Додатки

Додаток А (Код на мові Асемблер)

Prog1.asm

```
.386
.model flat, stdcall
option casemap :none

include masm32\include\windows.inc
include masm32\include\kernel32.inc
include masm32\include\masm32.inc
include masm32\include\user32.inc
include masm32\include\msvcrt.inc
includelib masm32\lib\kernel32.lib
includelib masm32\lib\masm32.lib
includelib masm32\lib\user32.lib
includelib masm32\lib\msvcrt.lib

.DATA
;===User
Data===============================================================
=============
    Aaaaa_      dd      0
    Bbbbb_      dd      0
    Xxxxx_      dd      0
    Yyyyy_      dd      0

    DivErrMsg   db      13, 10, "Division: Error: division by zero", 0
    ModErrMsg   db      13, 10, "Mod: Error: division by zero", 0
    String_0    db      "INPUT Aaaaa: ", 0
    String_1    db      "INPUT Bbbbb: ", 0
    String_2    db      "Aaaaa + Bbbbb: ", 0
    String_3    db      13, 10, "_AAAAAAAAAAAAAAAA - Bbbbb: ", 0
    String_4    db      13, 10, "_AAAAAAAAAAAAAAAA * Bbbbb: ", 0
    String_5    db      13, 10, "_AAAAAAAAAAAAAAAA / Bbbbb: ", 0
    String_6    db      13, 10, "_AAAAAAAAAAAAAAAA % Bbbbb: ", 0
    String_7    db      13, 10, "_XXXXXXXXXXXXXXXX = (Aaaaa - Bbbbb) * 10 +
(Aaaaa + Bbbbb) / 10", 13, 10, 0
    String_8    db      13, 10, "_YYYYYYYYYYYYYYYY = Xxxxx + (Xxxxx % 10)", 13,
10, 0

;===Addition
Data===============================================================
=========
    hConsoleInputdd     ?
    hConsoleOutput      dd      ?
    endBuff                     db      5 dup (?)
    msg1310                     db      13, 10, 0

    CharsReadNum        dd      ?
    InputBuf            db      15 dup (?)
    OutMessage          db      "%d", 0
    ResMessage          db      20 dup (?)

.CODE
start:
invoke AllocConsole
invoke GetStdHandle, STD_INPUT_HANDLE
mov hConsoleInput, eax
invoke GetStdHandle, STD_OUTPUT_HANDLE
mov hConsoleOutput, eax
    invoke WriteConsoleA, hConsoleOutput, ADDR String_0, SIZEOF String_0 - 1, 0, 0
    call Input_
    mov Aaaaa_, eax
    invoke WriteConsoleA, hConsoleOutput, ADDR String_1, SIZEOF String_1 - 1, 0, 0
    call Input_
    mov Bbbbb_, eax
    invoke WriteConsoleA, hConsoleOutput, ADDR String_2, SIZEOF String_2 - 1, 0, 0
    push Aaaaa_
    push Bbbbb_
    call Add_
    call Output_
    invoke WriteConsoleA, hConsoleOutput, ADDR String_3, SIZEOF String_3 - 1, 0, 0
    push Aaaaa_
    push Bbbbb_
    call Sub_
    call Output_
    invoke WriteConsoleA, hConsoleOutput, ADDR String_4, SIZEOF String_4 - 1, 0, 0
    push Aaaaa_
    push Bbbbb_
    call Mul_
    call Output_
    invoke WriteConsoleA, hConsoleOutput, ADDR String_5, SIZEOF String_5 - 1, 0, 0
    push Aaaaa_
    push Bbbbb_
    call Div_
    call Output_
    invoke WriteConsoleA, hConsoleOutput, ADDR String_6, SIZEOF String_6 - 1, 0, 0
    push Aaaaa_
    push Bbbbb_
    call Mod_
    call Output_
    push Aaaaa_
    push Bbbbb_
    call Sub_
    push dword ptr 10
    call Mul_
    push Aaaaa_
    push Bbbbb_
    call Add_
```

```
        push dword ptr 10
        call Div_
        call Add_
        pop Xxxxx_
        push Xxxxx_
        push Xxxxx_
        push dword ptr 10
        call Mod_
        call Add_
        pop Yyyyy_
        invoke WriteConsoleA, hConsoleOutput, ADDR String_7, SIZEOF String_7 - 1, 0, 0
        push Xxxxx_
        call Output_
        invoke WriteConsoleA, hConsoleOutput, ADDR String_8, SIZEOF String_8 - 1, 0, 0
        push Yyyyy_
        call Output_
    exit_label:
    invoke WriteConsoleA, hConsoleOutput, ADDR msg1310, SIZEOF msg1310 - 1, 0, 0
    invoke ReadConsoleA, hConsoleInput, ADDR endBuff, 5, 0, 0
    invoke ExitProcess, 0


    ;===Procedure
Add=================================================================
========
    Add_ PROC
        mov eax, [esp + 8]
        add eax, [esp + 4]
        mov [esp + 8], eax
        pop ecx
        pop eax
        push ecx
        ret
    Add_ ENDP
    ;=================================================================
======================


    ;===Procedure
