Міністерство освіти і науки України

Дніпропетровський національний університет

імені Олеся Гончара

ФФЕКС

Кафедра ЕОМ

**Звіз з лабораторної роботи № 5**

Виконав: студент 2 курсу

Шейко Р.О.

Науковий керівник:

Викладач кафедри EOM

Спірінцева Ольга Володимирівна

м. Дніпро

2022

Завдання

1. Визначити масив, елементи якого впорядковуватимуться. Тип масиву вибрати за таблицею №1 – масив №1.
2. Розробити функцію сортування масиву методами, вибраними за таблицею 3 відповідно до варіанта.
3. Будь-яким способом заповнити елементи масиву значення-ми.
4. Виконати сортування масиву першим алгоритмом та проконтролювати її результат. Перевірити всі варіанти вихідного заповнення масиву: випадковим чином відсортованого у зворотному порядку, відсортованого у потрібному порядку. Переконатись у правильності сортування у всіх випадках. Зробити висновки.
5. Повторити пункти 3 та 4 для другого алгоритму сортування

Хід роботи

1. Масив типу long;
2. Швидке сортування:

void swap(long\* a, long\* b) {

long t = \*a;

\*a = \*b;

\*b = t;

}

int partition(long array[], long low, long high) {

long pivot = array[high];

long i = (low - 1);

for (long j = low; j < high; j++) {

if (array[j] <= pivot) {

i++;

swap(&array[i], &array[j]);

}

}

swap(&array[i + 1], &array[high]);

return (i + 1);

}

void quickSort(long array[], long low, long high, int size) {

printArray(array, size);

Sleep(1000);

if (low < high) {

int pi = partition(array, low, high);

quickSort(array, low, pi - 1, size);

quickSort(array, pi + 1, high, size);

}

}

Сортування Insert:

void insertSort(long array[], int size) {

for (int step = 1; step < size; step++) {

long key = array[step];

int j = step - 1;

while (key < array[j] && j >= 0) {

printArray(array, size);

Sleep(1000);

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

--j;

}

array[j + 1] = key;

}

}

1. У программі було розроблено 3 способи заповнення масиву:
2. Заповнення через код;
3. Заповнення через рандом;
4. Заповнення через введення у консоль;
5. Та 5)

