

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»

Інститут прикладного системного аналізу

Кафедра математичних методів системного аналізу

Звіт

про виконання лабораторної роботи №3 з дисципліни «Алгоритми та структури даних»

Виконав:

студент I курсу, групи КА-07

Москаленко Максим Геннадійович

Прийняв:

Київ — 2020

|  |  |
| --- | --- |
| МЕТА ТА ВАРІАНТ | 1 |
| ЗАДАЧА 1 | 2 |
| ЗАДАЧА 2 | 5 |
| ВИСНОВОК | 8 |

МЕТА

Ознайомитись і дослідити елементарні алгоритми сортування: “бульбашкою”, вибором, вставками; алгоритм перетасовки Фішера-Йєтса та алгоритми пошуку: лінійний пошук, бінарний пошук. Набути навичок їх реалізації мовою програмування С/C++ та порівняти ці алгоритми.

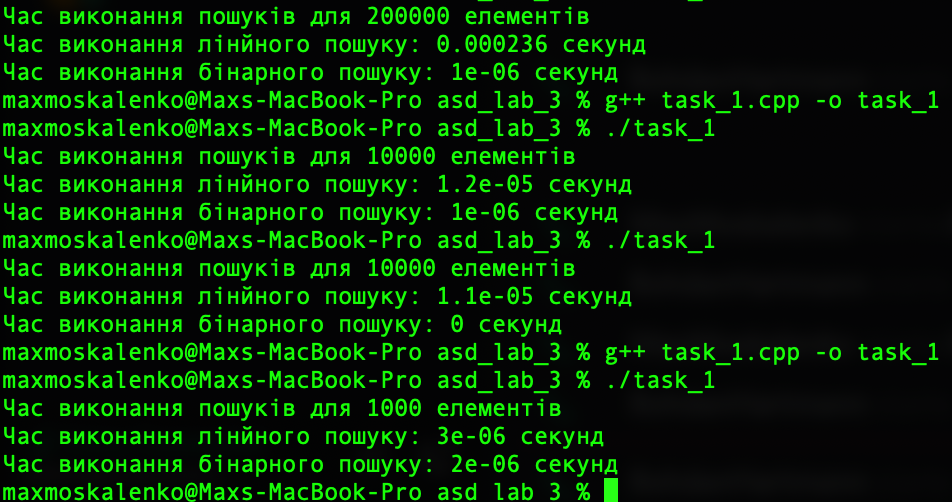
ВАРІАНТ 18

ЗАДАЧА 1

Базове сортування “бульбашкою”, пошук елементів у масиві

1.1 Створити масив на N = 1000 елементів та заповнити його випадковими цілими числами від -2000 до 2000.  
1.2 Написати функцію для сортування алгоритмом “бульбашкою”, відсортувати масив за зростанням.  
1.3 Написати функцію для пошуку елементу в масиві лінійним алгоритмом, яка повертає індекс знайденого елементу або -1, якщо такий елемент не знайдено.  
1.4 Написати функцію для пошуку елементу в масиві бінарним алгоритмом, яка повертає індекс знайденого елементу або -1, якщо такий елемент не знайдено.  
1.5 Порівняти швидкодію алгоритмів пошуку (пошук у масиві випадкових чисел від -2000 до 2000).

Результат виконання коду при 200 000 елементів, 10 000 елементів, 1 000 елементів:



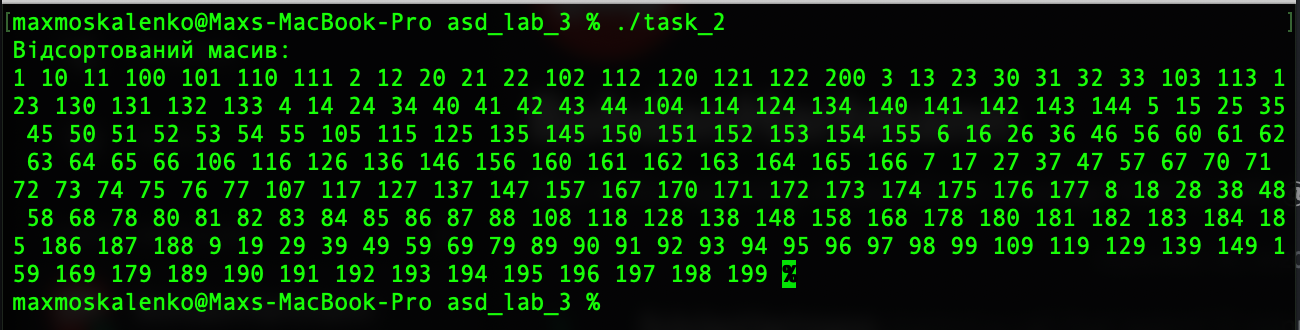
Код:

|  |
| --- |
| #include <ctime>  #include <iostream>  using namespace std;  const int ARRAY\_SIZE = 1000;  int \* bubble\_sort(int array[]);  int inline\_search(int element, int array[]);  int binary\_search(int element, int array[], int low, int high);  int main() {  srand(time(NULL));  int array[ARRAY\_SIZE];  //generates array  for (int i = 0; i < ARRAY\_SIZE; i++)  array[i] = (rand() % 4000) - 2000;  int \*sorted\_array = bubble\_sort(array);  cout << "Час виконання пошуків для " << ARRAY\_SIZE << " елементів\n";  clock\_t start = clock();  inline\_search(sorted\_array[728], sorted\_array);  clock\_t end = clock();  cout << "Час виконання лінйного пошуку: " << (float(end - start)) / CLOCKS\_PER\_SEC << " секунд\n";  start = clock();  binary\_search(sorted\_array[728], sorted\_array, 0, ARRAY\_SIZE-1);  end = clock();  cout << "Час виконання бінарного пошуку: " << (float(end - start)) / CLOCKS\_PER\_SEC << " секунд\n";  return 0;  }  int \* bubble\_sort(int array[]) {  int tmp;  for (int i = 0; i < ARRAY\_SIZE; i++)  //the biggest element moves to the last position, so we shouldn`t check it  for (int j = 0; j < ARRAY\_SIZE - i - 1; j++)  if(array[j] > array[j+1]){  tmp = array[j];  array[j] = array[j+1];  array[j+1] = tmp;  }  return array;  }  int inline\_search(int element, int array[]){  for (int i = 0; i < ARRAY\_SIZE; i++)  if (array[i] == element)  return i;  return -1;  }  int binary\_search(int element, int array[], int low, int high) {  if (high < low)  return -1;  int mid = (low + high)/2;  if (array[mid] > element)  return binary\_search(element, array, low, mid-1);  else if (array[mid] < element)  return binary\_search(element, array, mid+1, high);  else  return mid;  } |

ЗАДАЧА 2

Сортування елементарними алгоритмами по різним критеріям  
2.1 Створити масив на N = 200 елементів та заповнити його  
послідовними числами від 1 до N  
2.2 Написати функцію для перетасовки масиву алгоритмом Фішера-  
Йєтса, перетасувати масив  
2.3 Написати функцію для сортування масиву вибором за критерієм найбільшої цифри у числі у порядку зростання (напр. 67 < 19, адже 7 < 9), при рівності - спочатку менше за значенням.

Результат виконання коду:



Код:

|  |
| --- |
| #include <iostream>  using namespace std;  const int ARRAY\_SIZE = 200;  int \* fy\_shuffle(int array[]);  int biggest\_digit(int num);  bool is\_condition\_true(int old\_num, int new\_num);  int \* selection\_sort(int array[]);  int main() {  int array[ARRAY\_SIZE];  for (int i = 1; i <= ARRAY\_SIZE; i++)  array[i-1] = i;  int \* p\_array = fy\_shuffle(array);  p\_array = selection\_sort(p\_array);  cout << "Відсортований масив: " << endl;  for (int i = 0; i < ARRAY\_SIZE; i++)  cout << p\_array[i] << " ";  return 0;  }  int \* fy\_shuffle(int array[]) {  srand(time(NULL));  int j, tmp;  for (int i = ARRAY\_SIZE-1; i >= 1; i--) {  j = rand()%(i + 1);  // element swap  tmp = array[i];  array[i] = array[j];  array[j] = tmp;  }  return array;  }  int \* selection\_sort(int array[]) {  int min\_index, tmp;  for (int i = 0; i < ARRAY\_SIZE-1; i++) {  min\_index = i;  for (int j = i+1; j < ARRAY\_SIZE; j ++)  if (is\_condition\_true(array[min\_index], array[j]))  min\_index = j;  tmp = array[i];  array[i] = array[min\_index];  array[min\_index] = tmp;  }  return array;  }  bool is\_condition\_true(int old\_num, int new\_num){  if (biggest\_digit(new\_num) < biggest\_digit(old\_num))  return true;  if (biggest\_digit(new\_num) == biggest\_digit(old\_num))  return (new\_num < old\_num) ? true : false;  return false;  }  int biggest\_digit(int num) {  int biggest\_digit = 0, numeric\_digit[3] = {100, 10, 1};  for (int i = 0; i < 3; i++) {  if(num/numeric\_digit[i] > biggest\_digit)  biggest\_digit = num/numeric\_digit[i];  num = num%numeric\_digit[i];  }  return biggest\_digit;  } |

ВИСНОВКИ

У результаті лабораторної роботи було створено та реалізовано алгоритми: сортування “бульбашкою”, сортування вибором, лінійного пошуку, бінарного пошуку, Фішера-Йєтса. Також було застосовано вказівники для передачі масивів між функціями, рекурсивну функцію для бінарного пошуку. Також під час виконання лабораторної роботи було зроблено висновок, що бінарний пошук набагато швидший за лінійний, особливо при великих розмірах даних. Було використано константу (ARRAY\_SIZE) для швидкої зміни всіх потрібних параметрів програми, що значно спрощує процес тестування.