ĆWICZENIE NR 1

Zadanie: napisać w assemblerze program który w zależności od ilości podanych parametrów:
a) otworzy plik o nazwie [parametr1] i wyświetli jego zawartość w oknie konsoli
Przykład:
```
c:\program1.exe parametr1
```
b) utworzy plik o nazwie [parametr1] i zapisze do niego zawartość [parametr2]
Przykład:
```
c:\program1.exe parametr1 parametr2
```
c) wyświetli komunikat o nieprawidłowej liczbie parametrów
Przykład:
```
c:\program1.exe parametr1 parametr2 parametr3
c:\program1.exe
```

Przykład 1: wyświetlenie zawartości pliku w oknie konsoli.
Na początku przydziela się programowi okno konsoli.
Przykład:
```
invoke AllocConsole
```
Kolejnym krokiem jest pobranie uchwytu, w którym ustawiono standardowe wyjście na konsolę. Zapisujemy uchwyt do zmiennej. Parametry przekazujemy po przecinku zgodnie z zasadą: od lewej do prawej lub od góry w dół. Większość funkcji Windows API zwraca wartość poprzez rejestr `eax`.
Przykład:
```
invoke GetStdHandle, STD_OUTPUT_HANDLE
mov hInstance, eax
```

Funkcja `CreateFile`, w zależności od podanych parametrów, otwiera plik. Funkcja ta przyjmuje parametry: nazwę pliku, tryb otwarcia pliku, sposób współdzielenia pliku, atrybuty bezpieczeństwa, tryb tworzenia pliku, dodatkowe flagi, dodatkowe atrybuty tworzonego pliku. W tym przypadku najważniejsze parametry to nazwa pliku, którą przechowujemy w zmiennej `fName`, tryb otwarcia pliku ustawiony na odczyt, oraz tryb tworzenia pliku ustawiony na otwarcie istniejącego.
Przykład:
```
invoke CreateFile,ADDR fName, GENERIC_READ, 0, NULL,
OPEN_EXISTING,FILE_ATTRIBUTE_ARCHIVE,0
mov hFile, eax
```

`lstrlen` zwraca długość napisu zakończonego znakiem NULL. Wartość ta jest przekazana do `WriteConsole`. Funkcja ta wyświetla napis w konsoli. Parametry to: uchwyt okna konsoli, wskaźnik na zmienną z tekstem, długość tekstu, wskaźnik na zmienną do której funkcja zapisuje ilość wyświetlonych znaków. Ostatni parametr jest zarezerwowany i zawsze wynosi NULL. Funkcja wyświetla wyłącznie tekst zakodowany w formacie ANSI. Dla formatu utf16 używa się funkcji `WriteConsoleW`.

Przykład:
```
invoke lstrlen, OFFSET txt3
invoke WriteConsole, hInstance,OFFSET txt3, eax,ADDR tmpV, NULL
```

`GetFileSize` zwraca rozmiar otwartego pliku. Pierwszy parametr jest uchwytem do otwartego pliku. Drugi zawsze wynosi zero.
Przykład:
```
invoke GetFileSize,hFile,0
mov fSize, eax
```

Znając rozmiar pliku należy zaalokować bufor, do którego plik będzie odczytany. Pierwszy parametr to miejsce alokacji. NULL oznacza wybór miejsca przez system operacyjny. `eax` przekazuje rozmiar bufora. `MEM_COMMIT` inicjalizuje pamięć jako zero. Ostatni parameter to typ ochrony pamięci. Zwracany jest wskaźnik na bufor zaalokowanej pamięci.
Przykład:
```
invoke VirtualAlloc,0,eax,MEM_COMMIT,PAGE_READWRITE
```

`ReadFile` odczytuje plik. Pierwszy parametr do uchwyt do otwartego pliku, następnie wskaźnik na bufor do którego odczytujemy, ilość bajtów do odczytania, zmienna przechowująca ilość bajtów wczytanych, wskaźnik na strukturę współdzielenia lub NULL.
Przykład:
```
invoke ReadFile, hFile, eax, fSize, OFFSET readBytes, 0
```

Zwolnienie bufora pamięci oraz zamknięcie pliku.
Przykład:
```
invoke VirtualFree,eax,OFFSET fSize,MEM_RELEASE
invoke CloseHandle, hFile
```

Zwolnienie konsoli oraz wyjście z procesu.
Przykład:
```
invoke FreeConsole
invoke ExitProcess,0
```

