МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ім. І. Сікорського

Кафедра

інформатики та програмної інженерії

(повна назва кафедри, циклової комісії)

**КУРСОВА РОБОТА**

з Основ програмування

(назва дисципліни)

на тему: пошук заданих елементів у масиві

Студента 1 курсу, групи ІП-11

Прищепи Владислава Станіславовича

Спеціальності 121 «Інженерія програмного забезпечення»

Керівник Головченко Максим Миколайович

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

Кількість балів: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

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

Київ- 2022 рік

КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ім. І. Сікорського

(назва вищого навчального закладу)

Кафедра інформатики та програмної інженерії

Дисципліна Основи програмування

Напрям "ІПЗ"

Курс 1 Група ІП-11 Семестр 2

ЗАВДАННЯ

на курсову роботу студента

Прищепи Владислава Станіславовича

(прізвище, ім’я, по батькові)

1. Тема роботи пошук заданих елементів у масиві

2. Строк здачі студентом закінченої роботи 12.06.2022

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 10.02.2022

**КАЛЕНДАРНИЙ ПЛАН**

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Назва етапів курсової роботи | Термін виконання етапів роботи | Підписи керівника, студента |
| 1. | Отримання теми курсової роботи | 10.02.2022 |  |
| 2. | Підготовка ТЗ | 02.05.2022 |  |
| 3. | Пошук та вивчення літератури з питань курсової роботи | 03.05.2022 |  |
| 4. | Розробка сценарію роботи програми | 04.05.2022 |  |
| 6. | Узгодження сценарію роботи програми з керівником | 04.05.2022 |  |
| 5. | Розробка (вибір) алгоритму рішення задачі | 04.05.2022 |  |
| 6. | Узгодження алгоритму з керівником | 04.05.2022 |  |
| 7. | Узгодження з керівником інтерфейсу користувача | 05.05.2022 |  |
| 8. | Розробка програмного забезпечення | 06.05.2022 |  |
| 9. | Налагодження розрахункової частини програми | 06.05.2022 |  |
| 10. | Розробка та налагодження інтерфейсної частини програми | 07.05.2022 |  |
| 11. | Узгодження з керівником набору тестів для контрольного прикладу | 25.05.2022 |  |
| 12. | Тестування програми | 26.05.2022 |  |
| 13. | Підготовка пояснювальної записки | 05.06.2022 |  |
| 14. | Здача курсової роботи на перевірку | 12.06.2022 |  |
| 15. | Захист курсової роботи | 15.06.2022 |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Студент

(підпис)

Керівник Головченко Максим Миколайович

(підпис) (прізвище, ім’я, по батькові)

"\_\_\_" \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2022 р.

**АНОТАЦІЯ**

Пояснювальна записка до курсової: 61 сторінки, 6 рисунків, 12 таблиць, 2 посилань.

Об’єкт дослідження: пошук заданого елемента у масиві.

Мета роботи: дослідження методів пошуку заданого елемента у масиві, розробка програмного забезпечення для генерації масиву та пошуку заданих елементів у ньому.

Опановано розробку програмного забезпечення з використанням ООП. Приведені змістовні постановки задач, їх індивідуальні математичні моделі, а також описано детальний процес розв’язання кожної з них.

Виконана програмна реалізація програми пошуку заданого елементу у масиві.

ЗМІСТ

[ВСТУП 6](file:///C:\Users\User\Downloads\Telegram%20Desktop\ПЗ%20КР%20з%20ОП%20Рябов%20ІП-11.docx#_Toc105166659)

[1 ПОСТАНОВКА ЗАДАЧІ 6](file:///C:\Users\User\Downloads\Telegram%20Desktop\ПЗ%20КР%20з%20ОП%20Рябов%20ІП-11.docx#_Toc105166660)

[2 ТЕОРИТИЧНІ ВІДОМОСТІ 8](file:///C:\Users\User\Downloads\Telegram%20Desktop\ПЗ%20КР%20з%20ОП%20Рябов%20ІП-11.docx#_Toc105166661)

[3 ОПИС АЛГОРИТМІВ 9](file:///C:\Users\User\Downloads\Telegram%20Desktop\ПЗ%20КР%20з%20ОП%20Рябов%20ІП-11.docx#_Toc105166662)

[3.1 Алгоритм генерації масиву 10](file:///C:\Users\User\Downloads\Telegram%20Desktop\ПЗ%20КР%20з%20ОП%20Рябов%20ІП-11.docx#_Toc105166663)

[3.2 Алгоритм пошуку заданого елемента у масиві 11](file:///C:\Users\User\Downloads\Telegram%20Desktop\ПЗ%20КР%20з%20ОП%20Рябов%20ІП-11.docx#_Toc105166664)

[3.3 Алгоритм лінійного пошуку 13](file:///C:\Users\User\Downloads\Telegram%20Desktop\ПЗ%20КР%20з%20ОП%20Рябов%20ІП-11.docx#_Toc105166665)

[3.4 Алгоритм бінарного пошуку 13](file:///C:\Users\User\Downloads\Telegram%20Desktop\ПЗ%20КР%20з%20ОП%20Рябов%20ІП-11.docx#_Toc105166666)

[3.5 Алгоритм однорідного бінарного пошуку 14](file:///C:\Users\User\Downloads\Telegram%20Desktop\ПЗ%20КР%20з%20ОП%20Рябов%20ІП-11.docx#_Toc105166667)

[3.6 Алгоритм пошуку методом Шарра 15](file:///C:\Users\User\Downloads\Telegram%20Desktop\ПЗ%20КР%20з%20ОП%20Рябов%20ІП-11.docx#_Toc105166668)

[4 ОПИС ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 18](file:///C:\Users\User\Downloads\Telegram%20Desktop\ПЗ%20КР%20з%20ОП%20Рябов%20ІП-11.docx#_Toc105166669)

[4.1 Діаграма класів програмного забезпечення 18](file:///C:\Users\User\Downloads\Telegram%20Desktop\ПЗ%20КР%20з%20ОП%20Рябов%20ІП-11.docx#_Toc105166670)

[4.2 Опис методів частин програмного забезпечення 18](file:///C:\Users\User\Downloads\Telegram%20Desktop\ПЗ%20КР%20з%20ОП%20Рябов%20ІП-11.docx#_Toc105166671)

[4.2.1 Стандартні методи 19](file:///C:\Users\User\Downloads\Telegram%20Desktop\ПЗ%20КР%20з%20ОП%20Рябов%20ІП-11.docx#_Toc105166672)

[4.2.2 Користувацькі методи 21](file:///C:\Users\User\Downloads\Telegram%20Desktop\ПЗ%20КР%20з%20ОП%20Рябов%20ІП-11.docx#_Toc105166673)

[5 ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 26](file:///C:\Users\User\Downloads\Telegram%20Desktop\ПЗ%20КР%20з%20ОП%20Рябов%20ІП-11.docx#_Toc105166674)

[5.1 План тестування 26](file:///C:\Users\User\Downloads\Telegram%20Desktop\ПЗ%20КР%20з%20ОП%20Рябов%20ІП-11.docx#_Toc105166675)

[5.2 Приклади тестування 26](file:///C:\Users\User\Downloads\Telegram%20Desktop\ПЗ%20КР%20з%20ОП%20Рябов%20ІП-11.docx#_Toc105166676)

[6 ІНСТРУКЦІЯ КОРИСТУВАЧА 32](file:///C:\Users\User\Downloads\Telegram%20Desktop\ПЗ%20КР%20з%20ОП%20Рябов%20ІП-11.docx#_Toc105166677)

[6.1 Робота з програмою 32](file:///C:\Users\User\Downloads\Telegram%20Desktop\ПЗ%20КР%20з%20ОП%20Рябов%20ІП-11.docx#_Toc105166678)

[6.2 Системні вимоги 34](file:///C:\Users\User\Downloads\Telegram%20Desktop\ПЗ%20КР%20з%20ОП%20Рябов%20ІП-11.docx#_Toc105166680)

[ВИСНОВОК 36](file:///C:\Users\User\Downloads\Telegram%20Desktop\ПЗ%20КР%20з%20ОП%20Рябов%20ІП-11.docx#_Toc105166681)

[ПЕРЕЛІК ПОСИЛАНЬ 37](file:///C:\Users\User\Downloads\Telegram%20Desktop\ПЗ%20КР%20з%20ОП%20Рябов%20ІП-11.docx#_Toc105166682)

[ДОДАТОК А ТЕХНІЧНЕ ЗАВДАННЯ 38](file:///C:\Users\User\Downloads\Telegram%20Desktop\ПЗ%20КР%20з%20ОП%20Рябов%20ІП-11.docx#_Toc105166683)

[ДОДАТОК Б ТЕКСТИ ПРОГРАМНОГО КОДУ 41](file:///C:\Users\User\Downloads\Telegram%20Desktop\ПЗ%20КР%20з%20ОП%20Рябов%20ІП-11.docx#_Toc105166684)

[Main.py 42](file:///C:\Users\User\Downloads\Telegram%20Desktop\ПЗ%20КР%20з%20ОП%20Рябов%20ІП-11.docx#_Toc105166685)

[Data.py 43](file:///C:\Users\User\Downloads\Telegram%20Desktop\ПЗ%20КР%20з%20ОП%20Рябов%20ІП-11.docx#_Toc105166686)

[Interface.py 45](file:///C:\Users\User\Downloads\Telegram%20Desktop\ПЗ%20КР%20з%20ОП%20Рябов%20ІП-11.docx#_Toc105166687)

[MENU.py 51](file:///C:\Users\User\Downloads\Telegram%20Desktop\ПЗ%20КР%20з%20ОП%20Рябов%20ІП-11.docx#_Toc105166688)

[SM.py 54](file:///C:\Users\User\Downloads\Telegram%20Desktop\ПЗ%20КР%20з%20ОП%20Рябов%20ІП-11.docx#_Toc105166689)

[TF.py 59](file:///C:\Users\User\Downloads\Telegram%20Desktop\ПЗ%20КР%20з%20ОП%20Рябов%20ІП-11.docx#_Toc105166690)

[Tumbler.py.cs 60](file:///C:\Users\User\Downloads\Telegram%20Desktop\ПЗ%20КР%20з%20ОП%20Рябов%20ІП-11.docx#_Toc105166691)

# **ВСТУП**

Дана робота присвячена розробці програми «Пошук елемента у заданому масиві» з використанням об’єктно-орієнтованого програмування. Задача програмного забезпечення полягає в графічному та текстовому відображенні згенерованого масиву, елементів масиву, що були перевірені, послідовності перевірки та шуканого елемента при його наявності в масиві при виконані пошуку.

# **1** **ПОСТАНОВКА ЗАДАЧІ**

Розробити програмне забезпечення, що буде виконувати один вказаний з чотирьох алгоритмів пошуку вказаного елемента у згенерованому масиві унікальних значень.

Вхідними даними є введені значення розміру масиву та шуканого елемента, а також обраний метод пошуку.

