República Bolivariana de Venezuela
Universidad Nacional Experimental Politécnica
"Antonio José de Sucre"
Vice Rectorado Barquisimeto
Departamento de Ingeniería Electrónica

Practica 5
laboratorio de diseño de sistemas de computación

Integrantes:
Gerardo Alfonzo Campos Fonseca
V. 27085179
José Andrés Cortez Teran
V. 26540824

Barquisimeto, Agosto del 2021

Índice

Índice de figuras

1. Programa lectura.asm

El código del programa se encuentra en un único archivo llamado **lectura.asm**. Este programa funciona mediante una aplicación de consola usando la api de 32 bits de Windows. Para la lectura del archivo se usan las funciones CreateFile en modo lectura, GetFileSize y ReadFile el tamaño. GetFileSize se utiliza para crear un bloque de memoria lo suficientemente grande para almacenar el archivo en memoria, mas el nuevo registro. Para escribir el archivo una vez modificado se utilizan lstrlen para poder saber la cantidad de bytes a escribir y finalmente se usa WriteFile.

El programa se puede dividir en los siguientes pasos:

- Apertura y lectura del archivo.
- Apertura del archivo en el que se va a escribir.
- Conteo de lineas.
- Pedir por pantalla los datos que se van a ingresar.
- Creacion del nuevo elemento que se anexara al archivo.
- Localizacion de la posicion en la que se escribira.
- Escritura y cierre del fichero.

Apertura y lectura del archivo

Para esto se pide por teclado el nombre del fichero a abrir, mediante un macro **Get_Input** el cual hace uso de las funciones **StdOut** y **StdIn** para dar como salida un mensaje y leer por teclado. Y se utilizan las funciones **CreateFile, GetFileSize, GlobalAlloc, ReadFile** de la misma forma que en la Campos y col., 2021 para abrir y almacenar la informacion contenida en el archivo y **CloseHandle** para cerrarla.

Apertura del archivo en el que se va a escribir

Una vez tenida la informacion, se abre un nuevo fichero, en este caso en modo escritura, para almacenar allí la informacion. Esto se logra con:

```
"invoke  CreateFile,ADDR new_file ,GENERIC_WRITE,0,0,\
         CREATE_ALWAYS,FILE_ATTRIBUTE_NORMAL,0
 mov     hFileWrite,eax"
```

De donde resalta el uso de un nuevo “handler” y los parametros “ADDR new_file”, “GENERIC_WRITE” y “CREATE_ALWAYS” las cuales implican el nombre con el que se creara el archivo, el metodo de escritura y la opcion de que, aunque el archivo ya exista, siempre se creara desde 0.

Conteo de lineas

Para el calculo de lineas, se utiliza como contador el registro **ecx** inicializado en 1, la direccion de la informacion en el registro **esi** y se hace un ciclo de chequeo, en el que se recorren todos los caracteres del archivo buscando el caracter de salto de linea \ **n** o el numero 10 por ascii. Cada vez que hay una coincidencia se incrementa en 1 el contador, ya que hay una nueva linea, de esta forma cada coincidencia implicaria el comienzo de la 2da, 3ra, $\cdots$, n-esima linea.

Una vez recorrido todo el archivo, se procede a agregar la linea adicional, la nueva a agregar, y por esto se incrementa el contador nuevamente. Se guarda el valor obtenido en la pila, mediante un push y se llama al procedimiento NumbToStr para guardar el numero convertido en forma de string y poder mostrarlo por pantalla cuando se requieran los valores a ingresar.

Este procedimiento se basa en la construccion del string mediante la obtencion individual de cada digito. Esto se logra mediante divisiones sucesivas en-por base 10. Primero se calcula la direccion del ultimo caracter posible al sumar la memoria del puntero al string + la cantidad de digitos que se pueden agregar, en este caso 10 digitos. Se agrega el caracter de terminacion, “0”, como ultimo elemento y luego se hace un ciclo, mientras exista algun digito (el resultado sea distinto de 0) se divide, **eax** entre **ebx**, el resultado queda en **eax** y el residuo en **dl** se guarda el residuo en la direccion correspondiente, esta se decrementa y se repite el proceso. Finalmente, se devuelve en eax la direccion del primer elemento del string resultante.

