Programación Avanzada

Pontificia Universidad
JAVERIANA
Colombia

Ing. Orlando Vasquez

Manejo de archivos

Los archivos en C++ son una de las soluciones mas sencillas para persistir la información resultado de la ejecución de un programa.

Por ende un archivo de datos es una colección de información que se almacena como soporte de la ejecución y en la mayoría de los casos utiliza una misma estructura en todas sus líneas.

Funciones FSTREAM

- **fstream** Clase de flujo de C++ de entrada / salida para manejo de archivos

Los objetos de esta clase mantienen un objeto filebuf como su buffer de flujo interno, que realiza operaciones de entrada / salida en el archivo con el que están asociados.

- **ifstream** Clase de flujo de entrada en C++para operar en archivos.
- **ofstream** Clase de flujo de salida en C++ para operar en archivos

Funciones FSTREAM

- Función **open**

Abre el archivo identificado por nombre de archivo de argumento, asociándolo con el objeto de flujo, de modo que las operaciones de entrada / salida se realicen en su contenido. El modo de argumento especifica el modo de apertura.

Sintaxis

void open (const char* filename,ios_base::openmode mode = ios_base::in | ios_base::out);

Modos de apertura:
ios::in abre el archivo para lectura
ios::out abre el archivo para escritura
ios::app abre el archivo para añadir datos, se ubica al final del archivo antes de cada operación de salida
ios::binary // El archivo se abre en modo binario
ios::ate// Se ubica al final del archivo en el momento de la apertura del archivo

Ejemplo:
```
ifstream input_file;
input_file.open ("estudiantes1.dat", ios::binary|ios::app|ios::ate);
```

Funciones FSTREAM

- Función **is_open**

Función booleana asociada a un archivo que permite comprobar si un archivo está abierto.

Sintaxis

bool is_open();

Ejemplo:

```
ifstream input_file("estudiantes1.dat", ios::binary);
If (input_file.is_open()){.....
```

- Función **eof()**

Función booleana que devuelve veradero si el indicador de estado de error eofbit está establecido para el archivo

Sintaxis

bool eof();

Ejemplo:

```
ifstream input_file("estudiantes1.dat", ios::binary);
while (!input_file.eof()){.....
```

Tener en cuenta validar si la lectura del registro es **true**

if (input_file.read((char*)&maestra, sizeof(maestra))) {.....Existe el registro

Funciones FSTREAM

- Función **close:** Cierra el archivo asociado al objeto **fstream**. Cualquier secuencia de salida pendiente se escribe en el archivo.

Sintaxis

void close();

Ejemplos

```
#include <fstream>
using namespace std;
int main () {
  fstream fs;
  fs.open ("test.txt", ios::in | ios::out | ios::app);
  fs << " hola mundo";
  fs.close();
  return 0;
}
```

Pontificia Universidad JAVERIANA Colombia

Funciones FSTREAM

- Funciones **seekg** **y** **seekp**

Las funciones seekg y seekp mueven la posición del cursor del archivo a la posición relativa del archivo indicada por pos, donde pos es un entero. La función **seekg** se usa si el archivo es de lectura (g=get), **seekp** se usa si el archivo es de escritura (p=put)

Sintaxis

istream& seekg (streampos pos);
istream& seekg (streamoff off, ios_base::seekdir way);

ios_base::seekdir way: .ios::cur=Posicion actual, ios::beg=inicio, ios::end=final

streampos: Posicion dentro del archivo
Streamoff: Numero de posiciones a mover desde la posicion relativa indicada por ios_base

- Funciones **tellg** **y** **tellp**

Las funciones tellg() (entrada) y tellp() (salida) devuelven la posición relativa actual del cursor asociado al fichero de entrada y devuelve un –1 en caso de existir algún error. La función **tellg** se usa si el archivo es de lectura (g=get), **tellp** se usa si el archivo es de escritura (p=put)

Sintaxis

streampos tellp();

streampos tellg();

Funciones FSTREAM

- Función **write:** Inserta los primeros n caracteres de la matriz señalados por s en la secuencia. Esta función simplemente copia un bloque de datos, sin verificar su contenido: La matriz puede contener caracteres nulos, que también se copian sin detener el proceso de copia.