Необхідно реалізувати кнопку генерації масиву, яка буде генерувати масив на вказаний розмір масиву; якщо розмір масиву менше 1000, то програма змінить його на 1000; якщо до цього вже була виведена інформація, то очистити її. Вихідними даними буде текст у вигляді переліку всіх елементів згенерованого масиву:

0, 3, 12, 15, 16, 19, 20, 23, …

Необхідно реалізувати кнопку пошуку вказаного елемента масиву, яка буде виконувати один з чотирьох обраних алгоритмів пошуку над масивом; якщо до цього вже була виведена інформація, то очистити її. Вихідними даними буде текст у вигляді переліку всіх елементів згенерованого масиву з підсвіченими елементами, що були перевірені, та шуканим, за наявності:

0, 3, 12, 15, 16, 19, 20, 23, …

повідомлення про наявність шуканого елемента у масиві:

Yes

а також тексту, що буде представляти собою послідовність перевірки елементів:

1.[4](16)

2.[1](3)

3.[2](12)

Необхідно реалізувати меню, що буде надавати змогу очистити всі введені дані, отримати коротку інформацію про програму та завершити виконання програми.

# **2 ТЕОРИТИЧНІ ВІДОМОСТІ**

**Послідовний (лінійний) пошук** – метод пошуку, при якому кожен елемент у масиві послідовно з початку перевіряється до тих пір, поки шуканий елемент не буде знайдено у масиві або масив не закінчиться.

**Бінарний пошук** – метод пошуку, при якому застосовується дроблення масиву на половини. Дроблення масиву полягає в тому, щоб вказати перший (first) й останній (last) індекс елементів у масиві, де може бути шуканий елемент, знайти індекс серединного (mid) елемента проміжку, визначити, більше чи менше шукане значення від ключа серединного елемента і перейти до лівої половини (last=mid-1), якщо шукане значення менше, або до правої(first=mid+1), якщо шукане значення більше за перевірене, потім повторити процедуру. Виконується алгоритм у відсортованому масиві до тих пір, поки шукане не буде знайдено, або поки проміжок пошуку не сягне одного елемента і він не виявиться шуканим (first>last).

**Однорідний бінарний пошук** – метод пошуку, що є модифікацією бінарного пошуку, в якому використовується тільки покажчик поточного елемента (i), який, при умові, що індексація елементів починається з нуля, спочатку дорівнює покажчику зміни (b), котрий, у свою чергу, спочатку дорівнює цілій частині від ділення довжини масиву на два (b=N//2, N-довжина масиву). Кожного разу покажчик зміни ділиться на два (b//2), а покажчик зміни зміщується на нову b+1 залежно від того, шукане менше за поточне чи більше. Метод використовується для відсортованого масиву й виконується доти, поки покажчик зміни не прирівняється до нуля, або поки поточний елемент не стане шуканим.

**Метод Шарра** – метод пошуку, що базується на однорідному бінарному пошуку. Спочатку порівнюється K і Кі , де і=2^k-1 , k = [log2(N)]. Якщо К < Ki , використовується однорідний бінарний пошук з послідовністю b = 2^(k−1) , 2^(k−2) , .... 1, 0. При К > Ki , і N > 2^k , встановлюємо і = і`=N-2^l , де, l = [log2(N-2^k+1)] і роблячи вигляд, що першим було порівняння К > Ki , використовуємо однорідний пошук з b= 2^(l−1), 2^(l−2) .

**3 ОПИС АЛГОРИТМІВ**

Перелік всіх основних змінних та їхнє призначення наведено в таблиці

## Таблиця 3.1– Основні змінні та їхні призначення

|  |  |
| --- | --- |
| Змінна | Призначення |
| leng | Об’єкт, що містить введене значення розміру масиву |
| quan | Значення розміру масиву |
| GTL | Масив імен тегів червоного кольору |
| t1 | Текстове поле для виводу згенерованого масиву |
| ST | Ім’я зеленого тегу |
| Array | Масив згенерований |
| Str1 | Текст для t1 |
| t2 | Текстове поле для виводу послідовності перевірки |
| mes4 | Напис, що повідомляє про успіх чи невдачу пошуку |
| D | Об’єкт класу Data, що зберігає введені та отримані дані |
| snum | Об’єкт, що містить введене значення шуканого елемента |
| skn | Значення шуканого елемента |
| var | Значення обраного варіанту метода пошуку |
| meth | Об’єкт, що містить варіант обраного варіанту метода пошуку |
| Liss | Об’єкт класу SM |
| Lis | Масив індексів перевірених значень |
| LT | Масив початків і кінців перевірених елементів у Str1 |
| Str2 | Текст для t2 |
| a1 | Індекс початку шуканого елемента у Str1 |
| a2 | Індекс кінця шуканого елемента у Str1 |
| itt | Ітерація для лінійного пошуку |
| flag | Булеве значення, що відображає, чи продовжується пошук |
| num | Розмір згенерованого масиву як аргумент методів пошуку |
| seeken | Шукане як аргумент методів пошуку |
| Arr | Масив згенерований як аргумент методів пошуку |
| posb | Індекс початку проміжку, що розглядується |
| pose | Індекс кінця проміжку, що розглядується |
| mid | Серединне значення проміжку, що розглядується |
| b | Величина зміни для поточного стану |
| i | Поточний стан однорідного бінарного пошуку |
| k | Степінь, необхідна для розрахунку поточного стану |
| l | Степінь, необхідна для розрахунку поточного стану |
| LG | Масив індексів перевірених значень як атрибут класу SM |

## 3.1 Алгоритм генерації масиву

1. ПОЧАТОК
2. Зчитування leng
3. ЯКЩО leng складається лише з цифр:

3.1 quan := int(leng)

3.1 ЦИКЛ обходу масиву GTL (с-поточний елемент)

3.1.1 Видалити з t1 тег с

3.2 ЯКЩО ST не пустий:

3.2.1 Видалити з t1 тег ST

3.3 GTL := []

3.4 ST := “”

3.5 ЯКЩО quan < 1000:

3.5.1 quan := 1000

3.5.2 Вивести у поле вводу значення quan

3.6 Array := random.sample(range(quan\*5), quan).sort()

3.7 Str1 := str(Array)[1:len(str(Array))-1]

3.8 Вивести Str1 в t1

3.9 Вивести “None” в t2

3.10 Вивести “None” в mes4

3.11 Зберегти Array та quan у D

4. ІНАКШЕ:

4.1 Вивести у поле вводу “Wrong input!”

5. КІНЕЦЬ

## 3.2 Алгоритм пошуку шуканого елемента

1. ПОЧАТОК

2. Зчитування snum

3. ЯКЩО snum складається лише з цифр:

3.1 skn := snum

3.2 Взяти Array та quan з D

3.3 Взяти var з meth

3.4 Зберегти skn у D

3.5 Створити Liss як об’єкт класу SM

3.6 ЦИКЛ обходу масиву GTL (с-поточний елемент)

3.6.1 Видалити з t1 тег с

3.7 ЯКЩО ST не пустий:

3.7.1 Видалити з t1 тег ST

3.8 GTL := []

3.9 ST := “”

3.10 ЯКЩО var == 1:

3.10.1 Liss.Linear\_Search(Array, quan, skn) (3.3)

3.11 ІНАКШЕ ЯКЩО var == 2:

3.11.1 Liss.Binary\_Search(Array, quan, skn) (3.4)

3.12 ІНАКШЕ ЯКЩО var == 3:

3.12.1 Liss.O\_B\_Search(Array, quan, skn) (3.5)

3.13 ІНАКШЕ ЯКЩО var == 4:

3.13.1 Liss.Sharrah\_Search(Array, quan, skn) (3.6)

3.14 Зберегти у D значення атрибуту Lis з Liss

3.15 Створити пусті масив LT та строки Str1, Str2

3.16 Створити індекси а1 та а2 для шуканого елемента у строкі Str1 зі значеннями -1

3.17 ЦИКЛ обходу масиву Array (с-поточний елемент)

3.17.1 ЯКЩО Array[c] належить Liss:

3.17.1.1 ЯКЩО Array[c]==skn:

3.17.1.1.1 Встановити індекси а1 та а2 як початковий й кінцевий індекси шуканого елемента

3.17.1.1.2 Додати Array[c] у Str1

3.17.1.2 ІНАКШЕ:

3.17.1.2.1 Додати до LT індекси початку й кінця Array[c]

3.17.1.2.2 Додати Array[c] у Str1

3.17.2 ІНАКШЕ:

3.17.2.1 Додати Array[c] у Str1

3.18 Str1 := str(Str1)[1:len(Str1)-1]

3.19 ЦИКЛ обходу масиву Liss (с-поточний елемент)

3.19.1 Str2 += str(с+1)+".["+str(Liss.GetLis()[с])+"]("+str(Array[Liss. GetLis() [с]]) + ")\n"

3.20 Str2:=Str2[:len(Str2)-1]

3.21 ЯКЩО Str2 пустий:

3.21.1 Str2 := “Choose method!”

3.22 Вивести Str1 в t1

3.23 Вивести Str2 в t2

3.24 ЦИКЛ обходу масиву LT (с-поточний елемент, шаг обходу масиву - 2):

3.24.1 Створити тег червоного кольору з початком у LT[c] та кінцем у LT[c+1]

3.24.2 Додати у GTL створений тег

3.25 ЯКЩО а1 != -1:

3.25.1 Створити тег зеленого кольору з початком у а1 та кінцем у а2

3.25.2 Присвоїти ST ім’я створеного тега

3.26 ЯКЩО len(Liss.GetLis())>0 та arr[Liss.GetLis()[len(Liss.GetLis())-1]]==skn:

3.26.1 Тексту mes4 присвоїти “Yes” зеленого кольору

3.27 ІНАКШЕ:

3.27.1 Тексту mes4 присвоїти “No” червоного кольору

4. ІНАКШЕ:

4.1 Вивести у поле вводу “Wrong input!”

