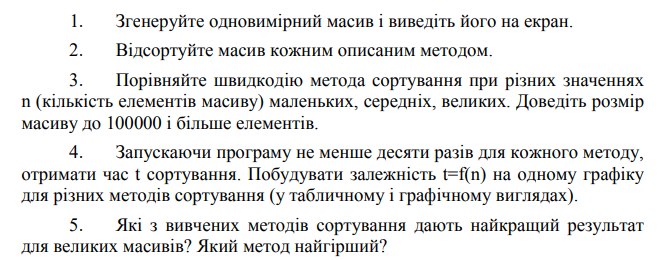
**ЛАБОРАТОРНА РОБОТА № 13-14**

**Сортування одновимірного масиву**

**Мета:** набуття практичних навиків роботи з одновимірними масивами, а саме сортування елементів масиву різними методами. Здійснення порівняння та аналізу ефективності використовуваних методів сортування.

**Хід роботи:**

**Завдання:**



Лістинг программи:

#include <iostream>

#include <math.h>

#include <windows.h>

#include <time.h>

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

srand(time(0));

int arr[250000], method, a, b, count, i, temp1, fl, iMIN, temp2, j, temp3, step, temp4, countNUM = 0;

float time;

clock\_t start, finish;

printf("1 - Сортування обміном\n2 - Сортування методом вибору\n3 - Сортування вставками\n4 - Сортування методом Шелла\n");

printf("Введіть яке саме сортування ви хочете виконати:"); scanf\_s("%d", &method);

printf("Введіть цілий розмір масиву A:"); scanf\_s("%d", &count);

printf("Введіть початкове значення масиву a (ціле):"); scanf\_s("%d", &a);

printf("Введіть кінцеве значення масиву b (ціле):"); scanf\_s("%d", &b);

if (a == b) {

printf("Помилка. Значення а не > b.");

}

printf("Масив до сортування:");

// Будуємо псевдорандомний масив та переносимо на кожному 100 єлементі наступні на новий рядок.

for (i = 0; i < count; i++) {

arr[i] = a + rand() % (b - a + 1);

if (i % 100 == 0) {

printf("\n%d ", arr[i]);

countNUM++; // Обрахунок кількості елементів у масиві(для себе)

}

else {

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

countNUM++; // Обрахунок кількості елементів у масиві(для себе)

}

}

// Виконання кожного методу сортування.

switch (method) {

case 1:

start = clock(); // Початок відрахунку часу

do

{

fl = 0;

for (i = 1; i < count; i++) // Проходимо по кожному елементу у масиві

if (arr[i - 1] > arr[i]) // Ящко елемент який стоїть перед і-тим елементом масиву більше за і-того, робимо наступний код

{

temp1 = arr[i]; // Присвоюємо тимчасовій змінній значення елемента i масиву

arr[i] = arr[i - 1]; // Присвоюємо змінній i масиву значення елемента стоячого до нього

arr[i - 1] = temp1; // Значення елемента і присвоюємо елементу стоячого до нього(тобто змінюємо іх місцями у масиві)

fl = 1; // Прапор для завершення циклу коли він підійде до кінця

}

} while (fl);

finish = clock(); // Зупинка відрахунку часу

time = (float)(finish - start) / CLOCKS\_PER\_SEC; // Обрахунок кількості пройденого часу

printf("\nМасив після сортування:");

for (i = 0; i < count; i++) { // Проходимо по кожному елементу у масиві

if (i % 100 == 0) { // Переносимо на кожному 100 єлементі наступні на новий рядок

printf("\n%d ", arr[i]);

}

else {

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

}

}

printf("\nЧас виконання:%.2f", time); // Вивід кількості пройденого часу пройденого за виконання програми(основного коду)

printf("\nКількість елементів у масиві:%d", countNUM); // Обрахунок кількості елементів у масиві(для себе)

break;

case 2:

start = clock(); // Початок відрахунку часу

for (i = 0; i < count - 1; i++) // Проходимо по кожному елементу у масиві

{

iMIN = i; // Присвоєння і-того елемента мінімуму

for (j = i + 1; j < count; j++) // Проходження по кожному елементу у масиві починаючи з другого елемента

if (arr[j] < arr[iMIN]) iMIN = j; // Знаходження мінімуму другий раз для того щоб змінити місцями з другим елементом

temp2 = arr[i]; // 84-86 - елементи змінюються місцями

arr[i] = arr[iMIN];

arr[iMIN] = temp2;

}

finish = clock(); // Зупинка відрахунку часу

time = (float)(finish - start) / CLOCKS\_PER\_SEC; // Обрахунок кількості пройденого часу

printf("\nМасив після сортування:");

for (i = 0; i < count; i++) { // Проходимо по кожному елементу у масиві

if (i % 100 == 0) { // Переносимо на кожному 100 єлементі наступні на новий рядок

printf("\n%d ", arr[i]);

