LICENCIATURA EN INFORMATICA

PROGRAMADOR UNIVERSITARIO

UNIVERSIDAD NACIONAL DE TUCUMÁN
facet
FACULTAD DE CIENCIAS EXACTAS Y TECNOLOGÍA

PROGRAMACIÓN

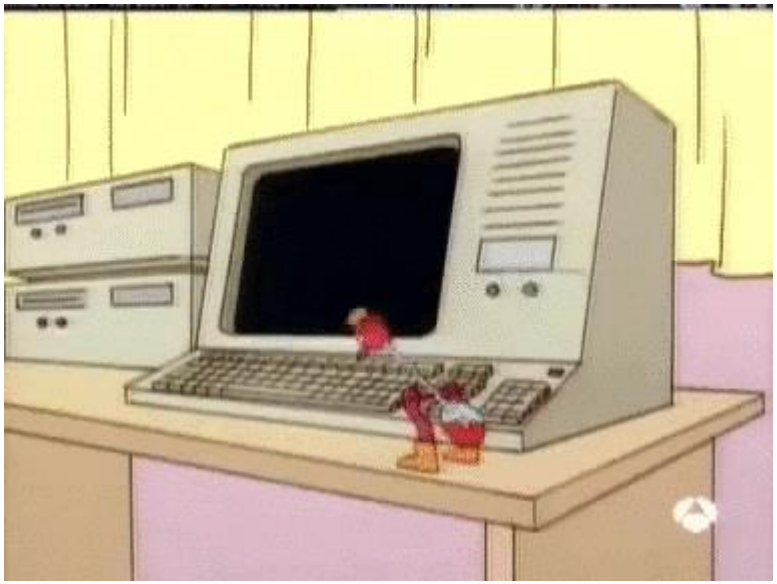
Unidad 8: Archivos en Disco.

Lic. Mariela A. Velázquez

Comenzamos el último Tema de la Materia

Introducción

La manera de capturar la información y mostrar los resultados producidos por los programas usada hasta ahora fue a través de la entrada y salida estándar (teclado y monitor)



Introducción

Los programas obtenían sus datos por medio de las funciones **scanf()** y **gets().**

Sin embargo, con los grandes volúmenes de datos que deben procesar la mayoría de las aplicaciones de uso habitual, se necesita una manera mejor de almacenar dichos datos. La solución más adecuada la constituyen los archivos.

¿Qué es un archivo?

- Un archivo es un flujo o secuencia de bytes
- La información almacenada se reconoce como registros.
- La forma en que está agrupada la información en cada registro depende del programador .
- Cuando un archivo se abre se establecen canales de comunicación (flujos) entre el programa y el archivo.

- Los programas que acceden a los archivos pueden:
    - Crear archivos (Apertura)
    - Modificar archivos
    - Borrar archivos
    - Cerrar archivos (Cierre)

Archivos en C

- Un archivo es una variable de tipo FILE, un puntero de archivo.
- File es un tipo predefinido, estructurado definido en <stdio.h>.
- La estructura FILE tiene información estadística del archivo.
- Declaración del archivo:
  - FILE *nom-archivo; // nombre interno del archivo
- Existen dos tipos de archivos: Archivos de acceso secuenciales y archivos de acceso aleatorios

De la organización física de los archivos

Archivos Secuenciales de datos

En contra:

- El acceso a registros individuales es lento.
- Las operaciones de inserción y eliminación de registros solo pueden hacerse al final del archivo.

Ventajas:

- Es más sencilla de manejar.
- Es un método más rápido.
- No deja espacios entre registros y registros, por lo que se optimiza el uso del espacio en la memoria secundaria.

Del acceso a archivos

El tipo de acceso elegido indica el procedimiento considerado para acceder a un registro de información y efectuar alguna transacción con el.

Acceso relativo	Acceso secuencial
• Permite acceder al registro buscando en forma veloz y directa. • Las operaciones de lectura y escritura se pueden hacer en cualquier parte del archivo.	• Es más lento. • Se adapta perfectamente al proceso de tratamiento de secuencias con marca final.

Tipos de archivos

- Archivos de texto (text), ASCII compatible. Se producen traducciones internas de los caracteres con la posible pérdidas de información. **Para guardar información editable.**

- Archivos binarios (bin). La secuencia de bytes que lo conforman no está procesada. Los datos están en formato interno, no hay traducción alguna. **Para guardar grandes volúmenes de información.**

Pasos para Trabajar con un archivo

1. Declarar la variable archivo con el nombre interno elegido.

   FILE *archi; //archi1 es el nombre interno, en el programa.

2. Hacer la apertura del archivo, indicando nombre y modo de trabajo.

   Se usará la función de la biblioteca stdio.h:

   FILE *fopen(char *nombre, char *modo); //la función retorna un puntero al sector de memoria del archivo.

   char *nombre: cadena de caracteres, es el nombre del archivo y la ruta de acceso.

   char *modo: cadena de caracteres que indica el modo de uso del archivo.

