

**PRYWATNA WYŻSZA SZKOŁA NAUK  
SPOŁECZNYCH, KOMPUTEROWYCH I MEDYCZNYCH**

---

**WYDZIAŁ NAUK SPOŁECZNYCH I  
TECHNIK KOMPUTEROWYCH**

**Ćwiczenie  
z programowania niskopoziomowego**

**„Operacje na plikach i katalogach”**

Wariant N 8

**Opracował**

Grzegorz Makowski

III rok Informatyki  
Studia niestacjonarne

Prowadzący  
Prof. dr hab. inż. Aleksandr Timofiejew

Warszawa 2019/2020

## Spis treści

Zadanie a	3
Tworzenie plików i katalogów.	3
Opracowanie zadania	3
Testowanie	3
Zadanie b	6
Operacje na plikach	6
Opracowanie zadania	6
Testowanie	15

## Zadanie a

### Tworzenie plików i katalogów.

Opracować program, który:

1. Tworzy katalog „DANE”.
2. Tworzy w tym katalogu plik z nazwą „test” z atrybutem „do czytania i zapisu”.
3. Zapisuje do pliku 100 liczb całkowitych z zakresu od -99 do +99 wygenerowanych losowo.
4. Wyświetla liczby po 10 w jednym wierszu.
5. Zamyka plik.

### Opracowanie zadania

;Kod programu w zadaniu B.

### Testowanie

```
D:\workspace\Programowanie\rok3\asm>cw7\cw7
```

```
Autor aplikacji Grzegorz Makowski i53
```

#### Zadanie a

```
Ścieżki do katalogu DANE i pliku test.txt.
```




```
D:\workspace\Programowanie\rok3\asm\DANE
```

```
D:\workspace\Programowanie\rok3\asm\DANE\test.txt
```

```
-65 -58 -25 -37 -43 -36 74 70 -83 4
-50 62 67 25 -53 59 89 -70 99 73
26 5 -2 -96 -60 -96 -73 1 31 0
98 -68 -83 20 -82 -3 -81 -77 -53 -52
13 14 -49 -27 -55 79 59 76 -50 29
20 -99 83 -84 62 53 31 -36 -89 -94
-68 30 -55 -46 24 16 9 52 61 21
49 -46 17 50 -59 -13 -30 69 28 -76
-84 -7 4 -30 -26 -69 -39 19 -16 -77
9 45 4 80 -2 -56 -3 94 -23 -83
```

Widok

» USER (D:) » workspace » Programowanie » rok3 » asm » DANE

Nazwa	Data modyfikacji	Typ	Rozmiar
 plik1.txt	29.04.2020 21:36	Dokument tekstowy	1 KB
 plik2.txt	29.04.2020 21:36	Dokument tekstowy	1 KB
 test.txt	29.04.2020 21:36	Dokument tekstowy	1 KB

Zawartość pliku test.txt:

-65  
-58  
-25  
-37  
-43  
-36  
74  
70  
-83  
4  
-50  
62  
67  
25  
-53  
59  
89  
-70  
99  
73  
26  
5  
-2  
-96  
-60  
-96  
-73  
1  
31  
0  
98  
-68  
-83  
20  
-82  
-3  
-81  
-77  
-53  
-52  
13  
14  
-49  
-27  
-55  
79

59  
76  
-50  
29  
20  
-99  
83  
-84  
62  
53  
31  
-36  
-89  
-94  
-68  
30  
-55  
-46  
24  
16  
9  
52  
61  
21  
49  
-46  
17  
50  
-59  
-13  
-30  
69  
28  
-76  
-84  
-7  
4  
-30  
-26  
-69  
-39  
19  
-16  
-77  
9  
45  
4  
80  
-2  
-56  
-3  
94  
-23

## Zadanie b

### Operacje na plikach

Opracować program, który:

1. Otworzy z atrybutem „tylko czytanie” plik „test” z katalogu „DANE” stworzony w punkcie a).
2. Tworzy w katalogu „DANE” pliki z nazwami „plik1” i „plik2” z atrybutem „do czytania i zapisu”
3. Przypisuje z pliku „test” do plików „plik1” i „plik2” dane według reguły „Co 8 parzysta i nieparzysta.”
4. Wyświetla liczby z plików „plik1” i „plik2” po 10 w jednym wierszu.
5. Zamyka wszystkie pliki.

