**СТРУКТУРА**

**звіту з лабораторної роботи**

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ**

**“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ**

**імені ІГОРЯ СІКОРСЬКОГО”**

**ЗВІТ**

**з лабораторної роботи №4**

**з навчальної дисципліни “Програмування складних алгоритмів”**

**Тема: Методи пошуку в масиві**

**Варіант №1**

**Виконав студент групи ТР–15**

Руденко Владислав Ігорович

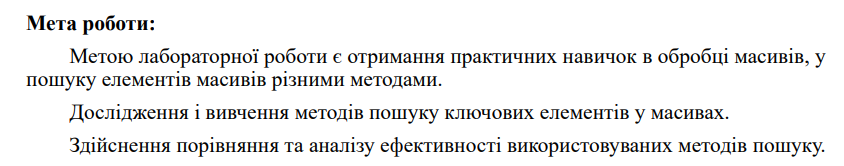
з оцінкою \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

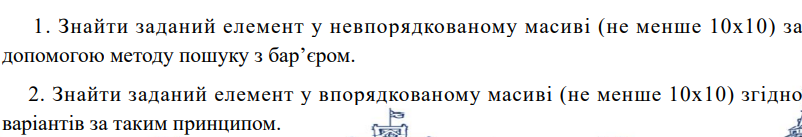
**Перевірив доцент кафедри**

Андрій ОНИСЬКО\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_\_\_ року

**Київ 2022**

**І. Завдання + Мета:**

**Завдання:**

**Теоретична частина.***Пошук* – процес знаходження у множині даних якоїсь інформації. Зазвичай у даних містяться ключі. Ключ пошуку – це поле запису, використовуючи значення якого, відбувається пошук. Їх використовують щоб розрізняти записи. Загальна мета пошуку – знайти усі збіги записів з певним значенням ключа.

Пошук є однією з найчастіше використовуваних операцій у програмуванні. Взагалі, різноманітних видів пошуку дуже багато і вони розрізняються між собою способом організації даних. Але у кожному алгоритмі пошуку є свої  переваги і недоліки.

Загальний алгоритм пошуку можна описати так:

-Обчислення елемента, що часто передбачає отримання значення елемента, ключа елемента і т.д.

-Порівняння елемента з еталоном або порівняння двох елементів.

-Перебір елементів множини, тобто проходження по елементах масиву.

Різняться види пошуку методом перебору і стратегіями пошуку. В лінійних структурах існують такі основні алгоритми.

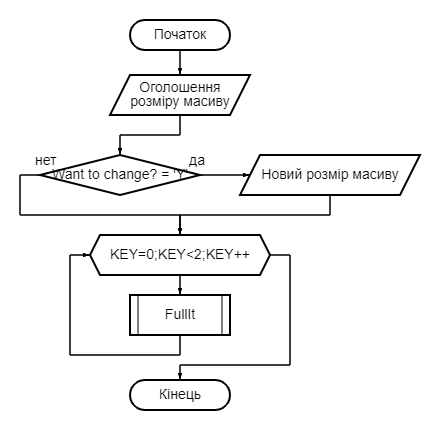
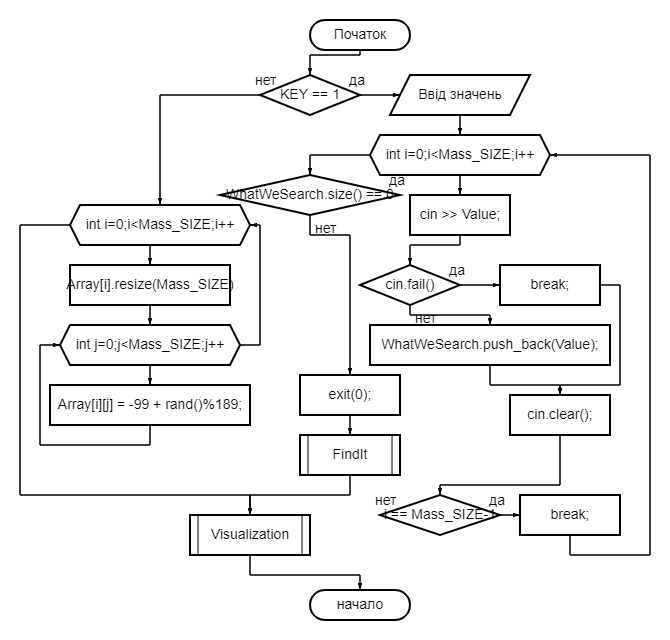
*Послідовний або лінійний пошук -* Це самий простий вид пошуку деякого елемента серед інших, що здійснюється за допомогою перевірки кожного елемента до тих пір, поки вони не будуть збігатися. Загальна ідея цього виду пошуку така: усі елементи розглядаються послідовно, один за одним. Це дає змогу не пропустити жодного елемента. Якщо збіг буде знайдено, то пошук припиняється і його результат є позитивним. Якщо не знайдено, то результат буде негативним.