5. КІНЕЦЬ

3.3 Алгоритм послідовного пошуку

1.ПОЧАТОК

2.Lis :=[]

3.itt :=0

4. flag :=1

5. ПОКИ flag == 1:

5.1 ЯКЩО itt == num:

5.1.1 flag :=0

5.2 ІНАКШЕ ЯКЩО Arr[itt] == seeken:

5.2.1 flag :=0

5.2.2 Lis.append(itt)

5.3 ІНАКШЕ:

5.3.1 Lis.append(itt)

5.3.2 itt := itt+1

6. LG := Lis

7. КІНЕЦЬ

3.4 Алгоритм бінарного пошуку

1.ПОЧАТОК

2.Lis :=[]

3.posb :=0

4.pose := num-1

5. flag :=1

6. ПОКИ flag == 1:

6.1 ЯКЩО pose>=posb:

6.1.1 mid := (posb+pose)//2

6.1.2 ЯКЩО Arr[mid]==seeken:

6.1.2.1 Lis.append(mid)

6.1.2.2 flag :=0

6.1.3 ІНАКШЕ ЯКЩО Arr[mid] > seeken:

6.1.3.1 Lis.append(mid)

6.1.3.2 pose := mid-1

6.1.4 ІНАКШЕ:

6.1.4.1 Lis.append(mid)

6.1.4.2 posb := mid+1

6.2 ІНАКШЕ:

6.2.1 flag :=0

7. LG := Lis

8. КІНЕЦЬ

3.5 Алгоритм однорідного бінарного пошуку

1.ПОЧАТОК

2.Lis :=[]

3. b :=int(num)//2

4. i :=b

5. flag :=1

6. ПОКИ flag == 1:

6.1 ЯКЩО b>0:

6.1.1 ЯКЩО i>=num:

6.1.1.1 i:=i-b//2+1

6.1.1.2 b:=b//2

6.1.2 ІНАКШЕ ЯКЩО i<0:

6.1.2.1 i:=i+b//2+1

6.1.2.2 b:=b//2

6.1.3 ІНАКШЕ ЯКЩО Arr[int(i)]==seeken:

6.1.3.1 Lis.append(int(i))

6.1.3.2 flag:=0

6.1.4 ІНАКШЕ ЯКЩО Arr[int(i)]<seeken:

6.1.4.1 Lis.append(int(i))

6.1.4.2 i:=i+b//2+1

6.1.4.3 b:=b//2

6.1.5 ІНАКШЕ:

6.1.5.1 Lis.append(int(i))

6.1.5.2 i:=i-b//2+1

6.1.5.3 b:=b//2

6.2 ІНАКШЕ:

6.2.1 ЯКЩО i<num та i>=0:

6.2.1.1 Lis.append(int(i))

6.2.2 flag:=0

7. LG := Lis

8. КІНЕЦЬ

3.6 Алгоритм пошуку методом Шарра

1.ПОЧАТОК

2.Lis :=[]

3. flag :=1

4. k:=(math.log2(num))//1

5. i:=(2^k)-1

6. ЯКЩО seeken <= Arr[int(i)]:

6.1 b:=(i+1)

6.2 ПОКИ flag == 1:

6.2.1 ЯКЩО b>0:

6.2.1.1 ЯКЩО i>=num:

6.2.1.1.1 i:=i-b//2+1

6.2.1.1.2 b:=b//2

6.2.1.2 ІНАКШЕ ЯКЩО i<0:

6.2.1.2.1 i:=i+b//2+1

6.2.1.2.2 b:=b//2

6.2.1.3 ІНАКШЕ ЯКЩО Arr[int(i)]==seeken:

6.2.1.3.1 Lis.append(int(i))

6.2.1.3.2 flag:=0

6.2.1.4 ІНАКШЕ ЯКЩО Arr[int(i)]<seeken:

6.2.1.4.1 Lis.append(int(i))

6.2.1.4.2 i:=i+b//2+1

6.2.1.4.3 b:=b//2

6.2.1.5 ІНАКШЕ:

6.2.1.5.1 Lis.append(int(i))

6.2.1.5.2 i:=i-b//2+1

6.2.1.5.3 b:=b//2

6.2.2 ІНАКШЕ:

6.2.2.1 ЯКЩО i<num та i>=0:

6.2.2.1.1 Lis.append(int(i))

6.2.2.2 flag:=0

7.ІНАКШЕ:

7.1 l:=math.log2(num-i)

7.2 i:=num-2^l

7.3 b:=2^(l)

7.4 ПОКИ flag == 1:

7.4.1 ЯКЩО b>0:

7.4.1.1 ЯКЩО i>=num:

7.4.1.1.1 i:=i-b//2+1

7.4.1.1.2 b:=b//2

7.4.1.2 ІНАКШЕ ЯКЩО i<0:

7.4.1.2.1 i:=i+b//2+1

7.4.1.2.2 b:=b//2

7.4.1.3 ІНАКШЕ ЯКЩО Arr[int(i)]==seeken:

7.4.1.3.1 Lis.append(int(i))

7.4.1.3.2 flag:=0

7.4.1.4 ІНАКШЕ ЯКЩО Arr[int(i)]<seeken:

7.4.1.4.1 Lis.append(int(i))

7.4.1.4.2 i:=i+b//2+1

7.4.1.4.3 b:=b//2

7.4.1.5 ІНАКШЕ:

7.4.1.5.1 Lis.append(int(i))

7.4.1.5.2 i:=i-b//2+1

7.4.1.5.3 b:=b//2

7.4.2 ІНАКШЕ:

7.4.2.1 ЯКЩО i<num та i>=0:

7.4.2.1.1 Lis.append(int(i))

7.4.2.2 flag:=0

8. LG := Lis

9. КІНЕЦЬ

# **ОПИС ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ**

## **Діаграма класів програмного забезпечення**

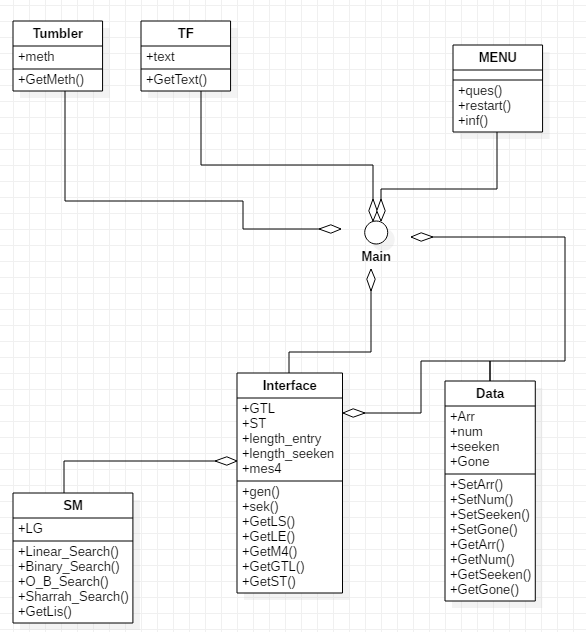


Рисунок 4.1 – Діаграма класів

## **Опис методів частин програмного забезпечення**

### Користувацькі методи

Таблиця 4.1 – Користувацькі методи

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № п/п | Назва классу | Назва функції | Призначення функції | Опис вхідних параметрів | Опис вихідних параметрів |
| 1 | Tumbler | \_\_init\_\_ | Конструктор класу | self, canvas | - |
| 2 | Tumbler | GetMeth | Гетер атрибута meth | self | meth(IntVar) |
| 3 | TF | \_\_init\_\_ | Конструктор класу | self, canvas | - |
| 4 | TF | GetText | Гетер атрибута text | self | text(Text) |
| 5 | MENU | \_\_init\_\_ | Конструктор класу | self, canvas, leng\_seek, leng\_en, m4, meth, GTL, ST, t1, t2 | - |
| 6 | MENU | ques | Завершення роботи | self, canvas | - |
| 7 | MENU | restart | Очищення введених полів | self, leng\_seek, leng\_en, m4, meth, GTL, ST, t1, t2 | - |
| 8 | MENU | inf | Вивід короткої довідки про програму | self | - |
| 9 | SM | \_\_init\_\_ | Конструктор класу | self | - |
| 10 | SM | Linear\_Search | Послідовний пошук | self, Arr, num, seeken | - |
| 11 | SM | Binary\_Search | Бінарний пошук | self, Arr, num, seeken | - |
| 12 | SM | O\_B\_Search | Однорідний бінарний пошук | self, Arr, num, seeken | - |
| 13 | SM | Sharrah\_Search | Пошук методом Шарра | self, Arr, num, seeken | - |
| 14 | SM | GetLis | Гетер атрибута LG | self | LG(list) |
| 15 | Interface | \_\_init\_\_ | Конструктор класу | self, canvas, D, leng, snum, meth, t1, t2 | - |
| 16 | Interface | gen | Генерація масиву | self, D, leng, t1, t2 | - |
| 17 | Interface | sek | Пошук шуканого елементу | self, D, snum, meth, t1, t2 | - |
| 18 | Interface | GetLS | Гетер атрибута length\_seeken | self | length\_seeken (Entry) |
| 19 | Interface | GetLE | Гетер атрибута length\_entry | self | length\_entry (Entry) |
| 20 | Interface | GetM4 | Гетер атрибута mes4 | self | mes4(Label) |
| 21 | Interface | GetGTL | Гетер атрибута GTL | self | GTL(list) |
| 22 | Interface | GetST | Гетер атрибута ST | self | ST(str) |
| 23 | Data | \_\_init\_\_ | Конструктор класу | self | - |
| 24 | Data | SetArr | Сетер атрибута Arr | ArrN | - |
| 25 | Data | SetNum | Сетер атрибута num | numN | - |
| 26 | Data | SetSeeken | Сетер атрибута seeken | seekenN | - |
| 27 | Data | SetGone | Сетер атрибута Gone | GoneN | - |
| 28 | Data | GеtArr | Гетер атрибута Arr | self | Arr (list) |
| 29 | Data | GetNum | Гетер атрибута num | self | num (int) |
| 30 | Data | GetSeeken | Гетер атрибута seeken | self | seeken (int) |
| 31 | Data | GetGone | Гетер атрибута Gone | self | Gone (list) |

### Стандартні методи

Таблиця 4.2

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № п/п | Назва классу | Назва функції | Призначення функції | Опис вхідних параметрів | Опис вихідних параметрів |
| 1 | TK | \_\_init\_\_ | Ініціалізація вікна програми | self | - |
| 2 | TK | title | Встановлення заголовку вікна | str | - |
| 3 | TK | geometry | Встановлення розміру вікна | str | - |
| 4 | Frame | \_\_init\_\_ | Ініціалізація рамки | canvas, padx, pady | - |
| 5 | tkinter | grid | Розташування у вікні віджета | row, column, sticky | - |
| 6 | StringVar | \_\_init\_\_ | Ініціалізація строки для введення через віджети | self | - |
| 7 | TK | mainloop | Запуск вікна програми | self | - |
| 8 | Label | \_\_init\_\_ | Ініціалізація напису | canvas, text, padx, pady, font | - |
| 9 | Entry | \_\_init\_\_ | Ініціалізація строки вводу | canvas, textvariable, font | - |
| 10 | Button | \_\_init\_\_ | Ініціалізація кнопки | canvas, text, command, font | - |
| 11 | - | int | Ініціалізація цілого числа | str | int |
| 12 | - | range | Створення списку | int | list |
| 13 | - | len | Підрахунок довжини строки чи розміру масиву | str/list | int |
| 14 | Text | tag\_delete | Видалення тегу з текстового поля | str | - |
| 15 | Text, Entry | delete | Видалення вмісту поля | pos, pos | - |
| 16 | Text, Entry | insert | Введення у поле вмісту | pos, str | - |
| 17 | random | sample | Вибір з масиву випадковим чином вказану к-сть елементів | list, int | list |
| 18 | list | sort | Сортування масиву | list | - |
| 19 | - | str | Ініціалізація строки | int | str |
| 20 | IntVar, StringVar | get | Отримання числового значення чи строки для подальшого використання | self | int/str |
| 21 | list | append | Додавання у кінець списку | Any | - |
| 22 | Text | tag\_add | Додавання тегу у текстове поле | Str, pos1, pos2 | - |
| 23 | Text | tag\_config | Налаштування тегу | Str, bg, fg | - |
| 24 | tkinter | config | Налаштування віджета | text, fg | - |
| 25 | Menu | \_\_init\_\_ | Ініціалізація меню програми | self | - |
| 26 | Menu | add\_cascade | Додання пункту до меню | label, command, font | - |
| 27 | tkinter | askyesno | Вивід повідомлення із запитанням | Name of box, Text (str) | bool |
| 28 | TK | destroy | Закриття вікна програми | self | - |
| 29 | tkinter | messagebox.showinfo | Вивід повідомлення | Name of box, Text (str) | - |
| 30 | Math | log2 | Логарифм за основою від 2 | float | float |
| 31 | Text | \_\_init\_\_ | Ініціалізація текстового поля | canvas, padx, pady, font | - |
| 32 | Scrollbar | \_\_init\_\_ | Ініціалізація повзунка | canvas, command | - |
| 33 | Intvar | \_\_init\_\_ | Ініціалізація цілого числа для введення через віджети | self | - |
| 34 | Radiobutton | \_\_init\_\_ | Ініціалізація позиції перемикача | canvas, text, value, variable, padx, pady, font | - |
| 35 | Text | yview | Можливість прокручувати текстове поле | self | - |
| 36 | Scrollbar | set | Встановлення повзунка у іншого віджета | self | - |