### Opracowanie zadania

;plik cw7.asm

```
;-----|
;   cw7.asm      |
;   Operacje na plikach i katalogach.  |
;-----|
;   Autor: Grzegorz Makowski |
;   MASM ver: 6.14.8444      |
;   ost. akt. 29.04.2020     |
;-----|
.586p
.model flat, stdcall
;-----|
;   wczytanie plikow zewnetrznych      |
;-----|
;   wczytanie własnych makr z pliku     |
;-----|
include mojemakra.mac ; Makra
include cw7.mac       ; Makra dedykowane do zad 7
;-----|
;   biblioteki systemowe i masm        |
;-----|
includelib .\lib\user32.lib
includelib .\lib\kernel32.lib
includelib .\lib\masm32.lib
;-----|
;   stale z pliku .\include\windows.inc |
;-----|
STD_INPUT_HANDLE equ -10
STD_OUTPUT_HANDLE equ -11
;-----|
;   stale do obsługi plikow            |
;-----|
GENERIC_READ equ 80000000h
GENERIC_WRITE equ 40000000h
CREATE_NEW equ 1
CREATE_ALWAYS equ 2
OPEN_EXISTING equ 3
OPEN_ALWAYS equ 4
TRUNCATE_EXISTING equ 5
FILE_FLAG_WRITE_THROUGH equ 80000000h
FILE_FLAG_OVERLAPPED equ 40000000h
FILE_FLAG_NO_BUFFERING equ 20000000h
FILE_FLAG_RANDOM_ACCESS equ 10000000h
FILE_FLAG_SEQUENTIAL_SCAN equ 8000000h
FILE_FLAG_DELETE_ON_CLOSE equ 4000000h
FILE_FLAG_BACKUP_SEMANTICS equ 2000000h
FILE_FLAG_POSIX_SEMANTICS equ 1000000h
FILE_ATTRIBUTE_READONLY equ 1h
FILE_ATTRIBUTE_HIDDEN equ 2h
FILE_ATTRIBUTE_SYSTEM equ 4h
FILE_ATTRIBUTE_DIRECTORY equ 10h
FILE_ATTRIBUTE_ARCHIVE equ 20h
FILE_ATTRIBUTE_NORMAL equ 80h
```

```

FILE_ATTRIBUTE_TEMPORARY equ 100h
FILE_ATTRIBUTE_COMPRESSED equ 800h
FORMAT_MESSAGE_ALLOCATE_BUFFER equ 100h
FORMAT_MESSAGE_IGNORE_INSERTS equ 200h
FORMAT_MESSAGE_FROM_STRING equ 400h
FORMAT_MESSAGE_FROM_HMODULE equ 800h
FORMAT_MESSAGE_FROM_SYSTEM equ 1000h
FORMAT_MESSAGE_ARGUMENT_ARRAY equ 2000h
FORMAT_MESSAGE_MAX_WIDTH_MASK equ 0FFh

;-----|
;      stale      |
;-----|
mbuf = 512
;--- funkcje API Win32 z pliku .\include\user32.inc ---
CharToOemA proto :dword,:dword
;--- z pliku .\include\kernel32.inc ---
GetStdHandle proto :dword
ReadConsoleA proto :dword,:dword,:dword,:dword,:dword
WriteConsoleA proto :dword,:dword,:dword,:dword,:dword
ExitProcess proto :dword
wsprintfA proto c :vararg
;; int wsprintf(LPTSTR lpOut, // pointer to buffer for output
;; LPCTSTR lpFmt, // pointer to format-control string
;; ... // optional arguments );
lstrlenA proto :dword
GetCurrentDirectoryA proto :dword,:dword
;; nBufferLength, lpBuffer; zwraca length
CreateDirectoryA proto :dword,:dword
;; lpPathName, lpSecurityAttributes; zwraca 0 jeśli bład
lstrcatA proto :dword,:dword
;; lpString1, lpString2; zwraca lpString1
CreateFileA proto :dword,:dword,:dword,:dword,:dword,:dword,:dword
;; LPCTSTR lp.szName, DWORD fdwAccess,
;; DWORD fdwShareMode, LPSECURITY_ATTRIBUTES lpSa, DWORD fdwCreate,
;; DWORD fdwAttrsAndFlags, HANDLE hTemplateFile
lstrcpyA proto :dword,:dword
;; LPTSTR lpString1 // address of buffer, LPCTSTR lpString2 // address of string to copy
CloseHandle proto :dword
;; BOOL CloseHandle(HANDLE hObject)
WriteFile proto :dword,:dword,:dword,:dword,:dword
;; BOOL WriteFile(
;; HANDLE hFile, // handle to file to write to
;; LPCVOID lpBuffer, // pointer to data to write to file
;; DWORD nNumberOfBytesToWrite, // number of bytes to write
;; LPDWORD lpNumberOfBytesWritten, // pointer to number of bytes written
;; LPOVERLAPPED lpOverlapped // pointer to structure needed for overlapped I/O
;;);
ReadFile proto :dword,:dword,:dword,:dword,:dword
;; BOOL ReadFile(
;; HANDLE hFile, // handle of file to read
;; LPVOID lpBuffer, // address of buffer that receives data
;; DWORD nNumberOfBytesToRead, // number of bytes to read
;; LPDWORD lpNumberOfBytesRead, // address of number of bytes read
;; LPOVERLAPPED lpOverlapped // address of structure for data
;;);
CopyFileA proto :dword,:dword,:dword
;; BOOL CopyFile(
;; LPCTSTR lpExistingFileName, // pointer to name of an existing file
;; LPCTSTR lpNewFileName, // pointer to filename to copy to
;; BOOL bFailIfExists // flag for operation if file exists
;;);
GetLastError proto
;--- z pliku .\include\masm32.inc ---
nrandom proto :dword
;--- funkcje
ScanInt proto C adres:dword

;-----|
; Początek segmentu danych      |
;-----|
_data segment
hout dd 0
nl db 0Dh, 0Ah, 0; nowa linia
nl2 db 0Dh,0Ah,20h,0; nowa inne formatowanie
nxt db 13,10,0; następny wiersz
naglow db "Autor aplikacji Grzegorz Makowski i53",0

