



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



## Práctica 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

<i>ÍNDICE</i>	II
---------------	----

## Índice

Índice	II
--------	----

Índice de figuras	1
-------------------	---

## Índice de figuras

1.	Abriendo el archivo . . . . .	9
2.	Agregando un nuevo registro . . . . .	10
3.	Mostrando en pantalla el archivo resultante . . . . .	11

## 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.

```

1  .386
2  .MODEL flat, stdcall
3  OPTION CASEMAP:NONE
4
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
26 Main PROTO
27
28 Print_Text Macro txt:REQ
29     Invoke StdOut,ADDR txt
30 EndM
31
32 Get_Input Macro prompt:REQ,buffer:REQ
33     Invoke StdOut,ADDR prompt
34     Invoke StdIn,ADDR buffer, LengthOf buffer
35 EndM
36
37 .DATA
38 Msg1    DB "Please Type the file is name or path: ",0AH,0DH,0
39 Msg4    DB "Press Enter to Exit",0AH,0DH,0

```

```

40 CRLF      DB 0DH,0AH,0
41 MsgPos    DB "Please type the position where the new register will be inserted. ",0AH,0DH,0
42 Msgaux    DB "should be almost 1 and less than: ",0
43 MsgNom     DB "Please type the name: ",0AH,0DH,0
44 MsgCed     DB "Please type the ID: ",0AH,0DH,0
45 MsgN1     DB "Please type first grade: ",0AH,0DH,0
46 MsgN2     DB "Please type second grade: ",0AH,0DH,0
47 MsgN3     DB "Please type third grade: ",0AH,0DH,0
48 coma      DB ", ",0
49
50 Cant_lineas DB 1
51 Aux_string DB 100 dup(0) ;para el nuevo campo de la bbdd
52 new_file    DB "BBDD.txt",0
53
54 .DATA?
55 inbuf DB 100 DUP (?)
56 hFile      DD ?
57 hFileWrite DD ?
58 FileSize   DD ?
59 hMem        DD ?
60 BytesRead  DD ?
61 pos        DB 10 DUP(?)
62 ID          DB 23 DUP(?)
63 nam         DB 80 DUP(?)
64 N1          DB 10 DUP(?)
65 N2          DB 10 DUP(?)
66 N3          DB 10 DUP(?)
67
68 bytewr      DD ?           ;variable adicional creada por necesidad para el proc
69 BufferSize  DD ?
70 Buffer_div_size DD ?
71 hMem_div    DD ?
72
73 .CODE
74 Start:
75
76 ;*****
77 ;                               handling the files                               *
78 ;*****
79
80 ; file to read
81 Get_Input Msg1, inbuf
82 ;se usa la api de windows para abrir la fila
83 invoke CreateFile,ADDR inbuf,GENERIC_READ,0,0,\
84 OPEN_EXISTING,FILE_ATTRIBUTE_NORMAL,0
85 mov      hFile,eax
86
87 ;obtencion del tamaño de la fila para pedir memoria dinamica
88 invoke GetFileSize,eax,0

```

```

89  mov     FileSize,eax
90  inc     eax
91  ;pedir memoria dinamica
92  invoke  GlobalAlloc,GMEM_FIXED,eax
93  mov     hMem,eax
94  add     eax,FileSize
95  mov     BYTE PTR [eax],0    ; Set the last byte to NULL so that StdOut
96  ; can safely display the text in memory.
97  ;finalmente se lee la fila
98  invoke  ReadFile,hFile,hMem,FileSize,ADDR BytesRead,0
99
100 ;se escribe el fichero
101 invoke  StdOut,hMem
102 Print_Text CRLF ;salto de linea
103 Print_Text CRLF ;salto de linea
104 invoke  CloseHandle,hFile
105
106 ;*****
107 ;*                               file to write                               *
108 ;*****
109
110 ;ADDR inbuf
111 invoke  CreateFile,ADDR inbuf ,GENERIC_WRITE,0,0,\
112         CREATE_ALWAYS,FILE_ATTRIBUTE_NORMAL,0
113 mov     hFileWrite,eax
114 ;invoke CreateFile,lpName,GENERIC_WRITE,NULL,NULL,CREATE_ALWAYS,
115         FILE_ATTRIBUTE_NORMAL,NULL
116
117 ;cantidad de bytes se consigue restando 2 direcciones de memoria xD
118
119
120 ;*****
121 ;                               calculo de lineas                               *
122 ;*****
123
124
125 mov ecx, 1 ;inicializamos el contador en 1 posible linea
126 mov esi, hMem ; vamos a trabajar con la fila, por eso usamos el handler de memoria...
127 mov eax, 0 ; limpiamos eax, solo usaremos al (byte...)
128 cont_lineas:
129     mov al, [esi]
130
131     cmp eax, 10 ;compara buscando caracter de salto de linea '\r' '\n' con \n = 10
132     decimal A hex, gracias olly
133     jne n_linea_nueva
134     inc ecx
135
136 n_linea_nueva:

```

```

136
137     cmp eax , 0 ; compara con el caracter de fin de archivo, end buffer...
138 je     f_lineas
139
140     inc esi
141
142 jmp cont_lineas
143
144 f_lineas:
145
146 mov  edi, OFFSET Cant_lineas; guardamos la cantidad de lineas que hay! primero la
    memoria a un registro
147 inc  ecx ; cantidad de lineas +1
148 push ecx ;guardamos el valor numerico
149
150 xor  ecx, 30H; mascara para llevarlo a ascii
151 mov  [edi] , ecx ; luego el valor a la direccion
152
153 ;*****
154 ;*                                lectura de valores                                *
155 ;*****
156
157 ;*****
158 ;*                                verificacion de posicion                            *
159 ;*****
160
161 verificacion:
162     Print_Text MsgPos
163     Print_Text Msgaux
164     Print_Text Cant_lineas
165     Get_Input CRLF, pos; pedir la posicion
166
167     mov  edi, OFFSET pos
168     xor  eax,eax ; limpiamos eax
169     mov  eax, [edi]
170     and  eax, 15 ;llevamos eax a numerico
171
172     cmp  eax,0
173     jg   segundo
174     jmp  verificacion
175
176 segundo:
177     pop  ecx ;recuperamos el valor numerico
178     push ecx ;lo volvemos a guardar para uso posterior al revisar
179     cmp  ecx,eax
180     jl   verificacion
181
182     push eax; Guardando el valor en el que se desa ingresar para recuperarlo
183 ;mas tarde facilmente

```

```

184
185 ;*****
186 ;*                               Pedir el resto de los datos                               *
187 ;*****
188
189 Get_Input MsgCed, ID; pedir la cedula
190 Get_Input MsgNom, nam; pedir el nombre
191 Get_Input MsgN1, N1; pedir la nota 1
192 Get_Input MsgN2, N2; pedir la nota 2
193 Get_Input MsgN3, N3; pedir la nota 3
194
195
196 ;*****
197 ;*                               Creacion de la esttring                               *
198 ;*****
199
200 invoke lstrcat,offset Aux_string,OFFSET ID
201 invoke lstrcat,offset Aux_string,OFFSET coma
202 invoke lstrcat,offset Aux_string,OFFSET nam
203 invoke lstrcat,offset Aux_string,OFFSET coma
204 invoke lstrcat,offset Aux_string,OFFSET N1
205 invoke lstrcat,offset Aux_string,OFFSET coma
206 invoke lstrcat,offset Aux_string,OFFSET N2
207 invoke lstrcat,offset Aux_string,OFFSET coma
208 invoke lstrcat,offset Aux_string,OFFSET N3
209
210 Invoke lstrlen, offset Aux_string
211 mov BufferSize,eax
212
213
214 ;*****
215 ;                               Escritura en el archivo                               *
216 ;*****
217
218 pop eax ; recuperamos el valo de la linea donde vamos a insertar
219 pop ecx ;recuperamos el valor de la cantidad de lineas que hay
220
221 cmp eax,1
222 je first_line
223 cmp ecx, eax
224 je last_line
225 jmp in_line
226
227 first_line:
228
229 invoke WriteFile,hFileWrite,offset Aux_string,BufferSize,ADDR bytewr,NULL;escritura
de la cadena
230 Invoke WriteFile,hFileWrite,offset CRLF,2,ADDR bytewr,NULL;escritura del salto de
linea

