МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ДЕРЖАВНИЙ ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД

«УЖГОРОДСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ»

ІНЖЕНЕРНО-ТЕХНІЧНИЙ ФАКУЛЬТЕТ

КАФЕДРА КОМП’ЮТЕРНИХ СИСТЕМ ТА МЕРЕЖ

**КУРСОВА РОБОТА**

з дисципліни **«**Системне програмування**»**

на тему:

«Програмна реалізація алгоритму сортування – Bubble sort (сортування бульбашкою)»

Студента 2-го курсу

Напрям підготовки 6.050102

«Комп’ютерна нженерія»

Головкевича Валентина

Керівник: ст.викл. Далекорей А.В.

Національна шкала\_\_\_\_\_\_\_\_\_\_

Кількість балів: \_\_ Оцінка: ECTS\_\_

Члени комісії \_\_\_\_\_\_\_\_\_\_ ст.викл. Самусь Є.І.

\_\_\_\_\_\_\_\_\_\_ ст.викл. Далекорей А.В.

**ЗМІСТ**

ВСТУП …………………………………..………………………….………... 3

1 ОСНОВНІ ПОНЯТТЯ ПРО СОРТУВАННЯ МАСИВІВ …….…………. 5

1.1 Алгоритми сортування ……………………………….................…… 5

1.2 Алгоритм сортування бульбашкою …………………………….….... 6

2 РОЗРОБКА ПРОГРАМИ ………………………………………………….. 7

2.1 Опис алгоритму ....………...………………………………….………. 7

2.2 Інструкція для користувача ……..………………….………………... 9

2.3 Тестові приклади …………………..…………...………………...…. 13

ВИСНОВКИ ……………………………………………….………....…...… 15

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ ………….…........................... 16

ДОДАТКИ …………………………………………………………….....….. 17

ДОДАТОК А. Лістинги програми .………………….………….....…… 17

ДОДАТОК Б. Блок-схема алгоритму …………….…………….....…… 25

ІНДИВІДУАЛЬНЕ ТЕХНІЧНЕ ЗАВДАННЯ

**Тема:** Програмна реалізація алгоритму сортування – Bubble sort (сортування бульбашкою).

Мета: Розробити програму на мові Асемблер та отримати необхідні навички програмування на даній мові.

Вимоги до виконання роботи:

1. Розробити алгоритм отриманий згідно варіанта.
2. Розробити процедури для:

* Зчитування даних з клавіатури та файлу;
* Реалізації алгоритму;
* Виводу результатів на екран та запису в файл;

1. Мова розробки – Асемблер.
2. Наявність користувацького інтерфейсу.
3. Передбачити перевірку коректності введених даних.
4. Передбачити перевірку коректності даних, які виводяться.
5. Передбачити можливість вибору початкових даних із файлу.
6. Реалізувати можливість зберігати дані у файл.

**ВСТУП**

В наш час нові інформаційні технології посідають дуже важливе місце не тільки в спеціалізованих, але й в повсякденних сферах життя. Комп'ютери застосовуються в бізнесі, менеджменті, торгівлі, навчанні та багатьох інших сферах діяльності людини.

Комп'ютерні технології дуже зручні для виконання різноманітних операцій, але в різних сферах застосування ці операції різні. Тому, кожна окрема галузь, яка використовує специфічні технічні засоби, потребує свої власні програми, які забезпечують роботу комп'ютерів.

Розробкою програмного забезпечення займається така галузь науки, як програмування. Вона набуває все більшого і більшого значення останнім часом, адже з кожним днем ​​комп'ютер стає все більш необхідним, все більш повсякденним явищем нашого життя. Адже обчислювальна техніка минулих років вже майже повністю вичерпала себе і не задовольняє тим потребам, які з'являються перед людством.

Таким чином, нові інформаційні технології дуже актуальні в наш час і потребують більшої уваги для подальшої розробки та вдосконалення. Поруч з цим, велике значення має також і програмування, яке є одним з фундаментальних розділів інформатики і тому не може залишатися осторонь.

