

Міністерство освіти та науки України

Національний технічний університет України «Київський політехнічний інститут»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Лабораторна робота №10

з дисципліни «Системне програмування – 1»

на тему: «Використання у проекті C++ модулів на асемблері»

Виконав:
студент 2-го курсу ФІОТ
групи ІВ-71
Мазан Я. В.
Перевірив:
Старший викладач
Порєв В. М.

Київ – 2019

Мета:

Навчитися створювати програми на C++ з використанням модулів на асемблері.

Завдання:

Варіант	Що потрібно обчислити	Розрядність (біт)
9	$(1 - A)(B + C)$	768

Код програми:

main.cpp:

```
#include <iostream>
#include "longop.h"

using namespace std;

int main() {
    unsigned int a[24];
    unsigned int b[24];
    unsigned int c[24];
    unsigned int oneMinusA[25];
    unsigned int bPlusC[25];
    unsigned int result[50];
    /*unsigned int var1[1] = {3};
    unsigned int var2[1] = {2};
    unsigned int result[2];
    char text_buffer[5];*/
    fill_n(oneMinusA, 25, 0);
    fill_n(bPlusC, 25, 0);
    fill_n(result, 50, 0);

    unsigned int one[24];
    fill_n(one, 23, 0);
    one[23] = 1;
    /*fill_n(a, 23, 0);
    fill_n(b, 23, 0);
    fill_n(c, 23, 0);
    a[23] = 0;
    b[23] = 0xf;
    c[23] = 1;*/
    unsigned int var = 0x80000000;
    for (int i = 0; i < 24; i++) {
        a[i] = var;
        b[i] = 0x80000000;
        c[i] = 0xffffffff;
        var += 0x00001000;
    }

    subLongop(oneMinusA, a, one, 24);    //oneMinusA = 1 - a
    addLongop(bPlusC, c, b, 24);
    mulN_x_N(result, bPlusC, oneMinusA, 25);
    for (int i = 0; i < 50; i++)
        cout << hex << result[i] << " ";

    //subLongop(result, var2, var1, 1);    result = var1 - var2

    //strDec(result, text_buffer, 2);
```

```
    return 0;
}
```

longop.asm:

```
section .bss
```

```
    i : resd 1
```

```
    j : resd 1
```

```
section .text
```

```
global mulN_x_N
```

```
global addLongop
```

```
global subLongop
```

```
addLongop:
```

```
    push ebp
```

```
    mov ebp, esp
```

```
    ;push esi
```

```
    ;push edx
```

```
    push ebx
```

```
    ;push ecx
```

```
    mov esi, dword [ebp+20]    ; length
```

```
    mov edx, dword [ebp+16]    ; A
```

```
    mov ebx, dword [ebp+12]    ; B
```

```
    mov ecx, dword [ebp+8]     ; C
```

```
    clc
```

```
    pushf
```

```
    dec esi
```

```
@addCycle:
```

```
    popf
```

```
    mov eax, dword [edx + esi*4]
```

```
    adc eax, dword [ebx + esi*4]
```

```
    pushf
```

```
    mov dword [ecx+esi*4+4], eax
```

```
    dec esi
```

```
    cmp esi, 0
```

```
    jge @addCycle
```

```
    xor eax, eax
```

```
    popf                ; забираємо наш прапор переносу зі стеку для того, щоби при наступному додаванні його використати
```

```
    adc eax, 0           ; запис прапору переносу в eax
```

```
    mov dword [ecx], eax
```

```
    ;pop ecx
```

```
    pop ebx
```

```
    ;pop edx
```

```
    ;pop esi
```

```
    leave
```

```
    ret                ; add esp, 16
```

```
subLongop:
```

```
    push ebp
```

```
    mov ebp, esp
```

```
    ;push esi
```

```
    ;push edx
```

```
    push ebx
```

```
    ;push ecx
```

```
    ;push eax
```

```
    mov esi, dword [ebp+20]    ; length
```

```
    mov edx, dword [ebp+16]    ; A
```

```
    mov ebx, dword [ebp+12]    ; B
```

```
    mov ecx, dword [ebp+8]     ; C
```

```
    clc
```

```

pushf
dec esi
@subCycle:
    popf
    mov eax, dword [edx + esi*4]
    sub eax, dword [ebx + esi*4]
    pushf
    mov dword [ecx+esi*4], eax
    dec esi
    cmp esi, 0
    jge @subCycle
    popf

;pop eax
;pop ecx
pop ebx
;pop edx
;pop esi

leave
ret ; add esp, 16