Linea 1 ...	\r	\n
Linea 2 ...	\r	\n
Linea 3 ...	\r	\n
Linea 4 ...	\r	\n
Linea 5 ...	\r	\n

Figura 1: Conteo de lineas

Pedir por pantalla los datos que se van a ingresar

Lo primero que se le pide al usuario es donde quiere ingresar el nuevo registro, se espera un numero en decimal, para esto el programa cuenta la cantidad de caracteres que son números del 0 al 9 y los pasa a un ciclo que convierte la cadena de caracteres en un entero mediante multiplicaciones consecutivas por 10 y la suma de cada numero de la cadena. Finalmente se piden el resto de caracteres que son simplemente cadenas de caracteres que se guardan en sus variables especificas.

Creacion del nuevo elemento que se anexara al archivo

Una vez obtenidos los datos, hace falta juntarlos de forma eficiente y en el formato indicato, esto se logra mediante la funcion de la api de windows "**lstrcat**" esta toma como parametros 2 direcciones a strings la primera es a la cual se anexará la segunda, de esta forma solo hace falta llamarla tantas veces sea necesaria y con los parametros adecuados para construir nuestro nuevo registro.

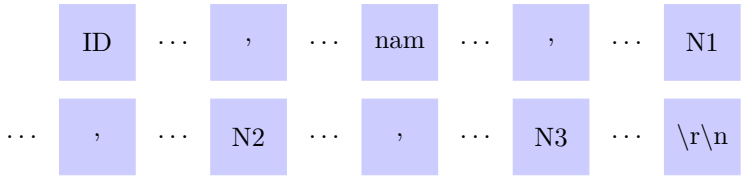

Figura 2: Creando registro

Localizacion de la posicion en la que se escribira

Para esto se utilizan los elementos guardados anteriormente, al hacer "pop" se retornan los valores numericos de donde se va a insertar el nuevo registro y cuantas lineas tiene el archivo. se hace una estructura de decision anidada en la cual se evaluan 3 posibilidades:

- El elemento se agregara en la primera posicion.
- El elemento se agregara en una posicion intermedia.
- El elemento se agregara en la ultima posicion.

Si se anexara al inicio o al final es una decision trivial, simplemente se estructura de esa forma la concatenacion y se envia al segmento de escritura correspondiente. En el caso de que el elemento sea en una posicion intermedia se utiliza un ciclo parecido al del conteo de lineas con la unica modificacion de que al conseguirse una nueva linea se evalua si es en esta linea en donde se anexara el registro creado anteriormente; si es asi, se va al procedimiento de escritura, si no se continua el ciclo hasta conseguir la linea deseada.

Escritura y cierre del fichero

Siguiendo las ideas anteriores, se presentan 3 posibilidades de escritura:

- Primera linea: se escribe 3 veces en el archivo:
  - Se escribe el nuevo registro.
  - Se escribe el salto de linea.
  - Se escribe el archivo leido.
- Linea intermedia: Ocurre en cuatro pasos:
  - Se cuenta la cantidad de bytes a escribir de acuerdo a la linea que se desea ingresar.

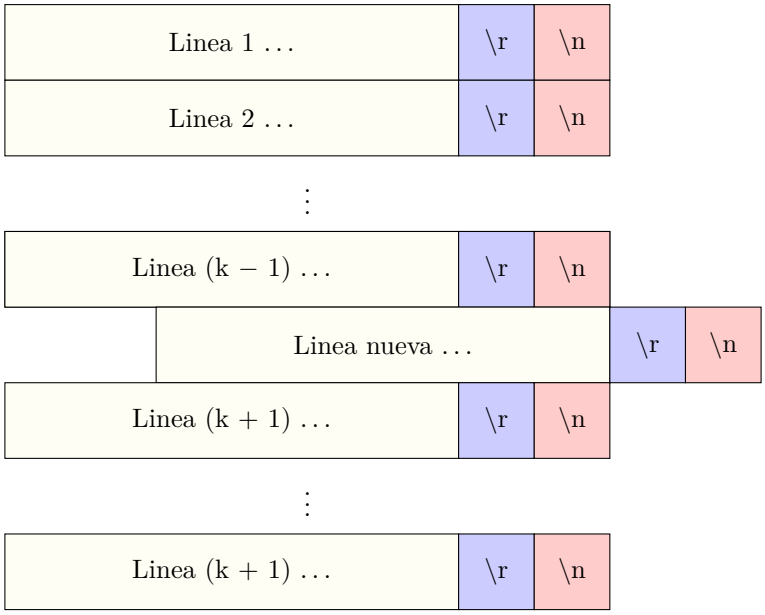

Figura 3: Colocando nuevo elemento en una posicion intermedia

- Se escribe esa cantidad bytes del archivo anterior.
- Se escribe el nuevo registro.
- Se escribe el resto del archivo.

