**Федеральное государственное бюджетное образовательное учреждение высшего образования "Белгородский государственный технологический университет им. В.Г. Шухова"**

Кафедра программного обеспечения вычислительной техники и

автоматизированных систем.

**Лабораторная работа № 8**

Способы вызова ассемблерных подпрограмм

в языках высокого уровня.

Вариант 13

Выполнил:

Студент группы КБ-211

Коренев Д.Н.

Принял:

Осипов О.В.

*Цель работы:* изучение способов вызова подпрограмм, написанных на разных языках программирования посредством dll-библиотек.

**Задание**

1. Написать и отладить подпрограммы на masm32 в разных стилях вызова для решения задачи соответствующего варианта. Глобальные переменные в подпрограммах использовать не разрешается. Если нужна дополнительная память, выделять её в стеке.

Варианты 8 - 16

Из массива a длиной length скопировать отрицательные числа в массив neg\_res, положительные – в массив pos\_res. Под массивы neg\_res и pos\_res в основной программе зарезервировать памяти столько, сколько занимает массив a. Полученные массивы отсортировать. Количество отрицательных чисел записать в выходной параметр neg\_count, положительных – в выходной параметр pos\_count. Исходный массив a оставить без изменений. Для удобства, можно реализовать в dll-библиотеке отдельную процедуру для сортировки одного массива.

Пример: a = {1, 3, 4, -5, 7, -2, -1, 3, 5, -5}, length = 10;

pos\_res = {1, 3, 3, 4, 5, 7} (сортировка по не убыванию);

neg\_res = {-1, -2, -5, -5} (сортировка по не возрастанию);

pos\_count = 6;

neg\_count = 4.

Пузырьковая сортировка по не убыванию.

int sort (int\* neg\_res, int\* a, int length, int\* pos\_res, int\* pos\_count).

neg\_count возвратить.

bubble\_sort\_32 proc parr:DWORD, cnt:DWORD

                   push   edi

                   push   esi

                   push   eax

                   push   edx

                   push   ecx

                   push   ebx

                   mov    ebx, cnt

                   sub    ebx, 1

    lbl0:

                   mov    esi, parr

                   xor    edx, edx

                   mov    edi, ebx

    lbl1:

                   mov    eax, [esi]

                   mov    ecx, [esi+4]

                   cmp    eax, ecx

                   jle    lbl2

                   mov    [esi], ecx

                   mov    [esi+4], eax

                   mov    edx, 1

    lbl2:

                   add    esi, 4

                   sub    edi, 1

                   cmp    edi, 0

                   jne    lbl1

                   test   edx, edx

                   jz     lbl3

                   sub    ebx, 1

                   jmp    lbl0

    lbl3:

                   pop    ebx

                   pop    ecx

                   pop    edx

                   pop    eax

                   pop    esi

                   pop    edi

                   ret    8

bubble\_sort\_32 endp

sort\_stdcall\_enum proc neg\_res:DWORD, a:DWORD, \_length:DWORD, pos\_res:DWORD,

    pos\_count:        DWORD

                      LOCAL  neg\_count:DWORD, pos\_start:DWORD, neg\_start:DWORD

                      push   ebx

                      push   esi

                      push   ecx

    ; Разделить массив на два массива: отрицательные и положительные числа

                      mov    neg\_count, 0

                      mov    eax, pos\_res

                      mov    pos\_start, eax

                      mov    eax, neg\_res

                      mov    neg\_start, eax

                      mov    eax, a

                      mov    ecx, \_length

                      inc    ecx

    \_loop:

                      dec    ecx

                      cmp    ecx, 0

                      je     \_end

                      mov    esi, dword ptr [eax]

                      cmp    esi, 0

                      jge    \_pos

                      jl     \_neg

                      jmp    \_end

    \_pos:

                      mov    ebx, dword ptr [pos\_res]

                      mov    dword ptr [ebx], esi

                      add    pos\_res, 4

                      add    eax, 4

                      mov    ebx, dword ptr [pos\_count]

                      inc    dword ptr [ebx]

                      mov    dword ptr [pos\_count], ebx

                      jmp    \_loop

    \_neg:

                      mov    ebx, dword ptr [neg\_res]

                      mov    dword ptr [ebx], esi

                      add    neg\_res, 4

                      add    eax, 4

                      inc    neg\_count

                      jmp    \_loop

    \_end:

    ; Сортировка массива положительных чисел

                      mov    ebx, dword ptr [pos\_count]

                      push   dword ptr [ebx]

                      push   pos\_start

                      call   bubble\_sort\_32

                      push   neg\_count

                      push   neg\_start

                      call   bubble\_sort\_32

                      mov    eax, neg\_count

                      pop    ecx

                      pop    esi

                      pop    ebx

                      ret    20

sort\_stdcall\_enum endp

         push   offset pos\_count

         push   offset pos\_res

         push   a\_len

         push   offset a

         push   offset neg\_res

         call   sort\_stdcall\_enum



sort\_cdecl\_enum proc c neg\_res:DWORD, a:DWORD, \_length:DWORD, pos\_res:DWORD,

    pos\_count:        DWORD

                      LOCAL  neg\_count:DWORD, pos\_start:DWORD, neg\_start:DWORD

                      push   ebx

                      push   esi

                      push   ecx

    ; Разделить массив на два массива: отрицательные и положительные числа

                      mov    neg\_count, 0

                      mov    eax, pos\_res

                      mov    pos\_start, eax

                      mov    eax, neg\_res

                      mov    neg\_start, eax

                      mov    eax, a

                      mov    ecx, \_length

                      inc    ecx

    \_loop:

                      dec    ecx

                      cmp    ecx, 0

                      je     \_end

                      mov    esi, dword ptr [eax]

                      cmp    esi, 0

                      jge    \_pos

                      jl     \_neg

                      jmp    \_end

    \_pos:

                      mov    ebx, dword ptr [pos\_res]

                      mov    dword ptr [ebx], esi

                      add    pos\_res, 4

                      add    eax, 4

                      mov    ebx, dword ptr [pos\_count]

                      inc    dword ptr [ebx]

                      mov    dword ptr [pos\_count], ebx

                      jmp    \_loop

    \_neg:

                      mov    ebx, dword ptr [neg\_res]

                      mov    dword ptr [ebx], esi

                      add    neg\_res, 4

                      add    eax, 4

                      inc    neg\_count

                      jmp    \_loop

    \_end:

    ; Сортировка массива положительных чисел

                      mov    ebx, dword ptr [pos\_count]

                      invoke bubble\_sort\_32, pos\_start, dword ptr [ebx]

                      invoke bubble\_sort\_32, neg\_start, neg\_count

                      mov    eax, neg\_count

                      pop    ecx

                      pop    esi

                      pop    ebx

                      ret

sort\_cdecl\_enum endp

         push   offset pos\_count

         push   offset pos\_res

         push   a\_len

         push   offset a

         push   offset neg\_res

         call   sort\_cdecl\_enum

         add    esp, 20



sort\_stdcall proc

                      LOCAL  \_neg\_count:DWORD, \_pos\_start:DWORD

                      LOCAL  \_neg\_start:DWORD

                      LOCAL  \_neg\_res:DWORD, \_a:DWORD, \_length:DWORD

                      LOCAL  \_pos\_res:DWORD, \_pos\_count:DWORD

                      push   ebx

                      push   esi

                      push   ecx

                      mov    eax, dword ptr [esp + 4 + 12\*4]

                      mov    \_neg\_res, eax

                      mov    eax, dword ptr [esp + 8 + 12\*4]

                      mov    \_a, eax

                      mov    eax, dword ptr [esp + 12 + 12\*4]

                      mov    \_length, eax

                      mov    eax, dword ptr [esp + 16 + 12\*4]

                      mov    \_pos\_res, eax

                      mov    eax, dword ptr [esp + 20 + 12\*4]

                      mov    \_pos\_count, eax

    ; Разделить массив на два массива: отрицательные и положительные числа

                      mov    \_neg\_count, 0

                      mov    eax, \_pos\_res

                      mov    \_pos\_start, eax

                      mov    eax, \_neg\_res

                      mov    \_neg\_start, eax

                      mov    eax, \_a

                      mov    ecx, \_length

                      inc    ecx

    \_loop:

                      dec    ecx

                      cmp    ecx, 0

                      je     \_end

                      mov    esi, dword ptr [eax]

                      cmp    esi, 0

                      jge    \_pos

                      jl     \_neg

                      jmp    \_end

    \_pos:

                      mov    ebx, dword ptr [\_pos\_res]

                      mov    dword ptr [ebx], esi

                      add    \_pos\_res, 4

                      add    eax, 4

                      mov    ebx, dword ptr [\_pos\_count]

                      inc    dword ptr [ebx]

                      mov    dword ptr [\_pos\_count], ebx

                      jmp    \_loop

    \_neg:

                      mov    ebx, dword ptr [\_neg\_res]

                      mov    dword ptr [ebx], esi

                      add    \_neg\_res, 4

                      add    eax, 4

                      inc    \_neg\_count

                      jmp    \_loop

    \_end:

    ; Сортировка массива положительных чисел

                      mov    ebx, dword ptr [\_pos\_count]

                      invoke bubble\_sort\_32, \_pos\_start, dword ptr [ebx]

                      invoke bubble\_sort\_32, \_neg\_start, \_neg\_count

                      mov    eax, \_neg\_count

                      pop    ecx

                      pop    esi

                      pop    ebx

                      ret    20

sort\_stdcall endp

         push   offset pos\_count

         push   offset pos\_res

         push   a\_len

         push   offset a

         push   offset neg\_res

         call   sort\_stdcall



sort\_cdecl proc c

                      LOCAL  \_neg\_count:DWORD, \_pos\_start:DWORD

                      LOCAL  \_neg\_start:DWORD

                      LOCAL  \_neg\_res:DWORD, \_a:DWORD, \_length:DWORD

                      LOCAL  \_pos\_res:DWORD, \_pos\_count:DWORD

                      push   ebx

                      push   esi

                      push   ecx

                      mov    eax, dword ptr [esp + 4 + 12\*4]

                      mov    \_neg\_res, eax

                      mov    eax, dword ptr [esp + 8 + 12\*4]

                      mov    \_a, eax

                      mov    eax, dword ptr [esp + 12 + 12\*4]

                      mov    \_length, eax

                      mov    eax, dword ptr [esp + 16 + 12\*4]

                      mov    \_pos\_res, eax

                      mov    eax, dword ptr [esp + 20 + 12\*4]

                      mov    \_pos\_count, eax

    ; Разделить массив на два массива: отрицательные и положительные числа

                      mov    \_neg\_count, 0

                      mov    eax, \_pos\_res

                      mov    \_pos\_start, eax

                      mov    eax, \_neg\_res

                      mov    \_neg\_start, eax

                      mov    eax, \_a

                      mov    ecx, \_length

                      inc    ecx

    \_loop:

                      dec    ecx

                      cmp    ecx, 0

                      je     \_end

                      mov    esi, dword ptr [eax]

                      cmp    esi, 0

                      jge    \_pos

                      jl     \_neg

                      jmp    \_end

    \_pos:

                      mov    ebx, dword ptr [\_pos\_res]

                      mov    dword ptr [ebx], esi

                      add    \_pos\_res, 4

                      add    eax, 4

                      mov    ebx, dword ptr [\_pos\_count]

                      inc    dword ptr [ebx]

                      mov    dword ptr [\_pos\_count], ebx

                      jmp    \_loop

    \_neg:

                      mov    ebx, dword ptr [\_neg\_res]

                      mov    dword ptr [ebx], esi

                      add    \_neg\_res, 4

                      add    eax, 4

                      inc    \_neg\_count

                      jmp    \_loop

    \_end:

    ; Сортировка массива положительных чисел

                      mov    ebx, dword ptr [\_pos\_count]

                      invoke bubble\_sort\_32, \_pos\_start, dword ptr [ebx]

                      invoke bubble\_sort\_32, \_neg\_start, \_neg\_count

                      mov    eax, \_neg\_count

                      pop    ecx

                      pop    esi

                      pop    ebx

                      ret

sort\_cdecl endp

         push   offset pos\_count

         push   offset pos\_res

         push   a\_len

         push   offset a

         push   offset neg\_res

         call   sort\_cdecl

         add    esp, 20



sort\_fastcall proc

                      LOCAL  \_neg\_count:DWORD, \_pos\_start:DWORD

                      LOCAL  \_neg\_start:DWORD

                      LOCAL  \_neg\_res:DWORD, \_a:DWORD, \_length:DWORD

                      LOCAL  \_pos\_res:DWORD, \_pos\_count:DWORD

                      push   ebx

                      push   esi

                      push   ecx

                      mov    \_neg\_res, eax

                      mov    \_a, edx

                      mov    eax, dword ptr [esp + 12 + 10\*4]