```



```

231     invoke WriteFile,hFileWrite,hMem,FileSize,ADDR bytewr,NULL ;escritura del archivo
232     jmp fin;listo
233
234 last_line:
235
236     invoke WriteFile,hFileWrite,hMem,FileSize,ADDR bytewr,NULL;escritura del archivo
237     Invoke WriteFile,hFileWrite,offset CRLF,2,ADDR bytewr,NULL;escritura del salto de
238     linea
239     invoke WriteFile,hFileWrite,offset Aux_string,BufferSize,ADDR bytewr,NULL ;
240     escritura del nuevo registro
241     jmp fin;listo
242
243 in_line:
244 ;eax contiene el lugar en el que se va a guardar, usamos un respaldo en edi
245 mov edi,eax
246
247 mov ecx, 1 ;inicializamos el contador en 1 posible linea
248 mov esi, hMem ; vamos a trabajar con el archivo, por eso usamos el handler de
249 memoria...
250 mov eax, 0 ; limpiamos eax, solo usaremos al (byte...)
251
252 cont_lineas2:
253     mov al, [esi]
254
255     cmp eax, 10 ;compara buscando caracter de salto de linea '\r' '\n' con \n = 10
256     decimal A hex
257     jne n_linea_nueva2
258 ;Se llego al punto en el que se escribiria la nueva linea
259     inc ecx
260     cmp edi, ecx
261     je escritura
262
263 n_linea_nueva2:
264     cmp eax, 0 ; compara con el caracter de fin de archivo, end buffer...
265     je f_lineas2
266
267     inc esi
268     jmp cont_lineas2
269
270 f_lineas2:
271     jmp fin
272
273 escritura:
274 ; eax tiene un caracter
275 ;esi direccion del hmem+cant caracteres sirve
276 ; edi tiene el numero de linea a donde va
277 ;ecx tiene el contador de cuantos lineas van
278     inc esi
279     push esi ; guardamos la direccion de hmem+cant_caracteres

```

```

276
277     mov ecx, hMem; direccion inicial del archivo
278     sub esi, ecx ;en esi se tiene cuantos caracteres hay
279
280     mov edi, offset Buffer_div_size
281     mov [edi] , esi ; luego el valor a la direccion
282
283     invoke WriteFile,hFileWrite,hMem,Buffer_div_size,ADDR bytewr,NULL;escritura del
284     archivo primera parte
285
286     invoke WriteFile,hFileWrite,offset Aux_string,BufferSize,ADDR bytewr,NULL ;
287     escritura del nuevo registro
288
289     Invoke WriteFile,hFileWrite,offset CRLF,2,ADDR bytewr,NULL;escritura del salto de
290     linea
291
292     pop eax ; guardamos en eax la direccion de donde me quede en el archivo
293     mov hMem_div,eax
294     mov ecx, FileSize
295     sub ecx, esi ; en ecx quedan cuantos caracteres faltan
296     mov [edi] , ecx ; luego el valor a la direccion
297     invoke WriteFile,hFileWrite,hMem_div,Buffer_div_size,ADDR bytewr,NULL;escritura del
298     archivo
299
300 fin:
301
302 ;*****
303 ;
304 ;*****
305
306 ;se cierran los ficheros
307 invoke CloseHandle,hFileWrite
308
309 ;se libera la memoria dinamica
310 invoke GlobalFree,hMem
311 ;espera enter para salir y poder leer
312
313 Print_Text CRLF ;salto de linea
314 Get_Input Msg4,inbuf ;mensaje de salida
315
316 ;sale del programa
317 Invoke ExitProcess,0
318
319 End Start

