " << datos.id << std::endl;
        std::cout << "Valor: " << datos.valor << std::endl;
        std::cout << "Nombre: " << datos.nombre << std::endl;
    }
```

Ejercicios propuestos

1. **Ejercicio de Registro de Empleados**:
   - ⊄ Crea una estructura llamada *Empleado* con campos como nombre, salario y antigüedad.
   - ⊄ Permite al usuario ingresar datos para varios empleados y guárdalos en un archivo binario.
   - ⊄ Luego, recupera la información de los empleados y calcula el promedio de salarios.
2. **Ejercicio de Registro de Estudiantes**:
   - ⊄ Define una estructura *Estudiante* con campos como nombre, edad y calificaciones.
   - ⊄ Permite al usuario ingresar datos para varios estudiantes y almacénalos en un archivo.
   - ⊄ Luego, lee los datos del archivo y calcula el promedio de calificaciones.
3. **Ejercicio de Registro de Ventas**:
   - ⊄ Crea una estructura *Venta* con campos como producto, cantidad y precio.
   - ⊄ Solicita al usuario ingresar datos de ventas y guarda la información en un archivo.
   - ⊄ Después, lee los datos del archivo y calcula el total de ventas y el producto más vendido.
4. **Ejercicio de Registro de Libros**:
   - ⊄ Define una estructura *Libro* con campos como título, autor y año de publicación.
   - ⊄ Permite al usuario ingresar detalles de varios libros y almacénalos en un archivo.
   - ⊄ Luego, recupera los datos y muestra los libros publicados antes de un año específico.
5. **Ejercicio de Registro de Alumnos**:
   - ⊄ Crea una estructura *Alumno* con campos como nombre, edad y notas.
   - ⊄ Pide al usuario que ingrese información sobre varios alumnos y guarda los datos en un archivo.
   - ⊄ Después, lee los datos y encuentra al estudiante con la calificación más alta.