                      mov    \_length, eax

                      mov    eax, dword ptr [esp + 16 + 10\*4]

                      mov    \_pos\_res, eax

                      mov    eax, dword ptr [esp + 20 + 10\*4]

                      mov    \_pos\_count, eax

    ; Разделить массив на два массива: отрицательные и положительные числа

                      mov    \_neg\_count, 0

                      mov    eax, \_pos\_res

                      mov    \_pos\_start, eax

                      mov    eax, \_neg\_res

                      mov    \_neg\_start, eax

                      mov    eax, \_a

                      mov    ecx, \_length

                      inc    ecx

    \_loop:

                      dec    ecx

                      cmp    ecx, 0

                      je     \_end

                      mov    esi, dword ptr [eax]

                      cmp    esi, 0

                      jge    \_pos

                      jl     \_neg

                      jmp    \_end

    \_pos:

                      mov    ebx, dword ptr [\_pos\_res]

                      mov    dword ptr [ebx], esi

                      add    \_pos\_res, 4

                      add    eax, 4

                      mov    ebx, dword ptr [\_pos\_count]

                      inc    dword ptr [ebx]

                      mov    dword ptr [\_pos\_count], ebx

                      jmp    \_loop

    \_neg:

                      mov    ebx, dword ptr [\_neg\_res]

                      mov    dword ptr [ebx], esi

                      add    \_neg\_res, 4

                      add    eax, 4

                      inc    \_neg\_count

                      jmp    \_loop

    \_end:

    ; Сортировка массива положительных чисел

                      mov    ebx, dword ptr [\_pos\_count]

                      invoke bubble\_sort\_32, \_pos\_start, dword ptr [ebx]

                      invoke bubble\_sort\_32, \_neg\_start, \_neg\_count

                      mov    eax, \_neg\_count

                      pop    ecx

                      pop    esi

                      pop    ebx

                      ret    20

sort\_fastcall endp

     push   offset pos\_count

     push   offset pos\_res

     push   a\_len

     mov    edx, offset a

     mov    eax, offset neg\_res

     call   sort\_fastcall



2. Подпрограммы собрать и скомпилировать в виде dll-библиотеки. Библиотека должна содержать:

подпрограммы в стилях stdcall, cdecl, fastcall, написанные на ассемблере без явного перечисления аргументов в заголовке;

Подпрограммы в стилях stdcall, cdecl, написанные, наоборот, с перечислением аргументов в заголовке подпрограммы.

3. Подключить все подпрограммы из dll-библиотеки к проектам на C# и С++ статическим и динамическим способом. Убедиться в правильности вызова всех подпрограмм.

4. Написать подпрограмму для решения задачи варианта с использованием ассемблерной вставки на языке C++.

int sort\_ins(int\* neg\_res, int\* a, int \_length, int\* pos\_res, int\* pos\_count)

{

int neg\_count;

int \*pos\_start = pos\_res, \*neg\_start = neg\_res;

\_\_asm {

mov eax, pos\_res

mov pos\_start, eax

mov eax, neg\_res

mov neg\_start, eax

push ebx

push edi

push ecx

; Разделить массив на два массива : отрицательные и положительные числа

mov neg\_count, 0

mov eax, a

mov ecx, \_length

inc ecx

\_loop :

dec ecx

cmp ecx, 0

je \_end

mov edi, dword ptr[eax]

cmp edi, 0

jge \_pos

jl \_neg

jmp \_end

\_pos :

mov ebx, dword ptr[pos\_res]

mov dword ptr[ebx], edi

add pos\_res, 4

add eax, 4

mov ebx, dword ptr[pos\_count]

inc dword ptr[ebx]

mov dword ptr[pos\_count], ebx

jmp \_loop

\_neg :

mov ebx, dword ptr[neg\_res]

mov dword ptr[ebx], edi

add neg\_res, 4

add eax, 4

inc neg\_count

jmp \_loop

\_end :

mov ebx, dword ptr[pos\_count]

mov eax, neg\_count

pop ecx

pop edi

pop ebx

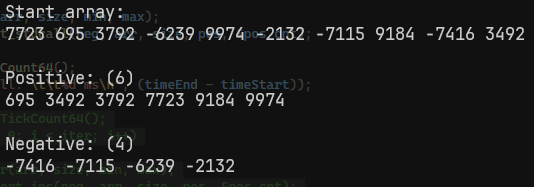
}

bubble\_sort\_32(neg\_start, neg\_count);

bubble\_sort\_32(pos\_start, \*pos\_count);

return neg\_count;