Modo de uso de un archivo

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

A los modos de apertura se le agrega el tipo de archivo a trabajar: binario o de texto (default).

r: lectura desde el comienzo del archivo.

w: escritura desde el comienzo.

a: añadir al final.

rb: lectura de un archivo binario.

wb: escritura en un archivo binario.

ab: añadir en un archivo binario.

Ejemplo

```
int main()
{
    FILE *archi1; //para escribir en el archivo binario
    ZOO animal; // tipo de dato predefinido

     archi1 = fopen("archi1.bin", "wb");//apertura del archivo
     if(archi1 == NULL) //¿el archivo existe? ¿está correcto?
     {
         printf("Imposible crear el archivo.\n");
         return 1;
     }
}
```

Cierre de un archivo

Se usará la función fclose definida en stdio.h:

int fclose(FILE *nombre_archivo);

Ejemplo

```
...

FILE *archi1; //apertura del archivo

if(archi1 = fopen("archi1.bin", "wb") == NULL)
{
    printf("Imposible crear el archivo. \n");
    return 1;
}

...

fclose(archi1);
```

Funciones importantes

Lectura y escritura, dos funciones de la biblioteca stdio.h: fread(), fwrite()

Ambas funciones permiten trabajar con bloques de datos, preferiblemente de tipo estructurado (registros). Previo a su uso, es necesario conocer el operador sizeof, que permite obtener la cantidad de bytes del tipo de su operando.

Ejemplo

```
…
ZOO animal; //tipo de dato predefinido
…
archi = fopen(“archi.bin”, “wb”); //apertura del archivo bin
if(archi==NULL)
{
   printf(“Imposible crear un archivo\n”);
   return 1;
}
…
fwrite(&animal, sizeof(ZOO), 1, archi); //graba el registro animal

//fwrite(arreglo, tamaño, contador, punteroDeArchivo);
//Permite escribir una gran cantidad de datos en una sola llamada a la función.

//fread(arreglo, tamaño, contador, punteroDeArchivo);
//Permite  leer una gran cantidad de datos en una sola llamada a la función.
```

Entrada/Salida de caracteres

Para la E/S de caracteres a través de archivos, 2 funciones de la biblioteca stdio.h: fgetc , fputc

Ejemplo: E/S interactiva

```
…
char car;
…
car = getchar(); //lectura interactiva
…
putchar(car);
```

Ejemplo: E/S por archivos

```
…
FILE *archi1, *archi2;
char car;
…
// apertura de archivos text
caracter1= fopen("archi1.text", “r");
caracter2= fopen("archi2.text", “w");

…
car = fgetc(archi1); // lectura desde archivo
fputc(archi2, car); //escritura desde archivo
…
```

Entrada/Salida de cadenas

Ejemplo: E/S interactiva

```
#define MAXIMO  100
...
char cadena[MAXIMO];
...
gets(cadena);
puts(cadena);
...
```

Ejemplo: E/S por archivos

```
#define MAXIMO  100
...
char cadena[MAXIMO];
FILE *cade1, *cade2;
...
// apertura de los archivos
...
fgets(cadena, MAXIMO, cade1);
fputs(cadena, cade2);
// cierre de ambos archivos
...
```

Entrada/Salida con formato

Ejemplo: E/S interactiva

```
...

int i;
...
printf("Números cuadrados\n") ;

for (i=1; i<20;i++)
{
    printf( "  %3d  \n",i);
}

...
```

Ejemplo: E/S por archivos

```
...
FILE *forma1;
int i;
...
// apertura del archivo en modo texto
...
fprintf(forma1, "20 numeros \n\n") ;

for (i=1; i<20;i++)
{
    fprintf(forma1, "  %3d   \n",i);
}

fclose(forma1);
...
```

Marca y función de fin de archivo.

El fin de datos de archivo se indica con un valor distintivo: EOF, de la biblioteca stdio.h

EOF esta en la tabla ASCII, y se usa para comparación con caracteres.

Se usará en los archivos de organización secuencial.

feof es una función que permite detectar el fin del archivo (stdio.h)

```
int feof(FILE *archivo);
```

Regresa un valor distinto de cero si ocurre el fin de archivo.

Ejemplo

```
int main ()
{       FILE *archi1;   // ARCHIVO BINARIO DEL CUAL SE VA A LEER "archi1.bin"
                        // QUE ESTA EN DISPOSITIVO EXTERNO
        ZOO animal;   // TIPO DE DATO PREDEFINIDO

        setvbuf(stdout, NULL, _IONBF, 0);
        archi1= fopen("E:\\archi1.bin", "rb"); // APERTURA DEL ARCHIVO

        // CONTROL DE APERTURA
        // LEE EL ARCHIVO BINARIO Y ESCRIBE EN LA SALIDA ESTANDAR
        fread(&animal, sizeof(ZOO), 1, archi1);
        while (!feof(archi1))
        {       printf("ESPECIE %s \n",animal.especie);
                printf("NRO DE JAULA %d\n",animal.numJaula);
                fread(&animal, sizeof(ZOO), 1, archi1);
        }
        fclose(archi1);
        return 0;
}
```

```c
#include <stdio.h>

int main()
{
    printf("Hasta la próxima clase!!\n");

    return 0;
}
```