НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ “ЛЬВІВСЬКА ПОЛІТЕХНІКА”

КАФЕДРА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ



**ЗВІТ**

**Про виконання лабораторної роботи № 8**

**з дисципліни «Алгоритми і структури даних»**

**Лектор:**

Доцент кафедри ПЗ

Коротєєва Т.О.

**Виконав:**

студ. групи ПЗ-23

Михалевич П.-І.В.

**Прийняла:**

Доцент кафедри ПЗ

Мельник Н.Б.

«\_\_\_» \_\_\_\_\_\_ 2021 р.

∑ = \_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_

Львів – 2021

**Тема роботи**: Метод лінійного пошуку.

**Мета роботи**: Навчитися застосовувати алгоритм лінійного пошуку при розв’язанні задач та перевірити його ефективність на різних масивах даних.

**Теоретичні відомості**

Нехай *A = {a1, a2, ...}* – послідовність однотипних елементів і *b* – деякий елемент, що має властивість *P*. Необхідно знайти місце елемента *b* у послідовності *А*.Оскільки представлення послідовності у пам’яті може бути здійснено у вигляді масиву, то задачі можуть бути уточнені як задачі пошуку елемента у масиві.

Наприклад:

1. Знайти максимальний елемент масива;

2. Знайти даний елемент масива;

3. Знайти *k*-тий за величиною елемент масива.

Найбільш прості і оптимальні алгоритми засновано на послідовному перегляді массива *A* з перевіркою властивості *P* на кожному елементі. Цей метод називають лінійним переглядом (пошуком).

Лінійний, або послідовний пошук – найпростіший алгоритм пошуку, який дозволяє знаходити задане значення на деякому відрізку. Цей алгоритм не накладає обмежень на значення, яке потрібно знайти.

Пошук виконується порівнянням заданого значення та значення поточного елемента масиву до тих пір, поки вони не співпадуть.

Якщо масив має довжину *n*, то складність алгоритму буде *O (n)*.

Недолік методу:

- низька ефективність.

Перевага методу:

- не потребує додаткової пам’яті.

**Постановка завдання**

Використовуючи алгоритм лінійного пошуку складіть програму, що шукає задане значення.  
5. Дано одновимірний масив цілих чисел A[і], де і =1,2,…,n. Знайти позиції елементів масиву, які кратні 5.

**Покроковий опис алгоритму лінійного пошуку:**

L1. Дано A = {x1,…, xn}. Цикл за індексом проходження i = 1..n. X - елемент пошуку.

L2. Якщо A[i] = X, виводимо його індекс i.

L3. Кінець. Вихід.

**Код програми**

**mainwindow.cpp**

#include "mainwindow.h"

#include "ui\_mainwindow.h"

#include <ctime>

#include <vector>

std::vector<int> g\_vect;

MainWindow::MainWindow(QWidget \*parent)

: QMainWindow(parent)

, ui(*new* Ui::MainWindow)

{

ui->setupUi(*this*);

connect(ui->pushButton\_clear,*SIGNAL*(clicked()),*this*,*SLOT*(MySlot()));

connect(ui->pushButton\_start,*SIGNAL*(clicked()),*this*,*SLOT*(MySlot()));

connect(ui->pushButton\_init,*SIGNAL*(clicked()),*this*,*SLOT*(MySlot()));

}

MainWindow::~MainWindow()

{

*delete* ui;

}

void MainWindow::MySlot()

{

QPushButton\* btn = (QPushButton\*)sender();

*if*(btn->text()=="Start"){

*for*(size\_t i = 0; i < g\_vect.size(); ++i){

*if*(g\_vect[i] % 5 == 0){

ui->textEdit\_result->append("Element found: index = " + QString::number(i+1) + " ;");

}

}

}*else* *if*(btn->text()== "Clear"){

ui->textEdit->clear();

ui->textEdit\_result->clear();