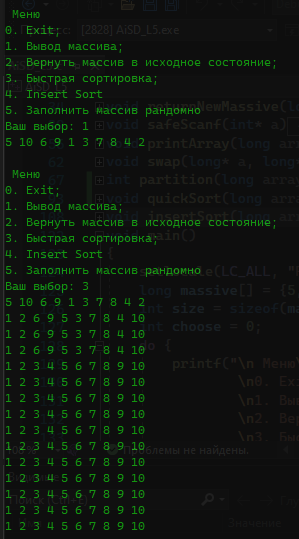


Рис 1. Показ псевдорандомного масиву. Швидке сортування

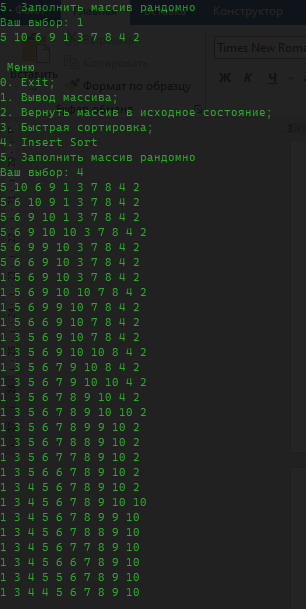


Рис 2. Сортування вставкою

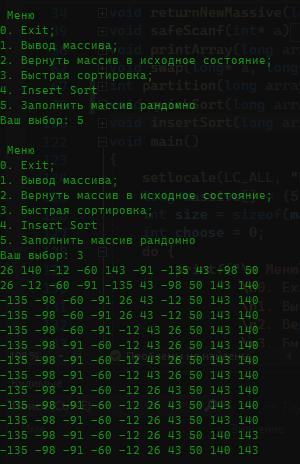


Рис 3. Швидке сортування з рандомними числами

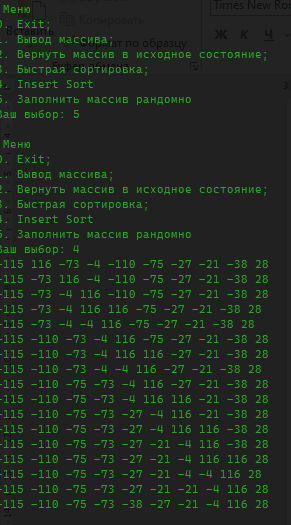


Рис 4. Insert сортування з рандомними числами

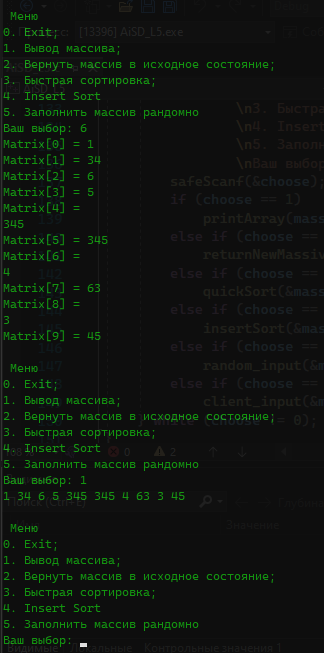


Рис 5. Заповнення масива користувачем

Висновок. Я вивчив загальні алгоритми сортування масивів та розробив програму, що використовує їх. Вони були перевіренні різними наборами даних, та вони працюють.

Лістинг з лабораторної роботи. Файл <main.c>

#include <stdio.h>

#include <stdlib.h>

#include <locale.h>

#include "windows.h"

#define N 50

void client\_input(int\* massive, int size) {

char temp[N];

int pomilka = 0;

for (int i = 0; i < size; i++)

do {

pomilka = 0;

printf("Мatrix[%d] = ", i);

scanf\_s("%s", &temp, N - 1);

for (int a = 0; a < strlen(temp); a++)

if (atoi(temp) == 0 && !(('0' <= temp[a]) && (temp[a] <= '9')))

pomilka = 1;

if (pomilka == 1)

printf("Вводите только числа!\n");

else {

massive[i] = atol(temp);

break;

}

} while (1);

}

void random\_input(long\* massive, int size) {

srand(time(NULL));

for (int i = 0; i < size; i++)

massive[i] = rand() % 301 - 150;

}

void returnNewMassive(long\* matrix, int size) {

long massive[] = { 5, 10, 6, 9, 1, 3 ,7 ,8, 4, 2 };

for (int i = 0; i < size; i++)

\*(matrix + i) = massive[i];

}

void safeScanf(int\* a) {

char temp[N];

int error = 0;

do {

error = 0;

scanf\_s("%s", &temp, N - 1);

for (int a = 0; a < strlen(temp); a++)

if (atoi(temp) == 0 && !(('0' <= temp[a]) && (temp[a] <= '9')))

error = 1;

if (error == 1)

printf("Вводите только числа!\n");

else {

\*a = atoi(temp);

break;

}

} while (1);

}

void printArray(long array[], int size) {

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

printf("%d ", array[i]);

}

printf("\n");

}

void swap(long\* a, long\* b) {

long t = \*a;

\*a = \*b;

\*b = t;

}

int partition(long array[], long low, long high) {

long pivot = array[high];

long i = (low - 1);

for (long j = low; j < high; j++) {

if (array[j] <= pivot) {

i++;

swap(&array[i], &array[j]);

}

}

swap(&array[i + 1], &array[high]);

return (i + 1);

}

void quickSort(long array[], long low, long high, int size) {

printArray(array, size);

Sleep(1000);

if (low < high) {

int pi = partition(array, low, high);

quickSort(array, low, pi - 1, size);

quickSort(array, pi + 1, high, size);

}

}

void insertSort(long array[], int size) {

for (int step = 1; step < size; step++) {

long key = array[step];

int j = step - 1;

while (key < array[j] && j >= 0) {

printArray(array, size);

Sleep(1000);

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

--j;

}

array[j + 1] = key;

}

}

void main()

{

setlocale(LC\_ALL, "Rus");

long massive[] = {5, 10, 6, 9, 1, 3 ,7 ,8, 4, 2};

int size = sizeof(massive) / sizeof(massive[0]);

int choose = 0;

do {

printf("\n Меню\

\n0. Exit; \

\n1. Вывод массива;\

\n2. Вернуть массив в исходное состояние;\

\n3. Быстрая сортировка;\

\n4. Insert Sort\

\n5. Заполнить массив рандомно\

\nВаш выбор: ");

safeScanf(&choose);

if (choose == 1)

printArray(massive, size);

else if (choose == 2)

returnNewMassive(&massive, size);

else if (choose == 3)

quickSort(&massive, 0, (long)size - 1, size);

else if (choose == 4)

insertSort(&massive, size);

else if (choose == 5)

random\_input(&massive, size);

else if (choose == 6)

client\_input(&massive, size);

} while (choose != 0);

}