UNIDAD CURRICULAR: ALGORITMICA Y PROGRAMACIÓN

UNIDAD X. ARCHIVOS

CONTENIDO:

UNIDAD X

ARCHIVOS

CONCEPTO

Es una colección de registros relacionados entre sí con aspectos en común y organizados para un propósito específico almacenados en algún dispositivo de almacenamiento externo. **Ejemplo:** archivo de nómina, archivo de inventario, archivo de inscripciones.

ESTRUCTURA Y ACCESO

C++ posee una librería denominada ***fstream*** que contiene una serie de funciones y operadores para trabajar con archivos tanto para entrada como para salida de datos.

Para tener acceso a un archivo hay que abrirlo. El proceso de la apertura se inicia identificando el nombre lógico del archivo (descriptor) que se va a utilizar y el modo de apertura: lectura o escritura. Es de hacer notar que este descriptor permitirá la asociación con el archivo físico que se guardará en el medio de almacenamiento secundario.

Apertura de un archivo para lectura: Si queremos abrir el archivo para leer su contenido se debe indicar el nombre lógico o descriptor del archivo. Para ello se utiliza la siguiente instrucción:

ifstream descriptor_o_nombre_logico_del_archivo;

Ejemplo:

ifstream archivo;

Apertura de un archivo para escritura: Si lo que se quiere es abrir el archivo para grabar datos en el archivo, se utiliza la siguiente instrucción:

ofstream descriptor_o_nombre_logico_del_archivo;

Ejemplo:

ofstream archivo;

TIPOS DE ARCHIVO: DATOS Y DE TEXTO.

C++ soporta dos tipos de archivos: texto y datos o binario.

Los archivos tipo texto almacenan datos como códigos ASCII. Los valores simples, tales como números y caracteres están separados por

espacios en blanco. Este tipo de archivo se utiliza para almacenar datos o crear salidas que se pueden imprimir después.

Los archivos binarios almacenan flujos de bits, sin prestar atención a los códigos ASCII o a la separación de espacios. Son adecuados para almacenar objetos y requiere usar la dirección de una posición de almacenamiento.

MÉTODOS PARA REALIZAR LA GESTIÓN DE ARCHIVOS.

Para la gestión de archivos texto, además de abrir el archivo para entrada o salida se utilizan las funciones miembro ***open y close*** y los operadores **<<** y **>>**.

La función miembro *open:* permite indicar el nombre físico del archivo al que se le asociará al nombre lógico.

Sintaxis:

```
descriptor.open("nombre_Fisico_del_archivo");
```

Ejemplo:

```
archivo.open("datos.txt");
```

Si se desea abrir el archivo para agregar datos al final del archivo se utiliza la siguiente instrucción:

```
descriptor.open("nombre_Fisico_del_archivo",ios::app);
```

Ejemplo:

```
archivo.open("datos.txt",ios::app);
```

La función miembro *close*: permite cerrar el archivo.

Sintaxis:

```
descriptor.close();
```

Ejemplo:

```
archivo.close();
```

El operador >>: se utiliza para la lectura de un archivo texto.

Sintaxis:

```
descriptor >> variable_que_guardará el valor;
```

Ejemplo:

archivo >> dato;

El operador <<: se utiliza para guardar un dato en un archivo texto.

Sintaxis:

descriptor << variable_o dato_a guardar;

Ejemplo:

archivo << dato;

EJERCICIOS RESUELTOS

a) Realice un programa que permita guardar en un archivo texto la cédula, nota definitiva y resultado de tres estudiantes.

```
#include <fstream> // Biblioteca para el manejo de  archivos o ficheros
#include <iostream>

using namespace std;
int main()
{
ofstream archout;
archout.open("notas.txt");
if (!archout)
     { cout << " No se puede crear o abrir el archivo "; }
else
     { // Escritura en el archivo
     archout << "15.432.152" << " " << 15.75 << " APROBADO" << endl;
     archout << "18.547.666" << " " << 8.75 << " PER" << endl;
     archout << "18.965.321" << " " << 5.0 << " REPROBADO " << endl;
     archout.close();
     }
}
```

b) Realice un programa que permita visualizar por pantalla el contenido del archivo creado en el ejercicio anterior.

```
#include <fstream> // Biblioteca para el manejo de ficheros
#include <iostream>
using namespace std;

int main()
```

```
{ int i;
  ifstream archin;

float nota;
char cadena[30];
archin.open("notas.txt");
if (archin.bad())
      { cout << "No se puede abrir el archivo " << endl; }
else
      { archin >> cadena;          // Lectura del primer valor en el archivo
        while (!archin.eof())      // mientras no se llegue al fin del archivo
          { cout << cadena << " ";
            archin >> nota;        // Lectura del siguiente valor en el archivo
            cout << nota << " ";
            archin >> cadena;
            cout << cadena << endl;
            archin >> cadena;
          }
        archin.close();
      }
}
```

c) Realice un programa que permita agregar la cédula, nota y resultado de un estudiante suministrada por teclado, en el mismo archivo notas.txt, sin perder la información almacenada anteriormente.

```
#include <fstream>
#include <iostream>
using namespace std;
int main()
{ char cedula[12], resultado[9];
  float nota;
  ofstream archout;
  archout.open("notas.txt",ios::app);  //apertura para agregar al final del archivo
  if (!archout)
       { cout << " No se puede  abrir el archivo"; }
  else
       { cout<<"Ingrese la cédula: ";
         cin>>cedula;
         cout<<"Ingrese la nota: ";
         cin>>nota;
         cout<<"Ingrese el resultado (APROBADO, REPROBADO, PER): ";
         cin>>resultado;
         archout << cedula << " " << nota << " "<<resultado << endl;
         archout.close();
       }
}
```

d) Modifique el programa anterior de manera que el resultado no se lea por teclado y se guarde de acuerdo al siguiente criterio. Si la nota es menor a 6.5 el resultado es REPROBADO, si la nota es mayor a 6.5 pero menor a 12 el resultado es PER. Si la nota es mayor o igual a 12, el resultado es APROBADO.

```
#include <fstream>
#include <iostream>
using namespace std;
int main()
{ char cedula[12];
  float nota;
  ofstream archout;
  archout.open("notas.txt",ios::app);  //apertura para agregar al final del archivo
  if (!archout)
     { cout << " No se puede  abrir el archivo"; }
  else
     { cout<<"Ingrese la cédula: ";
       cin>>cedula;
       cout<<"Ingrese la nota: ";
       cin>>nota;
       archout << cedula << " " << nota << " ";
       if (nota<6.5)
          archout << "REPROBADO"<<endl;
       else
          if (nota<12)
             archout << "PER"<<endl;
          else
            archout << "APROBADO"<<endl;
       archout.close();
     }
}
```

REFERENCIAS BIBLIOGRÁFICAS

Bassard, G y Bratley, P. (2010). Fundamentos de algoritmia. Prentice-Hall.

Joyanes, L. (2008). Fundamentos de programación. Algoritmos , Estructuras de datos y objetos. Mc Graw Hill. Tercera edición.

Joyanes, L. y Zahonero, I. (2005). Programación en C. Metodología, algoritmos y Estructura de datos. Mc Graw Hill. Segunda Edición

Martí, N. y Ortega, Y. (2004). Estructuras de datos y Métodos Algorítmicos. Ejercicios Resueltos. Pearson Education.