RESUMEN SEGUNDA PARTE

**Utilización de un *parser:***

haremos uso de este tipo de analizadores cuando necesitemos interpretar informacion que se encuentra almacenada bajo ciertas reglas o “formato” (carácter especial “,” o “enter”).

Creacion de un parser

un parser no sera mas que una funcion la cual abre un archivo y lo lee y llena un array pasado por parametro con la informacion leida.

***Funcion fscanf:***

mediante la funcion fscanf, podremos leer de un archivo una cadena de careteres mientras se cumpla una regla descripta, que debera ser pasada a la funcion como parametro. La funcion analizara la cadena y continuara leyendo mientras se cumple la regla que se le paso, dentro de ella le podemos que almacene los datos.

R = fscanf(pfile, “%d,%d,%d,%d\n”, &a, &b. &c, &d);

la funcion recibe como primer argumento el puntero al archivo, luego recibe la regla a aplicar sobre los datos leidos y luego recibe una catidad indefinida de punteros a las variables (a,b,c en este caso) en donde se almaneceran los numeros encontrados según lo especificado.

Si ponemos [^ ,] estamos diciendo que agregue cualquier carateter menos la coma.

Si ponemos [^ \n] menos el enter

ARCHIVOS

un archivo de texto contiene toda su informacion guardada en binario pero se interpreta como texto.

Un archivo binario se guardan datos con distinto formato, es decir se puede guardar caracteres mezclados con enteros y flotantes.

Trabajas con archivos binarios esta asociado al uso de estructuras

en c, existen 2 categorias de fuciones para trabajar con archivos y son “buffer” y las que acceden directamente al archivo.

El “buffer” no tiene acceso directo al archivo y cualquier operación que se desee realizar (lectura o escritura) va a ser hecha sobre el buffer. Cuando se llena o se vacia se actualizan los datos desde y hacia el archivo.

Para trabajar con archivo en C, las funciones ultilizan un puntero a estructura FILE, se encuentra definida en el archivo stdio.h

typedef struct

{

}FILE;

para el manejo de archivos es indespensable definir un puntero a la estructura FILE, (FILE \*pArch;)

apertura de un archivo:

se usa el fopen() y se escribe asi:

FILE \* fopen (const char \*nombre\_de\_archivo. Cons char \*modo)

nombre del archivo: es una cadena de caracteres que representa el archivo.

Valor retornado: si esta bien, devuelve un puntero a file sino devuelve null

tener en cuenta: se pueden abrir varios archivos siempre y cuando exista por lo menos un puntero a file para cada uno

la cantidad de archivos que se pueden abrir al mismo tiempo depende del sistema operativo

no se puede modificar el valor del puntero devuelto por fpoen().

Si rotorna null, significa que ha fallado en abrir el archivo

fclose() cierra el archivo, se retorna 0 cerro bien bien, sino devuelve -1

int fcloseall (void) cierra todo tambien devuelve 0 o -1

la funcion para una escritura es fwrite (void \* origen, size\_t tamaño, size\_t cantidad, file \*arch)

origen puntero de donde se obtienen los daros

tamño la cantdad de bytes del dato que se va a escribir

cantidad: es la cantidad de datos de longitud tamaño que se van a escribir

arch es el puntero asociado al archivos

devuelve el numeros de datos escritos (cantidad). Si el valor retornado es manor al que se indico, hubo un error en la escritura.

La funcion fwrite toma la cantidad de dats de longitud tamaño desde el origen y los escribe en el archivo asociado al puntero arch.

Antes de escribir el archivo se debe cargar el dato que se desea guardar

y para leer se usa el fread, los mismos parametros

para verificar que lllego al final del archivo se usa la funcion feof, cuando no se lee la cantidad de datos indicada puede haberse alcanzado el final de archivo o se puso haber producido un error

el portotipo es int feof(FILE\* arch)

para modificar archivos se usa el fseek y el rewind

rewind sirve para llevar el indicardor al principio del archivos

el prototipo es: void rewind(FILE\* arch)

el rewind, limpia los indicadores y el error que se encuentran en la estructura file

fseek permite desplazar el idicardor de posicoin del archivo a la posicion que el usuario quiere

int fseek (file \*arch, long despla. Int origien)

ftell te obtiene la posicion altual del indicador de posicion

long ftell()file \*arch)