Div=================================================================
========
    Div_ PROC
        pushf
        pop cx

        mov eax, [esp + 4]
        cmp eax, 0
        jne end_check
        invoke WriteConsoleA, hConsoleOutput, ADDR DivErrMsg, SIZEOF DivErrMsg - 1, 0, 0
        jmp exit_label
    end_check:
        mov eax, [esp + 8]
        cmp eax, 0
        jge gr
    lo:
        mov edx, -1
        jmp less_fin
    gr:
        mov edx, 0
    less_fin:
        mov eax, [esp + 8]
        idiv dword ptr [esp + 4]
        push cx
        popf

        mov [esp + 8], eax
        pop ecx
        pop eax
        push ecx
        ret
    Div_ ENDP
    ;============================================================
======================


    ;===Procedure
Input===========================================================
=======
    Input_ PROC
        invoke ReadConsoleA, hConsoleInput, ADDR InputBuf, 13, ADDR CharsReadNum, 0
        invoke crt_atoi, ADDR InputBuf
        ret
    Input_ ENDP
    ;============================================================
======================


    ;===Procedure
Mod=============================================================
=========
    Mod_ PROC
        pushf
        pop cx

        mov eax, [esp + 4]
        cmp eax, 0
        jne end_check
        invoke WriteConsoleA, hConsoleOutput, ADDR ModErrMsg, SIZEOF ModErrMsg - 1, 0,
0
        jmp exit_label
    end_check:
        mov eax, [esp + 8]
        cmp eax, 0
        jge gr
    lo:
        mov edx, -1
        jmp less_fin
    gr:
        mov edx, 0
    less_fin:
        mov eax, [esp + 8]
        idiv dword ptr [esp + 4]
        mov eax, edx
        push cx
        popf

        mov [esp + 8], eax
        pop ecx
        pop eax
        push ecx
        ret
    Mod_ ENDP
    ;=====================================================================
======================


    ;===Procedure
Mul===================================================================
========
    Mul_ PROC
        mov eax, [esp + 8]
        imul dword ptr [esp + 4]
        mov [esp + 8], eax
        pop ecx
        pop eax
        push ecx
        ret
    Mul_ ENDP
    ;=====================================================================
======================


    ;===Procedure
Output================================================================
=======
    Output_ PROC value: dword
        invoke wsprintf, ADDR ResMessage, ADDR OutMessage, value
        invoke WriteConsoleA, hConsoleOutput, ADDR ResMessage, eax, 0, 0
        ret 4
    Output_ ENDP
    ;==========================================================================
========================


    ;===Procedure
Sub===========================================================================
========
    Sub_ PROC
        mov eax, [esp + 8]
        sub eax, [esp + 4]
        mov [esp + 8], eax
        pop ecx
        pop eax
        push ecx
        ret
    Sub_ ENDP
    ;==========================================================================
========================
    end start
```

Prog2.asm

```
    .386
    .model flat, stdcall
    option casemap :none

    include masm32\include\windows.inc
    include masm32\include\kernel32.inc
    include masm32\include\masm32.inc
    include masm32\include\user32.inc
    include masm32\include\msvcrt.inc
    includelib masm32\lib\kernel32.lib
    includelib masm32\lib\masm32.lib
    includelib masm32\lib\user32.lib
    includelib masm32\lib\msvcrt.lib

    .DATA
    ;===User
Data==========================================================================
=============
        Aaaaa_      dd      0
        Bbbbb_      dd      0
        Ccccc_dd    0

        String_0    db      "INPUT Aaaaa: ", 0
        String_1    db      "INPUT Bbbbb: ", 0
        String_2    db      "INPUT Ccccc: ", 0
        String_3    db      13, 10, 0
        String_4    db      13, 10, 0
        String_5    db      13, 10, 0
    ;===Addition
Data=====================================================================
=========
        hConsoleInputdd     ?