```

```

align 4 ; przesuniecie do adresu podzielonego na 4
rozmN dd $ - naglow ;ilosc znaków w tablicy
zadanieA db "Zadanie a",0
align 4
rozmA dd $ - zadanieA ; ilosc znakow tekstu zadanieA
opisKatZadA db "Ściezki do katalogu DANE i pliku test.txt",0
align 4
rozmkatzadA dd $ - opisKatZadA ; ilosc znakow w opisie zadania a
align 4
opisKatZadA2 db "Wyświetlenie losowej zawrtości pliku test",0
align 4
rozmkatzadA2 dd $ - opisKatZadA2 ; ilosc znakow w opisie zadania a
zadanieB db "Zadanie b",0
align 4
rozmbB dd $ - zadanieB ; ilosc znakow tekstu zadanieB
align 4
opisKatZadB db "Ściezki do plików: plik1.txt i plik2.txt",0
align 4
rozmkatzadB dd $ - opisKatZadB ; ilosc znakow w opisie zadania a
align 4
rout dd 0
sciezka db mbuf dup(?)
nazwaDANE db "\DANE",0
nazwa db "test.txt",0
nazwa1 db "\plik1.txt",0
nazwa2 db "\plik2.txt",0
nazwat1 db 13,10,"Dane z plik1.txt - co 8 nieparzysta:",13,10,0
nazwat2 db 13,10,"Dane z plik2.txt - co 8 parzysta:",13,10,0
tesTxt db mbuf dup(?) ; bufor na sciezke dla pliku test.txt
tesTxt1 db mbuf dup(?) ; bufor na sciezke dla pliku test1.txt
tesTxt2 db mbuf dup(?) ; bufor na sciezke dla pliku test2.txt
katDane db mbuf dup(?) ; bufor katalogu
hfile dd 0 ; uchwyt do pliku test
hfile1 dd 0 ; uchwyt do pliku test1
hfile2 dd 0 ; uchwyt do pliku test2
tab dd 100 dup(0)
nbytes dd 0
liczba dd 0
licznik1 dd 0
licznik2 dd 0
bufor db mbuf dup(0)
leng dd 0
buf db mbuf dup(0) ; bufor pomocniczy
buff db mbuf dup(0) ; bufor pomocniczy
format1 db " %3ld",13,10,0
format2 db " %3ld",0
rsymb dd 0
_data ends

;-----|
; Koniec segmentu danych |
;-----|

;-----|
; Poczatek segmentu kodu |
;-----|

_text segment
start:
;--- nagłówek -----
Naglowek ; Makro
;---
zadA ; Makro
nowalinia nl, 2 ; MAKRO
invoke GetCurrentDirectoryA, mbuf, offset sciezka ; pobranie pełnej ścieżki
invoke lstrcpYA, offset katDane, offset sciezka ; łączenie stringow
invoke lstrcatA, offset katDane, offset nazwaDANE
invoke lstrlenA, offset katDane
mov leng, eax
wyswietl offset katDane, leng ; wyświetlenie pełnego katalogu DANE - MAKRO
;--- nowa linia
nowalinia nl, 2 ; nowa linia - MAKRO
;---
invoke CreateDirectoryA, offset katDane, 0 ; utworzenie katalogu
invoke lstrcpYA, offset tesTxt, offset katDane
invoke lstrcatA, offset tesTxt, offset nazwa
invoke lstrlenA, offset tesTxt
mov leng, eax