# **ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ**

## **План тестування**

Складемо план тестування програмного забезпечення, за допомогою якого протестуємо весь основний функціонал та реакцію на виключні ситуації

А) Тестування виводу інформації

1) Тестування генерації масиву

2) Тестування пошуку заданого елементу масиву

Б)Тестування меню

1)Тестування виведення вікна з інформацією

2)Тестування очищення полей введення

3)Тестування завершення програми

В) Тестування виняткових ситуацій

1) Тестування при вводі у поля вводу символів, які не є цифрами

2)Тестування при вводі у поле вводу розміру масиву, менше 1000

3) Тестування при відсутності вибору варіанту пошуку

Проведемо тестування( таблиці 5.1 – 5.8 )

## **Приклади тестування**

Таблиця 5.1 - Тестування генерації масиву

|  |  |
| --- | --- |
| Мета тесту | Перевірити вивід згенерованого масиву |
| Початковий стан програми | Відкрите вікно програми |
| Вхідні дані | 1002 |
| Схема проведення тесту | Ввести 1002 у поле для введення розміру масиву, натиснути кнопку Generate |
| Очікуваний результат | Виведення масиву у тестове поле для згенерованого масиву |
| Стан програми після проведення випробувань | Згенерований масив виведений у текстове поле для згенерованого масиву |

Таблиця 5.2 - Тестування пошуку заданого елементу масиву

|  |  |
| --- | --- |
| Мета тесту | Перевірити підсвічувансть перевірених елементів червоним і шуканого зеленим, вивід послідовності дій та повідомлення про успіх чи провал пошуку |
| Початковий стан програми | Відкрите вікно програми |
| Вхідні дані | 1003, 17, Бінарний Пошук |
| Схема проведення тесту | Ввести 1003 у поле для введення розміру масиву, натиснути кнопку Generate, ввести 17 у поле для введення шуканого елемента масиву, обрати у перемикачі Бінарний пошук, натиснути кнопку Search |
| Очікуваний результат | Виведення масиву у тестове поле для згенерованого масиву з підсвіченими червоним перевіреними елементами та, за наявності, зеленим шуканого, виведення послідовності дій у текстове поле для послідовності дій, виведення зеленого Yes чи червоного No у місце, де відображується результат пошуку |
| Стан програми після проведення випробувань | Згенерований масив виведений у текстове поле для згенерованих масивів з підсвіченими червоним перевіреними елементами і зеленим шуканого, послідовність дій виведена у текстове поле для послідовності дій, виведене зелене Yes |

Таблиця 5.3 - Тестування виведення вікна з інформацією

|  |  |
| --- | --- |
| Мета тесту | Перевірити вивід вікна з повідомленням, яке містить інформацію про використання програми |
| Початковий стан програми | Відкрите вікно програми |
| Вхідні дані | - |
| Схема проведення тесту | Натиснути в меню INFO |
| Очікуваний результат | Вивід вікна з інформацією про використання програми |
| Стан програми після проведення випробувань | Виведене вікно з інформацією про використання програми |

Таблиця 5.4 - Тестування очищення полей введення

|  |  |
| --- | --- |
| Мета тесту | Перевірити очищення полей введення |
| Початковий стан програми | Відкрите вікно програми |
| Вхідні дані | 1002, 7, Послідовний пошук |
| Схема проведення тесту | Ввести 1002 у поле для введення розміру масиву, натиснути кнопку Generate, ввести 7 у поле для введення шуканого елемента масиву, обрати у перемикачі Послідовний пошук, натиснути кнопку Search, натиснути у меню кнопку RESTART |
| Очікуваний результат | Повернення вікна програми до початкового стану |
| Стан програми після проведення випробувань | Вікно програми повернулося до початкового вигляду |

Таблиця 5.5 - Тестування завершення програми

|  |  |
| --- | --- |
| Мета тесту | Перевірити завершення виконання програми |
| Початковий стан програми | Відкрите вікно програми |
| Вхідні дані | - |
| Схема проведення тесту | Натиснути в меню EXIT |
| Очікуваний результат | Програма завершить свою роботу, вікно програми закриється |
| Стан програми після проведення випробувань | Програма завершила свою роботу, вікно програми закрилося |

Таблиця 5.6 - Тестування при вводі у поля вводу символів, які не є цифрами

|  |  |
| --- | --- |
| Мета тесту | Перевірити реакцію програми на введення у поля вводу розміру та шуканого символів, які не є цифрами |
| Початковий стан програми | Відкрите вікно програми |
| Вхідні дані | u, -0.1, Бінарний пошук |
| Схема проведення тесту | Ввести u у поле для введення розміру масиву, натиснути кнопку Generate, ввести -0.1 у поле для введення шуканого елемента масиву, обрати у перемикачі Бінарний пошук, натиснути кнопку Search |
| Очікуваний результат | Виведення у поля вводу розміру та шуканого “Wrong input!” |
| Стан програми після проведення випробувань | Виведено у поля вводу “Wrong input!” |

Таблиця 5.7 - Тестування при вводі у поле вводу розміру масиву, менше 1000

|  |  |
| --- | --- |
| Мета тесту | Перевірити здатність програми виправити число, введене у поле введення розміру, менше 1000 на 1000 і згенерувати масив |
| Початковий стан програми | Відкрите вікно програми |
| Вхідні дані | 999 |
| Схема проведення тесту | Ввести 999 у поле для введення розміру масиву, натиснути кнопку Generate |
| Очікуваний результат | У поле введення розміру виведеться 1000, у текстове поле виведення масиву виведеться масив на 1000 елементів |
| Стан програми після проведення випробувань | У поле введення розміру вивелася 1000, у текстове поле виведення масиву вивівся масив на 1000 елементів |

Таблиця 5.8 - Тестування при відсутності вибору варіанту пошуку

|  |  |
| --- | --- |
| Мета тесту | Перевірити реакцію програми а відсутність обраного методу пошуку |
| Початковий стан програми | Відкрите вікно програми |
| Вхідні дані | 1005, 21 |
| Схема проведення тесту | Ввести 1005 у поле для введення розміру масиву, натиснути кнопку Generate, ввести 21 у поле для введення шуканого елемента масиву, натиснути кнопку Search |
| Очікуваний результат | У текстове поле для виводу масиву виведеться масив, у текстове поле для послідовностей виведеться “Choose method!” |
| Стан програми після проведення випробувань | У текстове поле для виводу масиву вивівся масив, у текстове поле для послідовностей вивелося “Choose method!” |

# **6 ІНСТРУКЦІЯ КОРИСТУВАЧА**

## **6.1 Робота з програмою**

Після запуску виконавчого файлу з розширенням \*.exe, відкривається головне вікно програми (Рисунок 6.1).

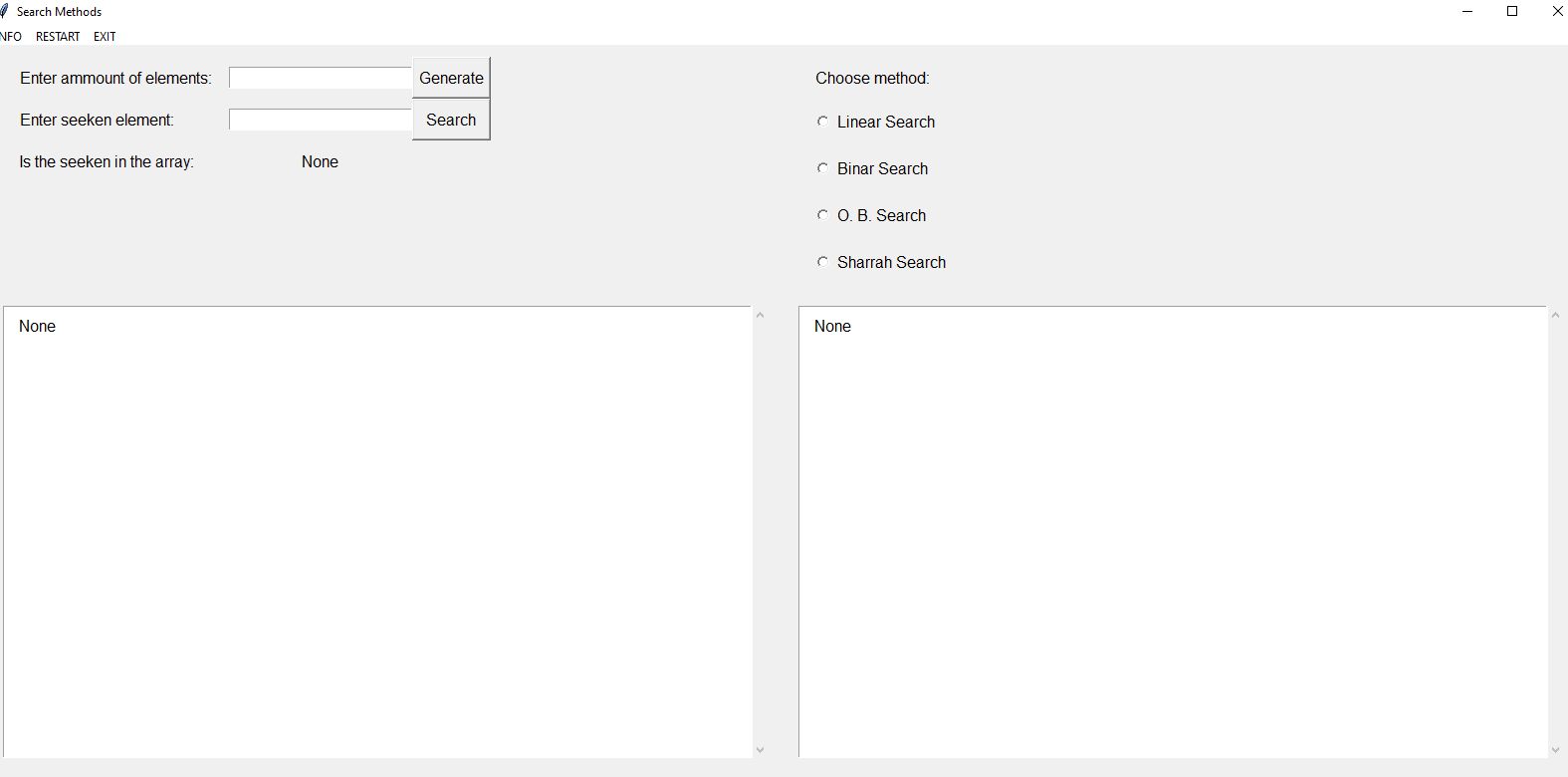


Рисунок 6.1 – Головне вікно програми

Далі вводимо розмір масиву (більше 1000) у поле для введення розміру масиву для генерації і натискаємо кнопку Generate (Рисунок 6.2).

