CURSO: Análisis y Diseño Orientado a Objetos

Fundamentos de Programación

Instructores:
MC. Gerardo Gálvez Gámez
Mayo 2023

Entrada y Salida con archivos

Manejo de archivos (ANSI)

- C cuenta con cuenta con funciones para trabajar con archivos: básicamente leer y escribir datos de diferentes formas.
- Antes de trabajar con un archivo, hay que abrirlo.
- Cuando se abre un archivo, se devuelve un puntero a una estructura de tipo **FILE** (definido en **stdio.h**).
- Esta estructura, llamada descriptor de archivo, servirá para manipular posteriormente el archivos
- Tras su uso, hay que cerrar el archivo.

Manejo de archivos (ANSI)

```
FILE* fd; //variable descriptor de archivo

// abre el archivo en modo lectura

fd = fopen ( “datos.txt", "rt" );

//... trabajar con el archivo ...

fclose (fd); /* cierra el archivo */
```

Abrir un archivo

FILE* fopen(char nombreArchivo[], char modo[]);

- El valor de retorno es un puntero a una estructura FILE.

- **nombreArchivo**: una cadena que contiene un nombre de archivo válido, esto depende del sistema operativo que estemos usando. El nombre puede incluir la ruta completo del archivo.

Modos de abrir un archivo

- El **modo** especifica en tipo de archivo que se abrirá o se creará y el tipo de datos que puede contener, de texto o binarios:

- **r**: sólo lectura. El archivo debe existir.

- **w**: se abre para escritura, se crea un archivo nuevo o se sobrescribe si ya existe.

- **a**: añadir, se abre para escritura, el cursor se sitúa al final del archivo. Si el archivo no existe, se crea.

- **r+**: lectura y escritura. El archivo debe existir.

Modos de abrir un archivo

- **w+**: lectura y escritura, se crea un archivo nuevo o se sobrescribe si ya existe.
- **a+**: añadir, lectura y escritura, el cursor se sitúa al final del archivo. Si el archivo no existe, se crea.
- **t**: tipo texto, si no se especifica "t" ni "b", se asume por defecto que es "t"
- **b**: tipo binario.

Cerrar archivo

int fclose(FILE *archivo);

- Un valor de retorno cero indica que el archivo ha sido correctamente cerrado, si ha habido algún error, el valor de retorno es la constante **EOF**.

- El parámetro es un puntero a la estructura **FILE** del archivo que queremos cerrar.

Leer una carácter desde un archivo

int fgetc(FILE *archivo);

- El valor de retorno es el carácter leído. Si no hay ningún carácter disponible, el valor de retorno es EOF.

- El parámetro es un puntero a una estructura FILE del archivo del que se hará la lectura.

Leer una carácter desde un archivo- Ejemplo

```
// Lee una archivo de texto y lo muestra en la pantalla
#include <stdio.h>
void main( ){
 char *RutaArchivo="NombreArchivo.txt";
 FILE *da;
 char car;
     da=fopen(RutaArchivo, "rt");
     if (da!=NULL){ // verifica si se abrió el archivo
         car = fgetc(da);
         while(car!=EOF){
                 printf("%c", car);
                 car=fgetc(da);
          }
         fclose(da);
     }
     else
        printf("Error al abrir archivo\n");
}
```

Escribe una carácter en un archivo

int fputc(int caracter, FILE *archivo);

- El valor de retorno es el carácter escrito, si la operación fue completada con éxito, en caso contrario será EOF.

- Los parámetros de entrada son el carácter a escribir, y un puntero a una estructura FILE del archivo en el que se hará la escritura.

Escribe una carácter en un archivo - Ejemplo

```
#include <stdio.h>

void main(){
    FILE *da_origen, *da_destino;
    char *RutaOrigen=”Main.txt";
    char *RutaDestino="Escritura.txt";
    char car;
    da_origen=fopen(RutaOrigen, "rt"); //Abre archivo origen
    da_destino=fopen(RutaDestino, "wt");//Abre archivo destino
   // verifica si se abrieron el archivo
   if (da_origen!=NULL && da_destino!=NULL){
      car=fgetc(da_origen);
      while(car !=EOF){
            fputc(car, da_destino);
            car=fgetc(da_origen);
        }
       fclose(da_origen); fclose(da_destino);
    }
    else
      printf("Error al abrir archivos\n");
}
```

Fin de archivo

int feof(FILE *archivo);

- El valor de retorno es distinto de cero sólo si se ha alcanzado el fin de archivo.
- parámetro es un puntero a la estructura **FILE** del archivo que queremos verificar.

Fin de archivo - Ejemplo

```
#include <stdio.h>

void main(){
     FILE *da_origen, *da_destino;
     char *RutaOrigen=”Main.txt";
     char *RutaDestino="Escritura.txt";
     char car;
     da_origen=fopen(RutaOrigen, "rt"); //Abre archivo origen
     da_destino=fopen(RutaDestino, "wt");//Abre archivo destino
    // verifica si se abrieron el archivo
    if (da_origen!=NULL && da_destino!=NULL){
        car=fgetc(da_origen);
        while(!feof(da_origen)){
                fputc(car, da_destino);
                car=fgetc(da_origen);
          }
          fclose(da_origen); fclose(da_destino);
      }
     else
        printf("Error al abrir archivos\n");
}
```

Leer una cadena desde un archivo

char *fgets(char *cadena, int n, FILE *archivo);

- Esta función está diseñada para leer cadenas de caracteres. Leerá hasta n-1 caracteres o hasta que lea un retorno de línea. En este último caso, el carácter de retorno de línea también es leído.