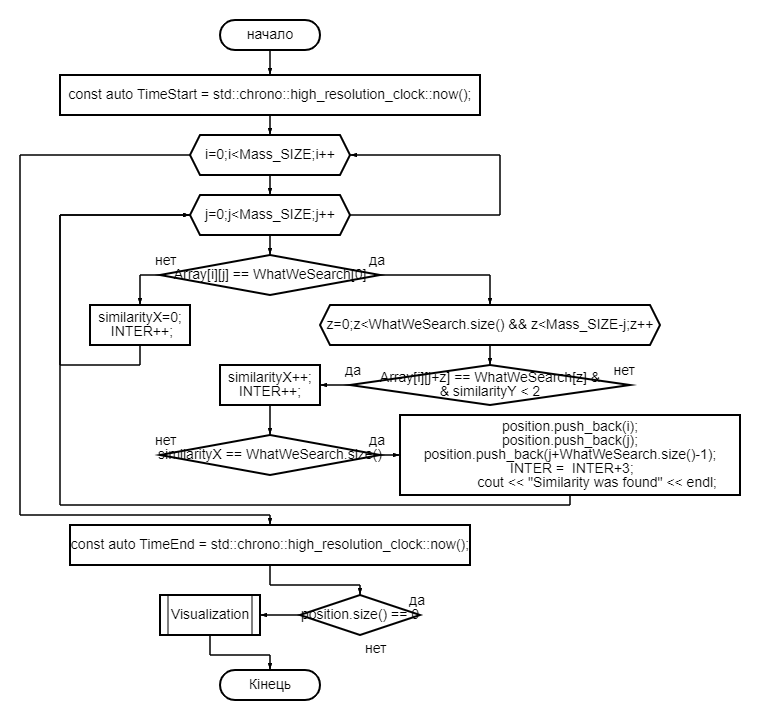
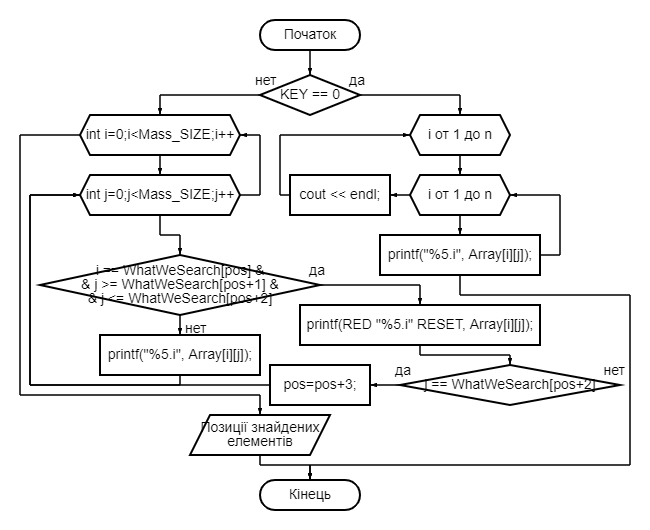
Перевагами такого пошуку є простота його реалізації, він не потребує додаткового об’єму пам’яті або додаткової роботи з функціями. Це дозволяє працювати вже під час отримання даних.

Також існує певний покращений послідовний алгоритм, який пришвидшує пошук. У множині встановлюється бар’єр, тобто елемент, який задовольняє пошуку. У циклі відпадає необхідність перевірки умови, зв’язаної з границями множини. Таким чином буде обмежена зміна індексу.