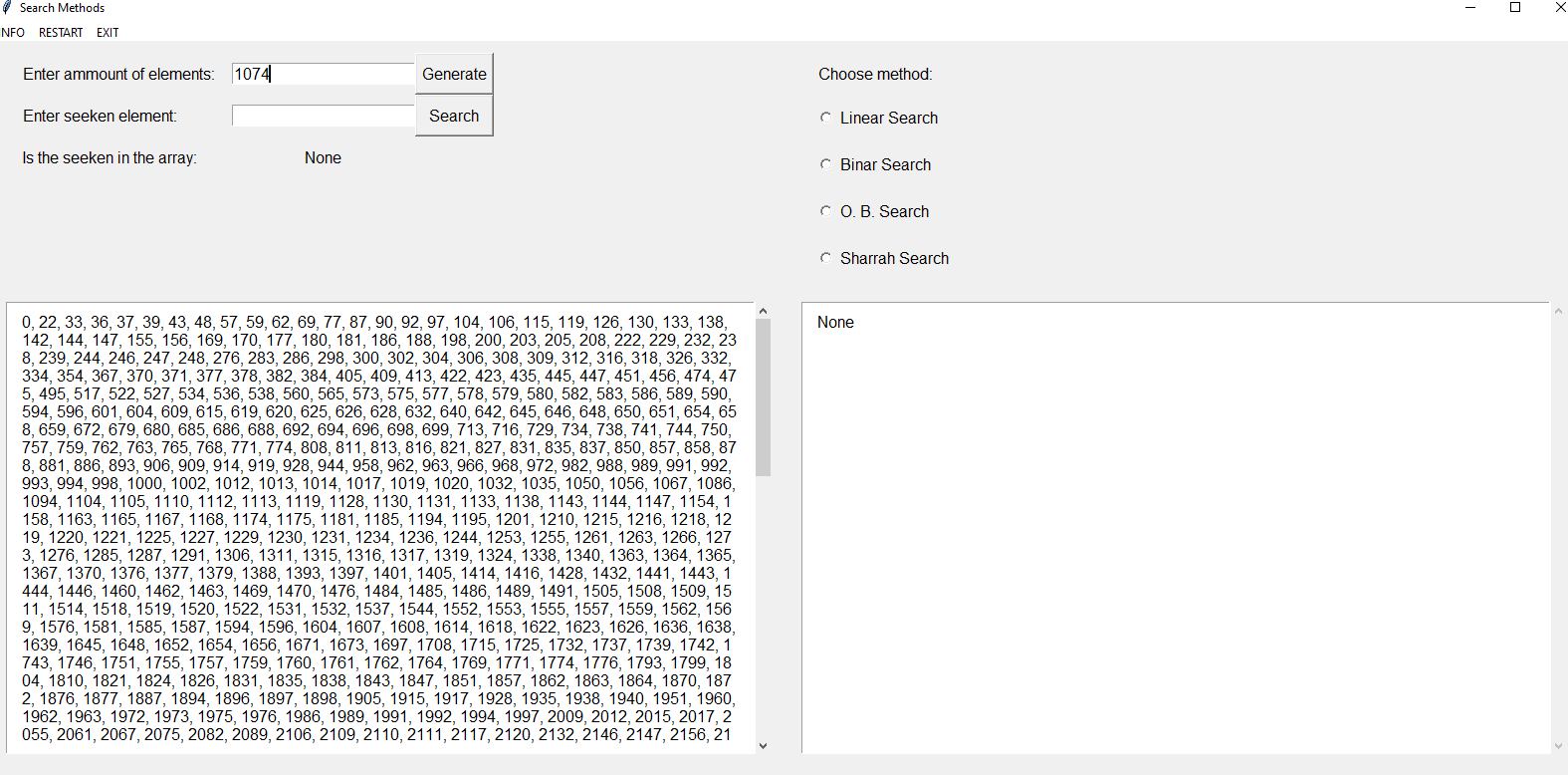


Рисунок 6.2 – Головне вікно програми після генерації масиву

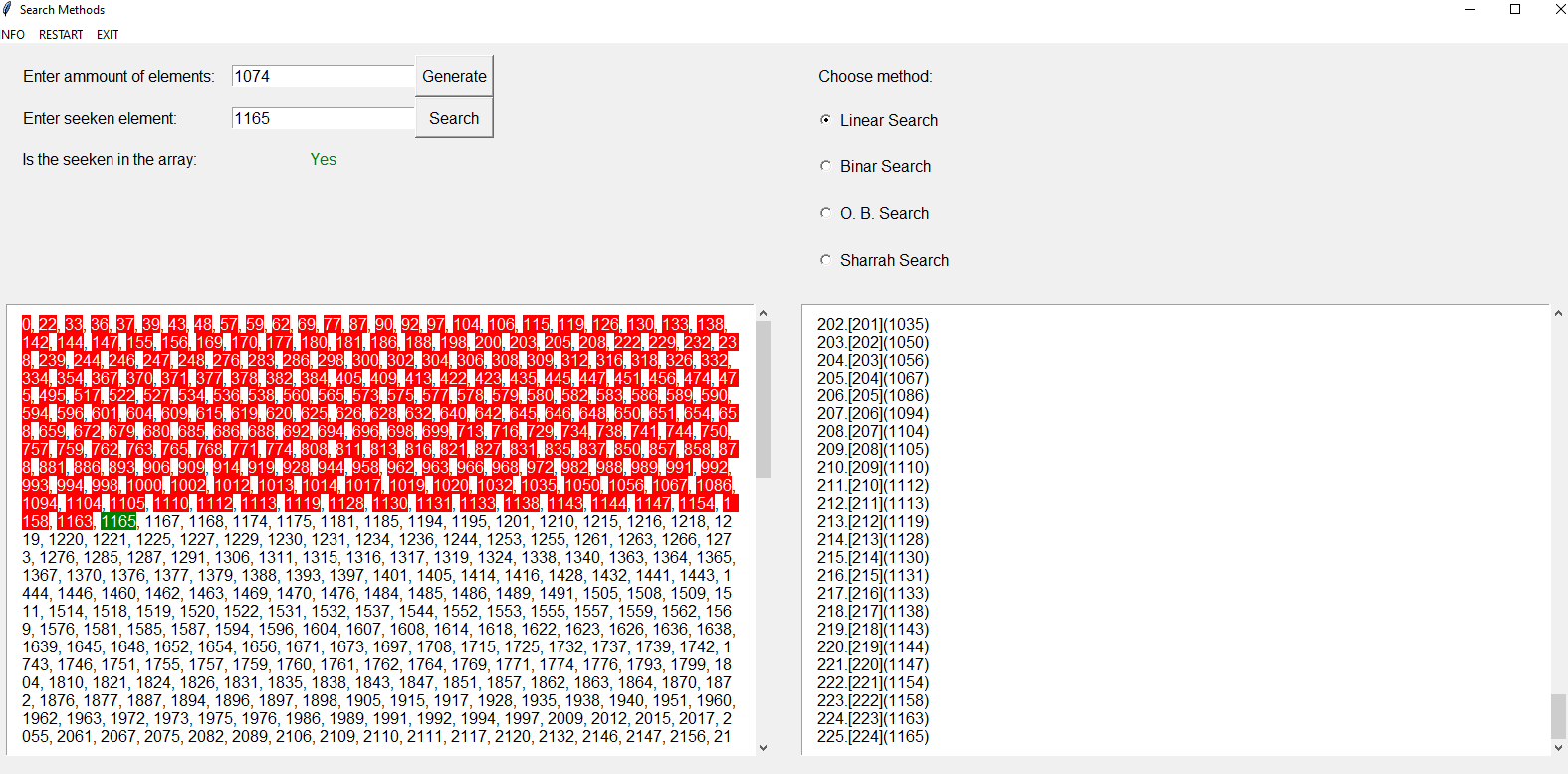
Далі вводимо значення шуканого елемента у поле введення шуканого елемента, обираємо метод пошуку, натискаємо кнопку Search (Рисунок 6.3). 

Рисунок 6.3 – Головне вікно програми після пошуку шуканого елемента

Для очищення вікна від введених даних та виведеного результату, натискаємо в меню RESTART(Рисунок 6.4).

