ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

Факультет компьютерных наук
Департамент программной инженерии

Консольная программа для сортировки по возрастанию методом сравнения и перестановки элементов массива слов длиной 15 элементов.

Пояснительная записка

Исполнитель:
студент группы БПИ199
Федченко В. А.

Москва 2020

Содержание

Задание

Разработать программу сортировки по возрастанию методом попарного сравнения и перестановки элементов массива слов длиной N=15 элементовх[1]

Применяемые расчетные методы

Алгоритм

Для сортировки массива используется алгоритм сортировки «пузырьком»[2]

Псевдокод данного алгоритма:

```
ЦИКЛ ДЛЯ J=1 ДО N-1 ШАГ 1
  F=0
  ЦИКЛ ДЛЯ I=0 ДО N-1-J ШАГ 1
    ЕСЛИ A[I] > A[I+1] ТО ОБМЕН A[I],A[I+1]:F=1
  СЛЕДУЮЩЕЕ I
  ЕСЛИ F=0 ТО ВЫХОД ИЗ ЦИКЛА
СЛЕДУЮЩЕЕ J
```

Где A — массив, A[i] – i-тый элемент массива A, N – длина массива

Стоит заметить, что программная реализация данного алгоритма на языке assambler несущественно, но отличается от вышеописанного псевдокода.

Входные данные

С помощью функции *printf* пользователю выводится предложение ввести i-ый элемент массива, где I = 0,N-1; N – размер массива (15 по условию задания)

Далее с помощью функции *scanf* считывается число, введенное пользователем. После этого введенное число кладется в нужную ячейку массива.

Допустимый диапазон значений входных данных

Программа работает только с целочисленными значениями в диапазоне [-2147483648, 2147483647] (int.MinValue и int.MaxValue соответственно). При вводе целочисленных значений не лежащих в данном диапазоне, строк или не целочисленных значений корректная работа программы не гарантируется.

Выходные данные

На выходе пользователю с помощью функции *printf* выводится в консоль отсортированный по возрастанию массив из 15 элементов, введенный пользователем ранее.

Тестовые примеры

Программа корректно работает на уже отсортированном массиве (см. рис. 1)

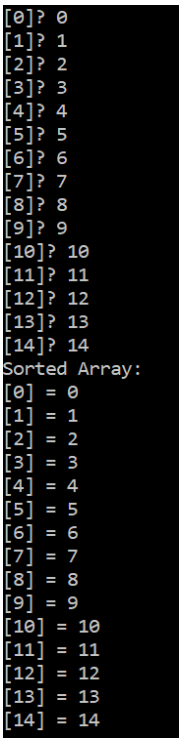

рис. 1

Программа корректно сортирует массив (см. рис. 2)

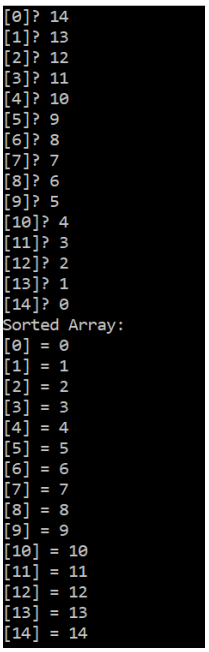

рис. 2

Программа корректно работает при нескольких вхождениях некоторых элементов в массиве (см. рис. 3)

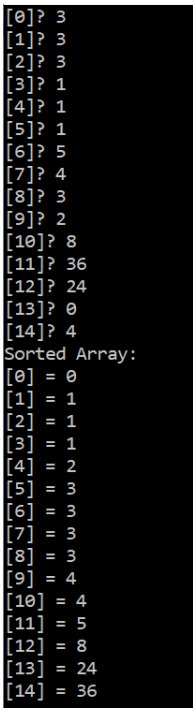

рис. 3

Программа корректно работает при наличии в массиве отрицательных элементов (см. рис. 4)

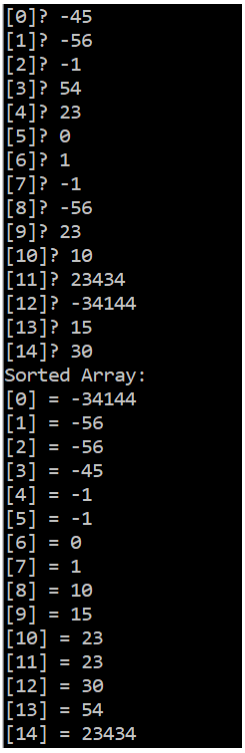

рис. 4

Программа корректно работает на всем диапазоне допустимых значений (см. рис. 5)

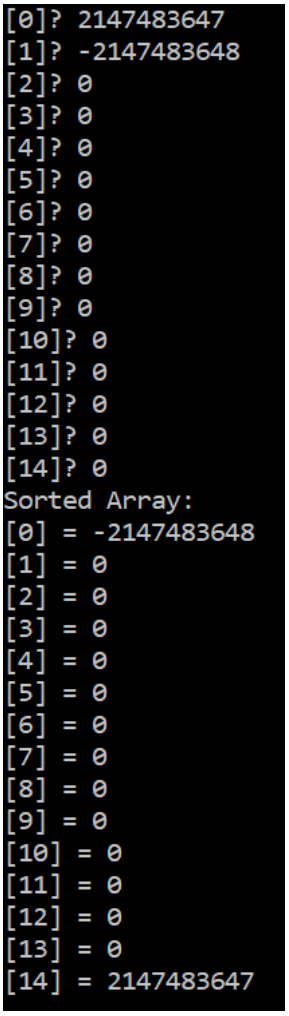

рис. 5

Список использованной литературы

[1] Программирование на языке ассемблера. Микропроект. Требования к оформлению. 2020-2021 уч.г. [Электронный ресурс]. // URL: http://softcraft.ru/edu/comparch/tasks/mp01/ (Дата обращения: 31.10.2020, режим доступа: свободный)

