НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

Кафедра автоматизованих систем обробки інформації та управління

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(повна назва кафедри, циклової комісії)

**КУРСОВА РОБОТА**

з                                           «Основи програмування»

(назва дисципліни)

на тему: «Пошук заданих елементів у масиві» .\_

Студента I курсу ІП-63 групи

напряму підготовки6.050103 «Програмна інженерія».

спеціальності «Програмне забезпечення високопродуктивних комп`ютерних систем та мереж».

Шелудько Д.М. .

(прізвище та ініціали)

КерівникГоловченко М.М. .

Асистент кафедри АСОІУ .

(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Національна оцінка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кількість балів: \_\_\_\_\_\_\_\_\_\_Оцінка: ECTS \_\_\_\_\_

|  |  |  |  |
| --- | --- | --- | --- |
| Члени комісії |  |  |  |
|  | (підпис) |  | (вчене звання, науковий ступінь, прізвище та ініціали) |
|  |  |  |  |
|  | (підпис) |  | (вчене звання, науковий ступінь, прізвище та ініціали) |

Київ - 2017

Національний технічний університет України “КПІ”

(назва вищого навчального закладу)

Кафедра автоматизованих систем обробки інформації і управління

Дисципліна Основи програмування

Напрям "Програмна інженерія"

Курс 1 Група ІП-63 Семестр 2

**ЗАВДАННЯ**

**на курсову роботу студента**

|  |
| --- |
| Шелудько Дмитра Максимовича |

(прізвище, ім’я, по батькові)

|  |  |
| --- | --- |
| 1. Тема роботи | «Пошук заданих елементів в масиві» |
|  | |
|  | |

|  |  |
| --- | --- |
| 2. Строк здачі студентом закінченої роботи |  |

|  |  |
| --- | --- |
| 3. Вихідні дані до роботи |  |
|  | |
|  | |
|  | |

4. Зміст розрахунково-пояснювальної записки (перелік питань, які підлягають розробці)

|  |
| --- |
|  |
|  |
|  |
|  |

5. Перелік графічного матеріалу ( з точним зазначенням обов’язкових креслень )

|  |
| --- |
|  |
|  |
|  |
|  |

|  |  |
| --- | --- |
| 6. Дата видачі завдання |  |

**КАЛЕНДАРНИЙ ПЛАН**

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Назва етапів курсової роботи | Термін виконання етапів роботи | Підписи керівника, студента |
| 1. | Отримання теми курсової роботи |  |  |
| 2. | Підготовка ТЗ |  |  |
| 3. | Пошук та вивчення літератури з питань курсової роботи |  |  |
| 4. | Розробка алгоритму вирішення задачі |  |  |
| 6. | Узгодження алгоритму з керівником |  |  |
| 5. | Розробка сценарію роботи програми |  |  |
| 6. | Узгодження сценарію роботи програми з керівником |  |  |
| 7. | Узгодження з керівником інтерфейсу користувача |  |  |
| 8. | Розробка програмного забезпечення |  |  |
| 9. | Налагодження розрахункової частини програми |  |  |
| 10. | Розробка та налагодження інтерфейсної частини програми |  |  |
| 11. | Узгодження з керівником набору тестів для контрольного прикладу |  |  |
| 12. | Тестування програми |  |  |
| 13. | Підготовка пояснювальної записки |  |  |
| 14. | Здача курсової роботи на перевірку |  |  |
| 15. | Захист курсової роботи |  |  |
|  |  |  |  |
|  |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Студент |  |  | Шелудько Д.М. |
|  | (підпис) |  | (прізвище, ім’я, по батькові) |

|  |  |  |  |
| --- | --- | --- | --- |
| Керівник |  |  | Головченко М.М. |
|  | (підпис) |  | (прізвище, ім’я, по батькові) |

"\_\_\_" \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2017 р.

**Анотація**

Пояснювальна записка до курсової роботи: сторінок, рисунків, таблиць, посилань, графіки.

Об’єкт дослідження: задача пошуку елементів у масиві.

Мета роботи: дослідження методів пошуку в масиві, створення програмного забезпечення для реалізації пошуку елементів в масиві методами послідовного, бінарного, однорідного бінарного пошуку і методом Шара.

Досліджено методи пошуку в масивах, проведений аналіз їх швидкості , ефективності і складності.

Виконана програмна реалізація алгоритмів пошуку.

МЕТОД ЛІНІЙНОГО ПОШУКУ, МЕТОД БІНАРНОГО ПОШУКУ, МЕТОД ОДНОРІДНОГО БІНАРНОГО ПОШУКУ, МЕТОД ШАРА

Зміст

[Вступ 6](#_Toc483137310)

[1 Постановка задачі 7](#_Toc483137311)

[2 Теоретичні відомості 8](#_Toc483137312)

[2.1 Послідовний пошук 8](#_Toc483137313)

[2.2 Бінарний пошук 8](#_Toc483137314)

[2.3 Однорідний бінарний пошук 9](#_Toc483137315)

[2.4 Метод Шара 9](#_Toc483137316)

[3 Опис алгоритмів 10](#_Toc483137317)

[3.1 Загальний алгоритм 10](#_Toc483137318)

[3.2 Послідовний пошук 11](#_Toc483137319)

[3.3 Бінарний пошук 11](#_Toc483137320)

3.4 [Однорідний бінарний пошук 12](#_Toc483137321)

[3.5 Метод Шара 12](#_Toc483137322)

[4 Опис програмного забезпечення 15](#_Toc483137323)

[Функціональна структура програмного забезпечення 15](#_Toc483137324)

[Опис функцій частин програмного забезпечення 15](#_Toc483137325)

[4.1 Користувацькі функції 15](#_Toc483137326)

[4.2 Стандартні функції 16](#_Toc483137327)

[5 Тестування програмного забезпечення 18](#_Toc483137328)

[5.1 План тестування 18](#_Toc483137329)

[5.2 Приклади тестування 19](#_Toc483137330)

[5.2.1 Тестування при введенні від’ємної кількості чисел для генерації. 19](#_Toc483137331)

[5.2.2 Тестування коректності роботи методів 20](#_Toc483137332)

[6 Інструкція користувача 23](#_Toc483137333)

[6.1 Робота з програмою 23](#_Toc483137334)

[Системні вимоги: 26](#_Toc483137335)

[7 Аналіз і узагальнення результатів 27](#_Toc483137336)

[7.1 Оцінка ефективності. 28](#_Toc483137337)

[7.2 Узагальнення 33](#_Toc483137338)

[Висновки 34](#_Toc483137339)

[Список літератури 35](#_Toc483137340)

[ДодатокА 36](#_Toc483137341)

[Додаток Б 39](#_Toc483137342)

**Вступ**

Кожного дня світ розвивається, і з’являються нові та нові технології, які потребують роботи із великою кількість даний.Ці дані потрібно обробляти, сортквати, змінювати, додавати, видаляти, та виконувати пошук.В рамках цієї роботи, я розгляну саме пошук елементів різними методами:

1. Послідовного пошуку;
2. Бінарного пошуку;
3. Однорідного бінарного пошуку;
4. Методом Шара.

Як результат, ми отримаємо програмний проект для роботи із набором даних, який реалізує пошук елемента в числовому масиві різними методами.

# Постановка задачі

Пошук заданих елементів у масиві. Масив (розмірність – 1000 унікальних випадкових елементів).

Методи пошуку:

* + послідовний(лінійний) метод (для порівняння);
  + бінарний метод;
  + однорідний бінарний метод;
  + метод Шара.

Вхідні дані: діапазон для генерування масиву 1000 унікальних елементів та елемент, за яким здійснюватиметься пошук.

Вихідні дані: порядковий номер елемента в масиві або -1 за його відсутності.

# Теоретичні відомості

Масив – це впорядкований набір фіксованої кількості однотипних елементів, що розсташовуюсть послідовно у пам’яті.

**Методи пошуку елемента в масиві**

Наступні алгоритми використовуються для пошуку по впорядкованих масивах.

## Послідовний пошук

Ідея алгоритму полягає в послідовному переборі всіх наявних елементів і порівнянні їх із шуканим значенням. Робота алгоритму продовжеється поки не буде знайдений елемент (успішний пошук) або не будуть переглянуті всі наявні елементи (невдалий пошук).

## Бінарний пошук

Ідея алгоритму полягає у порівнянні серединного елемента масиву з елементом пошуку, і повторенням даного алгоритму для правої чи лівої частини, залежно від результату порівняння: якщо серединний елемент рівний шуканому значенню, пошук завершується успішно; якщо серединний елемент менший, ніж шукане значення(тобто елемент знаходиться в правій частині), пошук повторюється для всіх елементів, що знаходяться праворуч серединного елемента ; якщо серединний елемент більший за шукане значення(тотбо елемент знаходиться в лівій частині) , пошук проводиться для всіх елементів, що знаходяться ліворуч серединного. Якщо ми отримали відрізок довжиною в 1 елемент так і не знайшли потрібний нам , то алгоритм завершує роботу, елемент відсутній у масиві.

## Однорідний бінарний пошук

Даний алгоритм є модифікацією алгоритму бінарного пошуку.Ідея алгоритму полягає в утворені таблиці індексів і подальшого використання елементів цієї таблиці індексів для визначення серединного елемента.Тобто, ми беремо послідовно(починаючи з першого елемента таблиці) індекси , що розбивають наш відрізок на 2 частини , при цьому індекс змінюється за формулою в залежності від позиції(праворуч або ліворуч) елемента пошуку.Модифікація полягає в тому, що ми не отримуємо індекси із спеціальної таблиці.