```

```

mulN_x_N: ; size, A, B, res
    push ebp
    mov ebp, esp

```

```

;push esi
push ebx
;push edi
;push ecx
;push eax

```

```

mov esi, dword [ebp + 20] ; size
dec esi ; Making pointer to point on start of
push esi ; save pointer on the last element
mov dword [i], esi ; i - iterates over A's dwords
mov dword [j], esi ; j - iterates over B's dwords
; base pointer: edi = i + j
mov esi, dword [ebp + 16] ; A
mov ebx, dword [ebp + 12] ; B
mov edi, dword [ebp + 8] ; result

```

```

@outerCycle:
    pop dword [j]
    push dword [j]
    @innerCycle:
        mov ecx, dword [i]
        mov eax, dword [esi + 4*ecx] ; eax = dword [A + i]
        mov ecx, dword [j]
        mul dword [ebx + 4*ecx] ; eax *= dword [B + j]
        add ecx, dword [i]
        add dword [edi + 4*ecx + 4], eax; res = dword [edi + i + j + 4], eax
        adc dword [edi + 4*ecx], edx ; res = dword [edi + i + j], edx
        push ecx
        pushf
        @addcarryflag: ; spread carry flag over all other digit numbers
            popf
            adc dword [edi + 4*ecx - 4], 0
            pushf
            dec ecx
            cmp ecx, 0
            jge @addcarryflag
        popf
        pop ecx
        mov eax, dword [j]
        dec eax
        mov dword [j], eax
        cmp eax, 0
        jge @innerCycle
    mov eax, dword [i]

```

```

dec eax
mov dword [i], eax
cmp eax, 0
jge @outerCycle

pop esi
;pop eax
;pop ecx
;pop edi
pop ebx
;pop esi

leave
ret                                ; add esp, 16

```

Асемблований main.cpp (main.s):

