


**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` вернуть.

```
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

```

```

-5 4 -2 3 1 0 2 -1 5 -3 4
0 1 2 3 4 4 5
-5 -3 -2 -1

```

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
```

```
-5 4 -2 3 1 0 2 -1 5 -3 4
0 1 2 3 4 4 5
-5 -3 -2 -1
```

```
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

```

```

-5 4 -2 3 1 0 2 -1 5 -3 4
0 1 2 3 4 4 5
-5 -3 -2 -1

```

```

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

```

```

-5 4 -2 3 1 0 2 -1 5 -3 4
0 1 2 3 4 4 5
-5 -3 -2 -1

```

```

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

```

```

-5 4 -2 3 1 0 2 -1 5 -3 4
0 1 2 3 4 4 5
-5 -3 -2 -1

```

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 C++ 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;
}

```

```

Start array:
7723 695 3792 -6239 9974 -2132 -7115 9184 -7416 3492

Count64():
Positive: (6)
695 3492 3792 7723 9184 9974

TickCount64():
Negative: (4)
-7416 -7115 -6239 -2132

```

5. Написать подпрограммы для решения задачи варианта с использованием обычного высокоуровневого 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;
}

```

```

Start array: size, pos, &pos_cnt);
4913 1532 -1554 -6772 5641 -388 -9094 -1185 -5400 -8291
\t\t%d ms\n", (timeEnd - timeStart));
Positive: (3)
1532 4913 5641
Negative: (7);
-9094 -8291 -6772 -5400 -1554 -1185 -388

```

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 `C++` 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.

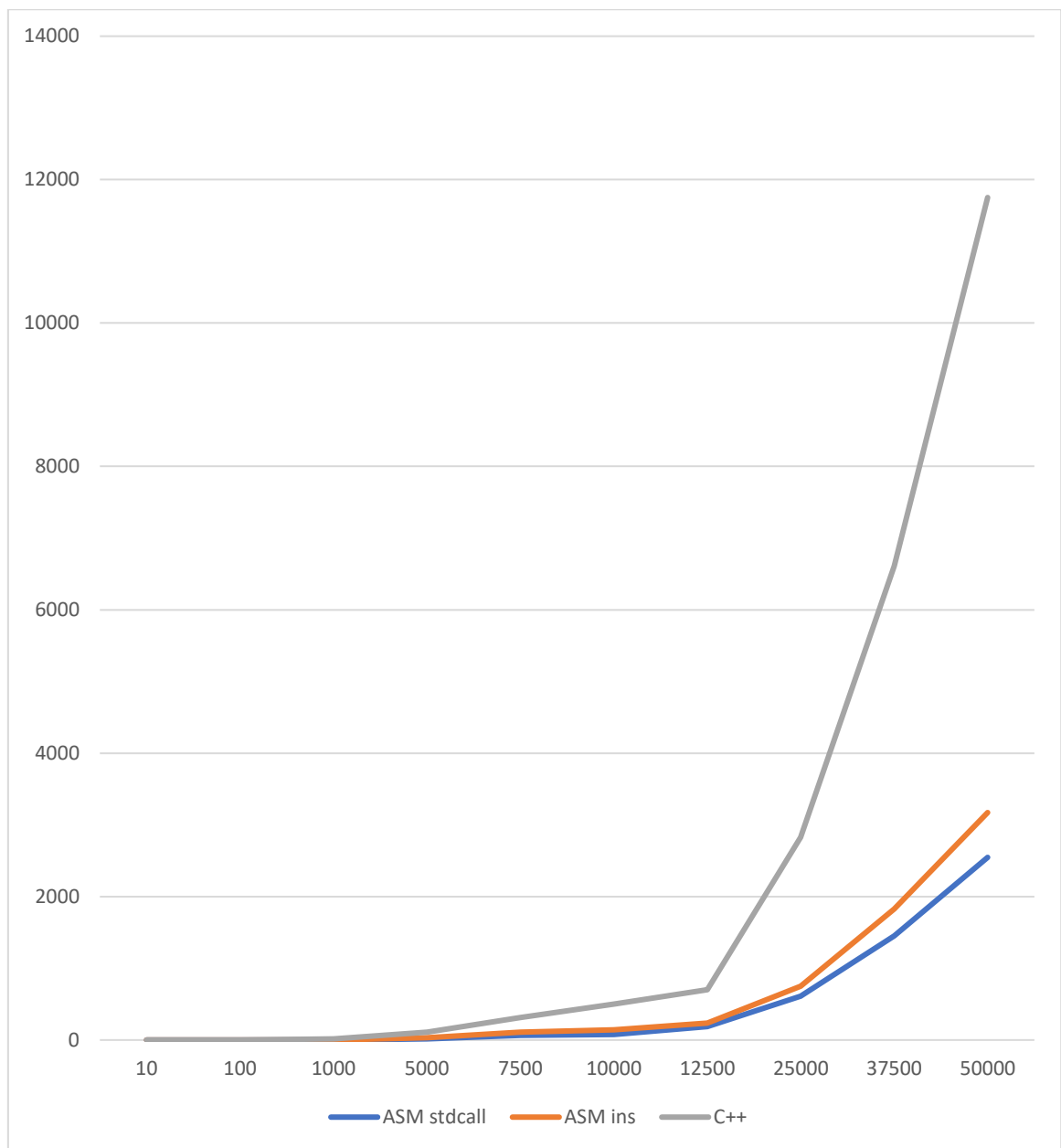
Array size:	10	Array size:	100
Iterations:	1	Iterations:	1
Array refilling time:	0 ms	Array refilling time:	0 ms
ASM stdcall:	0 ms	ASM stdcall:	0 ms
ASM insertions:	0 ms	ASM insertions:	0 ms
C++:	0 ms	C++:	0 ms

Array size:	1000	Array size:	5000
Iterations:	1	Iterations:	1
Array refilling time:	0 ms	Array refilling time:	0 ms
ASM stdcall:	0 ms	ASM stdcall:	16 ms
ASM insertions:	0 ms	ASM insertions:	31 ms
C++:	16 ms	C++:	110 ms

Array size:	7500	Array size:	10000
Iterations:	1	Iterations:	1
Array refilling time:	0 ms	Array refilling time:	0 ms
ASM stdcall:	63 ms	ASM stdcall:	78 ms
ASM insertions:	109 ms	ASM insertions:	141 ms
C++:	312 ms	C++:	500 ms

Array size:	12500	Array size:	25000
Iterations:	1	Iterations:	1
Array refilling time:	0 ms	Array refilling time:	0 ms
ASM stdcall:	187 ms	ASM stdcall:	610 ms
ASM insertions:	235 ms	ASM insertions:	750 ms
C++:	703 ms	C++:	2828 ms

Array size:	37500	Array size:	50000
Iterations:	1	Iterations:	1
Array refilling time:	0 ms	Array refilling time:	0 ms
ASM stdcall:	1453 ms	ASM stdcall:	2547 ms
ASM insertions:	1828 ms	ASM insertions:	3172 ms
C++:	6610 ms	C++:	11750 ms



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