## Метод Шара

Метод Шара є модифікованим методом бінарного пошуку, що буде працювати швидше на деяких комп’ютерах або за певної позиції елементів.Суть даного методу полягає в первинному порівнянні і , де , а (**K** – елемент пошуку ;**N** – розмірність масиву).Якщо елемент пошуку знаходиться в правій частині області, то ми ініціалізуємо , де і використовуємо бінарний пошук із .Якщо ж елемент пошуку знаходиться в лівій частині області, ми використовуємо однорідний бінарний пошук із .

# Опис алгоритмів

## ***Загальний алгоритм***

1. **Початок**.
2. Задати діапазон чисел масиву.
3. Перевірити довжину діапазону.
4. **Якщо** діапазон менше 1000 елементів
   1. Видати повідомлення про помилку.
5. Сгенерувати масив унікальних елементів із заданого діапазону.
6. Задати метод пошуку елемента в масиві.
7. **Якщо** вибрали лінійний пошук
   1. Продовжити роботу.
8. **В іншому випадку** 
   1. Відсортувати масив.
9. Вибрати елемент пошуку.
10. Викликати відповідну вибраному алгоритму функцію пошуку.
11. **Якщо** функція повернула -1
    1. Повідомити про відсутність елементу пошуку в масиві.
12. **В іншому випадку**
    1. Вивести позицію елемента пошуку в масиві
13. Вивести інформацію про кількість ітерацій та час роботи певного алгоритму
14. **Кінець**

## ***Послідовний пошук***

Таблица 3.1 - Вхідні та Вихідні дані

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Ім’я** | array | i | element | size |
| **Опис** | показчик на масив | ітератор | елемент пошуку | розмірність масиву |

**Алгоритм :**

1. **Початок**
2. Ініціалізуємо
3. Запускаємо цикл поки
   1. **Якщо**
      1. Програма не знайшла елемент
      2. Повернути -1
      3. Завершити роботу
   2. **В іншому випадку**
4. Програма знайшла елемент
5. Повернути його позицію
6. **Кінець**

## ***Бінарний пошук***

Таблица 3.2 - Вхідні та вихідні дані

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Ім’я | array | element | start | finish | size | pointer |
| Опис | показчик на масив | елемент пошуку | індекс початку відрізку | індекс кінця відрізку | розмірність масиву | індекс елементу в масиві |

**Алгоритм :**

1. **Початок**
2. Ініціалізувати позицію елемента в масиві
3. Запускаємо цикл поки виконуються умови і і
   1. **Якщо**
      1. Ініціалізувати , ,
   2. **В іншому випадку** 
      1. Ініціалізувати , ,
4. **Якщо**
   1. Програма не знайшла елемент
   2. Повернути -1
   3. Завершити роботу
5. **В іншому випадку** 
   1. Програма знайшла елемент
   2. Повернути його позицію
6. **Кінець**
   1. ***Однорідний бінарний пошук***

Таблица 3.3 – Вхідні та вихідні дані

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Ім’я | arr | c | D | size | K | i | j |
| Опис | показчик на масив | константа , що обчислюється за формулою  (пункт 2) | таблиця індексів | розмірність масиву | елемент пошуку | позиція елементу в масиві arr | позиція в таблиці індексів |

**Алгоритм** **:**

1. **Початок**
2. Ініціалізуємо змінну ***с*** за формулою
3. Динамічно створити масив ***D*** із розмірністю ***с***
4. Цикл з ітератором
   1. Ініціалізувати масив **D** за формулою
5. Ініціалізувати
6. Цикл з ітератором
   1. **Якщо**
      1. **Якщо** і і
         1. Елемент знайдено
         2. Повертаємо позицію :
      2. **В іншому випадку** 
         1. Елемент не знайдено
         2. Повернути -1
         3. Завершити роботу
   2. **В іншому випадку**
      1. **Якщо**
         1. Елемент знайдено
         2. Повернути його позицію
      2. **В іншому випадку**
         1. **Якщо**
            1. Ініціалізувати
         2. **В іншому випадку**
            1. Ініціалізувати
   3. **Якщо** і і
      1. Програма не знайшла елемент
      2. Повернути -1
      3. Завершити роботу
   4. **В іншому випадку** 
      1. Програма знайшла елемент
      2. Повернути його позицію
7. **Кінець**

## ***Метод Шара***

Таблица 3.4 – Вхідні та Вихідні дані

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Ім’я | array | K | n | k | i | sigma | it | l |
| Опис | показчик на масив | елемент пошуку | розмірність масиву | константа, що обчислюється за формулою  (пункт) | позиція в масиві | зміщення позиції в масиві | ітератор | консианта, що обчислюється за формулою |

**Алгоритм:**

1. **Початок**
2. Ініціалізувати  **,**
3. **Якщо**
   1. Ініціалізувати , ,
   2. Цикл поки i
      1. **Якщо**
         1. Елемент знайдено
         2. Повернути позицію
      2. **В іншому випадку**
         1. Ініціалізувати ,
         2. **Якщо**
            1. Ініціалізувати
         3. **В іншому випадку**
            1. Ініціалізувати
4. **В іншому випадку**
   1. Ініціалізувати
   2. Цикл поки i i
      1. **Якщо**
         1. Елемент знайдено
         2. Повернути позицію
      2. **В іншому випадку**
         1. **Якщо**
            1. Ініціалізувати
         2. **В іншому випадку**
            1. Ініціалізувати
            2. Інкрементувати
5. Елемнта не знайдено
6. Повернути
7. Программа завершила робота
8. **Кінець**

# Опис програмного забезпечення

Функціональна структура програмного забезпечення

Програмний додаток містить головний файл, 5 файлів реалізації, 5 заголовних файлів та заголовний файл для реалізації інтерфейсу (рисунок 4.1).

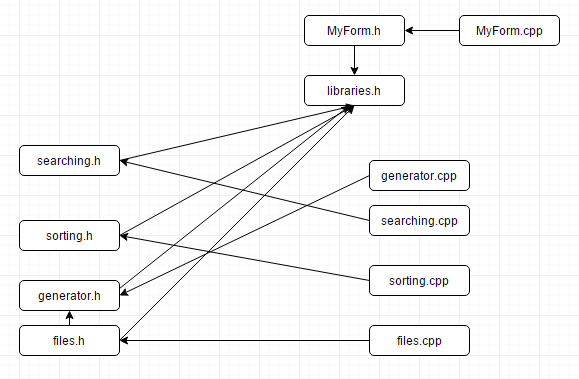


Рисунок 4.1– Функціональна структура програмного забезпечення

Опис користувацьких функцій програмного забезпечення

* **files.h** – реалізує обмін даними в програмі через файли
* **generator.h** – відповідає за генерацію 1000 випадкових унікальних чисел
* **searching.h** – відповідає за методи пошуку елемента в масиві
* **sorting.h** – відповідає за сорутвання елементів у масиві
* **libraries.h** – включає в себе всі бібліотеки використані в программі
* **MyForm.h** – реалізує взаємодію з користувачем

## Користувацькі функції

Таблица 4.1 – Користувацькі функції

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **№ п/п** | **Назва функції** | **Призначення функції** | **Опис вхідних параметрів** | **Опис вихідних параметрів** | **Заголовний файл** |
| **1** | InFile | Запис масиву у файл | Покажчик на масив і розмірність масиву | - | file.h |
| **2** | *FromFile* | Зчитування масиву з файлу | Покажчик на масив і розмірність масиву | - | files.h |
| **3** | *selectedIn* | Запис номера комірки у файл | Номер виділеної комірки | - | files.h |
| **4** | *selectedOut* | Зчитування номеру комірки з файлу | - | Номер виділеної комірки | files.h |
| **5** | *IterationIn* | Запис кількості ітерацій у файл | Кількість ітерацій | - | files.h |
| **6** | *IterationOut* | Зчитування кількості ітерацій з файлу | - | Кількість ітерацій | files.h |
| **7** | *Linear* | Лінійний пошук для генератора | Покажчик на масив , елемент пошуку , розмірність | Позицію знайденого елемента , чи -1 у випадку відсутності такого елемента | generator.h |
| **8** | *generate* | Генератор випадкових унікальних чисел | Покажчик на масив , межі генерування(з, по) | - | generator.h |

Таблица 4.1 - Продовження

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **9** | *Binary* | Бінарний пошук | Покажчик на масив , елемент пошуку , розмірність , межі масиву(як правило 0 і розмірність -1) | Позицію знайденого елемента , чи -1 у випадку відсутності такого елемента | *searchin.h* |
| **10** | *ABinary* | Однорідний бінарний пошук | Покажчик на масив , елемент пошуку , розмірність | Позицію знайденого елемента , чи -1 у випадку відсутності такого елемента | *searchin.h* |
| **11** | *linear* | Лінійний пошук | Покажчик на масив , елемент пошуку , розмірність | Позицію знайденого елемента , чи -1 у випадку відсутності 12такого елемента | *searchin.h* |
| **12** | *Shara* | Метод Шара | Покажчик на масив , елемент пошуку , розмірність | Позицію знайденого елемента , чи -1 у випадку відсутності такого елемента | *searchin.h* |
| **13** | *buble* | Сортування масиву | Покажчик на масив , розмірність | - | *sorting.h* |

## Стандартні функції

Таблиця 4.2 - Стандартні функцій

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **№ п/п** | **Назва функції** | **Призначення функції** | **Опис вхідних параметрів** | **Опис вихідних параметрів** | **Заголовний файл** |
| **1** | *clock()* | Отримати час роботи функції | - | Повертає час пройдений із моменту запуску программи | time.h |
| **2** | *log2* | Обчислення двійкового логарифму числа | Дійсне число | Значення двійкового логарифм учисла | math.h |
| **3** | *abs* | Обчислення абсолютного значення числа | Дійсне число | Абсолютне значення числа | math.h |
| **4** | *toString* | Переводить дані в рядок | Число/рядок/символ | Рядок | String |
| **5** | *toInt32* | Переводить рядок в ціле число, якщо це можливо | Рядок | «Так» і ціле число, якщо дія можлива, інакше «ні» | Int |

# Тестування програмного забезпечення

## План тестування

Помилки можуть виникнути через введення некоректних даних або відсутності даних.