<pre> .file "main.cpp" .intel_syntax noprefix .text .section .rodata .type _ZStL19piecewise_construct, @object .size _ZStL19piecewise_construct, 1 _ZStL19piecewise_construct: .zero 1 .local _ZStL8__ioint .comm _ZStL8__ioint,1,1 .text .globl main .type main, @function main: .LFB1490: .cfi_startproc push ebp .cfi_def_cfa_offset 8 .cfi_offset 5, -8 mov ebp, esp .cfi_def_cfa_register 5 push ebx and esp, -16 sub esp, 352 .cfi_offset 3, -12 call __x86.get_pc_thunk.bx add ebx, OFFSET FLAT:_GLOBAL_OFFSET_TABLE_ mov eax, DWORD PTR gs:20 mov DWORD PTR 348[esp], eax xor eax, eax mov DWORD PTR 40[esp], 3 mov DWORD PTR 44[esp], 2 mov DWORD PTR 20[esp], -2147483648 mov DWORD PTR 24[esp], 0 .L3: cmp DWORD PTR 24[esp], 23 jg .L2 mov eax, DWORD PTR 24[esp] mov edx, DWORD PTR 20[esp] mov DWORD PTR 56[esp+eax*4], edx mov eax, DWORD PTR 24[esp] mov DWORD PTR 152[esp+eax*4], -2147483648 add DWORD PTR 20[esp], 4096 inc DWORD PTR 24[esp] jmp .L3 .L2: mov DWORD PTR 12[esp], 24 lea eax, 56[esp] mov DWORD PTR 8[esp], eax lea eax, 152[esp] mov DWORD PTR 4[esp], eax lea eax, 248[esp] mov DWORD PTR [esp], eax </pre>	<pre> leave .cfi_restore 5 .cfi_def_cfa 4, 4 ret .cfi_endproc .LFE1730: .size _ZSt6fill_nIPjiiET_S1_T0_RKT1_, . _ZSt6fill_nIPjiiET_S1_T0_RKT1_ .section .text._ZSt12__niter_baseIPJET_S1_,"axG",@progbits,_ZSt12__niter_baseIPJET_S1_ .cfi_startproc push ebp .cfi_def_cfa_offset 8 .cfi_offset 5, -8 mov ebp, esp .cfi_def_cfa_register 5 call __x86.get_pc_thunk.ax add eax, OFFSET FLAT:_GLOBAL_OFFSET_TABLE_ mov eax, DWORD PTR 8[ebp] pop ebp .cfi_restore 5 .cfi_def_cfa 4, 4 ret .cfi_endproc .LFE1841: .size _ZSt12__niter_baseIPJET_S1_ _ZSt12__niter_baseIPJET_S1_ .type _ZSt12__niter_baseIPJET_S1_, @function _ZSt12__niter_baseIPJET_S1_: .LFB1841: .cfi_startproc push ebp .cfi_def_cfa_offset 8 .cfi_offset 5, -8 mov ebp, esp .cfi_def_cfa_register 5 call __x86.get_pc_thunk.ax add eax, OFFSET FLAT:_GLOBAL_OFFSET_TABLE_ mov eax, DWORD PTR 8[ebp] pop ebp .cfi_restore 5 .cfi_def_cfa 4, 4 ret .cfi_endproc .LFE1841: .size _ZSt10__fill_n_aIPjiiEN9__gnu_cxx11__enable_ifIXsrSt11__is_scalarIT1_E7__valueET_E6__typeES6_T0_RKS4_,"axG",@progbits,_ZSt10__fill_n_aIPjiiEN9__gnu_cxx11__enable_ifIXsrSt11__is_scalarIT1_E7__valueET_E6__typeES6_T0_RKS4_ _ZSt10__fill_n_aIPjiiEN9__gnu_cxx11__enable_ifIXsrSt11__is_scalarIT1_E7__valueET_E6__typeES6_T0_RKS4_ .type _ZSt10__fill_n_aIPjiiEN9__gnu_cxx11__enable_ifIXsrSt11__is_scalarIT1_E7__valueET_E6__typeES6_T0_RKS4_, @function _ZSt10__fill_n_aIPjiiEN9__gnu_cxx11__enable_ifIXsrSt11__is_scalarIT1_E7__valueET_E6__typeES6_T0_RKS4_: .LFB1842: .cfi_startproc push ebp .cfi_def_cfa_offset 8 .cfi_offset 5, -8 mov ebp, esp .cfi_def_cfa_register 5 sub esp, 16 call __x86.get_pc_thunk.ax </pre>
--	--