```



```

wyswietl offset tesTxt, leng ; MAKRO - wyswietlenie sciezki do pliku test.txt
;-- nowa linia
nowalinia nl, 2 ; nowa linia - MAKRO
invoke CreateFileA, offset tesTxt, GENERIC_WRITE , 0, 0, CREATE_ALWAYS, 0, 0 ; tworzenie pliku
mov hfile, eax
;--
invoke CloseHandle, hfile ; zamkniecie pliku test
;-- liczby pseudolosowe -> tablica
lea ebx, tab
mov edi, 0
mov ecx, 100
losowe:
push ecx
push ebx
...
invoke nrandom, 200
sub eax, 99
...
pop ebx
mov dword ptr [ebx], eax
add ebx, 4
pop ecx
loop losowe
;-----|
;   plik "test.txt"   |
;-----|
;-- nowa linia
nowalinia nxt, 2 ; nowa linia
invoke CreateFileA, offset tesTxt, GENERIC_READ OR GENERIC_WRITE , 0, 0, OPEN_EXISTING, 0, 0 ; tworzenie pliku
mov hfile, eax
;-- z tablicy do pliku ----
lea ebx, tab
mov ecx, 100
powt:
push ecx
push ebx
invoke sprintfA, offset buf, offset format1, DWORD PTR [ebx]
mov rsymb, eax
invoke WriteFile, hfile, offset buf, rsymb, offset nbytes, 0
pop ebx
add ebx, 4
pop ecx
loop powt
...
invoke CloseHandle, hfile
;-- pobranie z tablicy na ekran po 10 liczb na wierszu ----
lea ebx, tab
mov ecx, 100
mov licznik1, 0
powtE:
push ecx ;
push ebx
...
mov eax, licznik1
or eax, eax
jnz @F
;-- new line -----
nowalinia nl, 2 ; MAKRO
@@:
inc licznik1
cmp licznik1, 10
jb @F
mov licznik1, 0
@@:
pop ebx
push ebx
invoke sprintfA, offset buf, offset format2, DWORD PTR [ebx]
mov rsymb, eax
invoke WriteConsoleA, hout, offset buf, rsymb, offset nbytes, 0
...
pop ebx
add ebx, 4
pop ecx
loop powtE
;-- new line -----
nowalinia nl, 2 ; MAKRO