Передбачений випадок введення замалого діапазону, у цьому випадку, користувачу буде повідомлено про помилку, повідомдення відкриється в новому вікні, де буде детальніше описана помилка.

Передбачений випадок, коли користувач намагається згенерувати масив не задавши діапазон, у цьому випадку, користувачу буде повідомлено про помилку, повідомдення відкриється в новому вікні, де буде детальніше описана помилка.

Також программа передбачає спробу пошуку елемента у випадку, коли елемент пошуку не вибрано, чи не вибрано метод пошуку.

Останньою можливою помилкою може бути відсутність елементу пошуку в масиві чисел, у такому випадку, користавучу буде про це повідомлено.

1. Тестування правильності введених значень.
   1. Тестування при введенні замалого діапазону.
2. Тестування коректності роботи методів
   1. Перевірка коректності роботи послідовного пошуку.
   2. Перевірка коректності роботи бінарного пошуку.
   3. Перевірка коректності роботи однорідного бінарного пошуку.
   4. Перевірка коректності роботи методу Шара.
3. Тестування при відсутності методу пошуку, елемента пошуку, діапазону.

## Приклади тестування

### Тестування при введенні замалого діапазону.

Таблица 5.1 – Тестування при введені некоректного діапазону

|  |  |
| --- | --- |
| Мета тесту | Перевірити можливість введення неправильного діапазону |
| Початковий стан програми | Відкрите вікно програми |
| Вхідні дані | 1,100 |
| Схема проведення тесту | Введення значення в поля призначені для задання діапазону |
| Очікуваний результат | Відкриття вікна із детальним описом помилки |
| Стан програми після проведення випробувань | Відкрите вікно із повідомленням про помилку |

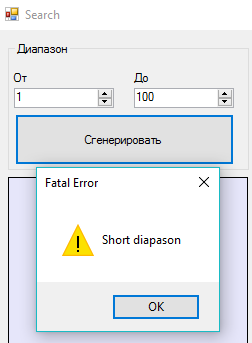


Рисунок 5.1 - Введення замалого діапазону

### Тестування коректності роботи методів

Таблиця 5.2 - Тестування коректності роботи методів

|  |  |
| --- | --- |
| Мета тесту | Перевірити коректність роботи методів |
| Початковий стан програми | Відкрите вікно програми |
| Вхідні дані | 544,545 |
| Схема проведення тесту | Введення елемента пошуку(перше число є в масиві, другого немає) |
| Очікуваний результат | Якщо елемент є в масиві – виділити його позицію в таблиці та вивести її;  Якщо елемент відсутній у масиві – повідомлення про відсутність елемента |
| Стан програми після проведення випробувань | Позиція елемента в масиві або повідомлення про його відсутність |

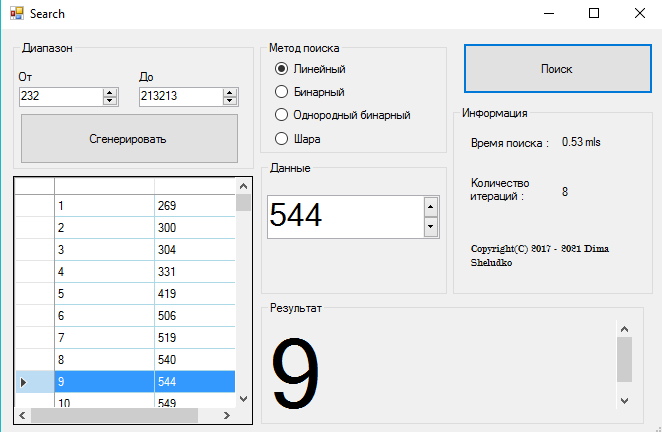


Рисунок 5.1 - Результат пошуку при наявності елемента(Лінійний)

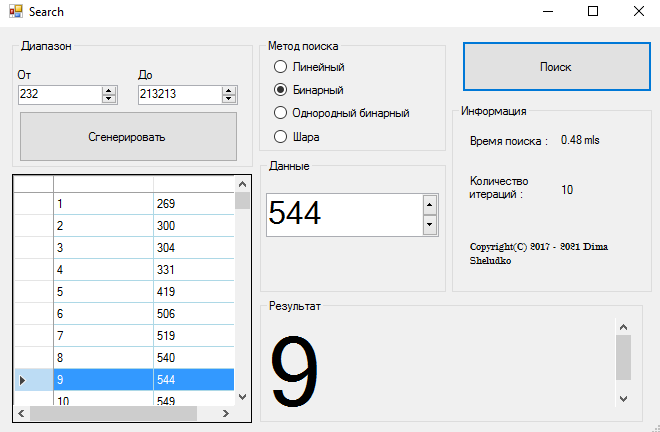


Рисунок 5.2 - Результат пошуку при наявності елемента(Бінарний)

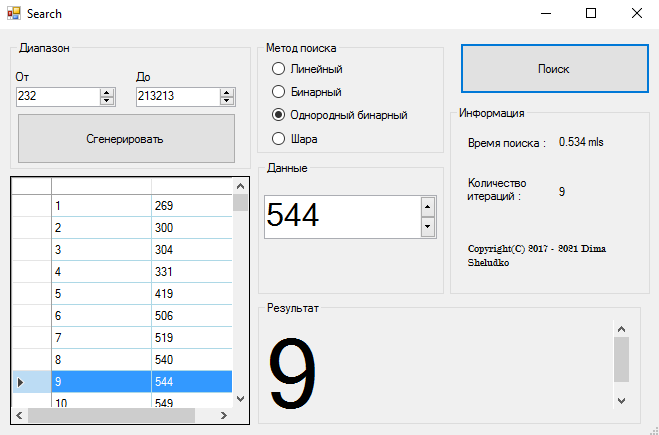


Рисунок 5.3 - Результат пошуку при наявності елемента(Однорідний бінарний)

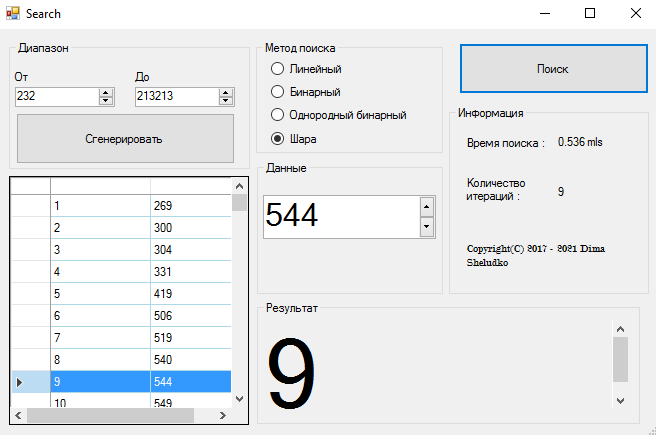


Рисунок 5.4 - Результат пошуку при наявності елемента(Метод Шара)

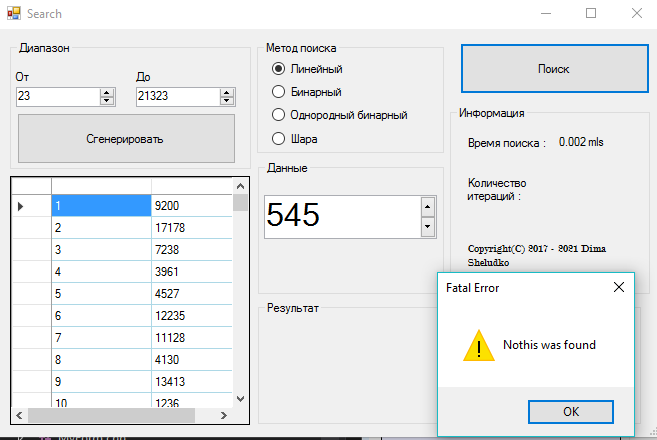


Рисунок 5.5 - Результат пошуку, за відсутності елементу

## 5.2.3 Тестування при відсутності методу пошуку, елемента пошуку, діапазону

Таблица 5.3 - Тестування відсутності методу пошуку, елемента пошуку, діапазону

|  |  |
| --- | --- |
| Мета тесту | Перевірити коректність роботи программи зі відсутності методу пошуку, елемента пошуку, діапазону |
| Початковий стан програми | Відкрите вікно програми |
| Вхідні дані | - |
| Схема проведення тесту | Не вводимо жодних данних у поля шо відповідають методу пошуку, діапазону, елемента пошуку |
| Очікуваний результат | Повідомлення про помилку |
| Стан програми після проведення випробувань | Повідомлення з детальним описом помилки |

# Інструкція користувача

## Робота з програмою

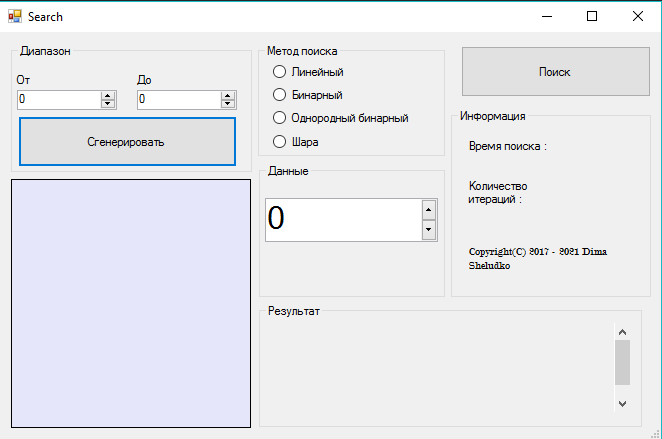
При запуску программи, користувач побачить наступний інтерфейс(Рисунок 6.1)

Рисунок 6.1- Вигляд программи

Далі в полі з назвою “Диапазон” слід вказати діапазон для генерування масиву. За умовою він має бути не менше 1000 (Рисунок 6.2).