Програмування містить цілий ряд важливих внутрішніх завдань. Однією з найбільш важливих завдань для програмування є задача сортування. Під сортуванням звичайно розуміють перестановки елементів будь-якої послідовності в певному порядку. Це завдання є однією з найважливіших тому, що її метою є полегшення подальшої обробки певних даних і, в першу чергу, завдання пошуку. В даній курсовій ми розглянемо і реалізуємо на мові Асемблер метод сортування бульбашкою.

Пояснювальна записка складається з пяти параграфів, а саме: індивідуальне технічне завдання, теоретична частина, опис алгоритму, інструкція для користувача, тестові приклади.

В параграфі «Індивідуальне технічне завдання» описано тему, мету, та вимоги до виконання роботи. Параграф «Теоретична частина» містить основні теоретичні відомості про сортування бульбашкою. З структурою алгоритму даного методу можна ознайомитись в параграфі «Опис алгоритму».

Для того щоб було легше користуватися програмою створений параграф «Інструкція для користувача». В параграфі «Тестові приклади» показані декілька прикладів сортування масиву в програмі створеній на мові Асемблер.

Також є окремий розділ – «Додатки» в котрому є лістинги програми.

1. **ТЕОРЕТИЧНА ЧАСТИНА**

**1.1 Алгоритм сортування**

*Алгоритм сортування* - це алгоритм для впорядкування елементів в списку. У разі, коли елемент списку має кілька полів, поле за яким здійснюється сортування, називається ключем сортування. На практиці, як ключ часто виступає число, а в інших полях зберігаються будь-які дані, що не впливають на роботу алгоритму.

Мабуть, жодна інша проблема не створила такої кількості найрізноманітніших рішень, як завдання сортування. Чи існує якийсь «універсальний», найкращий алгоритм? Взагалі кажучи, немає. Однак, маючи приблизні характеристики вхідних даних, можна підібрати метод, який працює оптимальним чином.

Оцінка алгоритму сортування. Для того, щоб обгрунтовано зробити такий вибір, розглянемо параметри, за якими буде проводитися оцінка алгоритмів.

*Час сортування* - основний параметр, що характеризує швидкодію алгоритму. Називається також обчислювальною складністю. Для сортування важливі найгірше, середнє і краще поведінки алгоритму в термінах розміру списку . Для типового алгоритму хорошу поведінку - це і погану поведінку - це . Ідеальну поведінку для сортування - . Алгоритми сортування, які використовують тільки абстрактну операцію порівняння ключів, завжди потребують, щонайменше, в порівняннях в середньому;

*Пам'ять* - ряд алгоритмів вимагає виділення додаткової пам'яті під тимчасове зберігання даних. При оцінці використовуваної пам'яті не буде враховуватися місце, яке займає вихідний масив і не залежать від вхідної послідовності витрати, наприклад, на зберігання коду програми.

*Стійкість (stability)* - стійка сортування не змінює взаємного розташування рівних елементів. Така властивість може бути дуже корисним, якщо вони складаються з декількох полів, а сортування відбувається по одному з них.

*Природність поведінки* - ефективність методу при обробці вже відсортованих, або частково впорядкованих даних. Алгоритм поводиться природно, якщо враховує цю характеристику вхідної послідовності і працює краще.

Ще однією важливою властивістю алгоритму є його сфера застосування. Тут основних типів сортування дві:

Внутрішнє сортування оперує з масивами, які цілком поміщаються в оперативній пам'яті з довільним доступом до будь-якого середовища. Дані зазвичай сортуються на тому ж місці, без додаткових витрат.