```

```

;--- new line -----
nowalinia nl, 2 ; MAKRO
;-----|
; Operacje na plikach |
;-----|
zadB ; Makro
invoke CreateFileA, offset tesTxt, GENERIC_READ, 0, 0, OPEN_EXISTING, 0, 0 ; plik test.txt
mov hfile, eax
invoke lstrcpyA, offset tesTxt1, offset katDane
invoke lstrcatA, offset tesTxt1, offset nazwa1
invoke lstrlenA, offset tesTxt1
mov leng, eax
wyswietl offset tesTxt1, leng ; wyswietlenie
nowalinia nxt, 2 ; nowa linia
invoke CreateFileA, offset tesTxt1, GENERIC_WRITE, 0, 0, CREATE_ALWAYS, 0, 0 ; stworzenie pliku
mov hfile1, eax
invoke lstrcpyA, offset tesTxt2, offset katDane
invoke lstrcatA, offset tesTxt2, offset nazwa2
invoke lstrlenA, offset tesTxt2
mov leng, eax
wyswietl offset tesTxt2, leng
nowalinia nxt, 2 ; nowa linia
invoke CreateFileA, offset tesTxt2, GENERIC_WRITE, 0, 0, CREATE_ALWAYS, 0, 0 ; stworzenie pliku
mov hfile2, eax
;-----
mov ecx, 100
mov licznik1, 8 ; co osma parzysta
mov licznik2, 8 ; co osma nieparzysta
powt2:
push ecx
invoke ReadFile, hfile, offset buf, 6, offset nbytes, 0 ;
cmp nbytes, 0
jnz @F
jmp zamyk
@@:
invoke ScanInt, offset buf ; tekst ASCII -> liczba
mov liczba, eax
mov eax, liczba
test eax, 1h
jz parz
;-- nieparzysta
dec licznik2
cmp licznik2, 0
je @F
jmp dalej
@@:
mov licznik2, 8
invoke sprintfA, offset buf, offset format1, liczba
mov rsymb, eax
invoke WriteFile, hfile1, offset buf, rsymb, offset nbytes, 0
jmp dalej
parz:
;-- parzysta
dec licznik1
cmp licznik1, 0
je @F
jmp dalej
@@:
mov licznik1, 8
invoke sprintfA, offset buf, offset format1, liczba
mov rsymb, eax
invoke WriteFile, hfile2, offset buf, rsymb, offset nbytes, 0
jmp dalej
dalej:
pop ecx
loop @F
jmp zamyk
@@:
jmp powt2
zamyk:
invoke CloseHandle, hfile
invoke CloseHandle, hfile1
invoke CloseHandle, hfile2
;-----
;-----
invoke lstrlenA, offset nazwa1
mov leng, eax

```

```

invoke WriteConsoleA, hout, offset nazwat1, leng , offset rout , 0
;-----|
;   plik "plik1.txt"   |
;-----|
invoke CreateFileA, offset tesTxt1,GENERIC_READ, 0, 0, OPEN_EXISTING, 0, 0 ; plik1.txt
mov hfile1, eax
mov licznik1,0
powtE1:
invoke ReadFile, hfile1, offset buf ,6 , offset nbytes, 0 ;;
cmp nbytes,0
jnz @F
jmp zamyk1
@@:
cmp licznik1,0
jnz @F
;--- new line -----
nowalinia nl, 2 ; MAKRO
@@:
inc licznik1
cmp licznik1,10
jb @F
mov licznik1,0
@@:
invoke WriteConsoleA, hout, offset buf ,4, offset nbytes, 0
jmp powtE1
zamyk1:
invoke CloseHandle, hfile1
;--- new line -----
nowalinia nl, 2 ; MAKRO
;-----|
;   plik "plik2.txt"   |
;-----|
invoke CreateFileA, offset tesTxt2,GENERIC_READ, 0, 0, OPEN_EXISTING, 0, 0 ; plik2.txt
mov hfile1, eax
mov licznik1,0
powtE2:
invoke ReadFile, hfile2, offset buf ,6 , offset nbytes, 0 ;;
cmp nbytes,0
jnz @F
jmp zamyk2
@@:
cmp licznik1,0
jnz @F
;--- new line -----
nowalinia nl, 2 ; MAKRO
@@:
inc licznik1
cmp licznik1,10
jb @F
mov licznik1,0
@@:
invoke WriteConsoleA, hout, offset buf ,4, offset nbytes, 0
jmp powtE2
zamyk2:
invoke CloseHandle, hfile2
;--- new line -----
nowalinia nl, 2 ; MAKRO
;-----|
;   kon:   |
;-----|
;---- wywołanie funkcji ExitProcess -----
invoke ExitProcess,0
;=====
;=== Podprogramy =====
;=====
ScanInt PROC C adres
;; funkcja ScanInt przekształca ciąg cyfr do liczby, którą jest zwracana przez eax
;; argument - zakończony zerem wiersz z cyframi
;; rejestry: ebx - adres wiersza, EDX - znak liczby, ESI - indeks cyfry w wierszu, EDI - tymczasowy
;--- początek funkcji
;--- odkładanie na stos
push ebx
push ecx