▪ Ultima linea: se escribe 3 veces en el archivo:

- Se escribe el archivo leido.
- Se escribe el salto de linea.
- Se escribe el nuevo registro.

```
1
2  .586
3  .MODEL flat, stdcall
4  OPTION CASEMAP:NONE
5  Include windows.inc
6  Include kernel32.inc
7  Include masm32.inc
8  include  user32.inc
9
10 IncludeLib kernel32.lib
11 IncludeLib masm32.lib
12 includelib user32.lib
13
14 ; lectura de archivo
15 ; calculos respectivos al archivo
16 ; escritura de archivo stdout
17
18 ; pedir informacion adicional
```

```
19
20 ; re estructurar todo
21 ; imprimir nuevo archivo
22
23 ;escribir y guardar en el nuevo archivo
24
25 NumbToStr   PROTO :DWORD,:DWORD
26
27
28 Main PROTO
29   Print_Text Macro txt:REQ
30   Invoke StdOut,ADDR txt
31 EndM
32
33 Get_Input Macro prompt:REQ,buffer:REQ
34   Invoke StdOut,ADDR prompt
35   Invoke StdIn,ADDR buffer, LengthOf buffer
36 EndM
37
38
39
40 .DATA
41 Msg1 DB "Please Type the file is name or path: ",0AH,0DH,0
42 Msg4 DB "Press Enter to Exit",0AH,0DH,0
43 CRLF DB 0DH,0AH,0
44
45 MsgPos DB "Please type the position where the new register will be inserted. ",0AH,0DH,0
46 Msgaux DB "should be almost 1 and less than: ",0
47
48 MsgNom DB "Please type the name: ",0AH,0DH,0
49 MsgCed DB "Please type the ID: ",0AH,0DH,0
50 MsgN1 DB "Please type first grade: ",0AH,0DH,0
51 MsgN2 DB "Please type second grade: ",0AH,0DH,0
52 MsgN3 DB "Please type third grade: ",0AH,0DH,0
53
54 coma db ",",0
55
56 Aux_string db 100 dup(0) ;para el nuevo campo de la bbdd
57
58 new_file DB "BBDD.txt",0
59
60
61 .DATA?
62 inbuf DB 100 DUP (?)
63
64 hFile         dd ?
65 hFileWrite      dd ?
66 FileSize      dd ?
67 hMem         dd ?
```

```
68  BytesRead   dd ?
69
70  pos     DB 10 dup(?)
71  ID      DB 23 DUP (?)
72  nam   DB 80 DUP (?)
73  N1      DB  10 dup(?)
74  N2    DB 10 dup(?)
75  N3      DB 10 dup(?)
76
77   bytewr    DD ?;variable adicional creada por necesidad para el proc
78
79  BufferSize DD ?
80  Buffer_div_size DD ?
81  hMem_div dd ?
82
83  Cant_lineas        db 11 dup(?)  ; variable para la conversion de cadenas, 11 elementos porque
        10 cubren max int y 1 caracter de terminacion
84  hCant_lineas      dd ?
85  .CODE
86  Start:
87      ;*************************************************
88      ;handling the files
89
90      ;******* file to read ************************
91    Get_Input Msg1, inbuf
92    ;se usa la api de windows para abrir la fila
93    invoke  CreateFile,ADDR inbuf,GENERIC_READ,0,0,\
94    OPEN_EXISTING,FILE_ATTRIBUTE_NORMAL,0
95    mov     hFile,eax
96
97    ;obtencion del tamano de la fila para pedir memoria dinamica
98    invoke  GetFileSize,eax,0
99    mov     FileSize,eax
100   inc     eax
101   ;pedir memoria dinamica
102   invoke  GlobalAlloc,GMEM_FIXED,eax
103   mov     hMem,eax
104   add     eax,FileSize
105   mov     BYTE PTR [eax],0   ; Set the last byte to NULL so that StdOut
106   ; can safely display the text in memory.
107   ;finalmente se lee la fila
108   invoke  ReadFile,hFile,hMem,FileSize,ADDR BytesRead,0
109
110   ;se escribe el fichero
111   invoke  StdOut,hMem
112   Print_Text CRLF ;salto de linea
113   Print_Text CRLF ;salto de linea
114   invoke  CloseHandle,hFile
115
```

```
116
117 ;********* file to write ************************
118 ;ADDR inbuf
119 invoke  CreateFile,ADDR new_file ,GENERIC_WRITE,0,0,\
120         CREATE_ALWAYS,FILE_ATTRIBUTE_NORMAL,0
121 mov     hFileWrite,eax
122   ;invoke CreateFile,lpName,GENERIC_WRITE,NULL,NULL,CREATE_ALWAYS,FILE_ATTRIBUTE_NORMAL,NULL
123
124
125 ;cantidad de bytes se consigue restando 2 direcciones de memoria xD
126
127
128 ;*******************************************************