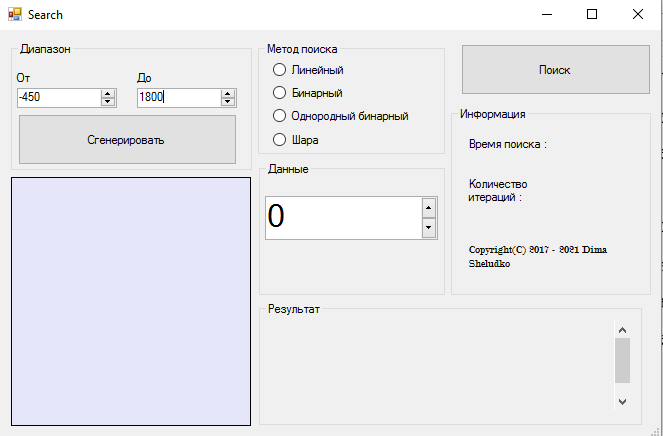


Рисунок 6.2 – Задання діапазону

Далі потрібно натиснути кнопку “Сгенерировать”, після чого у вікно таблиці буде виведено згенерований масив(Рисунок 6.3 – Згенерований масив).

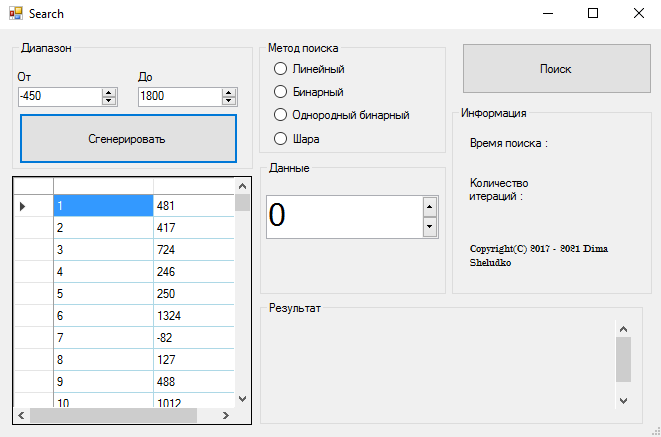
**

Рисунок 6.3 – Згенерований масив

Після цих кроків, треба в полі “Метод поиска” вибрати бажаний метод.Після вибору методу.У полі “Данные” потрібно ввести елемент пошуку, після чого натиснути кнопку “Поиск”.При правильному виконанні вказівок , ви отримаєте результат : елемент пошуку буде виділений у таблиці , буде виведено його позицію та інформацію про роботу алгоритму(Рисунок 6.4 - Результат)

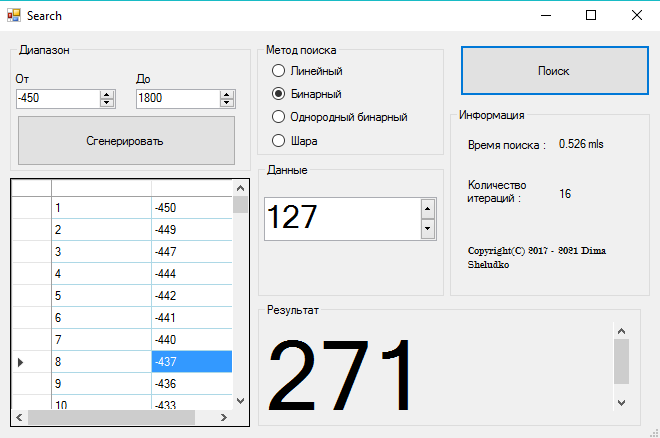


Рисунок 6.4 - Результат

Для завершення роботи з програмою потрібно закрити головне вікно програми.

Системні вимоги:

* Операційна система: Windows 7 і вище
* Процесор: 32-розрядний (x86) або 64-розрядний (x64)
* ОЗП: 512 Мб
* Монітор, клавіатура

# Аналіз і узагальнення результатів

1. Лінійний метод

Лінійний пошук є найпростішим із алгоритмів пошуку, дані не потребують первинної обробки, як наприклад бінарний пошук, який потребує первинне сортування масиву. Пошук значення здійснюється перебором усіх елементів масиву. Отже , асимптоматична складність алгоритму – . У зв'язку з малою ефективністю , алоритм використовується дуже рідко і зазвичай для невеликих за обсягом даних , оскільки показує себе на рівні , а інколи навіть швидше бінарного пошуку та його модифікацій.

1. Бінарний пошук

Бінарний пошук для кількості елементів, що знаходится в області:

(***N*** – розмірність масиву), потребує мінімум **1**, максимум ***k*** порівнянь.Отже асимптотична складність алгоритму складає .Бінарний пошук робить , у середньому порівнянь, що - набагато менше ніж послідовний пошук.

1. Однорідний бінарний пошук

Алгоритм однорідного бінарного пошуку, у випадку успішного завершення, виконує порівнянь в залежності від частини, де знаходиться елемент пошуку.Отже, асимптотична складність даного алгоритму .

Однорідний бінарний пошук є модифікацією бінарного, суть цієї модифікацї полягає у створенні таблиці коефіцієнтів, що потребує додаткових затрат пам’яті.

1. Метод Шара

Метод Шара ніколи не виконує більш ніж порівнянь, але цей метод потребує більшої кількості обчислень ніж однорідний бінарний ьа в деяких випадках бінарний.Асимптотична складність методу Шара – .

1. Оцінка ефективності.

Таблиця 7.1 - Час пошуку

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Позиція елементу пошука в масиві | Лінійний пошук | Бінарний пошук | Однорідний бінарний пошук | Метод Шара |
| 20 | 0.486 | 0.596 | 0.542 | 0.534 |
| 100 | 0.504 | 0.47 | 0.51 | 0.504 |
| 300 | 0.464 | 0.506 | 0.566 | 0.518 |
| 501 | 0.468 | 0.474 | 0.552 | 0.526 |
| 800 | 0.518 | 0.474 | 0.52 | 0.51 |
| 950 | 0.478 | 0.512 | 0.51 | 0.52 |

Таблиця 7.2 - Кількість ітерацій

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Позиція елементу пошука в масиві | Послідовний пошук | Бінарний пошук | Однорідний бінарний пошук | Метод Шара |
| 20 | 20 | 16 | 10 | 7 |
| 100 | 100 | 25 | 10 | 7 |
| 300 | 300 | 34 | 10 | 7 |
| 501 | 501 | 0 | 9 | 9 |
| 800 | 800 | 13 | 10 | 9 |
| 950 | 950 | 20 | 10 | 9 |

Таблиця 7.3 - Кількість порівнянь

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Позиція елементу пошука в масиві | Послідовний пошук | Бінарний пошук | Однорідний бінарний пошук | Метод Шара |
| 20 | 20 | 16 | 20 | 8 |
| 100 | 100 | 25 | 20 | 8 |
| 300 | 300 | 34 | 20 | 8 |
| 501 | 501 | 0 | 18 | 10 |
| 800 | 800 | 13 | 20 | 10 |
| 950 | 950 | 20 | 20 | 10 |

Таблиця 7.4 - Кількість операцій

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Позиція елементу пошука в масиві | Послідовний пошук | Бінарний пошук | Однорідний бінарний пошук | Метод Шара |
| 20 | 20 | 192 | 146 | 161 |
| 100 | 100 | 300 | 146 | 161 |
| 300 | 300 | 408 | 146 | 161 |
| 501 | 501 | 0 | 139 | 189 |
| 800 | 800 | 156 | 146 | 189 |
| 950 | 950 | 240 | 146 | 189 |

Нижче наведені графіки залежності часу виконання пошуку (графік7.1), кількості операцій (графік 7.2), кількості ітерацій(графік 7.3) та кількості порівнянь (графік 7.4) від кількості елементів масиву.

Графік 7.1- Залежність часу виконання пошуку від позиції елементу пошуку

Графік 7.2- Залежність кількості ітеріції від позиції елементу пошуку

Графік 7.3 - Залежність кількості операцій від позиції елементу пошуку

Графік 7.4 - Залежність кількості порівнянь від позиції елементу пошуку

На залежностях кількості ітерацій і кількості операцій від позиції елемента пошуку (графік 7.2 і графік 7.4 відповідно) лінійний пошук не зображено, оскільки в зв’язку із великими показниками, він унеможливлює порівняння інших методів.

