ICF 122 - Fundamentos de Programación
Ingeniería Civil Informática
Unidad 9: Archivos

Facultad de Ingeniería

Universidad Andrés Bello

Archivos

Introducción

- Hasta el momento toda la información (datos) que hemos sido capaces de gestionar, la hemos tomado de dos únicas fuentes:
  - datos del programa, o
  - datos que ingresa el usuario desde el teclado.
- Siempre que un programa ha obtenido un resultado, lo único que hemos hecho ha sido mostrarlo en pantalla.
- Sería muy conveniente poder almacenar en algún soporte informático esa información, y poder luego acceder a ese disco para volver a tomarla, para actualizar la información almacenada, para añadir o para eliminar todo o parte de ella.

Archivos

Introducción

- Un **archivo** o **fichero** es un tipo de dato con persistencia, es decir, su tiempo de vida no está ligado al de la ejecución del programa que lo crea o lo maneja.
- Un **archivo** existe desde que un programa lo crea y mientras que no sea destruido por este u otro programa.
- Dos formas de organización:
  - **Secuencial**: los registros del archivo se encuentran en un orden secuencial, por lo tanto, deben ser leídos de forma consecutiva.
  - **Indezado**: se dispone de un índice para obtener la ubicación de cada registro. Eso permite localizar cualquier registro del archivo sin tener que leer todos los que le preceden.

Archivos

Declaración

- La bilioteca `stdio.h` nos permite trabajar con archivos, las funciones, tipos de dato predefinidos y constantes.
- Para abrir un archivo, se indica mediante el nombre del archivo que se quiere acceder, y la función de apertura retorna una dirección de memoria que se utilizará en el programa.
- Esta dirección de memoria se denomina **puntero de archivo**.

```
FILE *ptrArchivo;
```

- La función **fopen** abre un archivo y retorna un puntero asociado al mismo.

```
FILE *fopen(const char *nombre, const char *modo);
```

donde **nombre** es el nombre del archivo, el cual debe ir en comillas dobles y **modo** es el modo de acceso:

- "r": abre un archivo para lectura. El archivo debe existir.
- "w": abre un archivo para escritura. Si existe ese archivo, lo borra y lo crea de nuevo. Los datos nuevos se escriben desde el principio.
- "a": abre un archivo de escritura y los datos nuevos se añaden al final. Si el archivo no existe lo crea.

```
FILE *ptrFile;
ptrFile = fopen("datos.txt", "w");
```

El archivo `datos.txt` ha sido abierto para escritura. Si ese archivo ya existe, se elimina y se crea otro nuevo y vacío.
Si la función no logra abrir el archivo, entonces retorna un puntero nulo.

```
FILE *ptrFile;
ptrFile = fopen ("datos.txt", "w");
if (ptrFile == NULL)
  printf("No se puede abrir el archivo \n");
```

Archivos

fclose

- La función **fclose** cierra el archivo que ha sido abierto mediante `fopen`.

```
int fclose(FILE *ptrArchivo);
```

- La función retorna valor cero si ha cerrado el archivo correctamente.

```
FILE *ptrFile;
ptrFile = fopen("datos.txt", "w");

/*
instrucciones
*/

fclose(ptrFile)
```

- La función **fputc** permite escribir un caracter en un archivo

```
int fputc(int c, FILE *ptrArchivo);
```

La función retorna el caracter escrito, en caso de éxito, sino retorna **EOF**, **E**nd **O**f **F**ile, en caso de error.

```
FILE *ptrFile;
ptrFile = fopen("datos.txt", "w");
fputc("a\n",ptrFile);
fputc("b\n",ptrFile);
fputc("c\n",ptrFile);
fclose(ptrFile);
```

- La función **fputs** permite escribir una cadena de caracteres en un archivo

```
int fputs(const char *s, FILE *ptrArchivo);
```

La función retorna un entero no negativo, en caso de éxito, sino retorna **EOF**, **E**nd **O**f **F**ile, en caso de error.
Otra opción es utilizar la función **fprintf**

```
int fprintf(FILE *ptrArchivo, const char *s);
```

```
FILE *ptrFile;
ptrFile = fopen("datos.txt", "w");
fputs(ptrFile, "1. Primera linea\n");
fputs(ptrFile, "2. Segunda linea\n");
fprintf("3. Tercera linea\n", ptrFile);
fclose(ptrFile);
```

- La función **fgetc** permite leer un caracter de un archivo

```
int fgetc(FILE *ptrArchivo);
```

La función retorna el caracter leído, en caso de éxito, sino retorna **EOF**, **E**nd **O**f **F**ile, en caso de error.

- La función `fgets` permite leer una cadena de carateres desde un archivo. Esta cadena la almacena en un buffer, colocando al final el caracter nulo.

```
char *fgets(char *buf, int n, FILE *ptrArchivo);
```

Si la función encuentra el salto de línea `\n`, antes de leer los `n` caracteres del buffer, entonces termina de leer y retorna solamente los caracteres que ha leído hasta ese momento.

- La función **fscanf** permite leer una cadena de carateres desde un archivo.

```
int *fscanf(FILE *ptrArchivo, const char *s);
```

Si la función encuentra un espacio termina de leer y retorna solamente los caracteres que ha leído hasta ese momento.
Esta función es muy útil cuando existen varios elementos separados por espacio en cada línea del archivo.

Suponga el archivo datos.txt

```
Valor 0000 --> 9.4667
Valor 0001 --> 30.4444
Valor 0002 --> 12.5821
Valor 0003 --> 0.2063
Valor 0004 --> 16.4545
```

```
unsigned int i = 0;
float val[5];
char c[100];
FILE *ptrFile;

ptrFile = fopen("datos.txt","r");
while(fscanf(ptrFile, "%s %s %s %f", c, c, c, val + i) != EOF)
  i++;

fclose(ptrFile);
```