Sintaxis

ostream& write (cont char* s, streamsize n);

- Función **read:** Extrae n caracteres de la secuencia y los almacena en la matriz señalada por s. Esta función simplemente copia un bloque de datos, sin verificar su contenido ni agregar un carácter nulo al final.

Sintaxis:

istream& read (char* s, streamsize n);

```
#include <fstream>
using namespace std;
int main () {
  ifstream infile ("test.txt",ios::binary);
  ofstream outfile ("nuevo.txt",ios::binary);

  // Obtiene tamaño del archivo
  infile.seekg (0,infile.end);
  /*Se ubica en el final movimiendose cero posiciones*/
  long size = infile.tellg();
  infile.seekg (0); /*Esto es igual a infile.seekg(0,ios::beg);

  char* buffer = new char[size];

  // Lee el contenido del archivo
  infile.read (buffer,size);

  // Escribe en el archivo de salida
  outfile.write (buffer,size);
  outfile.close();
  infile.close();
  return 0;
}
```

Funciones FSTREAM

❑ Escribir en un archivo de texto

Para escribir en un archivo de texto, podemos utilizar una sintaxis similar a la del **cout**

Ejemplo:

```
#include <fstream>
using namespace std;
int main()
{
        ofstream archivo("archivos1.txt“, ios::out);
        archivo << "Pedro Perez, 454545, 120.5" << endl;
        archivo << "Jaime Suarez, 35981, 2000.4" << endl;
        archivo << "Maria Gomez, 58478, 354.0" << endl;
        archivo.close();
}
```

Escritura de Texto

Lectura de Texto

Copiar un archivo en otro

❑ Leer línea a línea un archivo de texto

```
#include <fstream>
#include <iostream>
using namespace std;
int main() {
  fstream archivo;
  string line;
  archivo.open("archivos1.txt", ios::in|ios::out);
  if (!archivo){
     cout<<" Error apertura archivo\n";
  }
  else{
      while ( getline (archivo,line))
      {
           cout << line << '\n';
       }
    archivo.close();
}
```

Pontificia Universidad JAVERIANA Colombia

Funciones FSTREAM

Lectura y escritura de estructuras en un archivo

Escritura:

- **ofstream** variable_file("nombreArchivo.xxx", ios::binary|ios::app);
- variable_file.write((char*)**&variableTipoStruct, sizeof(variableTipoStruct));**
- variable_file**.close();**

Pontificia Universidad JAVERIANA Colombia

Funciones FSTREAM

Lectura y escritura de estructuras en un archivo binario

Lectura:

```
ifstream variable_file("nombreArchivo.dat", ios::binary);
variable_file.seekg(0, variable_file.end);     Ubicarse al final del archivo
variableTamano= variable_file.tellg();//Obtener el tamaño en bytes
variable_file.seekg(0, variable_file.beg);//Forzar la ubicacion en el primer byte
tipoStruct variableTipoStruct;
while (pos< variableTamano  ){
        variable_file.read((char*)& variableTipoStruct, sizeof(variableTipoStruct));
        cout << variableTipoStruct.atributo ;
        pos= variable_file.tellg();//Guarda en pos la posicion del ultimo byte leido
}
variable_file.close();
```

Codigo en DevC++

LecturaEscrituraBinarios.cpp

Pontificia Universidad JAVERIANA Colombia

Otras funciones de Alto nivel para manejo de archivos

Las funciones y tipos de datos básicos para el manejo de archivo son:

- Tipo de dato **FILE:** Es un tipo de dato que apunta a una estructura en la cual se guarda la información del archivo. Este tipo de datos se maneja como un apuntador.

Sintaxis: FILE *ptr_fich

Ejemplo: FILE *pfile =NULL;

- Función **fopen:** Se utiliza para abrir un archivo, ya sea para entrada, salida o entrada/salida

Sintaxis: FILE *fopen(const char *filename, const char *mode);

filename : ruta física y nombre del archivo

mode : Modo de apertura del archivo

r: Modo lectura

w: Modo escritura (sobreescribe el archivo)

rw: Archivo de lectura y escritura

a: Modo escritura. Si el archivo existe, se adiciona la información al final

a+: Modo lectura y escritura. Si el archivo existe, se adiciona la información al final

Otras funciones de Alto nivel para manejo de archivos

- Función **fclose :** Se utiliza para cerrar un archivo.