## Узагальнення

Основною метою курсової роботи було ознайомлення із різними методами пошуку елемента в масиві, детальний аналіз ефективності кожного із цих методів.

Порівняння ефективності методів дало очікуваний результат, узагальнюючи який ми отримали наступне:

* алгоритм послідовного пошуку доцільно використовувати лише при невелики об’ємах даних, оскільки він є найменш продуктивним із інших, але при невеликих об’мах даних, показує себе на рівні із іншими методоми;
* алгоритми бінарного працює швидше, на відману від лінійного, завдяки своїй головній ідеї – поділу області навпіл, але при невеликих об’ємах даних, може показувати себе на рівні із лінійним;
* алгоритм однорідного бінарного пошуку використовує додаткову таблицю індексів, що позитивно впливає на час його роботи, але потребує додаткової пам’яті для збереження;
* метод Шара один – найшвидший із усіх досліджених в рамках данної роботи, завдяки додатковим обчислення і порівнянням, ми отримаємо більш швидий метод.

Висновки

На етапі ознайомлення з теоретичними аспектами завдання даної курсової роботи, було детально проаналізовано кожен із методів, їх специфіка.

На етапі проектування програмного забезпечення була виконана реалізція кожного із методів.Досліджені проблеми, які можуть спіткати при реалізації цих методів та способи їх вирішень.

Також був розроблений простий інтуітиво-зрозумілий інтерфейс, який наочно демонструє роботу кожного із вищезгаданих методів.

Результатом виконання курсової роботи є Windows Forms программа, яку можна викоритсовувати для навчальних та порівняльних цілей, а самі алгоритми, можна використовувати для вирішення всіх задач, що потребують пошуку данних.

Список літератури

1. Дональд Е. Кнут “*Искусство программирования. Том 3*”.
2. Вірт Н. (1985), *Алгоритми та структури даних*, розділ 1.9.2.
3. [*Левитин А. В.*](https://www.wikidata.org/wiki/Q21694518) Глава 4. Метод декомпозиции: Бинарный поиск // [Алгоритмы. Введение в разработку и анализ](https://www.wikidata.org/wiki/Q21694522) — М.: [Вильямс](https://www.wikidata.org/wiki/Q21694521), 2006. — С. 180–183. — 576 с. — [ISBN 978-5-8459-0987-9](https://ru.wikipedia.org/wiki/%D0%A1%D0%BB%D1%83%D0%B6%D0%B5%D0%B1%D0%BD%D0%B0%D1%8F:%D0%98%D1%81%D1%82%D0%BE%D1%87%D0%BD%D0%B8%D0%BA%D0%B8_%D0%BA%D0%BD%D0%B8%D0%B3/9785845909879)
4. *Амосов А. А., Дубинский Ю. А., Копченова Н. П.* Вычислительные методы для инженеров. — М.: Мир, 1998.
5. *Бахвалов Н. С.,*[*Жидков.*](https://ru.wikipedia.org/wiki/%D0%96%D0%B8%D0%B4%D0%BA%D0%BE%D0%B2,_%D0%9D%D0%B8%D0%BA%D0%BE%D0%BB%D0%B0%D0%B9_%D0%9F%D0%B5%D1%82%D1%80%D0%BE%D0%B2%D0%B8%D1%87)*, Кобельков Г. Г.* Численные методы. — 8-е изд. — М.: Лаборатория Базовых Знаний, 2000.
6. *Вирт Н.* Алгоритмы + структуры данных = программы. — М.: «[Мир](https://ru.wikipedia.org/wiki/%D0%9C%D0%B8%D1%80_(%D0%B8%D0%B7%D0%B4%D0%B0%D1%82%D0%B5%D0%BB%D1%8C%D1%81%D1%82%D0%B2%D0%BE))», 1985. — С. 28.
7. *Волков Е. А.* Численные методы. — М.: Физматлит, 2003.
8. *Гилл Ф., Мюррей У., Райт М.* Практическая оптимизация. Пер. с англ. — М.: Мир, 1985.
9. [*Кормен, Т.*](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D1%80%D0%BC%D0%B5%D0%BD,_%D0%A2%D0%BE%D0%BC%D0%B0%D1%81)*,*[*Лейзерсон, Ч.*](https://ru.wikipedia.org/wiki/%D0%9B%D0%B5%D0%B9%D0%B7%D0%B5%D1%80%D1%81%D0%BE%D0%BD,_%D0%A7%D0%B0%D1%80%D0%BB%D1%8C%D0%B7_%D0%AD%D1%80%D0%B8%D0%BA)*,*[*Ривест, Р.*](https://ru.wikipedia.org/wiki/%D0%A0%D0%B8%D0%B2%D0%B5%D1%81%D1%82,_%D0%A0%D0%BE%D0%BD%D0%B0%D0%BB%D1%8C%D0%B4_%D0%9B%D0%B8%D0%BD%D0%BD)*,*[*Штайн, К.*](https://ru.wikipedia.org/wiki/%D0%A8%D1%82%D0%B0%D0%B9%D0%BD,_%D0%9A%D0%BB%D0%B8%D1%84%D1%84%D0%BE%D1%80%D0%B4) Алгоритмы: построение и анализ = Introduction to Algorithms / Под ред. И. В. Красикова. — 2-е изд. — М.: Вильямс, 2005. — 1296 с. — [ISBN 5-8459-0857-4](https://ru.wikipedia.org/wiki/%D0%A1%D0%BB%D1%83%D0%B6%D0%B5%D0%B1%D0%BD%D0%B0%D1%8F:%D0%98%D1%81%D1%82%D0%BE%D1%87%D0%BD%D0%B8%D0%BA%D0%B8_%D0%BA%D0%BD%D0%B8%D0%B3/5845908574).
10. *Корн Г., Корн Т.* Справочник по математике для научных работников и инженеров. — М.: Наука, 1970. — С. 575-576.
11. *Коршунов Ю. М., Коршунов Ю. М.* Математические основы кибернетики. — Энергоатомиздат, 1972.
12. *Максимов Ю. А., Филлиповская Е. А.* Алгоритмы решения задач нелинейного программирования. — М.: МИФИ, 1982.
13. *Роберт Седжвик.* Фундаментальные алгоритмы на C. Анализ/Структуры данных/Сортировка/Поиск = Algorithms in C. Fundamentals/Data Structures/Sorting/Searching. — СПб.: ДиаСофтЮП, 2003. — С. 672. — [ISBN 5-93772-081-4](https://ru.wikipedia.org/wiki/%D0%A1%D0%BB%D1%83%D0%B6%D0%B5%D0%B1%D0%BD%D0%B0%D1%8F:%D0%98%D1%81%D1%82%D0%BE%D1%87%D0%BD%D0%B8%D0%BA%D0%B8_%D0%BA%D0%BD%D0%B8%D0%B3/5937720814).

# ДОДАТОК А ТЕХНІЧНЕ ЗАВДАННЯ

МІНІСТЕРСТВО НАУКИ І ОСВІТИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ «КПІ»

Кафедра

автоматизованих систем обробки інформації та управління

Затвердив

Керівник Головченко М.М.

«\_\_» \_\_\_\_\_\_\_\_\_\_\_\_ 2016 р.

Виконавець:

Студент Шелудько Д.М. *\_\_\_\_*

«\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2016 р.

ТЕХНІЧНЕ ЗАВДАННЯ

на виконання курсової роботи

на тему: Розв'язання систем нелінійних рівнянь

з дисципліни:

«Основи програмування»

Київ 2016

1.1 Мета: Метою курсової роботи є розробка комплексу програм для *знаходження елемента масиву*

1.2 Найменування та галузь застосування об'єкта розробки:Дана робота присвячена розробці програмного забезпечення для використання в банківській справі , математиці , фізиці , програмуванні.

1.3 Підстава для проведення робіт:Підставою для розробки програмного забезпечення є навчальний план спеціальності 6.050301 «Програмна інженерія», робоча програма дисципліни „Основи програмування”, індивідуальне завдання.

1.4 Дата початку роботи: «6 » березня 2016 р.

1.5 Дата закінчення роботи: «23»» травня 2016 р.

1.6 Призначення розробки: Функціональним призначенням програми є пошук заданого елемента масиву лінійним методом , бінарним методом , однорідним бінарним методом , методом Шара . Експлуатаційним призначенням програми є швидкий пошук елементів у масивах даних.

1.7 Вимоги до програми та програмної документації: Все програмне забезпечення та супроводжуюча технічна документація повинні задовольняти наступним ГОСТ:

ИСО 5807 - 85 ГОСТ на розробку програмних документів, схем алгоритмів програм, даних та систем.

ГОСТ 19.781 - 74 - Вимоги до розробки програмного забезпечення.

ГОСТ 19.101-77 (СТ СЭВ 1626 - 79) - Держстандарт на розробку програмної документації, видів програм та програмних документів.

ГОСТ 29.401 - 78 - Текст програми. Вимоги до змісту та оформлення.

ГОСТ 19.106 - 78 - Вимоги до програмної документації.

ГОСТ 7.1 - 84 та ДСТУ 3008 - 95 - Розробка технічної документації.

1.8 Стадії та етапи розробки:

1. Аналіз методів вирішення поставленої задачі (до 17.04.2016 р.)

2. Розробка сценарію роботи програми (до 01.05.2016 р.)

3. Розробка алгоритмічного забезпечення (до 24.04.2016 р.)