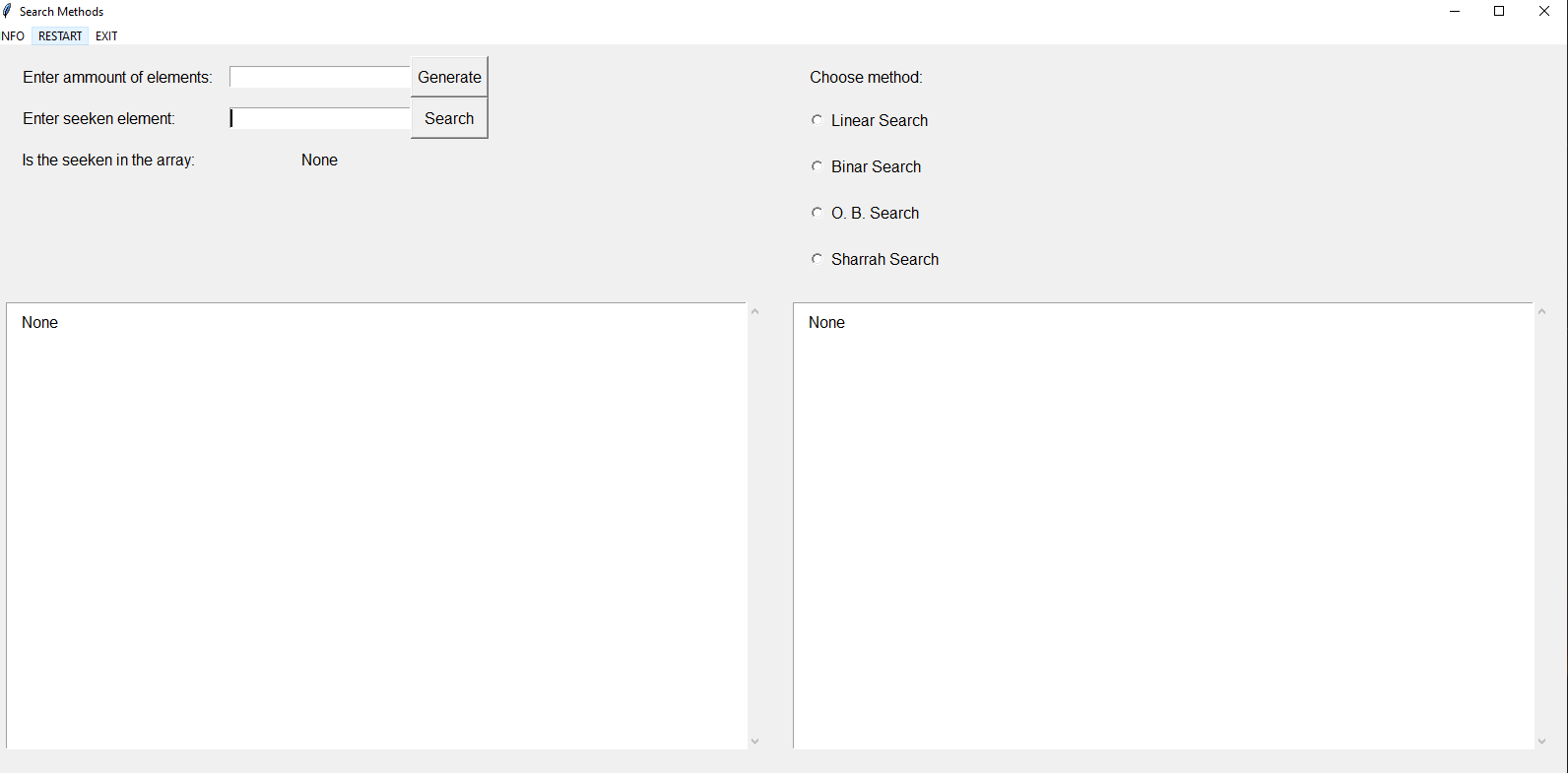


Рисунок 6.4 – Головне вікно програми після очищення

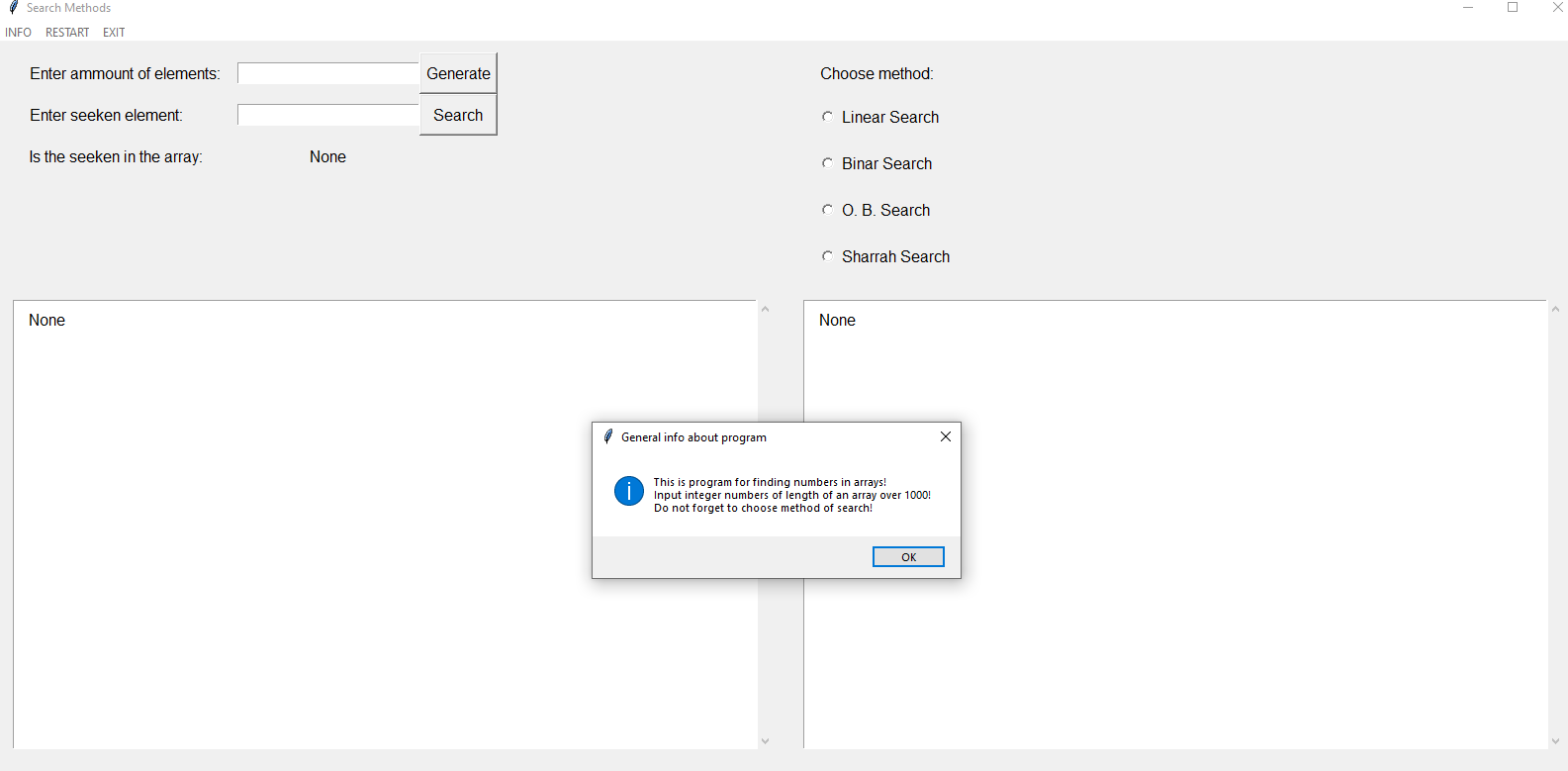
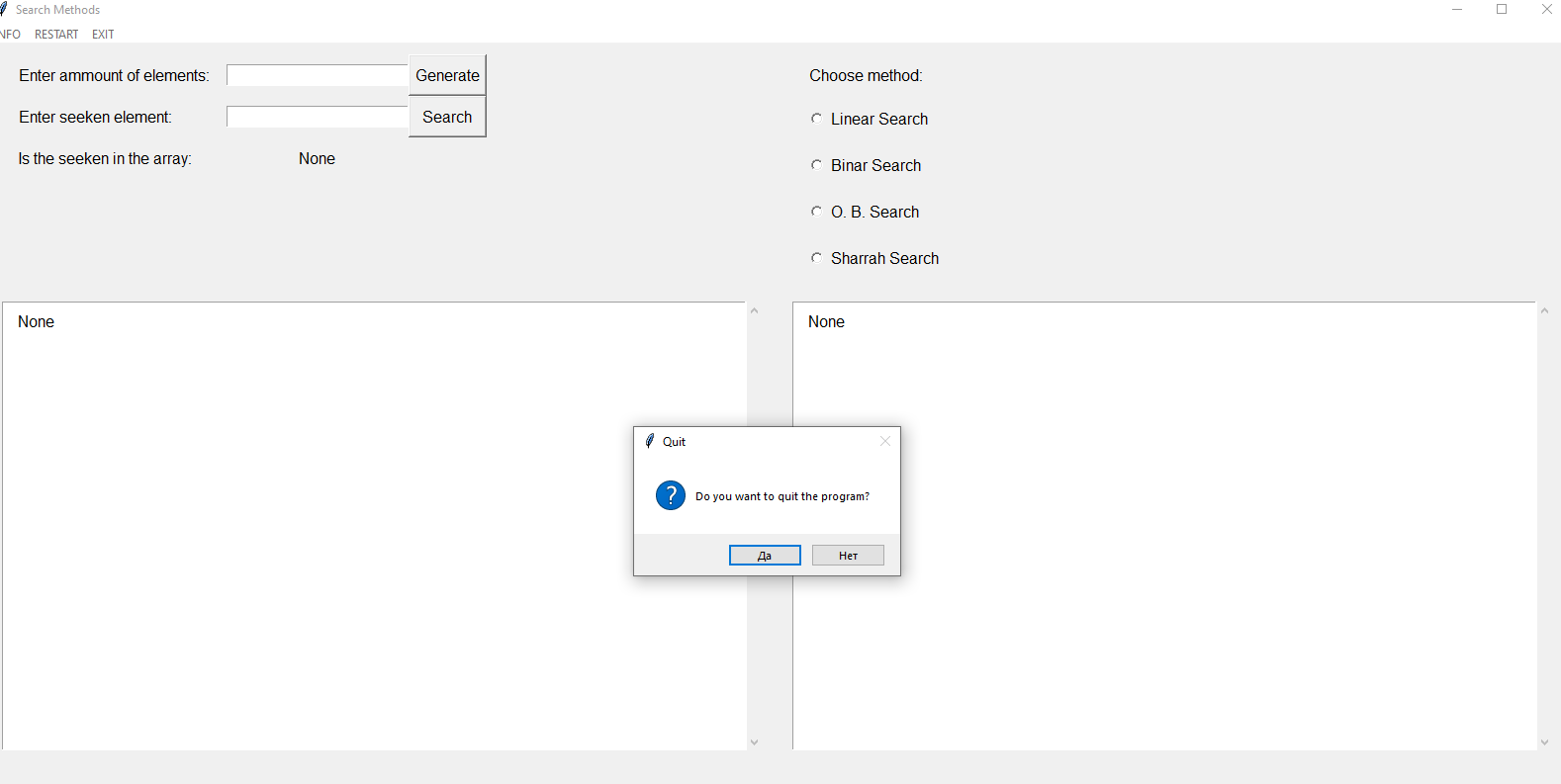
Для виводу інформації про користування програмою, натискаємо в меню INFO(Рисунок 6.5). 

Рисунок 6.5 – Головне вікно програми після натискання INFO

Для завершення роботи, натискаємо в меню EXIT(Рисунок 6.6).  Рисунок 6.6 – Головне вікно програми після натискання EXIT

## **Системні вимоги**

Системні вимоги до програмного забезпечення наведені в таблиці 10.1.

Таблиця 10.1 – Системні вимоги програмного забезпечення

|  |  |  |
| --- | --- | --- |
|  | Мінімальні | Рекомендовані |
| Операційна система | Windows® XP/Windows Vista/Windows 7/ Windows 8/Windows 10/Windows 11 (з останніми обновленнями) | Windows10/  Windows 11  (з останніми обновленнями) |
| Процесор | Intel® Pentium® ІІІ  1.0 GHz або  AMD Athlon™ 1.0 GHz | Intel® Pentium® D або AMD Athlon™ 64 X2 |
| Оперативна пам'ять | 256 MB RAM (для Windows® XP) / 1 GB RAM (для Windows Vista/Windows 7/  Windows 8/Windows 10) | 2 GB RAM |
| Відеоадаптер | Intel GMA 950 з відеопам'яттю об'ємом не менше 64 МБ (або сумісний аналог) | |
| Дисплей | 800х600 | 1024х768 або краще |

Висновок

Під час курсової роботи було вивчено метод розробки програмного забезпечення з використанням ООП на прикладі програми пошуку заданих елементів у масиві. Були наведені теоретичні відомості, що пояснюють методи пошуку; описано алгоритм, за яким комп’ютер виконує пошук; таблиці користувацьких і стандартних методів; діаграма класів; результати тестувань; інструкція користувача.