}*else* *if*(btn->text() == "Random input"){

ui->textEdit->clear();

ui->textEdit\_result->clear();

g\_vect.clear();

srand(time(*nullptr*));

*for*(*auto* i = 0; i < ui->spinBox->value(); ++i){

g\_vect.push\_back(rand() % 500);

}

int iter = 0;

int iterComa = 0;

QString line;

*for* (*const* *auto*& a : g\_vect) {

line += QString::number(a);

*if*(iterComa != g\_vect.size()-1){

line += ", ";

}

*if*(iter == 14 || iter == g\_vect.size()-1){

ui->textEdit->append(line);

line = "";

}

iter++;

iterComa++;

}

}

}

**mainwindow.h**

#ifndef MAINWINDOW\_H

#define MAINWINDOW\_H

#include <QMainWindow>

QT\_BEGIN\_NAMESPACE

*namespace* **Ui** { *class* **MainWindow**; }

QT\_END\_NAMESPACE

*class* **MainWindow** : *public* QMainWindow

{

Q\_OBJECT

*public*:

**MainWindow**(QWidget \*parent = *nullptr*);

~***MainWindow***();

*private* slots:

void **MySlot**();

*private*:

Ui::MainWindow \*ui;

};

#endif *//* *MAINWINDOW\_H*

**main.cpp**

#include "mainwindow.h"

#include <QApplication>

int main(int argc, char \*argv[])

{

QApplication a(*argc*, *argv*);

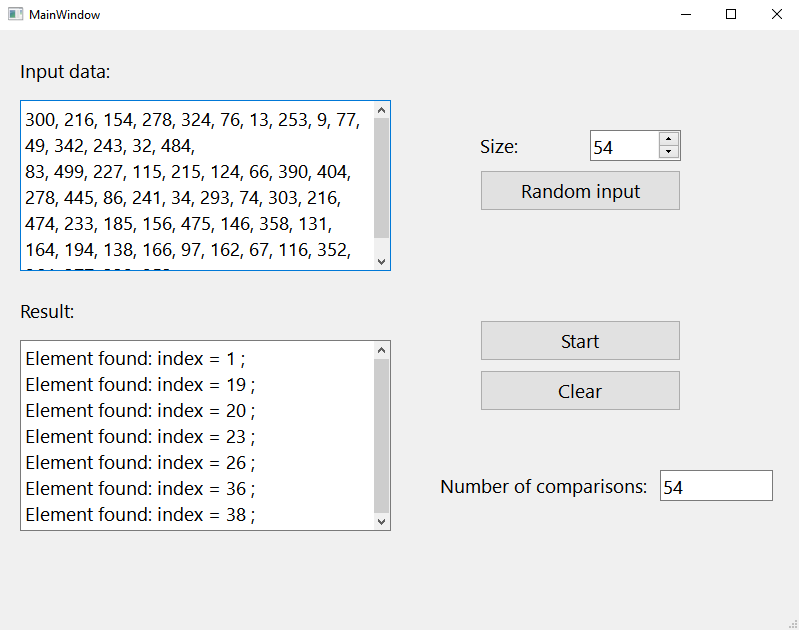
MainWindow w;

w.show();

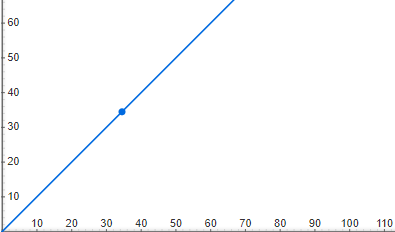
*return* a.exec();

}

**Результат програми**



Графік залежності кількості порівнянь від кількості елементів:



### Висновок

На цій лабораторній роботі я розробив програму, яка здійснює пошук елемента лінійним методом. Цей алгоритм є найпростішим і найменш ефективним водночас, адже ми порівнюємо кожен елемент масиву. Для пошуку я використовував одновимірний масив цілих чисел. Складність алгоритму дорівнює *O*(*n*), де *n* — кількість елементів для сортування.