4. Розробка програмного забезпечення (до 09.05.2016 р.)

5. Розробка інтерфейсу, планування, тестування розробленої програми (до 16.05.2016 р.)

6. Розробка пояснювальної записки (до 25.05.2016 р.).

7. Захист курсової роботи (до 12.06.2016 р.).

1.9 Порядок контролю та приймання. Поточні результати роботи над КР регулярно демонструються викладачу. Своєчасність виконання основних етапів графіку підготовки роботи впливає на оцінку за КР відповідно до критеріїв її оцінювання

# ДОДАТОК Б ТЕКСТИ ПРОГРАМНОГО КОДУ

*Тексти програмного коду програмного забезпечення*

«Пошук заданого елемента в масиві»

(Вид носія даних)

*DVD-RW*

(Обсяг програми (документа), арк., Кб)

*27 арк,60 Кb*

*студента групи ІП-63 І курсу*

Шелудька Дмитра Максимовича

Лістинг програми (функціональної частини):

## files.h

#pragma once

void InFile(int \*&array, int size);

void FromFile(int \*&array, int size);

void selectedIn(int number);

int selectedOut();

void IterationIn(int sum);

int IterationOut();

## files.cpp

#pragma once

#include <fstream>

#include <iostream>

using namespace std;

void InFile(int \*&array, int size)

{

ofstream out;

out.open("data.info");

for (int i = 0; i<size - 1; i++)

{

out << array[i] << " ";

}

out << array[size - 1];

}

void FromFile(int \*&array, int size)

{

ifstream in;

in.open("data.info");

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

{

in >> array[i];

}

}

void selectedIn(int number)

{

ofstream out;

out.open("selected.data");

out << number;

}

int selectedOut()

{

ifstream out("selected.data");

int result;

out >> result;

return result;

}

void IterationIn(int sum)

{

ofstream out;

out.open("iteration.data");

out << sum;

}

int IterationOut()

{

ifstream in("iteration.data");

int result = 0;

in >> result;

return result;

}

## generator.h

#pragma once

#define SIZE 1000

namespace Generator

{

int Linear(int \*array, int element, int size);

void generate(int \*&array, int from, int to);

}

## generator.cpp

#include <stdlib.h>

#include <ctime>

#include <time.h>

#include "files.h"

#include "generator.h"

//#define SIZE 1000

using namespace std;

namespace Generator

{

int Linear(int \*array, int element, int size)

{

int result = 0;

int iteration = 0;

while (array[result] != element)

{

iteration++;

result++;

if (result > size)

{

return -1;

}

}

IterationIn(iteration);

return result;

}

void generate(int \*&array, int from, int to)

{

//int SIZE = abs(to) - abs(from);

srand(time(0));

array = new int[SIZE];

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

{

array[i] = 0;

}

int curr = 0;

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

{

curr = rand() % (from - to + 1) + from;

while (Linear(array, curr, SIZE) != -1)

{

curr = rand() % (from - to + 1) + from;

}

array[i] = curr;

}

}

}

## libraries.h

#pragma once

#include <iostream>

#include <stdlib.h>

#include <cstdlib>

#include <stdio.h>

#include "files.h"

#include "sorting.h"

#include <ctime>

#include <time.h>

#include "searching.h"

#include "generator.h"

## searching.h

#pragma once

namespace Searching

{

int binary(int \*array, int element, int size, int start, int finish);

int Binary(int \*&array, int element, int sizeSS, int start, int finish);

int UBinary(int \*&array, int K, int i, int m);

int linear(int \*array, int element, int size);

int ubinary(int array[], int K, int i, int m);

int Alone(int \*&array, int K, int N);

int ABinary(int \*&array, int K, int size);

int Shar(int \*&array, int K, int N);

int Shara(int \*&array, int K, int n);

}

## searching.cpp

#include <math.h>

#include "files.h"

#include "searching.h"

namespace Searching

{

int Binary(int \*&array, int element, int size, int start, int finish)

{

int pointer = size / 2;//right

int iteration = 0;

while (pointer <= finish && pointer > 0 && array[pointer] != element)

{

iteration++;

if (array[pointer] > element)

{

size = size / 2;

finish = (pointer - 1);

pointer = (finish - start + 1) / 2;

}

else

{

start = pointer;

size = finish - pointer + 1;

pointer = size / 2 + start;

}

}

IterationIn(iteration);

if (array[pointer] != element)

{

return -1;

}

else

{

return pointer;

}

}

int linear(int \*array, int element, int size)

{

int result = 0;

int iteration = 0;

while (array[result] != element)

{

iteration++;

result++;

if (result > size)

{

return -1;

}

}

IterationIn(iteration);

return result;

}

int ABinary(int \*&array, int K, int size)

{

int \*arr = new int[size + 1];

for (int i = 1; i <= size; i++)

{

arr[i] = array[i - 1];

}

int c = log2(size) + 2;

int \*D = new int[c];

D[0] = 0;

for (int j = 1; j < log2(size) + 2; j++)

{

D[j] = (size + pow(2, j - 1)) / pow(2, j);

//cout << " j :" << j << ";" << D[j] << endl;

}

//D[9] = 2;

//D[10] = 1;

int i = D[1], j = 2;

while (i > 0 && i < size)

{

if (D[j] == 0)

{

if (i <= 10 && i >0 && D[i] == K)

return i - 1;

else

return -1;

}

else

{

if (arr[i] == K)

{

return i - 1;

}

else

{

if (arr[i] > K)

{

i = i - D[j];

j = j + 1;

}

else

{

i = i + D[j];

j = j + 1;

}

}

}

}

if (i > 0 && i < size && arr[i] != K)

{

return -1;

}

else

{

return i - 1;

}

}

int Shara(int \*&array, int K, int n)

{

int iterator = 0;

int \*arr = new int[n + 1];

arr[0] = 0;

for (int i = 1; i < n + 1; i++)

{

arr[i] = array[i - 1];

}

int k = log2(n);

int i = pow(2.,k);

int sigma = 0;

if (K < arr[i])

{

int it = 2;

sigma = pow(2., k - 1);

i = i - sigma;

while (sigma >= 0 && k - it >=0)

{

iterator++;

if(K == arr[i])

{

IterationIn(iterator);

return i - 1;

}

else

{

sigma = pow(2., k - it);

it++;

if (K < arr[i])

{

i = i - sigma;

}

else

{

i = i + sigma;

}

}

}

}

else

{

int l = log2(n - pow(2., k) + 1);

int i = n + 1 -pow(2., l);

int it = 1;

while (i <= n && i >= 0 && l - it >= 0)

{

iterator++;

if (K == arr[i])

{

IterationIn(iterator);

return i - 1;

}

else

{

if (K < arr[i])

{

i = i - pow(2., l - it);

}

else

{

i = i + pow(2., l - it);

}

it++;

}

}

}

int t = iterator;

IterationIn(iterator);

return -1;

}

}

## sorting.h

#pragma once

namespace Sorting

{

void buble(int \*&array, int size);

}

## sorting.cpp

#include "sorting.h"

using namespace std;

namespace Sorting

{

void buble(int \*&array, int size)

{

int curr = 0;

int c = 0;

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

{

curr = array[i];

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

{

if (curr > array[j])

{

curr = array[j];

c = array[i];

array[i] = curr;

array[j] = c;

}

}

}

}

}

## MyForm.h

#pragma once

#include "libraries.h"

/\*#include "files.h"

#include <iostream>

#include <ctime>

#include <time.h>

#include "searching.h"\*/

#define N 500 //for timing

//using namespace Searching;

namespace Cursach {

using namespace System;

using namespace System::ComponentModel;

using namespace System::Collections;

using namespace System::Windows::Forms;

using namespace System::Data;

using namespace System::Drawing;

/// <summary>

/// Summary for MyForm

/// </summary>

public ref class MyForm : public System::Windows::Forms::Form

{

public:

MyForm(void)

{

InitializeComponent();

//

//TODO: Add the constructor code here

//

}

protected:

/// <summary>

/// Clean up any resources being used.

/// </summary>

~MyForm()

{

if (components)

{

delete components;

}

}

private: System::Windows::Forms::GroupBox^ gbGenerate;

protected:

private: System::Windows::Forms::Button^ button1;

private: System::Windows::Forms::Label^ label2;

private: System::Windows::Forms::Label^ label1;

private: System::Windows::Forms::TextBox^ txtTo;

private: System::Windows::Forms::TextBox^ txtFrom;

private: System::Windows::Forms::GroupBox^ groupBox1;

private: System::Windows::Forms::RadioButton^ radioButton4;

private: System::Windows::Forms::RadioButton^ radioButton3;

private: System::Windows::Forms::RadioButton^ radioButton2;

private: System::Windows::Forms::RadioButton^ radioButton1;

private: System::Windows::Forms::GroupBox^ groupBox2;

private: System::Windows::Forms::Label^ label3;

private: System::Windows::Forms::Button^ button2;

private: System::Windows::Forms::TextBox^ textBox3;

private: System::Windows::Forms::GroupBox^ groupBox3;

private: System::Windows::Forms::Label^ label5;

private: System::Windows::Forms::GroupBox^ groupBox4;

private: System::Windows::Forms::Label^ Iteration;

private: System::Windows::Forms::Label^ TIME;

private: System::Windows::Forms::DataGridView^ Table;

private: System::Windows::Forms::RichTextBox^ richTextBox1;

private: System::Windows::Forms::Label^ label10;

protected:

protected:

private:

/// <summary>

/// Required designer variable.