**ІІ. Результати виконання лабораторної роботи.**

**Блок Схеми:**

****

*Блок-Схема 1 (Main - Основна частина) Блок-Схема 2 (Блок FullIt)*   

*Блок-Схема 3 (Блок FindIt)*  *Блок-Схема 4 (Visualization - Вивід маисву)*

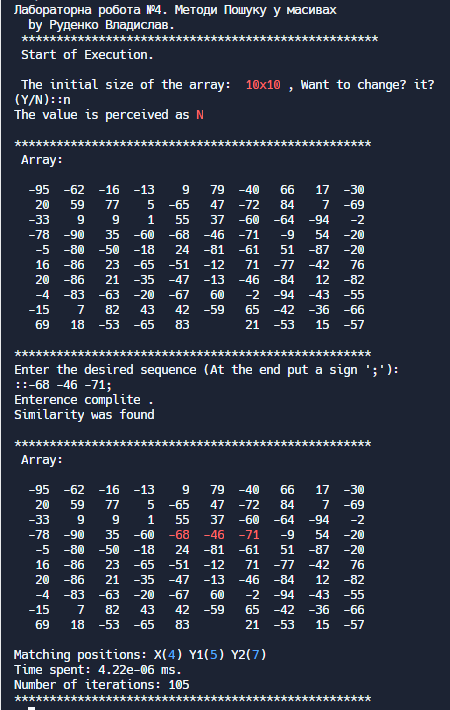
**Опис програми (написана на мові Сі++):**

Спочатку програма ініціалізує масив, який або задається оператором, або використовується значення за замовчуванням. Дані надсилаються до функції FullIt, яка заповнює масив випадковими числами, також в цій функції оператор обирає рядок, який будуть шукати в масиві. Після завершення викликаєтсья функція FindIt, в якій виконується пошук схожих значень в масиві. Під час пошуку виконується підрахунок використаного часу та кількості ітерацій, які будуть згодом обьявлені.

**Результати роботи:**

|  |  |  |
| --- | --- | --- |
| Поріврнянн часу виконання двох варіантів розв’язку | | |
| Розмір Масиву | Кількість ітерацій  Складність: | Час виконання  *ms* |
| **10x10** | **105** | **1.408e-05** |
| **50х50** | **2509** | **0.000118** |
| **100х100** | **10008** | **4.106e-05** |
| **500х500** | **250062** | **0.000704231** |

**Результати роботи у вигляді скріншотів:**



*Результати роботи - 1*

**Посилання на repl.it:**

<https://replit.com/join/rwcnwchgmr-hetik>

**Ш. Висновки.**

Під час виконання лабораторної роботи було спроектовану програму, яка виконує пошук заданого рядка в об’явленому масиві. Досліджено методи пошуку ключевих елементів у масиві. Був обраний метод пошуку послідовності елементів у масиві згідно варіанту. Виявилось що метод доволі повільний, оскільки складність його виконання це , що робить його неефективним при великих об’ємах даних. Було виявлено та виправлено деяку кількість помилок.

**Програмний код:***#include <iostream>*

*#include <chrono>*

*#include <vector>*

*using namespace std;*

*using milliseconds = chrono::duration<long long, micro>;*

*#define RED "\x1b[31m"*

*#define BLUE "\x1b[34m"*

*#define PINK "\033[1;35m"*

*#define RESET "\x1b[0m"*

*void Visualization(int Mass\_SIZE, vector<vector<int>>& Array, vector<int>& WhatWeSearch, int KEY, float UsedTime, int INTER);*

*void FindIt(int Mass\_SIZE, vector<vector<int>>& Array, vector<int>& WhatWeSearch, int KEY);*

*void FullIt(int Mass\_SIZE, vector<vector<int>>& Array, vector<int>& WhatWeSearch, int KEY);*

*int main()*

*{*

*char choose;*

*int Mass\_SIZE = 10,KEY;*

*srand(time(NULL));*

*cout << "Лабораторна робота №4. Методи Пошуку у масивах\n by Руденко Владислав."<< "\n \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \n Start of Execution.\n\n The initial size of the array: "<< RED << " 10х10 " << RESET << ", Want to change? it?\n(Y/N)::";*

*cin >> choose;*

*if (choose == 'Y' || choose == 'y')*

*{*

*cout << "Enter a new value"<< RED <<" (ROW length =COLUMN length) "<<RESET<<":\n:: ";*

*cin >> Mass\_SIZE;*

*}*

*else if (choose != 'N' || choose != 'n')*

*cout << "The value is perceived as"<<RED<<" N"<< RESET << endl;*

*vector<vector<int>> Array(Mass\_SIZE);*

*vector<int> WhatWeSearch;*

*for(KEY=0;KEY<2;KEY++)*

*FullIt(Mass\_SIZE, Array, WhatWeSearch, KEY);*

*}*

*void Visualization(int Mass\_SIZE, vector<vector<int>>& Array, vector<int>& WhatWeSearch, int KEY, float UsedTime, int INTER)*