Зовнішнє сортування оперує з пристроями, що запам'ятовують великого обсягу, але з доступом не довільно, а послідовним (сортування файлів), тобто в даний момент ми «бачимо» тільки один елемент, а витрати на перемотування в порівнянні з пам'яттю невиправдано великі. Це накладає деякі додаткові обмеження на алгоритм і призводить до спеціальних методів сортування, зазвичай використовують додатковий дисковий простір.

**1.2 Сортування бульбашкою**

*Сортування бульбашкою (англ. Bubble sort)* - простий алгоритм сортування. Алгоритм складається в повторюваних проходах по сортованого масиву. За кожен прохід елементи послідовно порівнюються попарно і, якщо порядок у парі невірний, виконується обмін елементів. Проходи по масиву повторюються до тих пір, поки на черговому проході не опиниться, що обміни більше не потрібні, що означає - масив відсортований. При проході алгоритму, елемент, що стоїть не так на своєму місці, «спливає» до потрібної позиції як бульбашка у воді, звідси і назва алгоритму.

Сортування методом бульбашки має складність . Для розуміння і реалізації це - найпростіший алгоритм сортування, але ефективний він лише для невеликих масивів.

# РОЗРОБКА ПРОГРАМИ

**2.1 Опис алгоритму**

Візьмемо масив чисел «», і за допомогою даного алгоритму, відсортуємо його від найменшого до найбільшого елементу. На кожному кроці, елементи, виділені **жирним** шрифтом будуть порівнюватись.

**Перша ітерація:**

- порівнює перші два елементи, і міняє їх місцями. - тут порівнювані елементи знаходяться на своїх місцях, тож алгоритм не міняє їх місцями.

**Друга ітерація:**

Тепер наш масив повністю відсортований, однак, алгоритм цього ще не знає. Йому потрібен ще один «пустий» прохід, під час якого він не поміняє місцями жодного елементу.

**Третя ітерація:**  
Нарешті, масив відсортовано, і алгоритм може припинити свою роботу.

**2.1 Інструкція для користувача**

1. Ми відкрили папку «Курсова робота». Тепер відкриваємо папку *«bubble\_sort»* (див. рисунок 1):

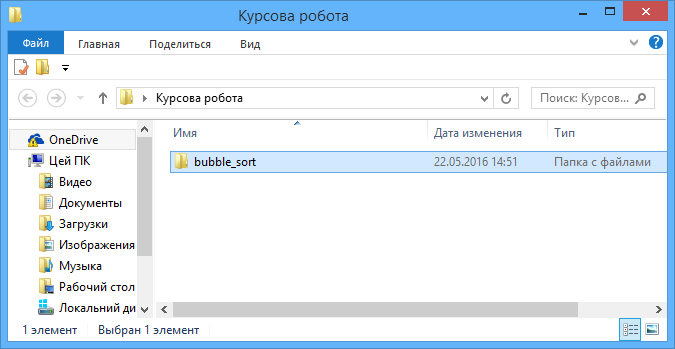


Рисунок 1 - Вікно папки «Курсова робота»

1. Далі запускаємо «*start.bat*» (див. рисунок 2):

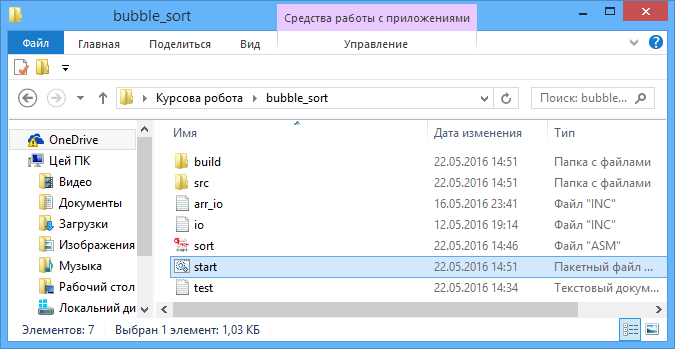


Рисунок 2 - Вікно - папки «bubble\_sort»