```

```

push EDX
push ESI
push EDI
;--- przygotowywanie cyklu
invoke strlenA, adres
mov EDI, eax ;ilość znaków
mov ecx, eax ;ilość powtórzeń = ilość znaków
xor ESI, ESI ; wyzerowanie ESI
xor EDX, EDX ; wyzerowanie EDX
xor eax, eax ; wyzerowanie eax
mov ebx, adres
;--- cykl -----
pocz: cmp BYTE PTR [ebx+ESI], 02Dh ;porównanie z kodem '-'
jne @F
mov EDX, 1
jmp nast
@@: cmp BYTE PTR [ebx+ESI], 030h ;porównanie z kodem '0'
jae @F
jmp nast
@@: cmp BYTE PTR [ebx+ESI], 039h ;porównanie z kodem '9'
jbe @F
jmp nast
;---
@@: push EDX ; do EDX procesor może zapisać wynik mnożenia
mov EDI, 10
mul EDI ;mnożenie eax * EDI
mov EDI, eax ; tymczasowo z eax do EDI
xor eax, eax ;zerowani eax
mov AL, BYTE PTR [ebx+ESI]
sub AL, 030h ; korekta: cyfra = kod znaku - kod '0'
add eax, EDI ; dodanie cyfry
pop EDX
nast: inc ESI
dec ecx
jz @F
jmp pocz
;--- wynik
@@: or EDX, EDX ;analiza znacznika EDX
jz @F
neg eax
@@:
;--- zdejmowanie ze stosu
pop EDI
pop ESI
pop EDX
pop ecx
pop ebx
;--- powrót
ret
ScanInt ENDP
.....
_text ends
end start
;-----|
; Koniec segmentu kodu |
;-----|

```

;plik mojemakra.mac

```
;-----|
;   mojemakra.mac           |
;   Operacje na plikach i katalogach.   |
;   ver 1.0                 |
;   Autor: Grzegorz Makowski |
;   MASM ver: 6.14.8444     |
;   ost. akt. 28.04.2020   |
;-----|
;-----|
;   Makro do deskryptorow   |
;-----|
podajdeskr macro handle, deskrypt
push handle
call GetStdHandle
mov deskrypt, eax ; deskryptor bufora konsoli
endm
```

```
;-----|
;   Konwersja polskich znakow         |
;   Przyjmuje tab znakowa i bufor      |
;-----|
plznaki macro text, bufor
invoke CharToOemA, addr text, addr bufor
endm
```

```
;-----|
; Wyświetlanie wyniku- tekstu na ekranie |
; bufor z plznaki lub kazy inny tekst   |
;-----|
```

```
wyswietl macro bufor, rozmiar
push 0 ; rezerwa, musi być zero
push offset rout ; wskaźnik na faktyczną ilość wyprowadzonych znaków
push rozmiar ; ilość znaków
push offset bufor ; wskaźnik na tekst w buforze
push hout ; deskryptor buforu konsoli
call WriteConsoleA ; wywołanie funkcji WriteConsoleA
endm
```

```
;-----|
;   Makro do robienia odstepu linii   |
;przyjmuje kod hex nowej linni i ilosc zn. |
;-----|
nowalinia macro nowa, ilzkn
push 0 ; rezerwa, musi być zero
push offset rout ; wskaźnik na faktyczną ilość wyprowadzonych znaków
push ilzkn ; ilość znaków
push offset nowa ; wskaźnik na tekst
push hout ; deskryptor buforu konsoli
call WriteConsoleA ; wywołanie funkcji WriteConsoleA
endm
```

