1. Write a program that keeps reading filenames from the keyboard until the word ‘end’ is introduced. From all the filenames read, identify the file that contains the maximum number of capital letters and print the filename and its content on the screen. The files can contain any number of characters. If there are more files with the same maximum number of capital letters, then the first such file (from the order in which they are read) must be chosen.

The program must contain two additional modules with the following functions:

* one function that gets a filename as a parameter and computes the number of capital letters from that file
* one function that gets a filename as a parameter and prints its name and content on the screen.

Note: The solution below requires improvements.

|  |  |  |
| --- | --- | --- |
| bits 32  global start  needed by our programe  extern exit  import exit msvcrt.dll  extern scanf  import scanf msvcrt.dll  extern compute, printResult  segment data use32 class=data  filename times 20 db 0  nrCapitalLett dd 0  filenameMax times 20 db 0  nrMax dd 0  final db 'end',0  format db '%s',0    segment code use32 class=code  start:  readFileNames:  push dword filename  push dword format  call [scanf]  add esp,4\*2    mov esi, filename  mov edi, final  compare:  cmp byte [esi],0  je check2 ;the first string is finished    cmpsb  je compare  jne processFile ;if they are not equal, then we have another file to process  check2:  cmp byte [edi],0  je done ;it means we read 'end' and the strings are equal    processFile:  push dword filename  push dword nrCapitalLett  call compute  add esp,4\*2    mov eax,[nrCapitalLett]  cmp eax, [nrMax]  jbe next  mov [nrMax],eax    ;we must also keep the name of the file with the max nr  mov esi, filename  mov edi, filenameMax  copy:  movsb  cmp byte [esi],0  jne copy  ;if we get here we are done copying (we reached 0), but we have to place the 0 in the destination as well  mov byte [edi],0    next:  jmp readFileNames    done:  ;call the function to print filename and content  push dword filenameMax  call printResult  add esp,4    push dword 0  call [exit] | bits 32  global compute  extern fopen,fread,fclose  import fopen msvcrt.dll  import fclose msvcrt.dll  import fread msvcrt.dll  segment data use32 class=data  readmode db 'r',0  buffer db 0  descriptor dd -1    segment code  compute:  ;esp+4 - nr  ;esp+8 - filename  mov eax,[esp+8]  mov ebx,0 ;for the count of capital letters  push dword readmode  push eax  call [fopen]  add esp,4\*2  mov [descriptor], eax  cmp eax, 0  je done2  reading:  ;int fread(void \* str, int size, int count, FILE \* stream)  push dword [descriptor]  push dword 1  push dword 1  push dword buffer  call [fread]  add esp, 4\*4  cmp eax, 0  je done2  cmp byte [buffer], 'A'  jb reading ;continue reading from the file  cmp byte [buffer], 'Z'  ja reading  inc ebx  jmp reading  done2:  mov eax, [esp+4] ;address for the count  mov [eax], ebx ;set the actual count  push dword [descriptor]  call [fclose]  add esp,4  ret | bits 32  global printResult  extern printf  import printf msvcrt.dll  extern fopen,fread,fclose  import fopen msvcrt.dll  import fclose msvcrt.dll  import fread msvcrt.dll  segment data use32 class=data  readmode db 'r',0  buffer times 101 db 0  descriptor dd -1  newLine db 10,13,0    segment code  printResult:  mov eax, [esp+4] ;address of filename  push dword eax  call [printf]  add esp,4  push dword newLine  call [printf]  add esp,4  mov eax, [esp+4] ;address of filename  push dword readmode  push dword eax  call [fopen]  add esp,4\*2  mov [descriptor], eax  cmp eax, 0  je final2  printFile:  ;int fread(void \* str, int size, int count, FILE \* stream)  push dword [descriptor]  push dword 100  push dword 1  push dword buffer  call [fread]  add esp, 4\*4  cmp eax,0  je closeFile  mov byte [buffer+eax],0 ;e important pentru cazul in care la final citim mai putin de 100 de carac si trebuie pus 0 pt afisare  push dword buffer  call [printf]  add esp,4  jmp printFile    closeFile:  push dword [descriptor]  call [fclose]  add esp,4  final2:  ret |

1. Read the name of a file from keyboard. Read all the decimal numbers (represented on doublewords) from the given file and print on the screen only the numbers for which the following property is true:
   * the low byte of the high word is a prime number

|  |  |
| --- | --- |
| bits 32  global start  extern exit  import exit msvcrt.dll  extern fopen,fclose,fscanf,printf ,scanf  import fopen msvcrt.dll  import fclose msvcrt.dll  import fscanf msvcrt.dll  import printf msvcrt.dll  import scanf msvcrt.dll    extern checkPrime  segment data use32 class=data  filename times 20 db 0  descriptor dd -1  nr dd 0  readmode db 'r',0  format db '%d',0  format2 db '%d',10,13,0  format3 db '%s',0  segment code use32 class=code  start:  push dword filename  push dword format3  call [scanf]  add esp,4\*2    push dword readmode  push dword filename  call [fopen]  add esp,4\*2    mov [descriptor], eax  cmp eax,0  je final    readFromFile:  ;int fscanf ( FILE \* stream, const char \* format, ... );  push dword nr  push dword format  push dword [descriptor]  call [fscanf]  add esp,4\*3  cmp eax, 0  **jle closeFile ;**we are done reading numbers    mov eax, [nr]  shr eax,16 ;ax = high word  mov ebx,0  mov bl, al ;ebx contains low byte from high word    push ebx  push dword 0 ;for the result: 0 = not prime, 1 = prime  call checkPrime    pop eax ;store the result of the prime function in eax  pop ebx ; restore the number to be printed (maybe ebx was used in the fct)    cmp eax, 0  je next    ;print prime nr on the screen  push dword [nr]  push dword format2  call [printf]  add esp,4\*2    next:  jmp readFromFile    closeFile:  push dword [descriptor]  call [fclose]  add esp,4    final:  push dword 0  call [exit] | bits 32  global checkPrime  segment code use32 class=code  checkPrime:  ;esp+4 - for the result  mov eax, [esp+8] ;it is actually byte  cmp eax,2  je prime  mov bl,2  dividing:  div bl  cmp ah, 0  je notPrime    mov eax, [esp+8] ;restore ax  inc bl  cmp bl,al  je prime  jmp dividing  notPrime:  mov dword [esp+4],0  jmp endFct  prime:  mov dword [esp+4],1  endFct:  ret |