```

## Funcionamiento del programa

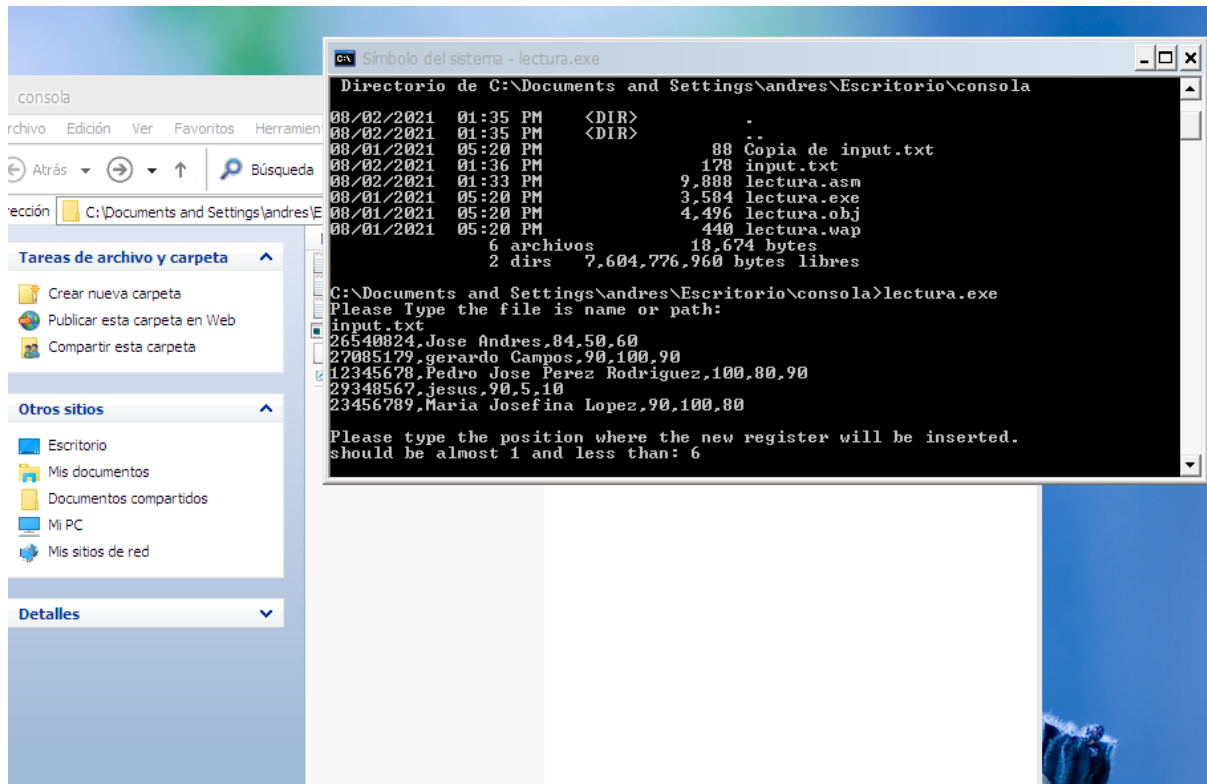


Figura 1: Abriendo el archivo

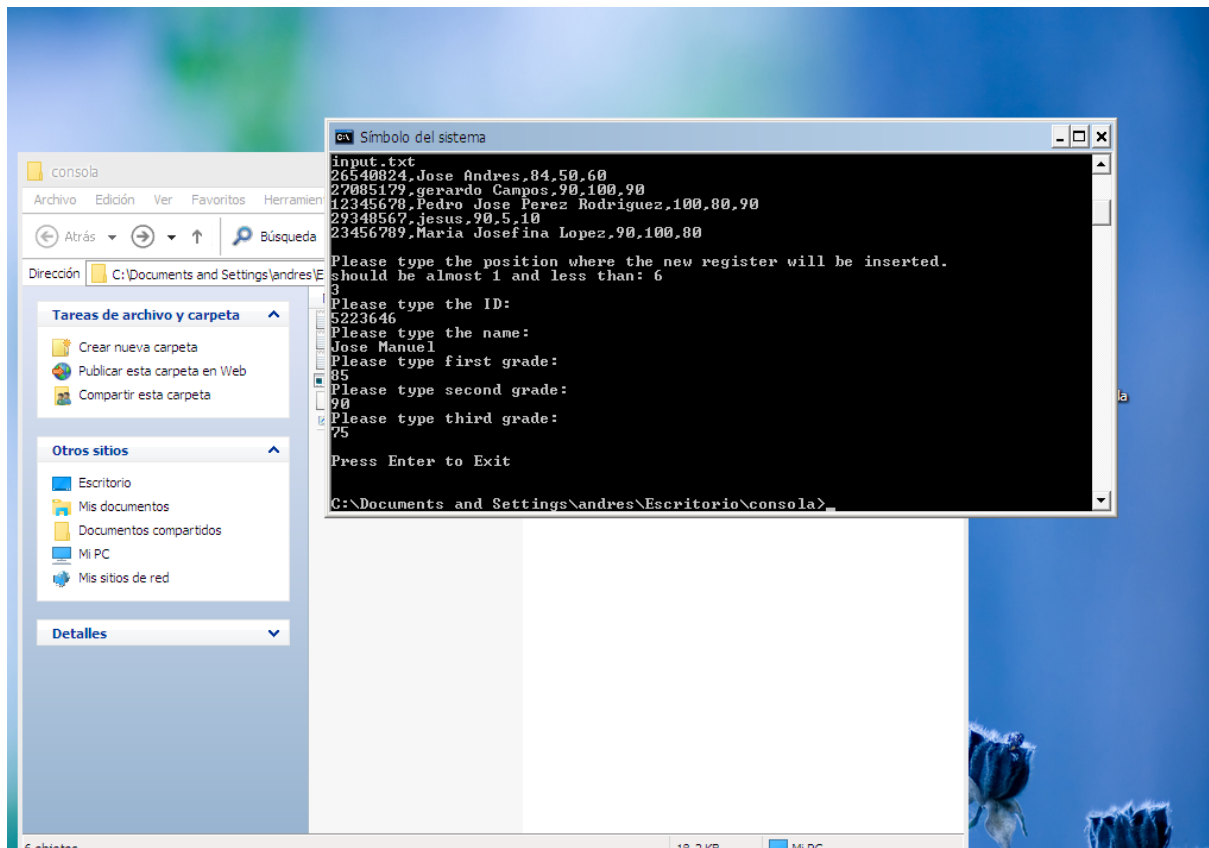


Figura 2: Agregando un nuevo registro

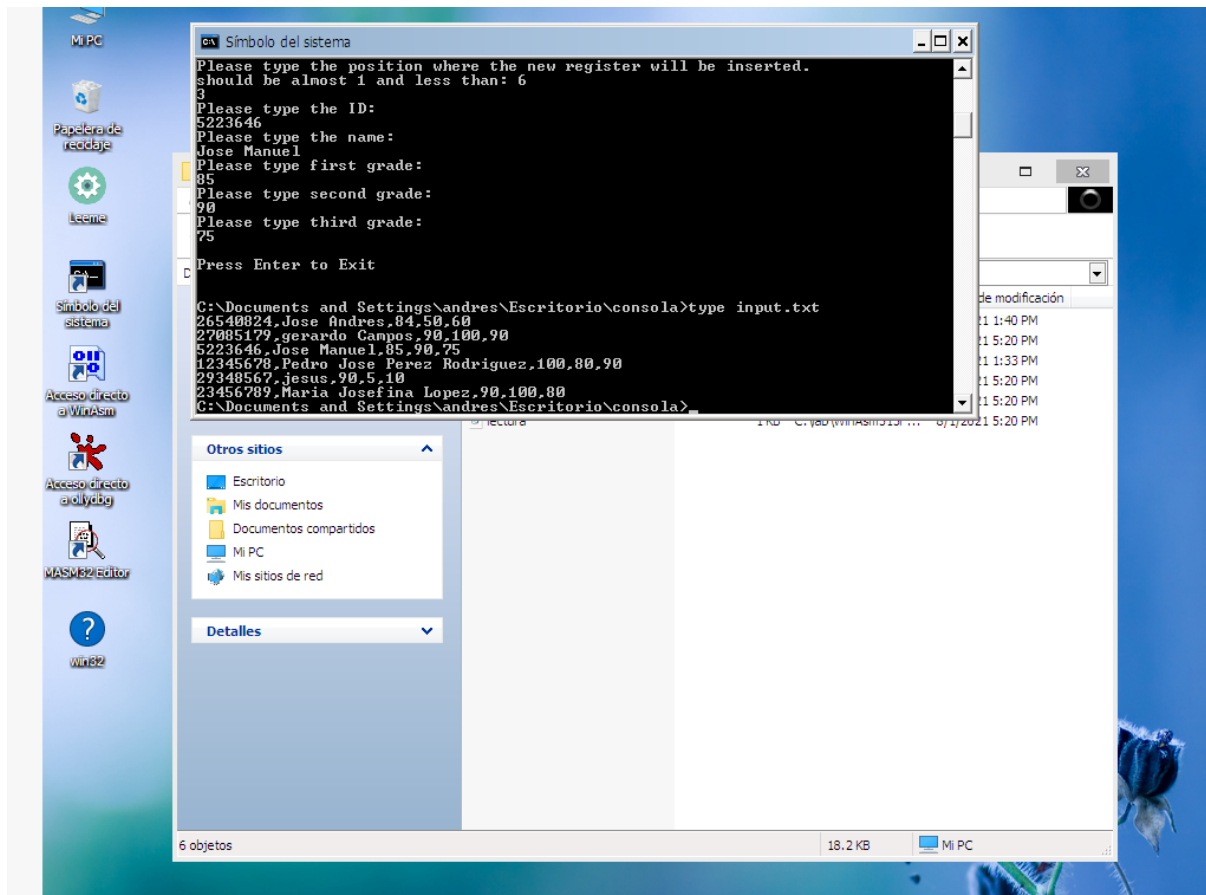


Figura 3: Mostrando en pantalla el archivo resultante

## Referencias Bibliográficas

- Iczelion. (s.f.). <http://www.movsd.com/icz.htm>
- Irvine, K. R. (2002). Assembly Language for Intel Assembly Language for Intel-Based Computers, 4th Edition Edition [fecha de consulta: 19/7/2021]. [https://www.csie.ntu.edu.tw/~acpang/course/asm\\_2004/slides/chapt\\_08Solve.pdf](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>