1. Відкривається вікно програми. Одразу вибераємо, яким чином будомо вводити дані до програми: з файлу (натисніть «1») чи з клавіатури (натисніть «2»). Обераємо введення з файлу. (див. рисунок *3 і*  рисунок *4* ):

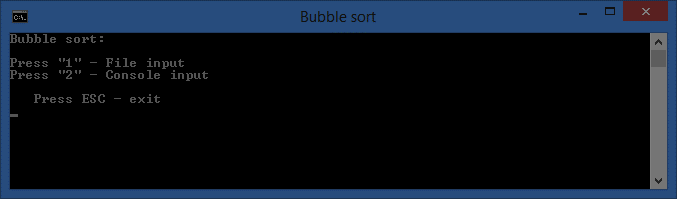


Рисунок 3 - Початкове вікно програми

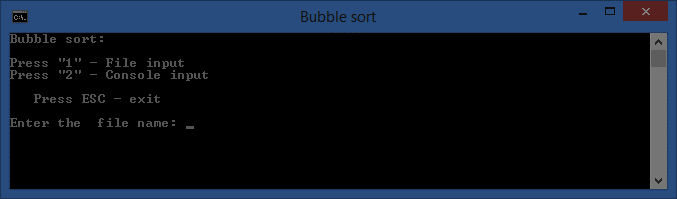


Рисунок 4 - Введення імені файлу

1. Далі вводимо ім’я файлу, з якого будемо зчитувати дані і програма виводить кількість елементів масиву, початковий і відсортований масив. (див. рисунок 5 і рисунок 6):

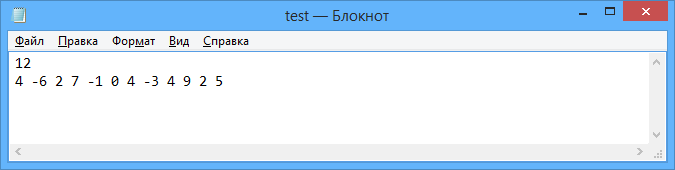


Рисунок 5 - Вікно файлу «test.txt» з якого зчитуємо дані

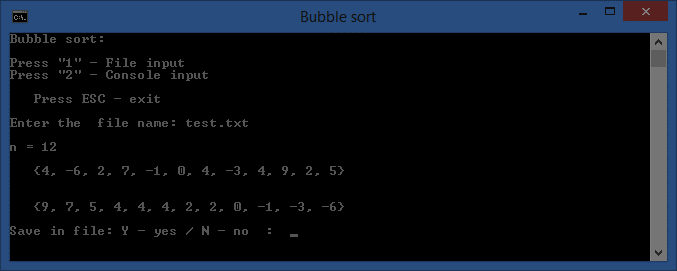


Рисунок 6 - Виведення відсортованого масиву

1. Далі, для того щоб зберегти відсортований масив елементів у новий файл натискаємо «y» і вводимо назву цього нового файлу (див. рисунок 7):

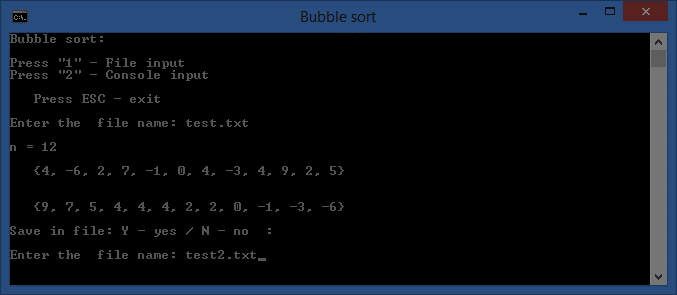


Рисунок 7 - Введення назви нового файла

1. Новий файл збережено у папці «bubble\_sort». Натискаємо будь-яку клавішу для виходу з програми. (див. рисунок 8 і рисунок 9):