}



5. Написать подпрограммы для решения задачи варианта с использованием обычного высокоуровнего языка C# и C++ (или любого другого).

void bubbleSort(int array[], int size) {

for (int i = 0; i < size - 1; i++) {

for (int j = 0; j < size - i - 1; j++) {

if (array[j] > array[j + 1]) {

int temp = array[j];

array[j] = array[j + 1];

array[j + 1] = temp;

}

}

}

}

int split(int\* neg\_res, int\* a, int \_length, int\* pos\_res, int\* pos\_count) {

int neg\_count = 0;

for (int i = 0; i < \_length; i++) {

if (a[i] < 0) {

neg\_res[neg\_count] = a[i];

neg\_count++;

}

else {

pos\_res[\*pos\_count] = a[i];

\*pos\_count = \*pos\_count + 1;

}

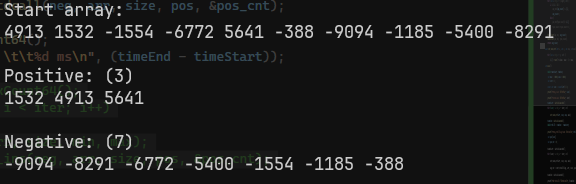
}

bubbleSort(neg\_res, neg\_count);

bubbleSort(pos\_res, \*pos\_count);

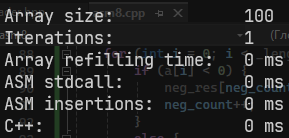
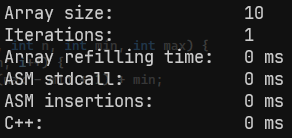
return neg\_count;

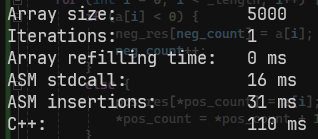
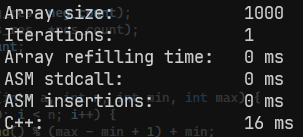
}

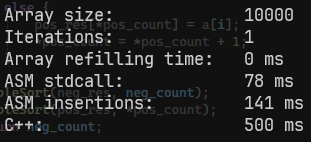
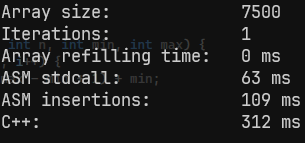


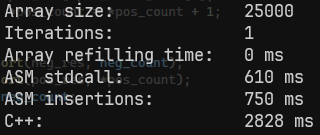
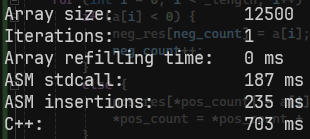
6. Сравнить скорость выполнения полученных подпрограмм на одних и тех же тестовых данных. Для сравнения выбрать:

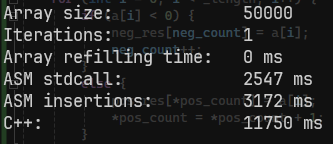
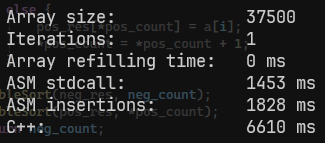
подпрограмму на ассемблере в masm32 (какую-нибудь одну из пяти), вызываемую из программы на языке C++ или C#; подпрограмму на C#; подпрограмму на C++; подпрограмму на С++ с использованием ассемблерной вставки. Построить на одной плоскости графики зависимости времени выполнения подпрограмм от длины массивов (не менее 10 точек для каждой подпрограммы). Для замера лучше передавать в подпрограммы массивы большой длины. Время замерять в миллисекундах с помощью API-функции GetTickCount(). Проверить, что подпрограммы при одинаковых тестовых данных выдают одинаковый результат. Для заполнения массивов использовать генератор случайных чисел.











**Вывод:** в ходе лабораторной работы мы изученили способы вызова подпрограмм, написанных на разных языках программирования посредством dll-библиотек.