        hConsoleOutput      dd      ?
        endBuff                     db      5 dup (?)
        msg1310                     db      13, 10, 0

        CharsReadNum        dd      ?
        InputBuf            db      15 dup (?)
        OutMessage          db      "%d", 0
        ResMessage          db      20 dup (?)

    .CODE
    start:
    invoke AllocConsole
    invoke GetStdHandle, STD_INPUT_HANDLE
    mov hConsoleInput, eax
    invoke GetStdHandle, STD_OUTPUT_HANDLE
    mov hConsoleOutput, eax
        invoke WriteConsoleA, hConsoleOutput, ADDR String_0, SIZEOF String_0 - 1, 0, 0
        call Input_
        mov Aaaaa_, eax
        invoke WriteConsoleA, hConsoleOutput, ADDR String_1, SIZEOF String_1 - 1, 0, 0
        call Input_
        mov Bbbbb_, eax
        invoke WriteConsoleA, hConsoleOutput, ADDR String_2, SIZEOF String_2 - 1, 0, 0
        call Input_
        mov Ccccc_, eax
        push Aaaaa_
        push Bbbbb_
        call Greate_
        pop eax
        cmp eax, 0
        je endIf2
        push Aaaaa_
        push Ccccc_
        call Greate_
        pop eax
        cmp eax, 0
        je elseLabel1
        jmp Aibig_
        jmp endIf1
    elseLabel1:
        push Ccccc_
        call Output_
        jmp Outif_
    Aibig_:
        push Aaaaa_
        call Output_
        jmp Outif_
    endIf1:
    endIf2:
        push Bbbbb_
        push Ccccc_
        call Less_
        pop eax
        cmp eax, 0
        je elseLabel3
        push Ccccc_
        call Output_
        jmp endIf3
    elseLabel3:
        push Bbbbb_
        call Output_
    endIf3:
    Outif_:
        invoke WriteConsoleA, hConsoleOutput, ADDR String_3, SIZEOF String_3 - 1, 0, 0
        push Aaaaa_
        push Bbbbb_
        call Equal_
        push Aaaaa_
        push Ccccc_
        call Equal_
        call And_
        push Bbbbb_
        push Ccccc_
        call Equal_
        call And_
        pop eax
        cmp eax, 0
        je elseLabel4
        push dword ptr 1
        call Output_
        jmp endIf4
    elseLabel4:
        push dword ptr 0
        call Output_
    endIf4:
        invoke WriteConsoleA, hConsoleOutput, ADDR String_4, SIZEOF String_4 - 1, 0, 0
        push Aaaaa_
        push dword ptr 0
        call Less_
        push Bbbbb_
        push dword ptr 0
        call Less_
        call Or_
        push Ccccc_
        push dword ptr 0
        call Less_
```

```
        call Or_
        pop eax
        cmp eax, 0
        je elseLabel5
        push dword ptr -1
        call Output_
        jmp endIf5
    elseLabel5:
        push dword ptr 0
        call Output_
    endIf5:
        invoke WriteConsoleA, hConsoleOutput, ADDR String_5, SIZEOF String_5 - 1, 0, 0
        push Aaaaa_
        push Bbbbb_
        push Ccccc_
        call Add_
        call Less_
        call Not_
        pop eax
        cmp eax, 0
        je elseLabel6
        push dword ptr 10
        call Output_
        jmp endIf6
    elseLabel6:
        push dword ptr 0
        call Output_
    endIf6:
    exit_label:
    invoke WriteConsoleA, hConsoleOutput, ADDR msg1310, SIZEOF msg1310 - 1, 0, 0
    invoke ReadConsoleA, hConsoleInput, ADDR endBuff, 5, 0, 0
    invoke ExitProcess, 0


    ;===Procedure
Add===========================================================================
========
    Add_ PROC
        mov eax, [esp + 8]
        add eax, [esp + 4]
        mov [esp + 8], eax
        pop ecx
        pop eax
        push ecx
        ret
    Add_ ENDP
    ;===========================================================================
======================
    ;===Procedure
And====================================================================
========
    And_ PROC
        pushf
        pop cx

        mov eax, [esp + 8]
        cmp eax, 0
        jnz and_t1
        jz and_false
    and_t1:
        mov eax, [esp + 4]
        cmp eax, 0
        jnz and_true
    and_false:
        mov eax, 0
        jmp and_fin
    and_true:
        mov eax, 1
    and_fin:
        push cx
        popf

        mov [esp + 8], eax
        pop ecx
        pop eax
        push ecx
        ret
    And_ ENDP
    ;====================================================================
========================


    ;===Procedure
Equal===================================================================
=======
    Equal_ PROC
        pushf