<div>.L5:</div> <div>call addLongop@PLT</div> <div>mov DWORD PTR 28[esp], 0</div> <div>cmp DWORD PTR 28[esp], 24</div> <div> <div>jg .L4</div> <div>mov eax, DWORD PTR 28[esp]</div> <div>mov eax, DWORD PTR 248[esp+eax*4]</div> <div>mov DWORD PTR 4[esp], eax</div> <div>mov eax, DWORD PTR _ZSt4cout@GOT[ebx]</div> <div>mov DWORD PTR [esp], eax</div> <div>call _ZNSolsEj@PLT</div> <div>inc DWORD PTR 28[esp]</div> <div>jmp .L5</div> </div> <div>.L4:</div> <div>mov eax, DWORD PTR _ZSt4endlcSt11char_traitslcEERSt13basic_ostreamIT_T0_ES6_@GOT[ebx]</div> <div>mov DWORD PTR 4[esp], eax</div> <div>mov eax, DWORD PTR _ZSt4cout@GOT[ebx]</div> <div>mov DWORD PTR [esp], eax</div> <div>call _ZNSolsEPFRSoS_E@PLT</div> <div>mov DWORD PTR 16[esp], 0</div> <div>lea eax, 16[esp]</div> <div>mov DWORD PTR 8[esp], eax</div> <div>mov DWORD PTR 4[esp], 25</div> <div>lea eax, 248[esp]</div> <div>mov DWORD PTR [esp], eax</div> <div>call _ZSt6fill_nIPjiiET_S1_T0_RKT1_</div> <div>mov DWORD PTR 12[esp], 24</div> <div>lea eax, 56[esp]</div> <div>mov DWORD PTR 8[esp], eax</div> <div>lea eax, 152[esp]</div> <div>mov DWORD PTR 4[esp], eax</div> <div>lea eax, 248[esp]</div> <div>mov DWORD PTR [esp], eax</div> <div>call subLongop@PLT</div> <div>mov DWORD PTR 32[esp], 0</div> <div>.L7:</div> <div>cmp DWORD PTR 32[esp], 24</div> <div> <div>jg .L6</div> <div>mov eax, DWORD PTR 32[esp]</div> <div>mov eax, DWORD PTR 248[esp+eax*4]</div> <div>mov DWORD PTR 4[esp], eax</div> <div>mov eax, DWORD PTR _ZSt4cout@GOT[ebx]</div> <div>mov DWORD PTR [esp], eax</div> <div>call _ZNSolsEj@PLT</div> <div>inc DWORD PTR 32[esp]</div> <div>jmp .L7</div> </div> <div>.L6:</div> <div>mov DWORD PTR 16[esp], 0</div> <div>lea eax, 16[esp]</div> <div>mov DWORD PTR 8[esp], eax</div> <div>mov DWORD PTR 4[esp], 5</div> <div>lea eax, 48[esp]</div> <div>mov DWORD PTR [esp], eax</div> <div>call _ZSt6fill_nIPjiiET_S1_T0_RKT1_</div> <div>mov eax, DWORD PTR _ZSt4endlcSt11char_traitslcEERSt13basic_ostreamIT_T0_ES6_@GOT[ebx]</div> <div>mov DWORD PTR 4[esp], eax</div> <div>mov eax, DWORD PTR _ZSt4cout@GOT[ebx]</div> <div>mov DWORD PTR [esp], eax</div> <div>call _ZNSolsEPFRSoS_E@PLT</div> <div>mov edx, DWORD PTR _ZSt4endlcSt11char_traitslcEERSt13basic_ostreamIT_T0_ES6_@GOT[ebx]</div> <div>mov DWORD PTR 4[esp], edx</div> <div>mov DWORD PTR [esp], eax</div> <div>call _ZNSolsEPFRSoS_E@PLT</div> <div>mov DWORD PTR 12[esp], 1</div> <div>lea eax, 40[esp]</div> <div>mov DWORD PTR 8[esp], eax</div> <div>lea eax, 44[esp]</div>	<div>add eax, OFFSET FLAT:_GLOBAL__OFFSET_TABLE_</div> <div>mov eax, DWORD PTR 16[ebp]</div> <div>mov eax, DWORD PTR [eax]</div> <div>mov DWORD PTR -4[ebp], eax</div> <div>mov eax, DWORD PTR 12[ebp]</div> <div>mov DWORD PTR -8[ebp], eax</div> <div>.L18:</div> <div>cmp DWORD PTR -8[ebp], 0</div> <div> <div>jle .L17</div> <div>mov edx, DWORD PTR -4[ebp]</div> <div>mov eax, DWORD PTR 8[ebp]</div> <div>mov DWORD PTR [eax], edx</div> <div>dec DWORD PTR -8[ebp]</div> <div>add DWORD PTR 8[ebp], 4</div> <div>jmp .L18</div> </div> <div>.L17:</div> <div>mov eax, DWORD PTR 8[ebp]</div> <div>leave</div> <div>.cfi_restore 5</div> <div>.cfi_def_cfa 4, 4</div> <div>ret</div> <div>.cfi_endproc</div> <div>.LFE1842:</div> <div>.size</div> <div>_ZSt10__fill_n_alPjiiEN9__gnu_cxx11__enable_ifIXsrSt11__is_scalarIT1_E7__valueET_E6__typeES6_T0_RKS4_,-</div> <div>_ZSt10__fill_n_alPjiiEN9__gnu_cxx11__enable_ifIXsrSt11__is_scalarIT1_E7__valueET_E6__typeES6_T0_RKS4_</div> <div>.text</div> <div>.type _Z41__static_initialization_and_destruction_0ii,@function</div> <div>_Z41__static_initialization_and_destruction_0ii:</div> <div>.LFB1980:</div> <div>.cfi_startproc</div> <div>push ebp</div> <div>.cfi_def_cfa_offset 8</div> <div>.cfi_offset 5, -8</div> <div>mov ebp, esp</div> <div>.cfi_def_cfa_register 5</div> <div>push ebx</div> <div>sub esp, 20</div> <div>.cfi_offset 3, -12</div> <div>call __x86.get_pc_thunk.bx</div> <div>add ebx, OFFSET FLAT:_GLOBAL__OFFSET_TABLE_</div> <div>cmp DWORD PTR 8[ebp], 1</div> <div> <div>jne .L22</div> <div>cmp DWORD PTR 12[ebp], 65535</div> <div>jne .L22</div> <div>lea eax, _ZStL8__joinit@GOTOFF[ebx]</div> <div>mov DWORD PTR [esp], eax</div> <div>call _ZNSt8ios_base4InitC1Ev@PLT</div> <div>lea eax, __dso_handle@GOTOFF[ebx]</div> <div>mov DWORD PTR 8[esp], eax</div> <div>lea eax, _ZStL8__joinit@GOTOFF[ebx]</div> <div>mov DWORD PTR 4[esp], eax</div> <div>mov eax, DWORD PTR _ZNSt8ios_base4InitD1Ev@GOT[ebx]</div> <div>mov DWORD PTR [esp], eax</div> <div>call __cxa_atexit@PLT</div> </div> <div>.L22:</div> <div>nop</div> <div>add esp, 20</div> <div>pop ebx</div> <div>.cfi_restore 3</div> <div>pop ebp</div> <div>.cfi_restore 5</div> <div>.cfi_def_cfa 4, 4</div> <div>ret</div> <div>.cfi_endproc</div> <div>.LFE1980:</div> <div>.size _Z41__static_initialization_and_destruction_0ii,-</div>
--	--