- El parámetro n nos permite limitar la lectura para evitar desbordar el espacio disponible en la cadena.

- El valor de retorno es un puntero a la cadena leída, si se leyó con éxito, y es NULL si se detecta el final del archivo o si hay un error

Leer una cadena desde un archivo - Ejemplo

```
#include <stdio.h>

int main()
{
    FILE *da;
char *RutaOrigen="main.c";
 char cadena[80];
     da=fopen(RutaOrigen,"rt"); // Abre archivo
      if (da!=NULL){ // verifica si se abrió el archivo
         fgets(cadena,80,da); // lee cadena de archivo
          while(!feof(da)){ // verifica fin de archivo
              printf("%s\n", cadena);
              fgets(cadena,80,da); // lee de nuevo cadena
          }
          fclose(da);
     }
     else
        printf("Error al abrir archivo\n");

    return 0;
}
```

Escribe una cadena en un archivo

int fputs(char *cadena, FILE *archivo);

- Escribe una cadena en un archivo. No se añade el carácter de retorno de línea ni el carácter nulo final.
- El valor de retorno es un número no negativo o EOF en caso de error.
- Los parámetros de entrada son la cadena a escribir y un puntero a la estructura FILE del archivo donde se realizará la escritura.

Escribe una cadena en un archivo - Ejemplo

```
#include <stdio.h>

int main()
{
    FILE *da_origen, *da_destino;
    char cadena[80];
    char *RutaOrigen="Main.txt";
    char *RutaDestino="Escritura.txt";
     da_origen=fopen(RutaOrigen, "rt"); // Abre archivos
     da_destino=fopen(RutaDestino, "wt");
     if (da_origen!=NULL && da_destino!=NULL){ // verifica si se abrieron
          fgets(cadena,80,da_origen);
          while(!feof(da_origen)){
                fputs(cadena,da_destino); // lee del primer archivo
                fgets(cadena,80,da_origen); // escribe en el segundo archivo
          }
      }
     else
        printf("Error al abrir archivo\n");
   fclose(da_origen);
   fclose(da_destino);
   return 0;
}
```

Actividades

ExtraClases

Objetivo:

El alumno demostrara la habilidad alcanzada en clases, para codificar algoritmos de diversos problemas, que utilizan procedimientos de solución que requieran el manejo de archivos.

Ejercicio

1. Desarrolle un programa que cuente de un archivo de texto cuantas vocales, consonantes y dígitos contiene.

1. Escriba un programa que convierta el contenido de archivo a mayúsculas o minúsculas a petición del usuario, la conversión debe guardarse en un segundo archivo.

Escritura con formato

int fprintf(FILE *archivo, const char *formato, argumento, ...);

La función **fprintf** funciona igual que **printf** en cuanto a parámetros, pero la salida se dirige a un archivo en lugar de a la pantalla.

Escritura con formato - Ejemplo

```
#include <stdio.h>

void main(){

   FILE *da;
   char *Ruta=”Archivo.txt";
   da = fopen(Ruta, "a+t");
   char nombre[20];
   int edad;
   float sueldo;

   printf("Nombre:");scanf("%s", nombre);
   printf("Edad:");scanf("%d", &edad);
   printf("Sueldo:");scanf("%f", &sueldo);

   fprintf(da, "%s %d %8.3f \n", nombre, edad, sueldo);

fclose(da);
}
```

Lectura con formato

int fscanf(FILE *archivo, const char *formato, argumento, ...);

La función **fscanf** funciona igual que **scanf** en cuanto a parámetros, pero la entrada se toma de un archivo en lugar del teclado.

Lectura con formato - Ejemplo

```
// Leer información de una archivo de texto con fscanf
#include <stdio.h>

void  main(){
  FILE *da;
 char *Ruta=”Archivo.txt";
 da= fopen(Ruta, "rt");
  char nombre[20];
  int edad;
  float sueldo, total=0;

  fscanf(da, "%s %d %f", nombre, &edad, &sueldo);
   while(!feof(da)){
       printf("%s %d %5.3f \n", nombre, edad, sueldo);
       total+=sueldo;
      fscanf(da, "%s %d %f", nombre, &edad, &sueldo);
}
 printf("---------\n Total ---> %f\n", total );
 fclose(da);
}
```

Lectura con formato - Ejemplo

```
// Leer información de una archivo de texto con fscanf y cuenta palabras buscadas

#include <stdio.h>
#include <string.h>
void  main(int argc, char *argv[]){

FILE *da;
char *RutaOrigen="Main.txt";
da= fopen(RutaOrigen, "rt");
char cadena[80];
int contador=0;
fscanf(da,"%s", cadena);
while(!feof(da)){
  if (strcmp(cadena,argv[2])==0)
       contador++;
   fscanf(da,"%s", cadena);

}
 printf("Total de palabras encontradas : %d\n",contador );
 fclose(da);
}
```

Actividades

ExtraClases

Objetivo:

El alumno demostrara la habilidad alcanzada en clases, para codificar algoritmos de diversos problemas, que utilizan procedimientos de solución que requieran el manejo de archivos.

Preguntas?