**Registros de Longitud Fija y Variable**

Profesor Heider Sanchez

**P1**: Dada la siguiente estructura del registro, en donde se encuentran dos atributos no textuales: ciclo y costo de mensualidad.

|  |
| --- |
| struct Alumno  {    char codigo [5];  char nombre [11];    char apellidos [20];  char carrera [15];    int ciclo;  float mensualidad;  }; |

Se le pide implementar las siguientes operaciones de **manipulación de archivo binario** con estrategia de eliminación **FreeList** en una clase llamada **FixedRecord**:

1. El constructor reciba el nombre del archivo.
2. Implemente la función ***vector<Alumno> load()*** para leer todos los registros del archivo.
3. Implemente la función ***void add(Alumno record)*** para agregar un nuevo registro al archivo.
4. Implemente la función ***Alumno readRecord(int pos)*** para obtener el registro de la posición “pos”.
5. Implemente la función ***bool delete(int pos)*** para eliminar un registro al archivo utilizando la estrategia **más óptima**.
6. Realice pruebas de cada función en el programa principal.

¿Por qué conviene usar archivo binario?

**P2:** Asuma que tiene un **archivo de texto** de registros de longitud variable. En donde, cada registro contiene cuatro campos de tamaño variable, los campos están delimitados por el caracter **|,** y los registros están delimitados por el salto de línea \**n**. La primera línea del archivo indica los nombres de cada campo por lo tanto ignore esta línea en la lectura.

|  |
| --- |
| Nombre|Apellidos|Carrera|Mensualidad  Howard|Paredes Zegarra|Computacion|1500.00  Penny|Vargas Cordero|Industrial|2550.50  Sheldon|Cooper Quizpe|Mecatronica|1850.00 |

|  |
| --- |
| struct Alumno  {  string Nombre;    string Apellidos;  string Carrera;  float mensualidad;  }; |

Encapsular las siguientes operaciones de manipulación del archivo en una clase llamada **VariableRecord**:

1. El constructor reciba el nombre del archivo.
2. Implemente la función ***vector<Alumno> load()*** para leer todos los registros del archivo.
3. Implemente la función ***void add(Alumno record)*** para agregar un nuevo registro al archivo.
4. Implemente la función ***Alumno readRecord(int pos)*** para obtener el registro de la posición “pos”.

**P3**: Implemente otro programa para leer y escribir registros de longitud variable en un **archivo binario usando el tamaño del dato como separador.** El registro debe tener los campos que se indican en la siguiente figura.

|  |
| --- |
| struct Matricula  {    string codigo;    int ciclo;  float mensualidad;  string observaciones;  }; |

Se le pide implementar la estructura adecuada para acceder directamente a los registros considerando lo siguiente:

1. Maneje un archivo adicional (metadata) para indicar la posición inicial de cada registro. Evalúe si es necesario guardar también el tamaño del registro.
2. Implemente adecuadamente la función ***vector<Matricula> load()*** para leer todos los registros del archivo.
3. Implemente adecuadamente la función ***void add(Matricula record)*** para agregar un nuevo registro al archivo.
4. Implemente la función ***Matricula readRecord(int pos)*** para obtener el registro de la posición “pos”.

**Entregable:**

* **Subir al canvas solo el código fuente de cada solución (p1.cpp, p2.cpp, p3.cpp).**
* **Los cuatro programas deben contener pruebas de funcionalidad en el main (debe correr).**
* **Compile con C++ versión 2017.**