МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний авіаційний університет

Кафедра прикладної інформатики

Лабараторна робота №9

З дисципліни “Теорія алгоритмів”

Виконав:

Студент 1 курсу ІКІТ

Групи УС-112

Лазоренко Максим Олександрович

Перевірив: Ходаков Д.В.

Київ 2016

**Лабораторная работа № `9.**

**Тема:** «Алгоритми злиття та пошуку»

Теоретичні відомості:

Алгоритми пошуку в одновимірних масивах

[Алгоритми](https://uk.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC%D0%B8) пошуку застосовуються для знаходження, наприклад, у масиві елемента з потрібними властивостями. Звичайно розрізняють постановки завдання пошуку для першого й останнього входження елемента. В усіх нижче викладених алгоритмах будемо вважати, що виробляється пошук у масиві A з N цілих чисел елемента, рівного X.

**Лінійний пошук**

Лінійний пошук здійснюється циклом ([while](https://uk.wikipedia.org/w/index.php?title=While&action=edit&redlink=1" \o "While (ще не написана))) з подвійною умовою. Перша умова контролює індекс на приналежність масиву, наприклад, (i<=N). Друга умова — це умова пошуку. У нашому випадку в циклі while ця умова продовження пошуку: (A[i]<>X), (A[i]=X). У тілі циклу звичайно пишеться тільки один оператор: зміна індексу в масиві. Після виходу із циклу необхідно перевірити, по якому з умов ми вийшли. В операторі if звичайно повторюють перша умова циклу. Можна говорити про успішний пошук із циклом while при виконанні цієї умови.

**Пошук бар'єром**

Ідея пошуку з бар'єром полягає в тому, щоб не перевіряти щораз у циклі умову, пов'язану із границями масиву. Це можна забезпечити, установивши в масив так званий бар'єр: будь-який елемент, що задовольняє умову пошуку. Тим самим буде обмежена зміна індексу. Вихід із циклу, у якому тепер залишається тільки умова пошуку, може відбутися або на знайденому елементі, або на бар'єрі. Таким чином, після виходу із циклу перевіряється, не чи бар'єр ми знайшли? Обчислювальна складність пошуку з бар'єром менше, ніж у лінійного пошуку, але також є величиною того ж порядку, що й N — кількість елементів масиву. Існує два способи установки бар'єра: додатковим елементом або замість крайнього елемента масиву.

**Двійковий (БІНАРНИЙ) пошук**

[Алгоритм](https://uk.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC) двійкового пошуку можна використати для пошуку елемента із заданою властивістю тільки в масивах, упорядкованих по цій властивості. Так при пошуку числа із заданим значенням необхідно мати масив, упорядкований по зростанню або по убуванню значень елементів. А, наприклад, при пошуку числа із заданою сумою цифр [масив](https://uk.wikipedia.org/wiki/%D0%9C%D0%B0%D1%81%D0%B8%D0%B2_(%D1%81%D1%82%D1%80%D1%83%D0%BA%D1%82%D1%83%D1%80%D0%B0_%D0%B4%D0%B0%D0%BD%D0%B8%D1%85)" \o "Масив (структура даних)) повинен бути впорядкований по зростанню або по убуванню сум цифр елементів. Ідея алгоритму полягає в тому, що масив щораз ділиться навпіл і вибирається та частина, де може перебувати потрібний елемент. Розподіл триває поки частина масиву для пошуку більше одного елемента, після чого залишається перевірити цей елемент, що залишився, на виконання умови пошуку. Існують дві модифікації цього алгоритму для пошуку першого й останнього входження. Все залежить від того, як вибирається середній елемент: округленням у меншу або більшу сторону. У першому випадку середній елемент ставиться до лівої частини масиву, а в другому — до правого. У процесі роботи алгоритму двійкового пошуку розмір фрагмента, де цей пошук повинен тривати, щораз зменшується приблизно у два рази. Це забезпечує обчислювальну складність алгоритму порядку логарифма N по підставі 2, де N — кількість елементів масиву.

**Завдання**: Знайти номер першого парного елементу масиву(N), і кількість елементів рівних «5»

Алгоритм:

.



Текст програми на с++

#include "stdafx.h"

#include <iostream>

using namespace std;

int main()

{

setlocale(LC\_ALL, "Russian");

int size; // розмір масиву

int k = 0;

cout << "Введите количество элементов в масиве" << endl;//ввід кількості елементів масиву

cout << "n = ";

cin >> size;

cout << "Введите поочерёдно элементы массива" << endl;

int \*mas = new int[size]; // виділення пам'яті падмасив

// заповнення масиву

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

cout << "mas[" << i << "] = ";

cin >> mas[i];

}

for (int j = 0; j < size; j++) {

if (mas[j] % 2 == 0) { cout <<"Номер первого парного елемена массива - "<< j << endl; break; }

}

for (int j = 0; j < size; j++) {

if (mas[j] == 5) { k++; }

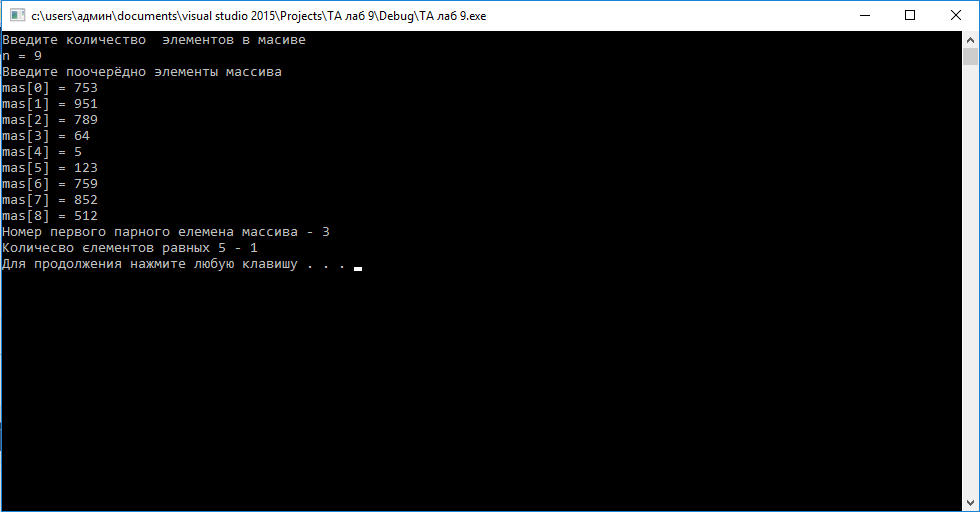
}

cout << "Количесво єлементов равніх 5 - " << k << endl;

system("pause");

return 0;

}



**Висновок**: Я навчився як можна краще використовувати всі най ефективніші методи сортування масивів, навчився розрізняти їх.