}

else {

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

}

}

printf("\nЧас виконання:%.2f", time); // Вивід кількості пройденого часу пройденого за виконання програми(основного коду)

printf("\nКількість елементів у масиві:%d", countNUM); // Обрахунок кількості елементів у масиві(для себе)

break;

case 3:

start = clock(); // Початок відрахунку часу

for (i = 1; i < count; i++) // Проходимо по кожному елементу у масиві

{

temp3 = arr[i];

for (j = i - 1; j >= 0 && arr[j] > temp3; j--) // Вставка елемента і масиву у потрібну позицію серед елементів arr[1], ..., arr[i-1], які вже впорядковані

{

arr[j + 1] = arr[j]; // Вставка елемента і масиву у потрібну позицію

arr[j] = temp3;

}

}

finish = clock(); // Зупинка відрахунку часу

time = (float)(finish - start) / CLOCKS\_PER\_SEC; // Обрахунок кількості пройденого часу

printf("\nМасив після сортування:");

for (i = 0; i < count; i++) { // Проходимо по кожному елементу у масиві

if (i % 100 == 0) { // Переносимо на кожному 100 єлементі наступні на новий рядок

printf("\n%d ", arr[i]);

}

else {

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

}

}

printf("\nЧас виконання:%.1f", time); // Вивід кількості пройденого часу пройденого за виконання програми(основного коду)

printf("\nКількість елементів у масиві:%d", countNUM); // Обрахунок кількості елементів у масиві(для себе)

break;

case 4:

start = clock(); // Початок відрахунку часу

step = count / 2; // Масив уявно ділиться на підмасиви, кожен з них впорядковується окремо

while (step > 0)

{

for (i = 0; i < (count - step); i++) // Масив йде саме до його половини розміру

{

j = i;

while (j >= 0 && arr[j] > arr[j + step]) // цикл для одночасного сортування 1 підмасиву за 2

{

temp4 = arr[j]; // 149-152 - зміна елементів місцями в межах підмасиву

arr[j] = arr[j + step];

arr[j + step] = temp4;

j--;

}

}

step = step / 2; // Підмасиви утворюються елементами через один.

}

finish = clock(); // Зупинка відрахунку часу

time = (float)(finish - start) / CLOCKS\_PER\_SEC; // Обрахунок кількості пройденого часу

printf("\nМасив після сортування:");

for (i = 0; i < count; i++) { // Проходимо по кожному елементу у масиві

if (i % 100 == 0) { // Переносимо на кожному 100 єлементі наступні на новий рядок

printf("\n%d ", arr[i]);

}

else {

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

}

}

printf("\nЧас виконання:%.1f", time); // Вивід кількості пройденого часу пройденого за виконання програми(основного коду)

printf("\nКількість елементів у масиві:%d", countNUM); // Обрахунок кількості елементів у масиві(для себе)

break;

}

return 0;

}

3. Табличка часу сортування масивів 4-ма різними способами:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Кількість елементів масиву | Сортування обміном | Сортування методом вибору | Сортування вставками | Сортування методом Шелла |
| 1000 | 0 | 0 | 0 | 0 |
| 5000 | 0.09 | 0.03 | 0 | 0 |
| 10000 | 0.38 | 0.13 | 0.1 | 0 |
| 20000 | 1.64 | 0.57 | 0.54 | 0.1 |
| 40000 | 6.53 | 2.01 | 1.3 | 0.2 |
| 60000 | 14.91 | 4.74 | 2.9 | 0.4 |
| 75000 | 24.82 | 7.27 | 4.4 | 0.6 |
| 85000 | 30.67 | 9.16 | 5.5 | 0.8 |
| 100000 | 41.51 | 12.92 | 9.5 | 1.3 |
| 250000 | 268.11 | 81.79 | 48.1 | 7.7 |

4.

5. Якщо порівняти всі випадки сортування масиву різними видами, то можна зробити висновок, що найкращим видом сортування великих масивів є метод Шелла ( 1000 ел. - 0, 10000 ел. - 0, 100000 ел. - 1.3, 250000 ел. - 7.7. ). Саме він показав найкращі результати сортування великих масивів.

2 місце займає метод сортування вставками (1000 ел. - 0, 10000 ел. - 0.1, 100000 ел. - 9.5, 250000 ел. - 48.1 ).

3 місце – метод сортування вибором (1000 ел. - 0, 10000 ел. - 0.13, 100000 ел. - 12.92, 250000 ел. - 81.79).

Останне місце – сортування «Бульбашка» (1000 ел. - 0, 10000 ел. - 0.38, 100000 ел. - 41.51, 250000 ел. - 268.11).

Мені найбільше сподобався останній метод, бо саме він найкраще оптимізує сортування масивів і пришвидшує роботу коду.

***Висновки:*** в ході виконання лабораторної роботи було ознайомлено з середовищем MS Visual Studio. Досліджено та отримано практичні навики щодо створення найпростішої програми.