Kod przykładu:

```
include \masm32\include\masm32rt.inc

.data?
hInstance dd ?
hFile dd ?
tmpV dd ?
fSize dd ?
hBuff dd ?
readBytes dd ?

argc dd ?
argv dd ?
.data

txt2 db "Blad otwarcia pliku.", NULL
txt3 db "Blad odczytu pliku.", NULL
txt4 db "Blad alokacji pamieci.", NULL
fName db "plik.txt"

.code
```

```
Start:
invoke AllocConsole
invoke GetStdHandle, STD_OUTPUT_HANDLE
mov hInstance, eax


invoke CreateFile,ADDR fName, GENERIC_READ, 0, NULL,
OPEN_EXISTING,FILE_ATTRIBUTE_ARCHIVE,0
mov hFile, eax

    .if eax==INVALID_HANDLE_VALUE
        invoke lstrlen, OFFSET txt2
        invoke WriteConsole, hInstance, OFFSET txt2, eax,ADDR tmpV, NULL
        jmp @end
    .endif

    invoke GetFileSize,hFile,0
    mov fSize, eax

    invoke VirtualAlloc,0,eax,MEM_COMMIT,PAGE_READWRITE
    .if eax == NULL
        invoke lstrlen, OFFSET txt4
        invoke WriteConsole, hInstance,OFFSET txt4, eax,OFFSET tmpV, NULL
    .else
        mov hBuff, eax
        invoke ReadFile, hFile, eax, fSize, OFFSET readBytes, 0

        .if eax == -1
            invoke lstrlen, OFFSET txt3
            invoke WriteConsole, hInstance,OFFSET txt3, eax,ADDR tmpV, NULL
        .else
            invoke WriteConsoleW, hInstance, hBuff, fSize,OFFSET tmpV, NULL
        .endif
    .endif

    invoke VirtualFree,eax,OFFSET fSize,MEM_RELEASE
    invoke CloseHandle, hFile

@end:

invoke FreeConsole
invoke ExitProcess,0

END Start
```

Przykład 2: wyświetlenie parametrów w oknie konsoli.

W dostępie do parametrów pomagają dwie funkcje: `GetCommandLineW` oraz `CommandLineToArgvW`. Pierwsza z nich zwraca wskaźnik do łańcucha linii poleceń. Można po użyciu tej funkcji odpowiednio analizować ten łańcuch i wyodrębnić parametry. Jednak wskaźnik ten jest argumentem funkcji `CommandLineToArgvW`, która zwaca znane z języka C zmienne argc oraz argv. Dla przypomnienia argc przechowuje ilość elementów w tablicy argv, która zawiera wskaźniki na kolejne parametry.

Przykład:

```
invoke GetCommandLineW
invoke CommandLineToArgvW,eax,OFFSET argc
```

Dzięki zmiennym argc i argv w łatwy sposób możemy w pętli wyświetlić wszystkie argumenty programu. W pętli zwiększamy rejestr `edi` aby poruszać się po elementach tablicy argv. Elementy tablicy argv wskazują na łańcuchy kolejnych parametrów które możemy wyświetlić używając `WriteConsoleW`.

Przykład:

```
mov argv,eax
mov edi, argv
xor ecx, ecx

@p1:
   mov eax,[edi]
   mov ebx, eax
   add edi, 4

   push ecx
   INVOKE lstrlenW, eax
   INVOKE WriteConsoleW, hInstance, ebx, eax,OFFSET tmpV, NULL
   INVOKE WriteConsole, hInstance, OFFSET cr, 2,OFFSET tmpV, NULL
   pop ecx
   inc ecx

cmp ecx, argc
jne @p1
```

Kod przykładu:

```
include \masm32\include\masm32rt.inc
.data?
hInstance dd ?
tmpV dd ?
argc dd ?
argv dd ?
.data
cr db 10,13,NULL

.code
Start:
INVOKE AllocConsole
INVOKE GetStdHandle, STD_OUTPUT_HANDLE
mov hInstance, eax

INVOKE GetCommandLineW
INVOKE CommandLineToArgvW,eax,OFFSET argc
mov argv,eax
mov edi, argv
xor ecx, ecx

@p1:
   mov eax,[edi]
   mov ebx, eax
   add edi, 4

   push ecx
   INVOKE lstrlenW, eax
   INVOKE WriteConsoleW, hInstance, ebx, eax,OFFSET tmpV, NULL
   INVOKE WriteConsole, hInstance, OFFSET cr, 2,OFFSET tmpV, NULL
   pop ecx
   inc ecx

cmp ecx, argc
jne @p1

INVOKE FreeConsole
INVOKE ExitProcess,0

END Start
```