/// </summary>

System::ComponentModel::Container ^components;

#pragma region Windows Form Designer generated code

/// <summary>

/// Required method for Designer support - do not modify

/// the contents of this method with the code editor.

/// </summary>

void InitializeComponent(void)

{

this->gbGenerate = (gcnew System::Windows::Forms::GroupBox());

this->button1 = (gcnew System::Windows::Forms::Button());

this->label2 = (gcnew System::Windows::Forms::Label());

this->label1 = (gcnew System::Windows::Forms::Label());

this->txtTo = (gcnew System::Windows::Forms::TextBox());

this->txtFrom = (gcnew System::Windows::Forms::TextBox());

this->groupBox1 = (gcnew System::Windows::Forms::GroupBox());

this->radioButton4 = (gcnew System::Windows::Forms::RadioButton());

this->radioButton3 = (gcnew System::Windows::Forms::RadioButton());

this->radioButton2 = (gcnew System::Windows::Forms::RadioButton());

this->radioButton1 = (gcnew System::Windows::Forms::RadioButton());

this->groupBox2 = (gcnew System::Windows::Forms::GroupBox());

this->label10 = (gcnew System::Windows::Forms::Label());

this->Iteration = (gcnew System::Windows::Forms::Label());

this->TIME = (gcnew System::Windows::Forms::Label());

this->label5 = (gcnew System::Windows::Forms::Label());

this->label3 = (gcnew System::Windows::Forms::Label());

this->button2 = (gcnew System::Windows::Forms::Button());

this->textBox3 = (gcnew System::Windows::Forms::TextBox());

this->groupBox3 = (gcnew System::Windows::Forms::GroupBox());

this->groupBox4 = (gcnew System::Windows::Forms::GroupBox());

this->richTextBox1 = (gcnew System::Windows::Forms::RichTextBox());

this->Table = (gcnew System::Windows::Forms::DataGridView());

this->gbGenerate->SuspendLayout();

this->groupBox1->SuspendLayout();

this->groupBox2->SuspendLayout();

this->groupBox3->SuspendLayout();

this->groupBox4->SuspendLayout();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->Table))->BeginInit();

this->SuspendLayout();

//

// gbGenerate

//

this->gbGenerate->Controls->Add(this->button1);

this->gbGenerate->Controls->Add(this->label2);

this->gbGenerate->Controls->Add(this->label1);

this->gbGenerate->Controls->Add(this->txtTo);

this->gbGenerate->Controls->Add(this->txtFrom);

this->gbGenerate->Location = System::Drawing::Point(12, 12);

this->gbGenerate->Name = L"gbGenerate";

this->gbGenerate->Size = System::Drawing::Size(241, 129);

this->gbGenerate->TabIndex = 0;

this->gbGenerate->TabStop = false;

this->gbGenerate->Text = L"Диапазон";

//

// button1

//

this->button1->Location = System::Drawing::Point(7, 72);

this->button1->Name = L"button1";

this->button1->Size = System::Drawing::Size(219, 51);

this->button1->TabIndex = 5;

this->button1->Text = L"Сгенерировать";

this->button1->UseVisualStyleBackColor = true;

this->button1->Click += gcnew System::EventHandler(this, &MyForm::button1\_Click\_1);

//

// label2

//

this->label2->AutoSize = true;

this->label2->Location = System::Drawing::Point(123, 29);

this->label2->Name = L"label2";

this->label2->Size = System::Drawing::Size(22, 13);

this->label2->TabIndex = 3;

this->label2->Text = L"До";

//

// label1

//

this->label1->AutoSize = true;

this->label1->Location = System::Drawing::Point(3, 29);

this->label1->Name = L"label1";

this->label1->Size = System::Drawing::Size(20, 13);

this->label1->TabIndex = 2;

this->label1->Text = L"От";

//

// txtTo

//

this->txtTo->Location = System::Drawing::Point(126, 45);

this->txtTo->Multiline = true;

this->txtTo->Name = L"txtTo";

this->txtTo->Size = System::Drawing::Size(100, 20);

this->txtTo->TabIndex = 1;

//

// txtFrom

//

this->txtFrom->Location = System::Drawing::Point(6, 45);

this->txtFrom->Multiline = true;

this->txtFrom->Name = L"txtFrom";

this->txtFrom->Size = System::Drawing::Size(100, 20);

this->txtFrom->TabIndex = 0;

//

// groupBox1

//

this->groupBox1->Controls->Add(this->radioButton4);

this->groupBox1->Controls->Add(this->radioButton3);

this->groupBox1->Controls->Add(this->radioButton2);

this->groupBox1->Controls->Add(this->radioButton1);

this->groupBox1->Location = System::Drawing::Point(259, 12);

this->groupBox1->Name = L"groupBox1";

this->groupBox1->Size = System::Drawing::Size(187, 113);

this->groupBox1->TabIndex = 2;

this->groupBox1->TabStop = false;

this->groupBox1->Text = L"Метод поиска";

//

// radioButton4

//

this->radioButton4->AutoSize = true;

this->radioButton4->Location = System::Drawing::Point(15, 89);

this->radioButton4->Name = L"radioButton4";

this->radioButton4->Size = System::Drawing::Size(52, 17);

this->radioButton4->TabIndex = 3;

this->radioButton4->TabStop = true;

this->radioButton4->Text = L"Шара";

this->radioButton4->UseVisualStyleBackColor = true;

//

// radioButton3

//

this->radioButton3->AutoSize = true;

this->radioButton3->Location = System::Drawing::Point(15, 65);

this->radioButton3->Name = L"radioButton3";

this->radioButton3->Size = System::Drawing::Size(142, 17);

this->radioButton3->TabIndex = 2;

this->radioButton3->TabStop = true;

this->radioButton3->Text = L"Однородный бинарный";

this->radioButton3->UseVisualStyleBackColor = true;

//

// radioButton2

//

this->radioButton2->AutoSize = true;

this->radioButton2->Location = System::Drawing::Point(15, 42);

this->radioButton2->Name = L"radioButton2";

this->radioButton2->Size = System::Drawing::Size(76, 17);

this->radioButton2->TabIndex = 1;

this->radioButton2->TabStop = true;

this->radioButton2->Text = L"Бинарный";

this->radioButton2->UseVisualStyleBackColor = true;

//

// radioButton1

//

this->radioButton1->AutoSize = true;

this->radioButton1->Location = System::Drawing::Point(15, 19);

this->radioButton1->Name = L"radioButton1";

this->radioButton1->Size = System::Drawing::Size(77, 17);

this->radioButton1->TabIndex = 0;

this->radioButton1->TabStop = true;

this->radioButton1->Text = L"Линейный";

this->radioButton1->UseVisualStyleBackColor = true;

//

// groupBox2

//

this->groupBox2->Controls->Add(this->label10);

this->groupBox2->Controls->Add(this->Iteration);

this->groupBox2->Controls->Add(this->TIME);

this->groupBox2->Controls->Add(this->label5);

this->groupBox2->Controls->Add(this->label3);

this->groupBox2->Location = System::Drawing::Point(452, 77);

this->groupBox2->Name = L"groupBox2";

this->groupBox2->Size = System::Drawing::Size(200, 189);

this->groupBox2->TabIndex = 3;

this->groupBox2->TabStop = false;

this->groupBox2->Text = L"Информация";

//

// label10

//

this->label10->Font = (gcnew System::Drawing::Font(L"Modern No. 20", 8.249999F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(0)));

this->label10->Location = System::Drawing::Point(15, 135);

this->label10->Name = L"label10";

this->label10->Size = System::Drawing::Size(176, 30);

this->label10->TabIndex = 7;

this->label10->Text = L"Copyright(C) 2017 - 2021 Dima Sheludko";

//

// Iteration

//

this->Iteration->Location = System::Drawing::Point(106, 79);

this->Iteration->Name = L"Iteration";

this->Iteration->Size = System::Drawing::Size(74, 14);

this->Iteration->TabIndex = 5;

//

// TIME

//

this->TIME->Location = System::Drawing::Point(106, 29);

this->TIME->Name = L"TIME";

this->TIME->Size = System::Drawing::Size(74, 14);

this->TIME->TabIndex = 3;

//

// label5

//

this->label5->Location = System::Drawing::Point(15, 70);

this->label5->Name = L"label5";

this->label5->Size = System::Drawing::Size(85, 48);

this->label5->TabIndex = 2;

this->label5->Text = L"Количество итераций :";

//

// label3

//

this->label3->AutoSize = true;

this->label3->Location = System::Drawing::Point(15, 30);

this->label3->Name = L"label3";

this->label3->Size = System::Drawing::Size(85, 13);

this->label3->TabIndex = 0;

this->label3->Text = L"Время поиска :";

//

// button2

//

this->button2->Location = System::Drawing::Point(462, 14);

this->button2->Name = L"button2";

this->button2->Size = System::Drawing::Size(190, 51);

this->button2->TabIndex = 4;

this->button2->Text = L"Поиск";

this->button2->UseVisualStyleBackColor = true;

this->button2->Click += gcnew System::EventHandler(this, &MyForm::button2\_Click);

//

// textBox3

//

this->textBox3->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 24, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(0)));

this->textBox3->Location = System::Drawing::Point(7, 44);

this->textBox3->Multiline = true;

this->textBox3->Name = L"textBox3";

this->textBox3->Size = System::Drawing::Size(173, 43);

this->textBox3->TabIndex = 6;