Sintaxis: fclose(FILE *stream);

Ej: fclose(pfile);

- Función **ferror :** Se utiliza para determinar si hay algún error en la variable del archivo (si es diferente de cero)

Sintaxis: ferror (FILE *stream);

Ej: ferror(pfile);

- Función **feof :** Comprueba si el indicador de fin de archivo tiene un valor de verdadero o falso

Sintaxis: feof (FILE *stream);

Ej: feof(pfile);

Otras funciones de Alto nivel para manejo de archivos

- Función **fprintf :** Permite escribir una cadena de caracteres en un archivo de texto abierto previamente

Sintaxis:

fprintf (FILE *ptr, const char *format [arg,] ,var list);

Ej: fprintf(pfile,"%s %s\n","1","pedro");

- Función **fscanf :** Permite leer información de cadena de caracteres escrita en un archivo de texto. La función fscanf almacena en una variable diferente cada vez que encuentra un espacio en una cadena

Sintaxis:

fscanf (FILE *ptr,const char *format [arg,] ,var list);

Ej: fscanf (pfile, "%s %s", id, cadena);

Ejemplos

```
#include <stdio.h>
int main (){
        FILE *pfile ;
        char cadena [100];
        char id[2];
        char titulo1[20];
        char titulo2[20];

        pfile= fopen("archivo1.txt","wt");
        printf ("Escribiendo archivo ...\n");
        fprintf(pfile,"Registro Estudiante\n");
        printf ("Escribiendo registro: 1 Pedro\n");
        fprintf(pfile,"%s %s\n","1","pedro");
        printf ("Escribiendo registro: 2 Juan\n");
        fprintf(pfile,"%s %s\n","2","juan");
        fclose(pfile);

        printf ("\n\nAbriendo archivo ...\n");
        pfile= fopen("archivo1.txt","r");
        fscanf (pfile, "%s %s", titulo1, titulo2);
        printf("%s %s\n\n",titulo1,titulo2);
        fscanf (pfile, "%s %s", id, cadena);
        while (!ferror(pfile)&& !feof(pfile)){
           printf("%s %s\n\n",id,cadena);
            fscanf (pfile, "%s %s", id, cadena);
        }
        fclose(pfile);
        return 0;
}
```

```
Escribiendo archivo ...
Escribiendo registro: 1 Pedro
Escribiendo registro: 2 Juan


Abriendo archivo ...
Registro Estudiante

1 pedro

2 juan
```

Ejemplos

```c
#include <stdio.h>
#include <string.h>
int main (){
   typedef struct {
        int registro;
        char nombre[20];
   } est;
   est estudiante;
   FILE *pfile =NULL;
   printf ("Abriendo archivo archivo1.bin....\n");
   pfile= fopen("archivo1.bin","wb");
   estudiante.registro=1;
   strcpy (estudiante.nombre,"juan gomez");
   printf ("Escribiendo registro 1.... %d %s\n",estudiante.registro,estudiante.nombre);
   fwrite(&estudiante,sizeof(est),1,pfile);
   estudiante.registro=2;
   strcpy (estudiante.nombre,"pedro araujo");
   printf ("Escribiendo registro 2.... %d %s\n",estudiante.registro ,estudiante.nombre);
   fwrite(&estudiante,sizeof(est),1,pfile);
   printf ("Cerrando archivo....\n");
   fclose(pfile);
   printf ("Abriendo archivo para lectura....\n");
   pfile= fopen("archivo1.bin","r");
   fread (&estudiante,sizeof(est),1,pfile);
   printf("\n\nLeyendo registro de estudiantes\n\n");
   while (!ferror(pfile)&& !feof(pfile)){
         printf ("Registro:%2d : Estudiante : %-20s\n",estudiante.registro,estudiante.nombre);
         fread (&estudiante,sizeof(est),1,pfile);
  }
  fclose(pfile);
  return 0;
}
```

```
Abriendo archivo archivo1.bin....
Escribiendo registro 1.... 1 juan gomez
Escribiendo registro 2.... 2 pedro arauj
Cerrando archivo....
Abriendo archivo para lectura....


Leyendo registro de estudiantes

Registro: 1 : Estudiante : juan gomez
Registro: 2 : Estudiante : pedro araujo
```