

Практична робота №2

Тема: Бінарний пошук.

Мета: Написати програму, що використовує алгоритм бінарного пошуку.

Теоретичні відомості

Бінарний пошук проводиться в упорядкованому масиві. Алгоритм може бути визначений в рекурсивній і нерекурсивній формі.

На кожному кроці здійснюється пошук середини відрізка за формулою

$$mid = (left + right) / 2$$

Якщо елемент, що шукають, дорівнює елементу з індексом *mid*, пошук завершується.

У разі якщо елемент, що шукають, менше елемента з індексом *mid*, на місце *mid* переміщається права межа розглянутого відрізка, в іншому випадку - ліва межа.

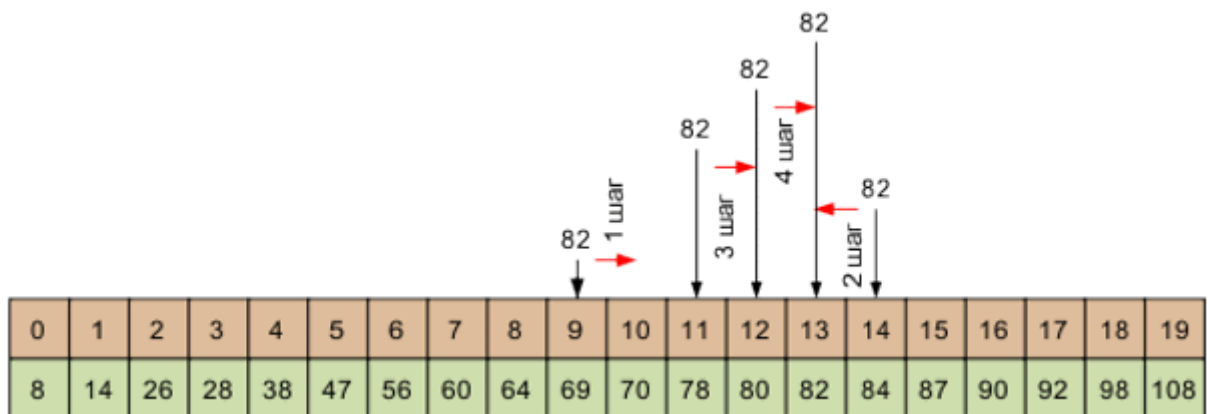


Рисунок 1 – Демонстрація роботи алгоритму бінарного пошуку

Підготовка. Перед початком пошуку встановлюємо ліву і праву межі масиву:

$$left = 0, right = 19$$

Крок 1. Шукаємо індекс середини масиву (округляємо в меншу сторону):

$$mid = (19 + 0) / 2 = 9$$

Порівнюємо значення за цим індексом з шуканим:

$$69 < 82$$

Зрушуємо ліву межу:

$$left = mid = 9$$

Крок 2. Шукаємо індекс середини масиву (округляємо в меншу сторону):

$$mid = (9 + 19) / 2 = 14$$

Порівнюємо значення за цим індексом з шуканим:

$$84 > 82$$

Зрушуємо праву межу:

$$right = mid = 14$$

Крок 3. Шукаємо індекс середини масиву (округляємо в меншу сторону):

$$mid = (9 + 14) / 2 = 11$$

Порівнюємо значення за цим індексом з шуканим:

$$78 < 82$$

Зрушуємо ліву межу:

$$left = mid = 11$$

Крок 4. Шукаємо індекс середини масиву (округляємо в меншу сторону):

$$mid = (11 + 14) / 2 = 12$$

Порівнюємо значення за цим індексом з шуканим:

$$80 < 82$$

Зрушуємо ліву межу

$$left = mid = 12$$

Крок 5. Шукаємо індекс середини масиву (округляємо в меншу сторону):

$$mid = (12 + 14) / 2 = 13$$

Порівнюємо значення за цим індексом з шуканим:

$$82 = 82$$

Рішення знайдено!

Щоб зменшити кількість кроків пошуку можна відразу зміщувати межі пошуку на елемент, наступний за серединою відрізка:

$$left = mid + 1$$

$$right = mid - 1$$

Реалізація бінарного пошуку на C++

```
#include <iostream>
#include <fstream>
#include <stdio.h>
#include <locale.h>
#include <stdlib.h>
using namespace std;

int main() {
    setlocale (LC_ALL, "Rus");
    int k[20]; // масив значень
    int key, i;
    // Ініціалізація масиву впорядкованими значеннями
    k[0] = 8; k[1] = 14; k[2] = 26; k[3] = 28; k[4] = 38; k[5] = 47;
    k[6] = 56; k[7] = 60; k[8] = 64; k[9] = 69; k[10] = 70; k[11] = 78;
    k[12] = 80; k[13] = 82; k[14] = 84; k[15] = 87; k[16] = 90; k[17] = 92;
    k[18] = 98; k[19] = 108;

    printf("Введіть key: "); // вводим значення, що будемо шукати
```

```

scanf("%d", &key);
int left = 0; // задаємо ліву та праву межу пошуку
int right = 19;
int search = -1; // індекс елементу дорівнює -1 (елемент не знайдений)
while (left <= right) // доки ліва межа не "перескочить" праву
{
    int mid = (left + right) / 2; // пошук середину відрізка
    if (key == k[mid]) { // якщо ключове поле дорівнює потрібному елементу
        search = mid; // знайшли потрібний елемент,
        break; // виходимо з циклу
    }
    if (key < k[mid]) // якщо ключове поле менше знайденої середини
        right = mid - 1; // зсуваємо праву межу, продовжуємо пошук у лівій частині
    else // інакше
        left = mid + 1; // зсуваємо ліву межу, продовжуємо пошук у правій частині
}
if (search == -1) // якщо індекс -1, то елемент не знайдений
    printf("Елемент не знайдений!\n");
else
    printf("key = %d, index = %d ", k[search], search);
return 0;
}

```

Практичне завдання

1. Заповнити масив 30 числами (від 0 до 99) у порядку зростання з використанням циклу for. Вивести у вигляді таблиці значення елементів масиву.
2. Згенерувати псевдовипадкове число A від 0 до 99 (<http://cppstudio.com/post/834/>).
3. Ввести з клавіатури число N після запиту програми
4. Перевірити за допомогою бінарного алгоритму чи є серед чисел масиву число N+A. Якщо є, то вивести індекс числа у масиві, інакше вивести повідомлення None.
5. Оформити звіт за виконанням практичної роботи.

Література

1. Бхаргава А. Грокаем алгоритмы. Иллюстрированное пособие для программистов и любопытствующих. СПб.: Питер, 2017. 288 с.
2. Бинарный поиск. URL <https://prog-cpp.ru/search-binary>.
3. Функция rand. URL: <http://cppstudio.com/post/834>