129 ; calculo de lineas
130
131
132 mov ecx, 1 ;inicializamos el contador en 1 posible linea
133 mov esi,  hMem ; vamos a trabajar con la fila, por eso usamos el handler de memoria...
134 mov eax, 0 ; limpiamos eax, solo usaremos al (byte...)
135 cont_lineas:
136   mov al, [esi]
137
138   cmp eax, 10 ;compara buscando caracter de salto de linea '\r' '\n' con \n = 10 decimal A
    hex, gracias olly
139   jne n_linea_nueva
140   inc ecx
141
142   n_linea_nueva:
143
144   cmp eax , 0 ; compara con el caracter de fin de archivo, end buffer...
145 je  f_lineas
146
147   inc esi
148
149 jmp cont_lineas
150
151 f_lineas:
152
153 mov  edi, OFFSET Cant_lineas; guardamos la cantidad de lineas que hay! primero la memoria a
    un registro
154 inc ecx ; cantidad de lineas +1
155 push ecx ;guardamos el valor numerico
156
157 ;conversion a ascii
158 invoke  NumbToStr, ecx, ADDR Cant_lineas
159
160 push eax
161 mov  esi, OFFSET hCant_lineas; guardamos la cantidad de lineas que hay! primero la memoria a
    un registro
```

```
162 mov     [esi],eax   ; store the character in the buffer
163 pop esi
164
165 ;mov edi,
166 ;*************************************************************************
167 ; lectura de valores
168
169 Invoke lstrlen, OFFSET pos  ; Guardamos la longitud del string en ECX
170       mov ecx, eax
171       dec ecx
172
173 ;  ****  verificacion de posicion ****
174
175 Print_Text MsgPos
176 Print_Text Msgaux
177 invoke  StdOut,esi
178 Get_Input CRLF,  pos; pedir la posicion
179
180 mov edi, OFFSET pos
181
182 xor ecx, ecx
183 mov bl, [edi]
184
185 .WHILE bl >= 30h && bl <= 39h
186 inc ecx
187 mov bl, [edi + ecx]
188 .ENDW
189
190 xor eax, eax
191
192 convertir:
193
194 mov bl, [edi]
195 imul eax, 10
196 sub bl, 30h
197 movzx ebx, bl
198 add eax, ebx
199 inc edi
200
201 loop convertir
202
203
204
205 push eax; Guardando el valor en el que se desa ingresar para recuperarlo
206 ;mas tarde facilmente
207
208
209 ;*********** Pedir el resto de los datos *************
210
```

```
211 Get_Input MsgCed, ID; pedir la cedula
212 Get_Input MsgNom, nam; pedir el nombre
213 Get_Input MsgN1, N1; pedir la nota 1
214 Get_Input MsgN2, N2; pedir la nota 2
215 Get_Input MsgN3, N3; pedir la nota 3
216
217
218 ;*************************************************
219 ;   Creacion de la estring
220 ;*************************************************
221
222 invoke lstrcat,offset Aux_string,OFFSET ID
223 invoke lstrcat,offset Aux_string,OFFSET coma
224 invoke lstrcat,offset Aux_string,OFFSET nam
225 invoke lstrcat,offset Aux_string,OFFSET coma
226 invoke lstrcat,offset Aux_string,OFFSET N1
227 invoke lstrcat,offset Aux_string,OFFSET coma
228 invoke lstrcat,offset Aux_string,OFFSET N2
229 invoke lstrcat,offset Aux_string,OFFSET coma
230 invoke lstrcat,offset Aux_string,OFFSET N3
231
232 Invoke lstrlen, offset Aux_string
233 mov BufferSize,eax
234
235
236 ;*****************************************************************
237 ; ***************  Escritura en el archivo ***********************
238
239 pop eax ; recuperamos el valo de la linea donde vamos a insertar
240 pop ecx ;recuperamos el valor de la cantidad de lineas que hay
241
242 cmp eax,1
243 je first_line
244 cmp ecx, eax
245 je last_line
246 jmp in_line
247
248 first_line:
249
250   invoke WriteFile,hFileWrite,offset Aux_string,BufferSize,ADDR bytewr,NULL;escritura de la
      cadena
251   Invoke WriteFile,hFileWrite,offset CRLF,2,ADDR bytewr,NULL;escritura del salto de linea
252   invoke WriteFile,hFileWrite,hMem,FileSize,ADDR bytewr,NULL   ;escritura del archivo
253   jmp fin;listo
254
255 last_line:
256