[2] Wikipedia. Bubble sort. [Электронный ресурс]. // URL: https://en.wikipedia.org/wiki/Bubble_sort (Дата обращения: 31.10.2020, режим доступа: свободный)

Текст программы

```
;Автор: Федченко В.А
;Группа: БПИ199
;Вариант: 26
;Задание: Разработать программу сортировки по
;возрастанию методом попарного сравнения и
;перестановки элементов массива слов длиной
;N=15 элементов

format PE console
entry start

include 'win32a.inc'
;--------------------------------------------------------------------------
section '.data' data readable writable
        strVecSize       db 'size of vector? ', 0
        strIncorSize     db 'Incorrect size of vector', 10, 0
        strIncorElelm    db 'Incorrect input', 10, 0
        strVecElemI      db '[%d]? ', 0
        strScanInt       db '%d', 0
        strVecElemOut    db '[%d] = %d', 10, 0
        strArrA          db 'Sorted Array:', 10, 0

        short_max_value  dd 32767
        short_min_value  dd -32768
        vec_size         dd 15
        size_minus_one   dd 14
        i                dd ?
        j                dd ?
        tmp              dd ?
        tmp2             dd ?
        tmpStack         dd ?
        vec              rd 15
;--------------------------------------------------------------------------
section '.code' code readable executable
start:
; 1) vector input
        call VectorInput
; 3) bubble sort vector
        call BubbleSort
; 4) vector out
        call VectorOut
```

```
finish:
        call [getch]
        push 0
        call [ExitProcess]
;--------------------------------------------------------------------------
;Код взят с сайта
;http://softcraft.ru/edu/comparch/practice/asm86/03-subprog/sum2-32/sum.asm
VectorInput:
getVector:
        xor ecx, ecx            ; ecx = 0
        mov ebx, vec            ; ebx = &vec
getVecLoop:
        mov [tmp], ebx
        cmp ecx, [vec_size]
        jge endInputVector      ; to end of loop

        ; input element
        mov [i], ecx
        push ecx
        push strVecElemI
        call [printf]
        add esp, 8

        push ebx
        push strScanInt
        call [scanf]
        add esp, 8

        ;i++
        mov ecx, [i]
        inc ecx

        ;vec.Next
        mov ebx, [tmp]
        add ebx, 4
        jmp getVecLoop
endInputVector:
        ret
;--------------------------------------------------------------------------
;Код взят с сайта
;http://softcraft.ru/edu/comparch/practice/asm86/03-subprog/sum2-32/sum.asm
VectorOut:
        mov [tmpStack], esp
```

```
        push strArrA
        call [printf]
        mov esp, [tmpStack]

        mov [tmpStack], esp
        xor ecx, ecx            ; ecx = 0
        mov ebx, vec            ; ebx = &vec
putVecLoop:
        mov [tmp], ebx
        cmp ecx, [vec_size]
        je endOutputVector      ; to end of loop
        mov [i], ecx

        ; output element
        push dword [ebx]
        push ecx
        push strVecElemOut
        call [printf]

        mov ecx, [i]
        inc ecx
        mov ebx, [tmp]
        add ebx, 4
        jmp putVecLoop
endOutputVector:
        mov esp, [tmpStack]
        ret
;------------------------------------------------------------------------
BubbleSort:
getVecSort:
        xor ecx, ecx                            ; i = 0;
        mov ebx, vec                            ; ebx = &vec
outerLoop:                                      ;
        mov [tmp2], ebx                         ; while(i < vec.Size)
        mov [i], ecx                            ; {
        cmp ecx, [vec_size]                     ;
        jge endSort     ; to end of outerLoop   ;
                                                ;
        xor edi, edi                            ;     j = 0;
                innerLoop:                      ;
                        mov [tmp], ebx          ;     while(j < vec.Size - 1)
                        mov [j], edi            ;     {
                        cmp edi, [size_minus_one]  ;
```

```
                        jge outerLoopNext          ;
                                                   ;
                        mov eax, dword[ebx]        ;         if(vec[i] < vec[i + 1])
                        cmp eax, [ebx + 4]         ;           continue;
                        jl next                    ;
                                                   ;
                        mov edx, dword[ebx]        ;         int temp = vec[i];    // На ассемблере не совсем
                        xchg edx, [ebx + 4]        ;         vec[i] = vec[i + 1];  // так, но суть такая же.
                        mov dword[ebx], edx        ;         vec[i + 1] = vec[i];
                next:                              ;
                        mov edi, [j]               ;         j++;
                        inc edi                    ;
                                                   ;
                        mov ebx, [tmp]             ;
                        add ebx, 4                 ;
                        jmp innerLoop              ;     }
                                                   ;
outerLoopNext:                                     ;
        mov ebx, [tmp2]                            ;
        mov ecx, [i]                               ;     i++;
        inc ecx                                    ;
                                                   ;
        jmp outerLoop                              ;}
endSort:
        ret
;------------------------------third act - including HeapApi-------------------------
section '.idata' import data readable
    library kernel, 'kernel32.dll',\
            msvcrt, 'msvcrt.dll',\
            user32,'USER32.DLL'

include 'api\user32.inc'
include 'api\kernel32.inc'
    import kernel,\
           ExitProcess, 'ExitProcess',\
           HeapCreate,'HeapCreate',\
           HeapAlloc,'HeapAlloc'
  include 'api\kernel32.inc'
    import msvcrt,\
           printf, 'printf',\
           scanf, 'scanf',\
           getch, '_getch'
```