;plik cw7.mac

```
;-----|
;   cw7.mac      |
;   Operacje na plikach i katalogach. |
;   Makro do opisu zadań      |
;   ver 1.0      |
;   Autor: Grzegorz Makowski |
;   MASM ver: 6.14.8444      |
;   ost. akt. 29.04.2020      |
;-----|
;-----|
;   Wyświetlenie nagłówka      |
;-----|
```

Naglowek macro

```
;--- wywołanie funkcji GetStdHandle
podajdeskr STD_OUTPUT_HANDLE, hout
plznaki naglow, buf ; konwersja
;--- wyświetlenie powitania -----
wyswietl buf, rozmN ; wyświetlenie
nowalinia nl, 2 ; nowa linia
nowalinia nl, 2 ; nowa linia
endm
```

```
;-----|
;   Wyświetlenie zadania a      |
;-----|
```

zadA macro

```
;--- wyświetlenie tekstu Zadanie a ----
plznaki zadanieA, buff ;
wyswietl buff, rozmA ; wyświetlenie -
nowalinia nl, 2 ; nowa linia - MAKRO
nowalinia nl, 2 ; nowa linia - MAKRO
plznaki opisKatZadA, buf ; MAKRO
wyswietl buf, rozmkatzadA ; wyświetlenie opisu zadania
endm
```

```
;-----|
;   Wyświetlenie zadania b      |
;-----|
```

zadB macro

```
plznaki zadanieB, buff
wyswietl buff, rozmB
nowalinia nxt,2 ; nowa linia
nowalinia nl,2
plznaki opisKatZadB, buff
wyswietl buff, rozmkatzadB
nowalinia nxt,2 ; nowa linia
endm
```

## Testowanie

```
D:\workspace\Programowanie\rok3\asm>cw7\cw7
Autor aplikacji Grzegorz Makowski i53
```

### Zadanie a

Ścieżki do katalogu DANE i pliku test.txt.

```
D:\workspace\Programowanie\rok3\asm\DANE
```

```
D:\workspace\Programowanie\rok3\asm\DANE\test.txt
```

```
-65 -58 -25 -37 -43 -36 74 70 -83 4
-50 62 67 25 -53 59 89 -70 99 73
26 5 -2 -96 -60 -96 -73 1 31 0
98 -68 -83 20 -82 -3 -81 -77 -53 -52
13 14 -49 -27 -55 79 59 76 -50 29
20 -99 83 -84 62 53 31 -36 -89 -94
-68 30 -55 -46 24 16 9 52 61 21
49 -46 17 50 -59 -13 -30 69 28 -76
-84 -7 4 -30 -26 -69 -39 19 -16 -77
9 45 4 80 -2 -56 -3 94 -23 -83
```

### Zadanie b

Ścieżki do plików: plik1.txt i plik2.txt

```
D:\workspace\Programowanie\rok3\asm\DANE\plik1.txt
```

```
D:\workspace\Programowanie\rok3\asm\DANE\plik2.txt
```

Dane z plik1.txt - co 8 nieparzysta:

```
-53 31 -27 31 -59 9
```

Dane z plik2.txt - co 8 parzysta:

```
-70 -68 -84 16 4 94
```

```
D:\workspace\Programowanie\rok3\asm>
```

Zawartość pliku plik1.txt:

-53

31

-27

31

-59

9

Zawartość pliku plik2.txt:

-70

-68

-84

16

4

94