257   invoke WriteFile,hFileWrite,hMem,FileSize,ADDR bytewr,NULL;escritura del archivo
258   Invoke WriteFile,hFileWrite,offset CRLF,2,ADDR bytewr,NULL;escritura del salto de linea
```

```
259     invoke WriteFile,hFileWrite,offset Aux_string,BufferSize,ADDR bytewr,NULL ; escritura del
        nuevo registro
260     jmp fin;listo
261
262
263   in_line:
264   ;eax contiene el lugar en el que se va a guardar, usamos un respaldo en edi
265   mov edi,eax
266
267   mov ecx, 1 ;inicializamos el contador en 1 posible linea
268   mov esi,  hMem ; vamos a trabajar con el archivo, por eso usamos el handler de memoria...
269   mov eax, 0 ; limpiamos eax, solo usaremos al (byte...)
270   cont_lineas2:
271     mov al, [esi]
272
273     cmp eax, 10 ;compara buscando caracter de salto de linea '\r' '\n' con \n = 10 decimal A
        hex
274     jne n_linea_nueva2
275     ;Se llego al punto en el que se escribiria la nueva linea
276     inc ecx
277     cmp edi, ecx
278     je escritura
279
280     n_linea_nueva2:
281     cmp eax , 0 ; compara con el caracter de fin de archivo, end buffer...
282   je  f_lineas2
283
284     inc esi
285   jmp cont_lineas2
286
287   f_lineas2:
288     jmp fin
289
290   escritura:
291     ; eax tiene un caracter
292     ;esi direccion del hmem+cant caracteres     sirve
293     ; edi tiene el numero de linea a donde va
294     ;ecx tiene el contador de cuantos lineas  van
295     inc esi
296     push esi ; guardamos la direccion de hmem+cant_caracteres
297
298     mov ecx, hMem; direccion inicial del archivo
299     sub esi, ecx ;en esi se tiene cuantos caracteres hay
300
301     mov edi, offset Buffer_div_size
302     mov [edi] , esi ; luego el valor a la direccion
303
304     invoke WriteFile,hFileWrite,hMem,Buffer_div_size,ADDR bytewr,NULL;escritura del archivo
        primera parte
```

```
305
306
307     invoke WriteFile,hFileWrite,offset Aux_string,BufferSize,ADDR bytewr,NULL ; escritura del
        nuevo registro
308     Invoke WriteFile,hFileWrite,offset CRLF,2,ADDR bytewr,NULL;escritura del salto de linea
309
310     pop eax ; guardamos en eax la direccion de donde me quede en el archivo
311     mov   hMem_div,eax
312     mov  ecx, FileSize
313     sub ecx, esi ; en ecx quedan cuantos caracteres faltan
314     mov [edi] , ecx ; luego el valor a la direccion
315     invoke WriteFile,hFileWrite,hMem_div,Buffer_div_size,ADDR bytewr,NULL;escritura del
        archivo
316
317
318   fin:
319   ;*************************************************
320   ; Fin Programa
321   ;se cierran los ficheros
322
323   invoke  CloseHandle,hFileWrite
324
325
326
327   ;se libera la memoria dinamica
328   invoke  GlobalFree,hMem
329   ;espera enter para salir y poder leer
330
331
332
333   Print_Text CRLF ;salto de linea
334   Get_Input Msg4,inbuf ;mensaje de salida
335
336
337   ;sale del programa
338   Invoke ExitProcess,0
339
340
341   NumbToStr PROC uses ebx x:DWORD,buffer:DWORD
342
343   mov     ecx,buffer
344   mov     eax,x
345   mov     ebx,10
346   add     ecx,ebx              ; ecx = buffer + max size of string
347   mov     BYTE PTR [ecx],0    ; store the character in the buffer
348   dec ecx
349   ciclo:
350     xor     edx,edx
351     div     ebx
```

```
352     add     edx,48              ; convert the digit to ASCII
353     mov     BYTE PTR [ecx],dl   ; store the character in the buffer
354     dec     ecx                 ; decrement ecx pointing the buffer
355     test    eax,eax             ; check if the quotient is 0
356   jnz     ciclo
357
358   inc     ecx
359   mov     eax,ecx             ; eax points the string in the buffer
360   ret
361
362 NumbToStr ENDP
363
364
365 End Start
```