*{*

*if(KEY==0)*

*{*

*cout << "\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n Array:\n\n";*

*for(int i=0;i<Mass\_SIZE;i++){*

*for(int j=0;j<Mass\_SIZE;j++){*

*printf("%5.i", Array[i][j]);*

*}*

*cout << endl;*

*}*

*cout << "\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n";*

*}*

*else if(KEY==2)*

*{*

*int pos=0;*

*cout << "\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n Array:\n\n";*

*for(int i=0;i<Mass\_SIZE;i++){*

*for(int j=0;j<Mass\_SIZE;j++){*

*if (i == WhatWeSearch[pos] && j >= WhatWeSearch[pos+1] && j <= WhatWeSearch[pos+2]){*

*printf(RED "%5.i" RESET, Array[i][j]);*

*if(j == WhatWeSearch[pos+2])*

*pos=pos+3;*

*}*

*else*

*printf("%5.i", Array[i][j]);*

*}*

*cout << endl;*

*}*

*for(pos=0;pos+2 < WhatWeSearch.size() ;pos=pos+3)*

*cout << "\nMatching positions: X(" << BLUE << WhatWeSearch[pos]+1 << RESET << ") Y1(" << BLUE << WhatWeSearch[pos+1]+1 << RESET << ") Y2(" << BLUE << WhatWeSearch[pos+2]+1 << RESET << ")";*

*cout << "\nTime spent: " << UsedTime << " ms." << "\nNumber of iterations: " << INTER << "\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n";*

*}*

*}*

*void FindIt(int Mass\_SIZE, vector<vector<int>>& Array, vector<int>& WhatWeSearch, int KEY)*

*{*

*int INTER = 0;*

*vector <int > position;*

*int i,j,z, similarityX = 0,similarityY = 0, pos = 0;*

*const auto TimeStart = std::chrono::high\_resolution\_clock::now();*

*for (i=0;i<Mass\_SIZE;i++)*

*{*

*for(j=0;j<Mass\_SIZE;j++)*

*{*

*if(Array[i][j] == WhatWeSearch[0])*

*{*

*for(z=0;z<WhatWeSearch.size() && z<Mass\_SIZE-j;z++)*

*if(Array[i][j+z] == WhatWeSearch[z] && similarityY < 2){*

*similarityX++;*

*INTER++;*

*}*

*if(similarityX == WhatWeSearch.size())*

*{*

*position.push\_back(i);*

*position.push\_back(j);*

*position.push\_back(j+WhatWeSearch.size()-1);*

*INTER = INTER+3;*

*cout << "Similarity was found" << endl;*

*}*

*}*

*else {*

*similarityX=0;*

*INTER++;*

*}*

*}*

*}*

*const auto TimeEnd = std::chrono::high\_resolution\_clock::now();*

*if(position.size() == 0)*

*cout << "Similarity was NOT found" << endl;*

*else*

*Visualization(Mass\_SIZE, Array, position, 2, chrono::duration<float>(TimeEnd - TimeStart).count(), INTER);*

*}*

*void FullIt(int Mass\_SIZE, vector<vector<int>>& Array, vector<int>& WhatWeSearch, int KEY)*

*{*

*if(KEY == 1){*

*cout << "Enter the desired sequence (At the end put a sign ';'):\n::";*

*int Value;*

*for(int i=0;i<Mass\_SIZE;i++)*

*{*

*cin >> Value;*

*if(cin.fail())*

*{*

*cout << "Enterence complite ." << endl;*

*break;*

*}*

*else*

*{*

*WhatWeSearch.push\_back(Value);*

*}*

*cin.clear();*

*if(i == Mass\_SIZE-1)*

*{*

*cout << "Enterence complite by value: "<< Value << endl;*

*break;*

*}*

*}*

*if(WhatWeSearch.size() == 0)*

*{*

*cout << "Sequences not entered. \nExit....\n";*

*exit(0);*

*}*

*FindIt(Mass\_SIZE, Array, WhatWeSearch, KEY);*

*}*

*else*

*{*

*for(int i=0;i<Mass\_SIZE;i++)*

*{*

*Array[i].resize(Mass\_SIZE);*

*for(int j=0;j<Mass\_SIZE;j++)*

*Array[i][j] = -99 + rand()%189;*

*}*

*}*

*Visualization(Mass\_SIZE, Array, WhatWeSearch, KEY, 0, 0);*

*}*