//

// groupBox3

//

this->groupBox3->Controls->Add(this->textBox3);

this->groupBox3->Location = System::Drawing::Point(260, 132);

this->groupBox3->Name = L"groupBox3";

this->groupBox3->Size = System::Drawing::Size(186, 134);

this->groupBox3->TabIndex = 7;

this->groupBox3->TabStop = false;

this->groupBox3->Text = L"Данные";

//

// groupBox4

//

this->groupBox4->Controls->Add(this->richTextBox1);

this->groupBox4->Location = System::Drawing::Point(260, 272);

this->groupBox4->Name = L"groupBox4";

this->groupBox4->Size = System::Drawing::Size(383, 124);

this->groupBox4->TabIndex = 8;

this->groupBox4->TabStop = false;

this->groupBox4->Text = L"Результат";

//

// richTextBox1

//

this->richTextBox1->BackColor = System::Drawing::SystemColors::Control;

this->richTextBox1->BorderStyle = System::Windows::Forms::BorderStyle::None;

this->richTextBox1->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 72, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(0)));

this->richTextBox1->Location = System::Drawing::Point(7, 19);

this->richTextBox1->Name = L"richTextBox1";

this->richTextBox1->Size = System::Drawing::Size(365, 89);

this->richTextBox1->TabIndex = 0;

this->richTextBox1->Text = L"";

//

// Table

//

this->Table->AllowUserToAddRows = false;

this->Table->AllowUserToDeleteRows = false;

this->Table->AllowUserToResizeColumns = false;

this->Table->AllowUserToResizeRows = false;

this->Table->BackgroundColor = System::Drawing::Color::Lavender;

this->Table->ColumnHeadersHeightSizeMode = System::Windows::Forms::DataGridViewColumnHeadersHeightSizeMode::AutoSize;

this->Table->GridColor = System::Drawing::Color::LightBlue;

this->Table->Location = System::Drawing::Point(12, 147);

this->Table->Name = L"Table";

this->Table->Size = System::Drawing::Size(240, 249);

this->Table->TabIndex = 9;

//

// MyForm

//

this->AutoScaleDimensions = System::Drawing::SizeF(6, 13);

this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;

this->ClientSize = System::Drawing::Size(662, 408);

this->Controls->Add(this->Table);

this->Controls->Add(this->button2);

this->Controls->Add(this->groupBox4);

this->Controls->Add(this->groupBox3);

this->Controls->Add(this->groupBox2);

this->Controls->Add(this->groupBox1);

this->Controls->Add(this->gbGenerate);

this->MaximumSize = System::Drawing::Size(678, 447);

this->MinimumSize = System::Drawing::Size(678, 447);

this->Name = L"MyForm";

this->SizeGripStyle = System::Windows::Forms::SizeGripStyle::Show;

this->Text = L"Search";

this->Load += gcnew System::EventHandler(this, &MyForm::MyForm\_Load);

this->gbGenerate->ResumeLayout(false);

this->gbGenerate->PerformLayout();

this->groupBox1->ResumeLayout(false);

this->groupBox1->PerformLayout();

this->groupBox2->ResumeLayout(false);

this->groupBox2->PerformLayout();

this->groupBox3->ResumeLayout(false);

this->groupBox3->PerformLayout();

this->groupBox4->ResumeLayout(false);

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->Table))->EndInit();

this->ResumeLayout(false);

}

#pragma endregion

private: System::Void button1\_Click(System::Object^ sender, System::EventArgs^ e) {

}

private: System::Void MyForm\_Load(System::Object^ sender, System::EventArgs^ e) {

}

private: System::Void button1\_Click\_1(System::Object^ sender, System::EventArgs^ e)

{

//DataGridViewRow row;

int from = 0, to = 0;

if (txtFrom->Text == "" || txtTo->Text == "")

{

MessageBox::Show("Enter a diapason", "Fatal Error",

MessageBoxButtons::OK, MessageBoxIcon::Exclamation);

}

else

{

from = Convert::ToInt32(txtFrom->Text);

to = Convert::ToInt32(txtTo->Text);

if (to < from)

{

MessageBox::Show("Short diapason", "Fatal Error",

MessageBoxButtons::OK, MessageBoxIcon::Exclamation);

}

else

{

if (abs(to - from) > 1000)

{

int \*array = 0;

Generator::generate(array, from, to);

Table->RowCount = SIZE;

Table->ColumnCount = 2;

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

{

Table->Rows[i]->Cells[0]->Value = i + 1;

Table->Rows[i]->Cells[1]->Value = array[i];

}

InFile(array, SIZE);

}

else

{

MessageBox::Show("Short diapason", "Fatal Error",

MessageBoxButtons::OK, MessageBoxIcon::Exclamation);

}

}

}

}

private: void Finding()

{

if (textBox3->Text == "")

{

MessageBox::Show("Enter an object", "Fatal Error",

MessageBoxButtons::OK, MessageBoxIcon::Exclamation);

}

else

{

int element = Convert::ToInt32(textBox3->Text);

if (!radioButton1->Checked && !radioButton2->Checked && !radioButton3->Checked && !radioButton4->Checked)

{

MessageBox::Show("Chose type of searching", "Fatal Error",

MessageBoxButtons::OK, MessageBoxIcon::Exclamation);

}

else

{

//time\_t seconds = time(NULL);

//tm \*time1 , \*time2;

clock\_t start, end;

int size = Table->RowCount;

int \*array = new int[size];

int result = 0;

FromFile(array, size);

if (radioButton1->Checked)

{

start = clock();

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

{

result = Searching::linear(array, element, size);

}

end = clock();

TIME->Text = Convert::ToString(((double)end - (double)start) / ((double)CLOCKS\_PER\_SEC)/N);

if (result == -1)

{

MessageBox::Show("Nothis was found", "Fatal Error",

MessageBoxButtons::OK, MessageBoxIcon::Exclamation);

}

else

{

int test = 0;

//int iter = 0;

test = selectedOut();

if (test > SIZE)

{

test = 0;

}

Table->Rows[test]->Selected = false;

Table->Rows[result]->Selected = true;

selectedIn(result);

richTextBox1->Text = Convert::ToString(result + 1);

int iter = IterationOut();

Iteration->Text = Convert::ToString(iter);

}

}

else

{

if (radioButton2->Checked)

{

start = clock();

Sorting::buble(array, SIZE);

InFile(array, SIZE);

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

{

Table->Rows[i]->Cells[1]->Value = array[i];

}

int test = 0;

test = selectedOut();

if (test > SIZE)

{

test = 0;

}

Table->Rows[test]->Selected = false;

//const int n = 500;

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

{

result = Searching::Binary(array, element, SIZE, 0, SIZE - 1);

}

end = clock();

if (result != -1)

{

TIME->Text = Convert::ToString(((double)end - (double)start) / ((double)CLOCKS\_PER\_SEC)/N);

richTextBox1->Text = Convert::ToString(result+1);

Table->Rows[result]->Selected = true;

selectedIn(result);

int iter = IterationOut();

Iteration->Text = Convert::ToString(iter);

}

else

{

MessageBox::Show("Nothis was found", "Fatal Error",

MessageBoxButtons::OK, MessageBoxIcon::Exclamation);

}

}

else

{

if (radioButton3->Checked)

{

start = clock();

Sorting::buble(array, SIZE);

InFile(array, SIZE);

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

{

Table->Rows[i]->Cells[1]->Value = array[i];

}

int test = 0;

test = selectedOut();

if (test > SIZE)

{

test = 0;

}

Table->Rows[test]->Selected = false;

result = 0;

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

{

result = Searching::ABinary(array, element, SIZE);

}

end = clock();

if (result != -1)

{

TIME->Text = Convert::ToString(((double)end - (double)start) / ((double)CLOCKS\_PER\_SEC)/N);

richTextBox1->Text = Convert::ToString(result + 1);

Table->Rows[result]->Selected = true;

selectedIn(result);

int iter = IterationOut();

Iteration->Text = Convert::ToString(iter);

}

else

{

MessageBox::Show("Nothis was found", "Fatal Error",

MessageBoxButtons::OK, MessageBoxIcon::Exclamation);

}

}

else

{

start = clock();

Sorting::buble(array, SIZE);

InFile(array, SIZE);

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

{

Table->Rows[i]->Cells[1]->Value = array[i];

}

int test = 0;

test = selectedOut();

if (test > SIZE)

{

test = 0;

}

Table->Rows[test]->Selected = false;

result = 0;

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

{

result = Searching::Sharа(array,element,SIZE);

}

end = clock();

if (result != -1)

{

TIME->Text = Convert::ToString(((double)end - (double)start) / ((double)CLOCKS\_PER\_SEC) / N);

richTextBox1->Text = Convert::ToString(result + 1);

Table->Rows[result]->Selected = true;

selectedIn(result);

int iter = IterationOut();

Iteration->Text = Convert::ToString(iter);

}

else

{

MessageBox::Show("Nothis was found", "Fatal Error",

MessageBoxButtons::OK, MessageBoxIcon::Exclamation);

}

}

}

}

}

}

}

private: System::Void button2\_Click(System::Object^ sender, System::EventArgs^ e)

{

Finding();

}

};

}

## MyForm.cpp

#include "MyForm.h"

using namespace System;

using namespace System::Windows::Forms;

[STAThread]

void main(cli::array<String^>^ arg) {

Application::EnableVisualStyles();

Application::SetCompatibleTextRenderingDefault(false);

Cursach::MyForm form;

Application::Run(%form);

}