Funcionamiento del programa

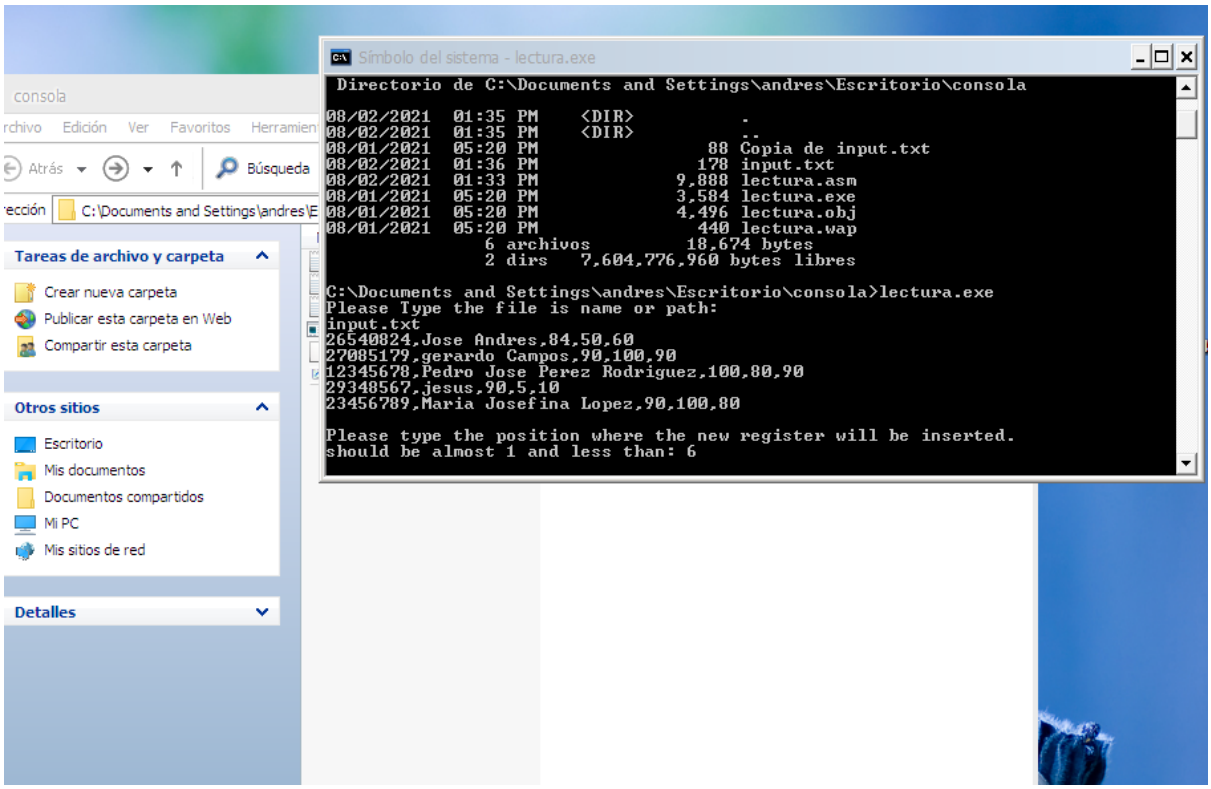

Figura 4: Abriendo el archivo

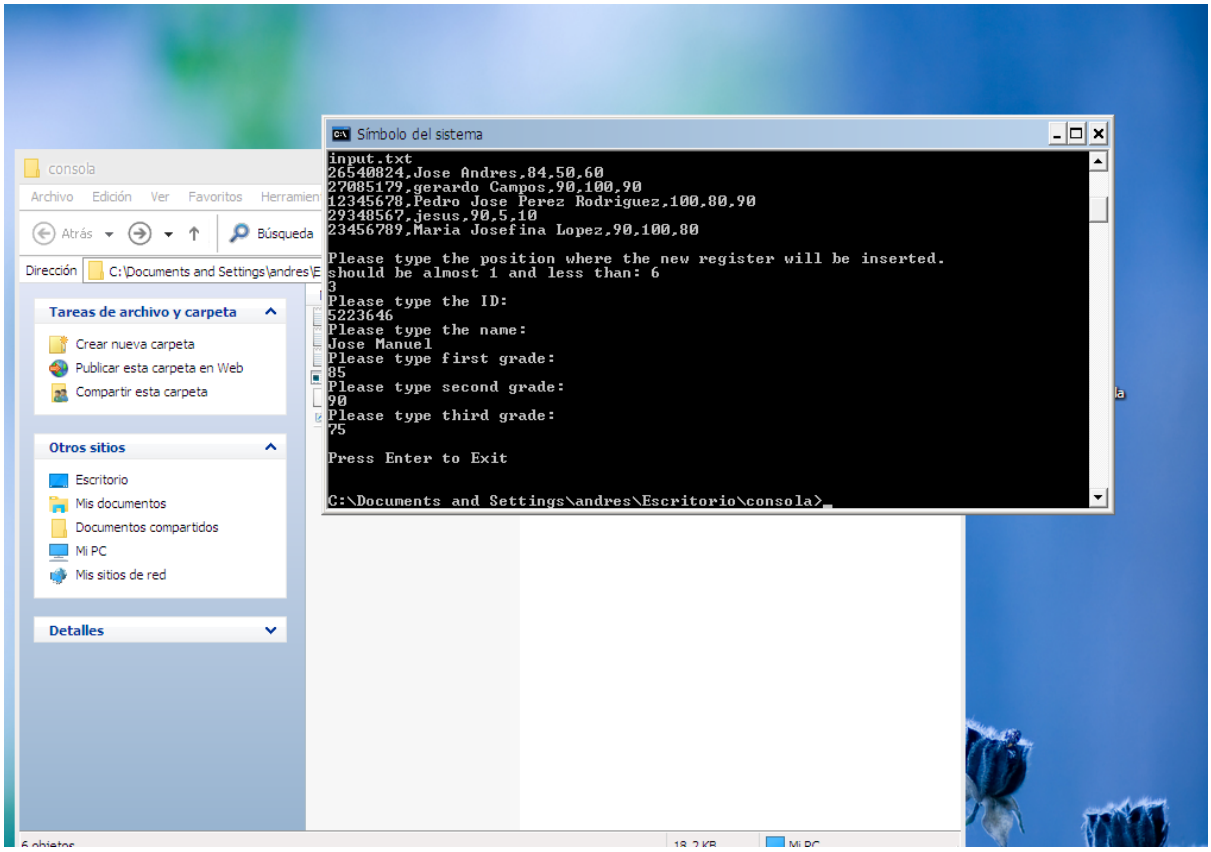

Figura 5: Agregando un nuevo registro

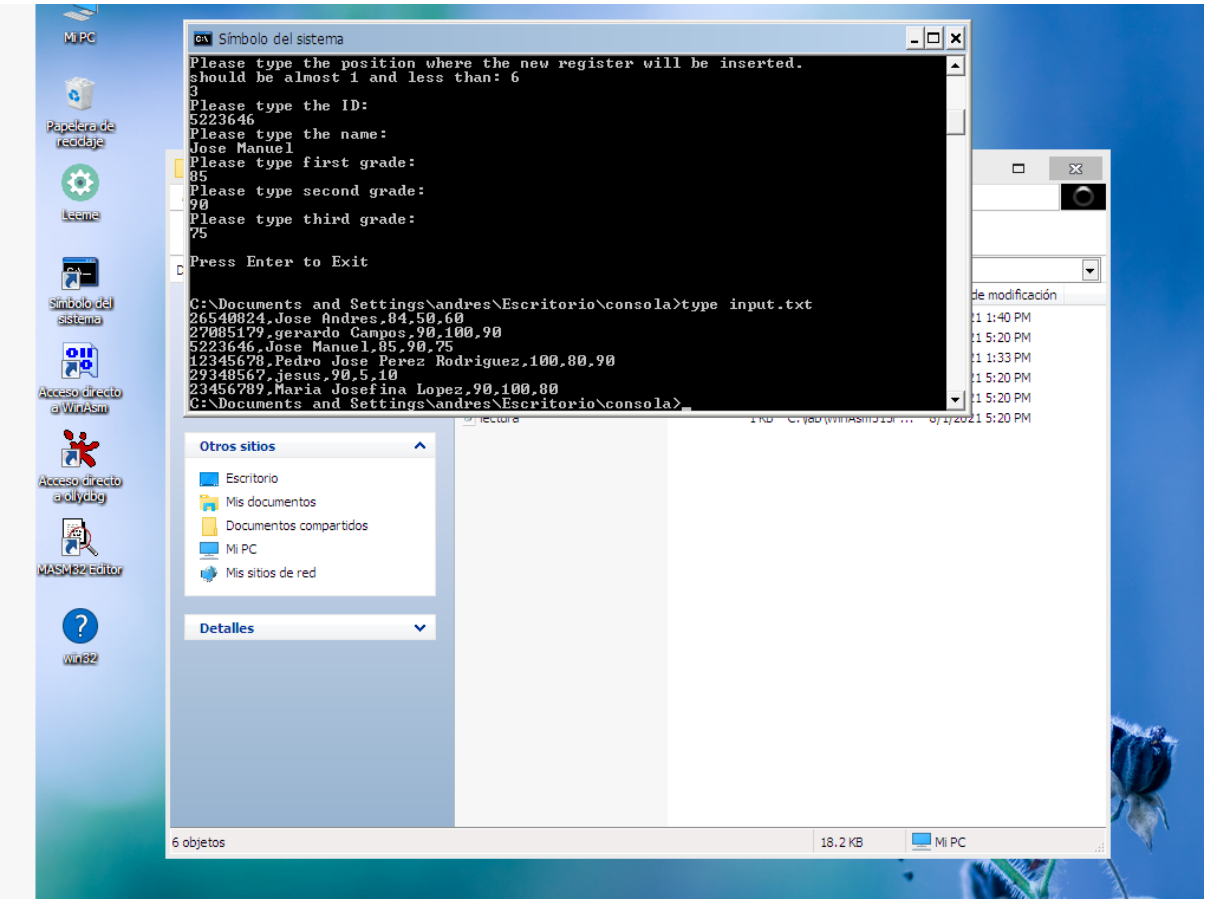

Figura 6: Mostrando en pantalla el archivo resultante

Referencias Bibliográficas

Campos, G. & Cortez, J. (2021). Practica 4 laboratorio de diseño de sistemas de computación.

Iczelion. (s.f.). http://www.movsd.com/icz.htm

Irvine, K. R. (2002). Assembly Language for Intel Assembly Language for Intel-Based Computers, 4 Computers, 4th Edition Edition [fecha de consulta: 19/7/2021]. https://www.csie.ntu.edu.tw/~acpang/course/asm_2004/slides/chapt_08Solve.pdf

Microsoft. (2017). CreateFileA function (fileapi.h) [fecha de consulta: 22/07/2021]. https://docs.microsoft.com/en-us/windows/win32/api/fileapi/nf-fileapi-createfilea

