НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

*Звіти до комп’ютерних практикумів дисципліни*

«Системне програмне забезпечення»

**Прийняв Виконав**

**доцент кафедри ІПІ Студент групи ІП-21**

**Лісовиченко О.І. Скрипець О.О.**

**“06” травня 2024 р.**

Київ – 2024

**Комп’ютерний практикум №3**

**Тема:** програмування розгалужених алгоритмів.

**Завдання**:

1. Написати програму, яка повинна мати наступний функціонал:

1. Можливість введення користувачем розміру одномірного масиву.

2. Можливість введення користувачем значень елементів одномірного масиву.

3. Можливість знаходження суми елементів одномірного масиву.

4. Можливість пошуку максимального (або мінімального) елемента одномірного масиву.

5. Можливість сортування одномірного масиву цілих чисел загального вигляду.

Текст програми

sseg *segment* para stack 'stack'

    db 64 dup ( 'STACK' )

sseg *ends*

dseg *segment* para public 'data'

    array              dw 50 dup(0)

    arraySize          dw 0

    number             dw 0

    temp               dw 0

    is\_negative        dw 0

    minLenArr          dw 1

    maxLenArr          dw 50

    maxNumber          dw 0

    enter\_arr\_size\_msg db 0dh, 0ah, 'enter length of array [1-50]: $'

    min\_error\_msg      db 0dh, 0ah, 'array has to have 1 element $'

    max\_error\_msg      db 0dh, 0ah, 'array can`t have more than 50 elements $'

    enter\_element\_msg  db 0dh, 0ah, 'enter in range of -32767 to 32767 : $'

    array\_elements\_msg db 0dh, 0ah, 'Sorted array : $'

    max\_element\_msg    db 0dh, 0ah, 'Max element: $'

    sum\_msg            db 0dh, 0ah, 'Sum of elements: $'

    overflow\_error     db 'Overflow $'

    input\_error\_msg    db 0dh, 0ah, 'the number you entered is too large$'

    out\_of\_range\_msg   db 0dh, 0ah, 'Number is out of range (-32768 to 32767)$'

    wrong\_char\_msg     db 0dh, 0ah, 'wrong character$'

    empty\_input\_msg    db 0dh, 0ah, 'empty input$'

    space              db ' $'

dseg *ends*

cseg *segment* para public 'code'

print *macro* string

                            push ax

                            lea dx,string

                            mov ah,09h

                            int 21h

                            pop ax

*endm*

main *proc*

    main\_loop:

                            assume cs: cseg, ds: dseg, ss: sseg

                            mov ax, dseg

                            mov ds, ax

                            call enter\_array

                            call print\_array

                            call sum\_of\_array

                            call find\_max\_element

                            call sort

                            call print\_array

                            jmp    main\_loop

main *endp*

enter\_array *proc*

                            print enter\_arr\_size\_msg

                            call read\_digit

                            mov ax, number

                            mov arraySize, ax ;встановлює arraySize рівним зчитаному числу

                            call array\_length ;перевіряє, чи введений розмір масиву відповідає допустимому діапазону

                            mov cx, ax ;встановлює розмір масиву в регістр cx

                            xor bx, bx

                            lea si, array ;завантажує в регістр si адресу масиву

    read\_array\_loop:

                            push cx ;зберігає кількість елементів які потрібно зчитати

                            print enter\_element\_msg

                            call read\_digit

                            mov ax, number

                            mov [si], ax

                            add si, 2 ; збільшення на 2 байти

                            pop cx

                            loop read\_array\_loop

                            ret

enter\_array *endp*

array\_length *proc*

                            xor ax,ax

                            mov ax,arraySize

                            cmp ax, minLenArr

                            jl min\_error

                            cmp ax, maxLenArr

                            jg max\_error

                            ret

    min\_error:

                            print min\_error\_msg

                            call main\_loop

    max\_error:

                            print max\_error\_msg

                            call main\_loop

array\_length *endp*

read\_digit *proc*

                            mov number,0

                            mov bx, 10

                            mov cx, 5

                            mov is\_negative,0

    read:

                            xor ax, ax

                            mov ah, 01h

                            int 21h

                            cmp al, 13

                            je stop

                            cmp al, 48

                            jl check\_sign

                            cmp al, 57

                            ja wrong\_character

                            sub al, '0'

                            sub ah, ah

                            mov temp, ax

                            mov ax, number

                            imul bx

                            jo input\_error

                            add ax, temp

                            jo input\_error

                            mov number, ax

                            loop read

                            cmp is\_negative, 1

                            je make\_negative

                            ret

    stop:

                            cmp cx, 5

                            je empty\_input

                            mov cx, 0

                            cmp is\_negative, 1

                            je make\_negative

                            ret

    make\_negative:

                            neg number

                            ret

    check\_sign:

                            cmp al, '-'

                            jne wrong\_character

                            cmp cx, 5

                            jne wrong\_character

                            mov is\_negative, 1

                            jmp read

    input\_error:

                            print input\_error\_msg

                            call main\_loop

    wrong\_character:

                            print wrong\_char\_msg

                            call main\_loop

    empty\_input:

                            print empty\_input\_msg

                            call main\_loop

read\_digit *endp*

find\_max\_element *proc*

                            xor si,si

                            xor dx,dx

                            mov dx, array[si] ;переміщує перший елемент масиву в регістр dx

                            mov maxNumber, dx ;встановлює maxNumber  рівним першому елементу масиву

                            mov cx, arraySize ;встановлює розмір масиву в регістр cx

    compare:

                            mov ax, array[si] ;зчитує поточний елемент масиву в регістр ax

                            cmp maxNumber, ax ;порівнює поточний елемент масиву з поточним максимальним числом

                            jg no\_change ;якщо менший

                            mov maxNumber, ax ;якщо більший

    no\_change:

                            add si, 2 ;переходимо до наступного елементу масиву

                            loop compare

    print max\_element\_msg

                            mov ax, maxNumber

                            call print\_result

                            ret

find\_max\_element *endp*

sort *proc*

                            push ax ;зберігають поточні значення регістрів на вершині стеку

                            push bx

                            push cx

                            push dx

                            push di

                            xor si,si

                            lea si, array ;вказує на масив

                            xor bx,bx

                            mov bx, arraySize

                            mov ax, si ; ax- поточний елемент

                            mov cx, bx ; сх - лічильник зовнішнього циклу (відповідає розміру масиву)

                            dec cx

    sort\_outer\_loop:

                            mov bx,cx ;bx- лічильник внутрішнього циклу

                            mov si,ax ;si поточний елемент

                            mov di,ax ;di наступний

                            add di,2

    sort\_inner\_loop:

                            mov dx,[si] ; зчитується поточний елемент

                            cmp dx, [di] ; зчитується наступний елемент

                            jl skip ; якщо поточний менший, то пропускаємо обмін

                            xchg dx,[di] ;якщо поточний більший

                            mov [si],dx ;обмінюються

    skip:

                            add si,2 ;перехід до наступного елементу

                            add di,2 ;перехід до наступного елементу

                            dec bx ;зменшення лічильника на 1

                            jnz sort\_inner\_loop ;якщо він не 0 повторення внутрішнього циклу

                            loop sort\_outer\_loop

                            pop di

                            pop dx

                            pop cx

                            pop bx

                            pop ax

                            ret

sort *endp*

sum\_of\_array *proc*

                            .386 ;директива асемблера? дозволяє використовувати 32-бітні регістри

                            mov cx, arraySize

                            xor ax, ax

                            xor bx, bx

                            xor si, si

    sum:

                            mov ax, array[si] ;зчитує поточний елемент масиву в регістр ax

                            cwde ;перетворює 16-бітне значення в регістрі ax на 32-бітне значення в регістрі eax потрібно для того, щоб можна було виконувати операції з 32-бітними числами.

                            add ebx, eax ;ebx використовується для зберігання проміжного результату суми.

                            add si, 2

                            loop sum

                            mov eax, ebx

    print sum\_msg

                            call print\_result

                            ret

    overflow1:

                            print overflow\_error

                            ret

sum\_of\_array *endp*

print\_array *proc*

                            print array\_elements\_msg

                            mov cx, arraySize ;встановлює розмір масиву в регістр cx

                            xor si, si

    display\_loop:

                            mov ax, array[si] ;зчитує поточний елемент масиву в регістр ax

                            push cx ;зберігає поточне значення регістра cx (кількість елементів, які потрібно вивести) на вершині стеку

                            push ax ; зберігає поточний елемент масиву на вершині стеку

                            print space

                            pop ax ;відновлює оригінальне значення регістра ax (поточний елемент масиву) з вершини стеку

                            call print\_result

                            pop cx ;відновлює оригінальне значення регістра cx (кількість елементів, які потрібно вивести) з вершини стеку

                            add si, 2

                            loop display\_loop

                            ret

print\_array *endp*

print\_result *proc*

                              mov bx, ax

                                or bx,bx

                              jns    positive\_res

                              mov    al, '-'

                              int    29h

                              neg    bx

    positive\_res:

                              mov    ax, bx

                              xor    cx, cx

                              mov    bx, 10

    transform\_res:

                              xor    dx, dx

                              div    bx

                              add    dl, '0'

                              push   dx

                              inc    cx

                              test   ax, ax

                              jnz    transform\_res

    print\_res:

                              pop    ax

                              int    29h

                              loop   print\_res

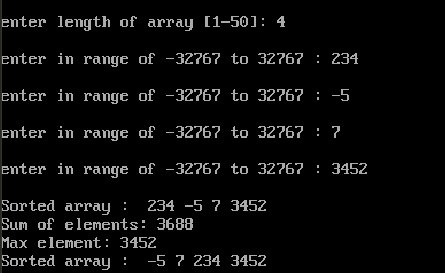
                              ret

print\_result *endp*

cseg *ends*

end main

**Перевірки:**



**Висновок:**

Висновок до лабораторної роботи:

У цій лабораторній роботі ми розглянули створення програми, яка включає в себе введення даних користувачем, створення одномірного масиву, обчислення суми елементів масиву, пошук максимального (або мінімального) елемента масиву, а також сортування масиву.

Ця робота також підкреслила важливість ретельного тестування та налагодження коду, щоб забезпечити його коректну роботу в усіх можливих сценаріях використання. Загалом, ця лабораторна робота була корисною для поглиблення наших знань про програмування та розуміння того, як можна ефективно структурувати та організувати код.

Ця робота також допомогла нам краще зрозуміти, як можна використовувати програмування для вирішення реальних проблем та виконання реальних завдань, що є важливим аспектом навчання програмуванню.