        pop cx

        mov eax, [esp + 8]
        cmp eax, [esp + 4]
        jne equal_false
        mov eax, 1
        jmp equal_fin
    equal_false:
        mov eax, 0
    equal_fin:
        push cx
        popf

        mov [esp + 8], eax
        pop ecx
        pop eax
        push ecx
        ret
    Equal_ ENDP
    ;==============================================================================================

    ;===Procedure Greate=============================================================================
    Greate_ PROC
        pushf
        pop cx

        mov eax, [esp + 8]
        cmp eax, [esp + 4]
        jle greate_false
        mov eax, 1
        jmp greate_fin
    greate_false:
        mov eax, 0
    greate_fin:
        push cx
        popf

        mov [esp + 8], eax
        pop ecx
        pop eax
        push ecx
        ret
    Greate_ ENDP
    ;==============================================================================================

    ;===Procedure Input=============================================================================
    Input_ PROC
        invoke ReadConsoleA, hConsoleInput, ADDR InputBuf, 13, ADDR CharsReadNum, 0
        invoke crt_atoi, ADDR InputBuf
        ret
    Input_ ENDP
    ;==============================================================================================
    ;===Procedure
Less====================================================================
========
    Less_ PROC
        pushf
        pop cx

        mov eax, [esp + 8]
        cmp eax, [esp + 4]
        jge less_false
        mov eax, 1
        jmp less_fin
    less_false:
        mov eax, 0
    less_fin:
        push cx
        popf

        mov [esp + 8], eax
        pop ecx
        pop eax
        push ecx
        ret
    Less_ ENDP
    ;==================================================================
=======================


    ;===Procedure
Not=====================================================================
========
    Not_ PROC
        pushf
        pop cx

        mov eax, [esp + 4]
        cmp eax, 0
        jnz not_false
    not_t1:
        mov eax, 1
        jmp not_fin
    not_false:
        mov eax, 0
    not_fin:
        push cx
        popf

        mov [esp + 4], eax
        ret
    Not_ ENDP
    ;==================================================================
======================


    ;===Procedure
Or==================================================================
========
    Or_ PROC
        pushf
        pop cx

        mov eax, [esp + 8]
        cmp eax, 0
        jnz or_true
        jz or_t1
    or_t1:
        mov eax, [esp + 4]
        cmp eax, 0
        jnz or_true
    or_false:
        mov eax, 0
        jmp or_fin
    or_true:
        mov eax, 1
    or_fin:
        push cx
        popf

        mov [esp + 8], eax
        pop ecx
        pop eax
        push ecx
        ret
    Or_ ENDP
    ;==================================================================
======================


    ;===Procedure
Output==============================================================
=======
    Output_ PROC value: dword
        invoke wsprintf, ADDR ResMessage, ADDR OutMessage, value
        invoke WriteConsoleA, hConsoleOutput, ADDR ResMessage, eax, 0, 0
        ret 4
    Output_ ENDP
    ;==================================================================
======================
end start
```
Prog3.asm
```
.386
.model flat, stdcall
option casemap :none

include masm32\include\windows.inc
include masm32\include\kernel32.inc
include masm32\include\masm32.inc
include masm32\include\user32.inc
include masm32\include\msvcrt.inc
includelib masm32\lib\kernel32.lib
includelib masm32\lib\masm32.lib
includelib masm32\lib\user32.lib
includelib masm32\lib\msvcrt.lib

.DATA
;===User
Data=====================================================================
=============
	Aaaa2_		dd	0
	Aaaaa_		dd	0
	Bbbbb_		dd	0
	Cccc1_		dd	0
	Cccc2_		dd	0
	Xxxxx_		dd	0

	String_0	db	"INPUT Aaaaa: ", 0
	String_1	db	"INPUT Bbbbb: ", 0
	String_2	db	"FOR TO do", 0
	String_3	db	13, 10, 0
	String_4	db	13, 10, "For DOWNTO do", 0
	String_5	db	13, 10, 0
	String_6	db	13, 10, "While Aaaaa * Bbbbb: ", 0
	String_7	db	13, 10, "Repeat UNTIL Aaaaa * Bbbbb: ", 0

;===Addition
Data=====================================================================
=========
	hConsoleInputdd	?