	<pre> mov DWORD PTR 4[esp], eax lea eax, 48[esp] mov DWORD PTR [esp], eax call mulN_x_N@PLT mov DWORD PTR 36[esp], 0 .L9: cmp DWORD PTR 36[esp], 1 jg .L8 mov eax, DWORD PTR 36[esp] mov eax, DWORD PTR 48[esp+eax*4] mov DWORD PTR 4[esp], eax mov eax, DWORD PTR _ZSt4cout@GOT[ebx] mov DWORD PTR [esp], eax call _ZNSolsEj@PLT inc DWORD PTR 36[esp] jmp .L9 .L8: mov eax, 0 mov ecx, DWORD PTR 348[esp] xor ecx, DWORD PTR gs:20 je .L11 call __stack_chk_fail_local .L11: mov ebx, DWORD PTR -4[ebp] leave .cfi_restore 5 .cfi_restore 3 .cfi_def_cfa 4, 4 ret .cfi_endproc .LFE1490: .size main, -main .section .text, _ZSt6fill_nIPjiiET_S1_T0_RKT1_, "axG", @progbits, _ZSt6fill_nIPjiiET_S1_T0_RKT1_, comdat .weak _ZSt6fill_nIPjiiET_S1_T0_RKT1_ .type _ZSt6fill_nIPjiiET_S1_T0_RKT1_ @function _ZSt6fill_nIPjiiET_S1_T0_RKT1_: .LFB1730: .cfi_startproc push ebp .cfi_def_cfa_offset 8 .cfi_offset 5, -8 mov ebp, esp .cfi_def_cfa_register 5 sub esp, 24 call __x86.get_pc_thunk.ax add eax, OFFSET FLAT:_GLOBAL__OFFSET_TABLE_ mov eax, DWORD PTR 8[ebp] mov DWORD PTR [esp], eax call _ZSt12__niter_baseIPjET_S1_ mov edx, DWORD PTR 16[ebp] mov DWORD PTR 8[esp], edx mov edx, DWORD PTR 12[ebp] mov DWORD PTR 4[esp], edx mov DWORD PTR [esp], eax call </pre>	<pre> _Z41__static_initialization_and_destruction_0ii .type _GLOBAL__sub_I_main, @function _GLOBAL__sub_I_main: .LFB1981: .cfi_startproc push ebp .cfi_def_cfa_offset 8 .cfi_offset 5, -8 mov ebp, esp .cfi_def_cfa_register 5 sub esp, 24 call __x86.get_pc_thunk.ax add eax, OFFSET FLAT:_GLOBAL__OFFSET_TABLE_ mov DWORD PTR 4[esp], 65535 mov DWORD PTR [esp], 1 call _Z41__static_initialization_and_destruction_0ii leave .cfi_restore 5 .cfi_def_cfa 4, 4 ret .cfi_endproc .LFE1981: .size _GLOBAL__sub_I_main, -_GLOBAL__sub_I_main .section .init_array,"aw" .align 4 .long _GLOBAL__sub_I_main .section .text.__x86.get_pc_thunk.ax,"axG",@progbits,__x86.get_pc_thunk.ax,comdat .globl __x86.get_pc_thunk.ax .hidden __x86.get_pc_thunk.ax .type __x86.get_pc_thunk.ax, @function __x86.get_pc_thunk.ax: .LFB1982: .cfi_startproc mov eax, DWORD PTR [esp] ret .cfi_endproc .LFE1982: .section .text.__x86.get_pc_thunk.bx,"axG",@progbits,__x86.get_pc_thunk.bx,comdat .globl __x86.get_pc_thunk.bx .hidden __x86.get_pc_thunk.bx .type __x86.get_pc_thunk.bx, @function __x86.get_pc_thunk.bx: .LFB1983: .cfi_startproc mov ebx, DWORD PTR [esp] ret .cfi_endproc .LFE1983: .hidden __dso_handle .hidden __stack_chk_fail_local .ident "GCC: (Ubuntu 7.4.0-1ubuntu1~18.04) 7.4.0" .section .note.GNU-stack,"",@progbits </pre>
	<pre> _ZSt10__fill_n_aIPjiiEN9__gnu_cxx11__enable_ifIXsrSt11__is_scalarIT1_E7__valueET_E6__typeES6_T0_RKS4_ </pre>	