Перелік посилань

1. Linear search URL:

<https://en.wikipedia.org/wiki/Linear_search#:~:text=In%20computer%20science%2C%20a%20linear,Linear%20search>

1. Binary search algorithm URL:

<https://en.wikipedia.org/wiki/Binary_search_algorithm#:~:text=In%20computer%20science%2C%20binary%20search,middle%20element%20of%20the%20array>.

Додаток А Технічне завдання

КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ім. І. Сікорського

Кафедра

інформатики та програмної інженерії

Затвердив

Керівник Головченко М. М.

«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2022 р.

Виконавець:

Студент Прищепа Владислав Станіславович

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2022 р.

ТЕХНІЧНЕ ЗАВДАННЯ

на виконання курсової роботи

на тему: пошук заданих елементів у масиві

з дисципліни:

«Основи програмування»

Київ 2022

* 1. *Мета*: Метою курсової роботи є розробка програми пошуку заданих елементів у масиві
  2. *Дата початку роботи*: «\_\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_202\_ р.
  3. *Дата закінчення роботи*: «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_ 202\_ р.
  4. *Вимоги до програмного забезпечення*.

1. Функціональні вимоги:

* можливість обирати метод пошуку
* вивід згенерованого масиву
* вивід послідовності перевірки елементів масиву
* висвітлення перевірених та шуканого елементів масиву після пошуку
* можливість очистити вікно програми від введених та виведених даних
* можливість завершити роботу програми через меню

1. Нефункціональні вимоги:

* Все програмне забезпечення та супроводжуюча технічна документація повинні задовольняти наступним ДЕСТам:

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 - 2015 - Розробка технічної документації.

* 1. *Стадії та етапи розробки*:

1. Об'єктно-орієнтований аналіз предметної області задачі (до\_\_.\_\_.202\_ р.)
2. Об'єктно-орієнтоване проектування архітектури програмної системи (до \_\_.\_\_.202\_р.)
3. Розробка програмного забезпечення (до \_\_.\_\_.202\_р.)
4. Тестування розробленої програми (до \_\_.\_\_.202\_р.)
5. Розробка пояснювальної записки (до \_\_.\_\_.202\_ р.).
6. Захист курсової роботи (до \_\_.\_\_.202\_ р.).
   1. *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 оцінювання.

Додаток Б Тексти програмного коду

*студента групи ІП-1 І курсу*

*Прищепи В.С.*

(Найменування програми(документа))

*Тексти програмного коду програми пошуку заданих елементів у масиві*

(Вид носія даних)

*Github*

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

*21 арк, 50,2 Кб*

B

Main.py

from Tumbler import \*

from TF import \*

from tkinter import \*

from Data import \*

from MENU import \*

from Interface import \*

#Основний файл, у якому створюються усі об'єкти та ініціалізуються налаштування програми

root = Tk()

root.title("Search Methods")

root.geometry("1600x750")

fram1=Frame(root, padx=15, pady=10)

fram1.grid(row=0, column=0, sticky='nsew')

fram2=Frame(root, padx=15, pady=10)

fram2.grid(row=1, column=0, sticky='nsew')

fram3=Frame(root, padx=15, pady=10)

fram3.grid(row=0, column=1, sticky='nsew')

fram4=Frame(root, padx=15, pady=10)

fram4.grid(row=1, column=1, sticky='nsew')

Dat =Data()

leng = StringVar()

snum = StringVar()

ChM=Tumbler(fram3)

text1=TF(fram2)

text2=TF(fram4)

Cons=Interface(fram1, Dat, leng, snum, ChM.GetMeth(), text1.GetText(), text2.GetText())

main\_menu = MENU(root, Cons.GetLS(), Cons.GetLE(), Cons.GetM4(), ChM.GetMeth(), Cons.GetGTL(), Cons.GetST(), text1.GetText(), text2.GetText())

root.mainloop()

DATA.py

#Клас, що представляє отримані дані в ході виконання програми

#Атрибути:

#Arr : list

#Масив з унікальних цілочисленних значень

#num : int

#К-сть елементів у масиві

#seeken : int

#Значення шуканого елемента масива

#Gone : list

#Масив індексів елментів, які були перевірені в ході виконання алгоритмів пошуку

# Методи

# \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# def \_\_init\_\_(self)

# Конструктор класу

# Аргументи:

# 1) self - об'єкт класу

# Повертає: None

# \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

class Data:

def \_\_init\_\_(self):

self.Arr=[]

self.num=0

self.seeken=0

self.Gone=[]

def SetArr(self, ArrN):

self.Arr = ArrN

def SetNum(self, numN):

self.num = numN

def SetSeeken(self, seekenN):

self.seeken = seekenN

def SetGone(self, GoneN):

self.Gone = GoneN

def GetArr(self):

return self.Arr

def GetNum(self):

return self.num

def GetSeeken(self):

return self.seeken

def GetGone(self):

return self.Gone

Interface.py

from tkinter import \*

import random

from SM import \*

#Клас, що представляє інтерфейс програми

#Атрибути:

#GTL : list

#Масив індексів початкового та останього індекса елементів, що були перевірені

#ST : str

#Ім'я тегу для підсвічування шуканого елемента

#length\_entry : Entry

#Поле вводу розміру масива

#length\_seeken : Entry

#Поле вводу значення шуканого елементу

#mes4 : Label

#Напис, що відображає результат пошуку шуканого елемента у масиві (Yes/No/None)

#Методи

# ------

# \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# def \_\_init\_\_(self, canvas, D, leng, snum, meth, t1, t2)

# Конструктор класу

# Аргументи:

# 1) self - об'єкт класу

# 2) canvas - рамка вікна програми, у яке виводиться інтерфейс

# 3) D - об'єкт класу Data, що зберігає введені дані

# 4) leng - розмір генерованого масиву

# 5) snum - шуканий елемент

# 6) meth - варіант метода пошуку

# 7) t1 - текстове поле для виведення згенерованого масива

# 8) t2 - текстове поле для виведення послідовності перевірки

# Повертає: None

# \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# def gen(self, D, leng, t1, t2)

# Генерація масива унікальних елементів

# Аргументи:

# 1) self - об'єкт класу

# 2) D - об'єкт класу Data, що зберігає введені дані

# 3) leng - розмір генерованого масиву

# 4) t1 - текстове поле для виведення згенерованого масива

# 5) t2 - текстове поле для виведення послідовності перевірки

# Повертає: None

# \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# def sek(self, D, snum, meth, t1, t2)

# Пошук шуканого елемента масива

# Аргументи:

# 1) self - об'єкт класу

# 2) D - об'єкт класу Data, що зберігає введені дані

# 3) snum - шуканий елемент

# 4) meth - варіант метода пошуку

# 5) t1 - текстове поле для виведення згенерованого масива

# 6) t2 - текстове поле для виведення послідовності перевірки

# Повертає: None

# \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

class Interface:

def \_\_init\_\_(self, canvas, D, leng, snum, meth, t1, t2):

self.GTL=[]

self.ST=""

mes1 = Label(canvas, text = "Enter ammount of elements:", padx=15, pady=10, font="14")

mes1.grid(row=0, column=0, sticky=W)

self.length\_entry=Entry(canvas, textvariable=leng, font="14")

self.length\_entry.grid(row=0, column=1, sticky='ew')

mes2 = Label(canvas, text = "Enter seeken element:", padx=15, pady=10, font="14")

mes2.grid(row=1, column=0, sticky=W)

self.length\_seeken=Entry(canvas, textvariable=snum, font="14")

self.length\_seeken.grid(row=1, column=1, sticky='ew')

mes3 = Label(canvas, text = "Is the seeken in the array:", padx=15, pady=10, font="14")

mes3.grid(row=2, column=0, sticky=W)

self.mes4 = Label(canvas, text = "None", fg='black', padx=15, pady=10, font="14")

self.mes4.grid(row=2, column=1, sticky='nsew')

but1=Button(canvas, text = "Generate", command=lambda: self.gen(D, leng, t1, t2), font="14")

but1.grid(row=0, column=2, sticky='nsew')

but2=Button(canvas, text = "Search", command=lambda: self.sek(D, snum, meth, t1, t2), font="14")

but2.grid(row=1, column=2, sticky='nsew')

def gen(self, D, leng, t1, t2):

quanstr=leng.get()

if quanstr.isnumeric():

quan=int(quanstr)

for i in range(len(self.GTL)):

t1.tag\_delete(self.GTL[i])

if self.ST!="":

t1.tag\_delete(self.ST)

self.GTL=[]

self.ST=""

if quan < 1000:

quan=1000

self.length\_entry.delete("0", "end")

self.length\_entry.insert(INSERT, "1000")

Array=random.sample(range(quan\*5), quan)

Array.sort()

Str1=str(Array)

Str1=Str1[1:len(Str1)-1]

t1.delete("1.0", "end")

t1.insert(INSERT, Str1)

t2.delete("1.0", "end")

t2.insert(INSERT, "None")

self.mes4.config(text="None")

D.SetArr(Array)

D.SetNum(quan)

else:

self.length\_entry.delete("0", "end")

self.length\_entry.insert(INSERT, "Wrong input!")

def sek(self, D, snum, meth, t1, t2):

sknstr=snum.get()

if sknstr.isnumeric():

skn=int(sknstr)

quan=D.GetNum()

arr=D.GetArr()

var=meth.get()

D.SetSeeken(skn)

Liss=SM()

for i in range(len(self.GTL)):

t1.tag\_delete(self.GTL[i])

if self.ST!="":

t1.tag\_delete(self.ST)

self.GTL=[]

self.ST=""

if var==1:

Liss.Linear\_Search(arr, quan, skn)

elif var==2:

Liss.Binary\_Search(arr, quan, skn)

elif var==3:

Liss.O\_B\_Search(arr, quan, skn)

elif var==4:

Liss.Sharrah\_Search(arr, quan, skn)

D.SetGone(Liss.GetLis())

Str1=""

Str2=""

LT = []

a1 = int(-1)

a2 = int(-1)

for i in range(len(arr)):

if Liss.GetLis().count(i)>0:

if(arr[i]==skn):

a1=len(Str1)

Str1+=str(arr[i])+", "

a2=len(Str1)-3

else:

LT.append(len(Str1))

Str1+=str(arr[i])+", "

LT.append(len(Str1)-3)

else:

Str1+=str(arr[i])+", "

Str1=Str1[:len(Str1)-2]

for i in range(len(Liss.GetLis())):

Str2+=str(i+1)+".["+str(Liss.GetLis()[i])+"]("+str(arr[Liss.GetLis()[i]])+")\n"

Str2=Str2[:len(Str2)-1]

if Str2=="":

Str2="Choose method!"

t1.delete("1.0", "end")

t1.insert(INSERT, Str1)

t2.delete("1.0", "end")

t2.insert(INSERT, Str2)

i=0

while i < len(LT):

C1="1."+str(LT[i])

C2="1."+str(LT[i+1]+1)

Name="Gone"+str(i//2+1)

t1.tag\_add(Name, C1, C2)

t1.tag\_config(Name, background="red", foreground="white")

self.GTL.append(Name)

i+=2

if a1 !=-1:

C1="1."+str(a1)

C2="1."+str(a2+1)

self.ST="Seeken"

t1.tag\_add(self.ST, C1, C2)

t1.tag\_config(self.ST, background="green", foreground="white")

if len(Liss.GetLis())>0 and arr[Liss.GetLis()[len(Liss.GetLis())-1]]==skn:

self.mes4.config(text="Yes", fg='green')

else:

self.mes4.config(text="No", fg='red')

else:

self.length\_seeken.delete("0", "end")

self.length\_seeken.insert(INSERT, "Wrong input!")

def GetLS(self):

return self.length\_seeken

def GetLE(self):

return self.length\_entry

def GetM4(self):

return self.mes4

def GetGTL(self):

return self.GTL

def GetST(self):

return self.ST

MENU.py

from tkinter import \*

from tkinter.messagebox import \*

from tkinter import messagebox

#Клас, що представляє меню програми

#Атрибути:

# Відсутні

# Методи

# ------

# \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# def \_\_init\_\_(self, canvas, leng\_seek, leng\_en, m4, meth, GTL, ST, t1, t2)

# Конструктор класу

# Аргументи:

# 1) self - об'єкт класу

# 2) canvas - рамка вікна програми, у яке виводиться інтерфейс

# 3) leng\_seek - поле вводу розміру масива

# 4) leng\_en - поле вводу значення шуканого елементу

# 5) m4 - напис, що відображає результат пошуку шуканого елемента у масиві (Yes/No/None)

# 6) meth - варіант метода пошуку

# 7) GTL - масив індексів початкового та останього індекса елементів, що були перевірені

# 8) ST - ім'я тегу для підсвічування шуканого елемента

# 9) t1 - текстове поле для виведення згенерованого масива

# 10) t2 - текстове поле для виведення послідовності перевірки

# Повертає: None

# \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# def ques(self, canvas)

# Завершення програми

# Аргументи:

# 1) self - об'єкт класу

# 2) canvas - рамка вікна програми, у яке виводиться інтерфейс

# Повертає: None

# \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# def restart(self, leng\_seek, leng\_en, m4, meth, GTL, ST, t1, t2)

# Очищення форми програми

# Аргументи:

# 1) self - об'єкт класу

# 2) leng\_seek - поле вводу розміру масива

# 3) leng\_en - поле вводу значення шуканого елементу

# 4) m4 - напис, що відображає результат пошуку шуканого елемента у масиві (Yes/No/None)

# 5) meth - варіант метода пошуку

# 6) GTL - масив індексів початкового та останього індекса елементів, що були перевірені

# 7) ST - ім'я тегу для підсвічування шуканого елемента

# 8) t1 - текстове поле для виведення згенерованого масива

# 9) t2 - текстове поле для виведення послідовності перевірки

# Повертає: None

# \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# def inf(self)

# Вивід короткої довідки про програму

# Аргументи:

# 1) self - об'єкт класу

# Повертає: None

# \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

class MENU:

def \_\_init\_\_(self, canvas, leng\_seek, leng\_en, m4, meth, GTL, ST, t1, t2):

MM=Menu()

MM.add\_cascade(label="INFO", command=lambda: self.inf(), font="14")

MM.add\_cascade(label="RESTART", command=lambda: self.restart(leng\_seek, leng\_en, m4, meth, GTL, ST, t1, t2), font="14")

MM.add\_cascade(label="EXIT", command=lambda: self.ques(canvas), font="14")

canvas.config(menu=MM)

def ques(self, canvas):

if askyesno("Quit", "Do you want to quit the program?"):

canvas.destroy()

def restart(self, leng\_seek, leng\_en, m4, meth, GTL, ST, t1, t2):

for i in range(len(GTL)):

t1.tag\_delete(GTL[i])

if ST!="":

t1.tag\_delete(ST)

leng\_seek.delete("0", "end")

leng\_en.delete("0", "end")

m4.config(text="None", fg='black')

meth.set(0)

t1.delete("1.0", "end")

t1.insert(INSERT, "None")

t2.delete("1.0", "end")

t2.insert(INSERT, "None")

def inf(self):

tline="This is program for finding numbers in arrays!\nInput integer numbers of length of an array over 1000!\nDo not forget to choose method of search!"

messagebox.showinfo("General info about program", tline)

SM.py

import math

#Клас, що представляє методи пошуку елементів у программі

#Атрибути:

#LG : list

#Масив індексів елментів, які були перевірені в ході виконання алгоритмів пошуку

# Методи

# ------

# \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# def \_\_init\_\_(self)

# Конструктор класу

# Аргументи:

# 1) self - об'єкт класу

# Повертає: None

# \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# def Linear\_Search(self, Arr, num, seeken)

# Послідовний пошук

# Аргументи:

# 1) self - об'єкт класу

# 2) Arr - масив з унікальних цілочисленних значень

# 3) num - к-сть елементів у масиві

# 4) seeken - значення шуканого елемента масива

# Повертає: None

# \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# def Binary\_Search(self, Arr, num, seeken)

# Бінарний пошук

# Аргументи:

# 1) self - об'єкт класу

# 2) Arr - масив з унікальних цілочисленних значень

# 3) num - к-сть елементів у масиві

# 4) seeken - значення шуканого елемента масива

# Повертає: None

# \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# def O\_B\_Search(self, Arr, num, seeken)

# Однорідний бінарний пошук

# Аргументи:

# 1) self - об'єкт класу

# 2) Arr - масив з унікальних цілочисленних значень

# 3) num - к-сть елементів у масиві

# 4) seeken - значення шуканого елемента масива

# Повертає: None

# \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# def Sharrah\_Search(self, Arr, num, seeken)

# Пошук методом Шарра

# Аргументи:

# 1) self - об'єкт класу

# 2) Arr - масив з унікальних цілочисленних значень

# 3) num - к-сть елементів у масиві

# 4) seeken - значення шуканого елемента масива

# Повертає: None

# \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

class SM:

def \_\_init\_\_(self):

self.LG=[]

def GetLis(self):

return self.LG

def Linear\_Search(self, Arr, num, seeken):

Lis=[]

itt = 0

flag=1

while flag == 1:

if itt == num :

flag = 0

elif Arr[itt] == seeken :

flag = 0

Lis.append(itt)

else:

Lis.append(itt)

itt+=1

self.LG=Lis

def Binary\_Search(self, Arr, num, seeken):

Lis= []

posb=0

pose=num-1

flag=1

while flag==1:

if pose>=posb:

mid = (posb+pose)//2

if Arr[mid]==seeken:

Lis.append(mid)

flag=0

elif Arr[mid] > seeken:

Lis.append(mid)

pose = mid-1

else:

Lis.append(mid)

posb = mid+1

else:

flag=0

self.LG=Lis

def O\_B\_Search(self, Arr, num, seeken):

Lis=[]

b=int(num)//2

i =b

flag=1

while flag==1:

if b>0:

if i>=num:

i-=b//2+1

b//=2

elif i<0:

i+=b//2+1

b//=2

elif Arr[int(i)]==seeken:

Lis.append(int(i))

flag=0

elif Arr[int(i)]<seeken:

Lis.append(int(i))

i+=b//2+1

b//=2

else:

Lis.append(int(i))

i-=b//2+1

b//=2

else:

if i<num and i>=0:

Lis.append(int(i))

flag=0

self.LG=Lis

def Sharrah\_Search(self, Arr, num, seeken):

Lis=[]

flag=1

k=(math.log2(num))//1

i=(2\*\*k)-1

if seeken <= Arr[int(i)]:

b=(i+1)

while flag==1:

if b>0:

if i>=num:

i-=b//2+1

b//=2

elif i<0:

i+=b//2+1

b//=2

elif Arr[int(i)]==seeken:

Lis.append(int(i))

flag=0

elif Arr[int(i)]<seeken:

Lis.append(int(i))

i+=b//2+1

b//=2

else:

Lis.append(int(i))

i-=b//2+1

b//=2

else:

if i<num and i>=0:

Lis.append(int(i))

flag=0

else:

l=math.log2(num-i)

i=num-2\*\*l

b=2\*\*(l)

while flag==1:

if b>0:

if i>=num:

i-=b//2+1

b//=2

elif i<0:

i+=b//2+1

b//=2

elif Arr[int(i)]==seeken:

Lis.append(int(i))

flag=0

elif Arr[int(i)]<seeken:

Lis.append(int(i))

i+=b//2+1

b//=2

else:

Lis.append(int(i))

i-=b//2+1

b//=2

else:

if i<num and i>=0:

Lis.append(int(i))

flag=0

self.LG=Lis

TF.py

from tkinter import \*

#Клас, що представляє отримані дані в ході виконання програми

#Атрибути:

#text : Text

#Текстове поле для виведення згенерованого масива чи послідовності перевірки елементів

# Методи

# ------

# \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# def \_\_init\_\_(self, canvas)

# Конструктор класу

# Аргументи:

# 1) self - об'єкт класу

# 2) canvas - рамка вікна програми, у яке виводиться текстове поле

# Повертає: None

# \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

class TF:

def \_\_init\_\_(self, canvas):

self.text=Text(canvas, padx=15, pady=10, font="14")

self.text.grid(row=0, column=0)

scroll=Scrollbar(canvas, command=self.text.yview())

scroll.grid(row=0, column=1, rowspan=2, sticky='ns')

scroll.config(command=self.text.yview)

self.text.config(yscrollcommand=scroll.set)

self.text.insert(INSERT, "None")

def GetText(self):

return self.text

Tumbler.py

from tkinter import \*

#Клас, що представляє поле вибору метода пошука елемента у масиві

#Атрибути:

#meth : IntVar

#Варіант метода пошуку

# Методи

# ------

# \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# def \_\_init\_\_(self, canvas)

# Конструктор класу

# Аргументи:

# 1) self - об'єкт класу

# 2) canvas - рамка вікна програми, у яке виводяться варіанти вибору методу пошука

# Повертає: None

# \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

class Tumbler:

def \_\_init\_\_(self, canvas):

self.meth = IntVar()

header = Label(canvas, text="Choose method:", padx=15, pady=10, font="14")

header.grid(row=0, column=0, sticky=W)

poslid\_checkbutton = Radiobutton(canvas, text="Linear Search", value=1, variable=self.meth, padx=15, pady=10, font="14")

poslid\_checkbutton.grid(row=1, column=0, sticky=W)

binary\_checkbutton = Radiobutton(canvas, text="Binar Search", value=2, variable=self.meth, padx=15, pady=10, font="14")

binary\_checkbutton.grid(row=2, column=0, sticky=W)

odnor\_binary\_checkbutton = Radiobutton(canvas, text="O. B. Search", value=3, variable=self.meth, padx=15, pady=10, font="14")

odnor\_binary\_checkbutton.grid(row=3, column=0, sticky=W)

Sharra\_checkbutton = Radiobutton(canvas, text="Sharrah Search", value=4, variable=self.meth, padx=15, pady=10, font="14")

Sharra\_checkbutton.grid(row=4, column=0, sticky=W)

def GetMeth(self):

return self.meth