	hConsoleOutput	dd	?
	endBuff			db	5 dup (?)
	msg1310			db	13, 10, 0

	CharsReadNum	dd	?
	InputBuf	db	15 dup (?)
	OutMessage	db	"%d", 0
	ResMessage	db	20 dup (?)

.CODE
```

```
start:
invoke AllocConsole
invoke GetStdHandle, STD_INPUT_HANDLE
mov hConsoleInput, eax
invoke GetStdHandle, STD_OUTPUT_HANDLE
mov hConsoleOutput, eax
    invoke WriteConsoleA, hConsoleOutput, ADDR String_0, SIZEOF String_0 - 1, 0, 0
    call Input_
    mov Aaaaa_, eax
    invoke WriteConsoleA, hConsoleOutput, ADDR String_1, SIZEOF String_1 - 1, 0, 0
    call Input_
    mov Bbbbb_, eax
    invoke WriteConsoleA, hConsoleOutput, ADDR String_2, SIZEOF String_2 - 1, 0, 0
    push Aaaaa_
    pop Aaaa2_
forPasStart1:
    push Bbbbb_
    push Aaaa2_
    call Less_
    call Not_
    pop eax
    cmp eax, 0
    je forPasEnd1
    invoke WriteConsoleA, hConsoleOutput, ADDR String_3, SIZEOF String_3 - 1, 0, 0
    push Aaaa2_
    push Aaaa2_
    call Mul_
    call Output_
    push Aaaa2_
    push dword ptr 1
    call Add_
    pop Aaaa2_
    jmp forPasStart1
forPasEnd1:
    invoke WriteConsoleA, hConsoleOutput, ADDR String_4, SIZEOF String_4 - 1, 0, 0
    push Bbbbb_
    pop Aaaa2_
forPasStart2:
    push Aaaaa_
    push Aaaa2_
    call Greate_
    call Not_
    pop eax
    cmp eax, 0
    je forPasEnd2
    invoke WriteConsoleA, hConsoleOutput, ADDR String_5, SIZEOF String_5 - 1, 0, 0
    push Aaaa2_
    push Aaaa2_
    call Mul_
    call Output_
    push Aaaa2_
    push dword ptr 1
    call Sub_
    pop Aaaa2_
    jmp forPasStart2
forPasEnd2:
    invoke WriteConsoleA, hConsoleOutput, ADDR String_6, SIZEOF String_6 - 1, 0, 0
    push dword ptr 0
    pop Xxxxx_
    push dword ptr 0
    pop Cccc1_
whileStart2:
    push Cccc1_
    push Aaaaa_
    call Less_
    pop eax
    cmp eax, 0
    je whileEnd2
    push dword ptr 0
    pop Cccc2_
whileStart1:
    push Cccc2_
    push Bbbbb_
    call Less_
    pop eax
    cmp eax, 0
    je whileEnd1
    push Xxxxx_
    push dword ptr 1
    call Add_
    pop Xxxxx_
    push Cccc2_
    push dword ptr 1
    call Add_
    pop Cccc2_
    jmp whileStart1
whileEnd1:
    push Cccc1_
    push dword ptr 1
    call Add_
    pop Cccc1_
    jmp whileStart2
whileEnd2:
    push Xxxxx_
    call Output_
    invoke WriteConsoleA, hConsoleOutput, ADDR String_7, SIZEOF String_7 - 1, 0, 0
    push dword ptr 0
    pop Xxxxx_
    push dword ptr 1
    pop Cccc1_
    repeatStart2:
        push dword ptr 1
        pop Cccc2_
    repeatStart1:
        push Xxxxx_
        push dword ptr 1
        call Add_
        pop Xxxxx_
        push Cccc2_
        push dword ptr 1
        call Add_
        pop Cccc2_
        push Cccc2_
        push Bbbbb_
        call Greate_
        call Not_
        pop eax
        cmp eax, 0
        je repeatEnd1
        jmp repeatStart1
    repeatEnd1:
        push Cccc1_
        push dword ptr 1
        call Add_
        pop Cccc1_
        push Cccc1_
        push Aaaaa_
        call Greate_
        call Not_
        pop eax
        cmp eax, 0
        je repeatEnd2
        jmp repeatStart2
    repeatEnd2:
        push Xxxxx_
        call Output_
    exit_label:
    invoke WriteConsoleA, hConsoleOutput, ADDR msg1310, SIZEOF msg1310 - 1, 0, 0
    invoke ReadConsoleA, hConsoleInput, ADDR endBuff, 5, 0, 0
    invoke ExitProcess, 0


    ;===Procedure
Add============================================================