Makefile:

```

CC=gcc
CPLUS=g++
ASM=nasm
LD=g++

```

#File types to compile

```
.SUFFIXES: .c .cpp .cxx .asm .o .a

OBJS=main.o longop.o

#Rules for compiling into .o files
.c.o:
    $(CC) -c -march=i686 -m32 -I. $*.c

.cpp.o:
    $(CPLUS) -c -march=i686 -m32 -I. $*.cpp

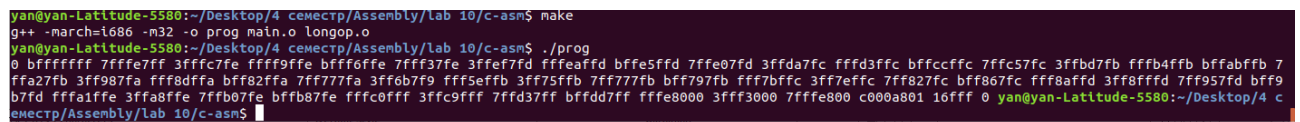
.cxx.o:
    $(CPLUS) -c -march=i686 -m32 -I. $*.cxx

.asm.o:
    $(ASM) -g -f elf32 $*.asm -o $*.o

#Rule for linking
prog: $(OBJS)
    $(LD) -march=i686 -m32 -o $@ $(OBJS)

#Cleaning project
clean:
    rm *.o prog
```

Результати виконання програми:



```
yan@yan-Latitude-5580:~/Desktop/4 семестр/Assembly/Lab 10/c-asm$ make
g++ -march=i686 -m32 -o prog main.o longop.o
yan@yan-Latitude-5580:~/Desktop/4 семестр/Assembly/Lab 10/c-asm$ ./prog
0 bffffff 7fffe7ff 3fffc7fe ffff9ffe bfff6ffe 7fff37fe 3ffef7fd fffeaafd bffe5ffd 7ffe07fd 3ffda7fc fffd3ffc bffccffc 7ffc57fc 3ffbd7fb fffb4ffb bffabffb 7
ffa27fb 3ff987fa fff8dffa bff82ffa 7ff777fa 3ff6b7f0 fff5effb 3ff75ffb 7ff777fb bff797fb fff7bffc 3ff7effc 7ff827fc bff867fc fff8affd 3ff8fffd 7ff957fd bff9
b7fd fffa1ffe 3ffa8ffe 7ffb07fe bffb87fe fffc0fff 3ffc9fff 7ffd37ff bffd77ff fffe8000 3fff3000 7fffe800 c000a801 16fff 0 yan@yan-Latitude-5580:~/Desktop/4 с
еместр/Assembly/Lab 10/c-asm$
```

Висновок:

Під час виконання даної лабораторної роботи були закріплені навички використання процедур написаних на асемблері в програмах C++. Під час реалізації розрахунку виразів були виявлені фатальні помилки в процедурах, але всі вони були вдало усунені.