Лабораторна робота (Комп’ютерний практикум) № 1:
«Методи сортування масиві»

1.1. Завдання

Ознайомитись з алгоритмами сортування масивів та способами їхньої реалізації.

У якості індивідуального завдання необхідно написати програмний код, у якому реалізується сортування масивів методами бульбашки, вставок, вибору, сортуванням Шелла, Гоара (швидкого сортування). Виконати порівняння ефективності вказаних методів сортування.

Звернення до елементів масиву реалізувати за допомогою вказівника на масив (C++).

Вихідні дані по варіантах у Додатку В-1.

РЕЗУЛЬТАТ РОБОТИ ПОТРІБНО:

1. Роздрукувати (вивести на екран) початковий масив та масиви після виконання сортування різними методами. Показати кількість операцій для виконання сортування різними методами.
2. Відкритий для редагування програмний код розмістити на сайті *https://replit.com/* (посилання через кнопку «*+ Invite*»).
3. Звіт до комп’ютерного практикуму № 1 додати в свій Клас на ресурсі *https://classroom.google.com/*.

1.2. Теоретичні відомості

Метод бульбашки (Bubble Sort)

Сутність методу полягає в багаторазовому проході по масиву. На кожному кроці послідовно порівнюються пари сусідніх елементів, і якщо порядок в такій парі невірний, то елементи в парі міняються місцями. При проході алгоритму, елемент, що стоїть не на своїй позиції, «спливає» до потрібної позиції як бульбашка, звідки і назва алгоритму.

Сортування зі вставками (Insertion Sort)

Сортований масив переглядається в порядку зростання номерів і кожен елемент вставляється в уже переглянуту частина масиву так, щоб зберегти порядок.

Сортування вибором

Спочатку відшукується найменший елемент масиву, потім він міняється місцями з елементом, що стоїть першим у сортованому масиві. Далі, знаходиться другий найменший елемент і міняється місцями з елементом, що стоїть другим у вихідному масиві. Цей процес триває до тих пір, поки весь масив не буде відсортований.

Метод Шелла

Цей метод полягає в порівнянні елементів масиву, розділених однаковою відстанню таким чином, щоб елементи на цій відстані були впорядковані. Потім ця відстань ділиться навпіл і процес триває. В кінці відстань рівна 1 і якщо змін немає, то масив відсортований.

Швидке сортування

Цей метод розглядає масив, як список значень. Спочатку виділяється середнє значення як сепаратор (фактор розбиття) списку. Список розбивається на два: в одному з них значення менше сепаратора, а в іншому - більше або рівні. Далі процедура сортування рекурсивно викликає саму себе для кожного з двох списків. Кожен раз при виклику сортування список елементів розбивається на два менших.

Швидке сортування Гоара

Цей метод ґрунтується на послідовному поділі набору даних на блоки меншого розміру таким чином, що між значеннями різних блоків забезпечується відношення впорядкованості (для будь-якої пари блоків всі значення одного з цих блоків не перевищують значень іншого блоку).

1.3. Описання програми

Програмний код реалізує метод сортування Бульбашка. Під час ініціалізації Для друку даних використовуються створені функції *void print_array_num(int num[], int size_num)* та *void print_array_abc(char abc[], int size_abc)*. В даному прикладі перевантаження функцій не застосовувалося.

1.4. Результати виконання програми

Копію екрана з результатами роботи програми наведено на рисунку 1.1.

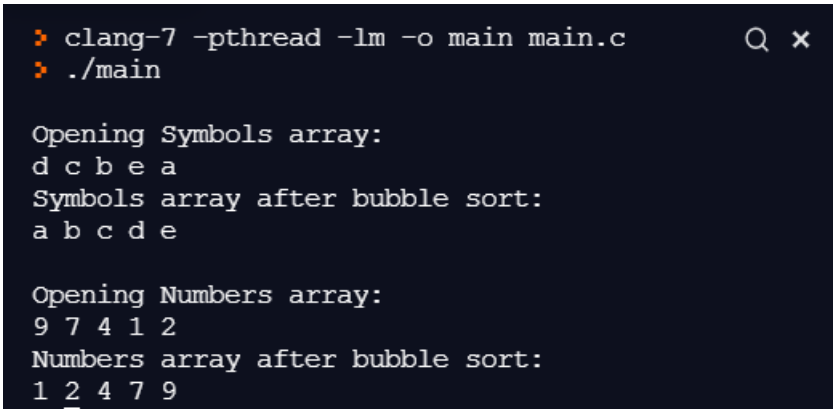

Рисунок 1.1 – екрана з результатами роботи програми

1.5. Висновки по роботі

У результаті виконання комп'ютерного практикуму було досліджено особливості програмної реалізації різних методів сортування масивів даних. У якості прикладу, виконано розробку коду програми, що реалізує сортування методом Бульбашка.

1.6. Лістинг програми

```
// lab1.c 2022
#include <stdio.h>
#include <stdlib.h>
#include<time.h>

#define N 5

// Symbols printing - print_array_abc
void print_array_abc(char abc[], int size_abc)
{
   int i;

   for (i = 0; i < size_abc; i++)
    {
     printf("%c ",  abc[i]);
    }
   printf("\n");
}

// Numbers printing - print_array_num
```

```
void print_array_num(int num[], int size_num)
{
   int i;

   for (i = 0; i < size_num; i++)
    {
     printf("%d ", num[i]);
    }
   printf("\n");
}

int main()
{
 char abc[] = {'d','c','b','e','a'};
 int size_abc = sizeof(abc)/sizeof(abc[0]);
 int i, j, temp;

 srand(time(NULL));

 printf("\nOpening Symbols array: \n");
 print_array_abc(abc, size_abc);

 // bubble sort for Symbols
 for (i = 0; i < size_abc - 1; i++)
    {
     for (j = 0; j < size_abc - i - 1; j++)
         if (abc[j] > abc[j + 1])
         {
          temp = abc[j];
          abc[j] = abc[j + 1];
          abc[j + 1] = temp;
         }
    }

 printf("Symbols array after bubble sort: \n");
 print_array_abc(abc, size_abc);

 int num[N];
 int size_num = sizeof(num)/sizeof(num[0]);

 for (int n = 0; n < N; n++)
```

```
    {
     num[n] = (rand() % 10)+1;
    }

printf("\nOpening Numbers array: \n");
print_array_num(num, size_num);

// bubble sort for Numbers

for (i = 0; i < size_num - 1; i++)
    {

     for (j = 0; j < size_num - i - 1; j++)
         if (num[j] > num[j + 1])
         {
          temp = num[j];
          num[j] = num[j+1];
          num[j+1] = temp;
         }
    }

printf("Numbers array after bubble sort: \n");
print_array_num(num, size_num);

return 0;
}
```

Контрольні запитання

1. У чому полягає алгоритм сортування зі вставками?
2. Навести алгоритм сортування вибором.
3. У чому полягає метод Шелла?
4. Алгоритм швидкого сортування.
5. Алгоритм швидкого сортування Гоара.