Microsoft. (2021a). CloseHandle function (handleapi.h) [fecha de consulta: 22/7/2021]. https://docs.microsoft.com/en-us/windows/win32/api/handleapi/nf-handleapi-closehandle

Microsoft. (2021b). GetDlgItemTextA function (winuser.h) [fecha de consulta: 24/7/2021]. https://docs.microsoft.com/en-us/windows/win32/api/winuser/nf-winuser-getdlgitemtexta

Microsoft. (2021c). GetFileSize function (fileapi.h) [fecha de consulta: 22/7/2021]. https://docs.microsoft.com/en-us/windows/win32/api/fileapi/nf-fileapi-getfilesize

Microsoft. (2021d). GlobalAlloc function (winbase.h) [fecha de consulta: 22/7/2021]. https://docs.microsoft.com/en-us/windows/win32/api/winbase/nf-winbase-globalalloc

Microsoft. (2021e). How to get the string value of editbox in the variable in win32 application? [fecha de consulta: 24/7/2021]. https://social.msdn.microsoft.com/Forums/vstudio/en-US/17c2d97a-011b-4fb1-9563-4f095d9321e4/how-to-get-the-string-value-of-editbox-in-the-variable-in-win32-application?forum=vcgeneral

Microsoft. (2021f). ReadFile function (fileapi.h) [fecha de consulta: 22/7/2021]. https://docs.microsoft.com/en-us/windows/win32/api/fileapi/nf-fileapi-readfile

OVERFLOW, S. (2015). Getting string input and displaying input with DOS interrupts MASM [fecha de consulta: 19/7/2021]. https://stackoverflow.com/questions/29504516/getting-string-input-and-displaying-input-with-dos-interrupts-masm

to 2012, T. M. F. A. 2. (s.f.). how to read from file in masm [fecha de consulta: 22/7/2021]. http://www.masmforum.com/board/index.php?topic=16266.0