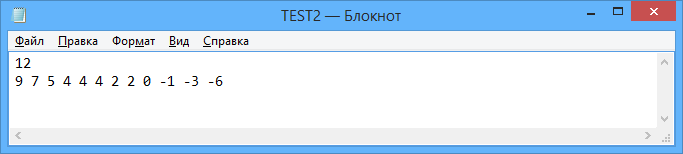


Рисунок 8 - Вікно файлу «test2.txt» в якому зберігаємо дані

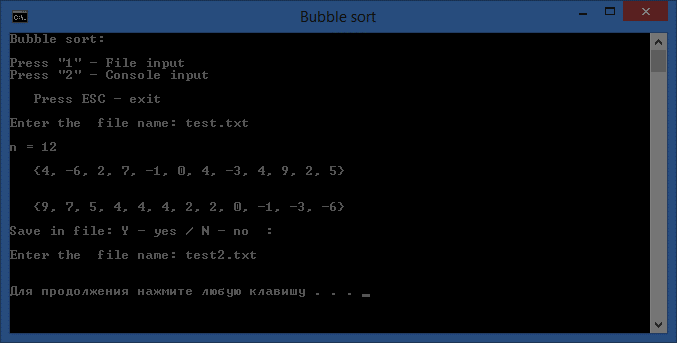
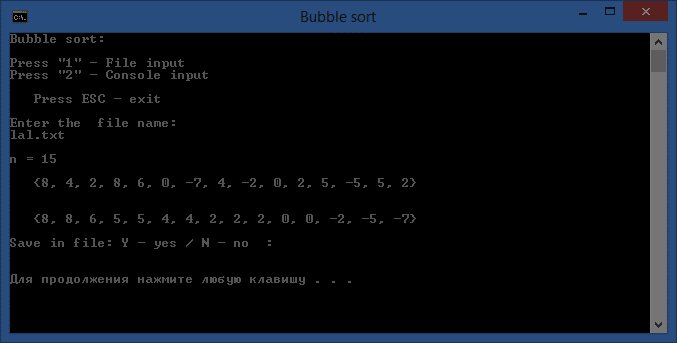


Рисунок 9 - Вихід з програми

**2.2 Тестові приклади**

1. Відсортувати масив:

Результат сортування в програмі (*Рис. 10*):

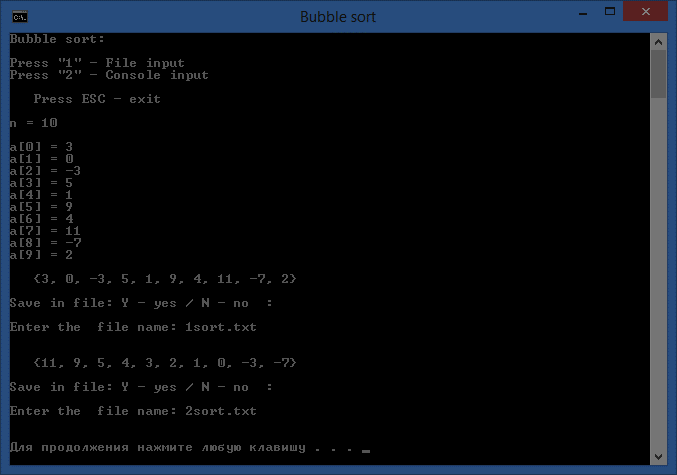


*Рис. 10*

Як бачимо, програма відсортувала правильно.

1. Відсортувати масив:

Сортування за допомогою програми: (*Рис.11*):



*Рис. 11*

Як бачимо, програма відсортувала правильно.

**ВИСНОВКИ**

В результаті виконання курсової роботи було розроблено програму на мові програмування Асемблер. Дана програма реалізовує сортування бульбашкою.

Було детально описано теоретичну частину і зроблено повний опис алгоритму. Розроблена інструкція для користувача, яка покроково пояснює, де можна знайти програму, як її запускати і використовувати. Також проілюстровано два приклади, які показують, що програма працює правильно.

**СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ**

1. **Вирт Н**. Алгоритмы и структуры данных М.: Мир. – 1989

2.

**ДОДАТКИ**

ДОДАТОК **А. Лістинги програми**

.model small

.386

.stack 3000h

.data

head db "Bubble sort:", endl, 0

str\_fin db 'Press "1" - File input', endl, 0

str\_cin db 'Press "2" - Console input', endl, 0

str\_n db 'n = ', 0

str\_id db ' Invalid data!', endl, 0

str\_nf db ' Not found!', endl, 0

exit\_tab db endl, ' Press ESC - exit', endl, 0

str\_arr\_beg db 'a[', 0

str\_arr\_end db '] = ', 0

str\_value db 'value = ', 0

file\_err db 'not exist!', endl, 0

prompt db 'Enter the file name: ', 0

str\_save db 'Save in file: Y - yes / N - no : ', 0

len dw ?

n dw 0

value dw 0

arr dw ?

handle dw ?

file\_name db 80 dup(0)

arr\_out db 80 dup(0)

tmp db 80 dup(0)

.code

LOCALS

include arr\_io.inc

;============================

cmpnum proc c

arg @a, @b

uses bx, si

mov si, @a

mov ax, word ptr[si]

mov si, @b

mov bx, word ptr[si]

cmp ax, bx

je @@ecv

jl @@less

mov ax, 1

jmp @@end

@@ecv:

xor ax, ax

jmp @@end

@@less:

mov ax, -1

@@end:

ret

cmpnum endp

;============================

;============================

; void buble\_sort(void \*arr, size\_t arr\_len, size\_t size\_elem, int (\*comparator)(const \*void, const \*void))

buble\_sort proc c

arg @arr, @arr\_len, @size\_elem, @comparator

uses ax, bx, cx, dx, si, di

mov cx, @arr\_len

@@cycle1:

cmp cx, 0

je @@end

mov dx, @arr\_len

dec dx

mov bx, @arr

@@cycle2:

cmp dx, 0

je @@ns0

mov si, bx

add si, @size\_elem

push bx

push si

call @comparator

add sp, 4

cmp ax, 0

jng @@ns1

push cx

mov cx, @size\_elem

mov di, bx

@@cycle3:

cmp cx, 0

je @@ns2

mov al, [di]

mov ah, [si]

mov [di], ah

mov [si], al

inc si

inc di

dec cx

jmp @@cycle3

@@ns2:

pop cx

@@ns1:

dec dx

add bx, @size\_elem

jmp @@cycle2

@@ns0:

dec cx

jmp @@cycle1

@@end:

ret

buble\_sort endp

;============================

;============================

file\_save proc c

uses ax

puts str\_save

@@fsave:

call \_getch

cmp al, 'y'

je @@save

cmp al, 'Y'

je @@save

cmp al, 'n'

je @@end

cmp al, 'N'

je @@end

jmp @@fsave

@@save:

putc endl

putc endl

puts prompt

push offset file\_name

call gets

add sp, 2

push WO

push offset file\_name

call fopen

add sp, 4

mov handle, ax

push 10

push offset tmp

push n

call itoa

add sp, 6

push handle

push offset tmp

call fputs

add sp, 4

fputc endl, handle

push handle

push n

push arr

call arr\_ifout

add sp, 6

push handle

call fclose

add sp, 2

@@end:

putc endl

putc endl

ret

file\_save endp

;============================

;============================

main proc

mov ax, @data

mov ds, ax

@@beg\_app:

; menu

puts head

putc endl

puts str\_fin

puts str\_cin

puts exit\_tab

@@check\_input:

call \_getch

cmp al, 31h

je @@fin

cmp al, 'f'

je @@fin

cmp al, 'F'

je @@fin

cmp al, 32h

je @@cin

cmp al, 'c'

je @@cin

cmp al, 'C'

je @@cin

cmp ah, 1

jne @@check\_input

\_exit

; file input

@@fin:

putc endl

puts prompt

push offset file\_name

call gets

add sp, 2

putc endl

putc endl

push RO

push offset file\_name

call fopen

add sp, 4

test ax, ax

jne @@ns2

putc ' '

putc ' '

putc ' '

putc '"'

puts file\_name

putc '"'

putc ' '

puts file\_err

putc endl

jmp @@beg\_app

@@ns2:

mov handle, ax

push offset tmp

push handle

call ifin

add sp, 4

cmp bl, 0

je @@ns3 ; file open

; error

puts str\_id

jmp @@beg\_app

@@ns3:

mov n, ax

add ax, ax

mov len, ax

sub sp, ax

mov arr, sp

puts str\_n

puts tmp

putc endl

putc endl

push handle

push n

push arr

call arr\_ifin

add sp, 6

push ax

putc endl

pop ax

cmp al, 0

je @@alg

; error

putc endl

puts str\_id

; memory free

add sp, len

; goto @@beg\_app

jmp @@beg\_app

; console input

@@cin:

putc endl

push offset str\_n

call icin

add sp, 2

mov n, ax

; reserved memory

add ax, ax

mov len, ax

sub sp, ax

mov arr, sp

putc endl

; input arr

push n

push arr

call arr\_icin

add sp, 4

putc endl

push n

push arr

call arr\_icout

add sp, 4

putc endl

call file\_save

@@alg:

putc endl

push offset cmpnum

push 2

push n

push arr

call buble\_sort

add sp, 8

push n

push arr

call arr\_icout

add sp, 4

putc endl

call file\_save

putc endl

@@end:

; memory free

add sp, n

add sp, n

@@end\_app:

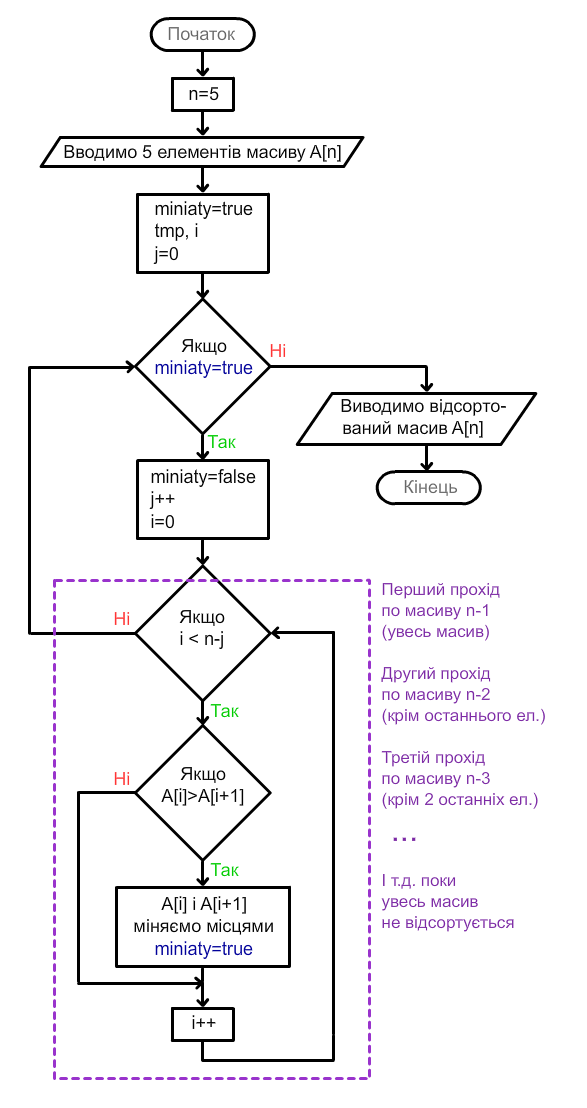
\_exit

main endp

;============================

end main

ДОДАТОК **Б. Блок-схема алгоритму**

****