========
    Add_ PROC
        mov eax, [esp + 8]
        add eax, [esp + 4]
        mov [esp + 8], eax
        pop ecx
        pop eax
        push ecx
        ret
    Add_ ENDP
    ;======================================================================
=====================


    ;===Procedure
Greate=================================================================
=======
    Greate_ PROC
        pushf
        pop cx

        mov eax, [esp + 8]
        cmp eax, [esp + 4]
        jle greate_false
        mov eax, 1
        jmp greate_fin
    greate_false:
        mov eax, 0
    greate_fin:
        push cx
        popf

        mov [esp + 8], eax
        pop ecx
        pop eax
        push ecx
        ret
    Greate_ ENDP
    ;======================================================================
=====================


    ;===Procedure
Input==================================================================
=======
    Input_ PROC
        invoke ReadConsoleA, hConsoleInput, ADDR InputBuf, 13, ADDR CharsReadNum, 0
        invoke crt_atoi, ADDR InputBuf
        ret
    Input_ ENDP
    ;======================================================================
=====================
    ;===Procedure
Less===================================================================
========
    Less_ PROC
        pushf
        pop cx

        mov eax, [esp + 8]
        cmp eax, [esp + 4]
        jge less_false
        mov eax, 1
        jmp less_fin
    less_false:
        mov eax, 0
    less_fin:
        push cx
        popf

        mov [esp + 8], eax
        pop ecx
        pop eax
        push ecx
        ret
    Less_ ENDP
    ;====================================================================
========================


    ;===Procedure
Mul====================================================================
========
    Mul_ PROC
        mov eax, [esp + 8]
        imul dword ptr [esp + 4]
        mov [esp + 8], eax
        pop ecx
        pop eax
        push ecx
        ret
    Mul_ ENDP
    ;====================================================================
========================


    ;===Procedure
Not===================================================================
========
    Not_ PROC
        pushf
        pop cx
        mov eax, [esp + 4]
        cmp eax, 0
        jnz not_false
    not_t1:
        mov eax, 1
        jmp not_fin
    not_false:
        mov eax, 0
    not_fin:
        push cx
        popf

        mov [esp + 4], eax
        ret
    Not_ ENDP
    ;=================================================================
=======================


    ;===Procedure
Output=============================================================
=======
    Output_ PROC value: dword
        invoke wsprintf, ADDR ResMessage, ADDR OutMessage, value
        invoke WriteConsoleA, hConsoleOutput, ADDR ResMessage, eax, 0, 0
        ret 4
    Output_ ENDP
    ;=================================================================
=======================


    ;===Procedure
Sub================================================================
========
    Sub_ PROC
        mov eax, [esp + 8]
        sub eax, [esp + 4]
        mov [esp + 8], eax
        pop ecx
        pop eax
        push ecx
        ret
    Sub_ ENDP
    ;=================================================================
=======================
    end start
```