Національний технічний університет України
"Київський політехнічний інститут імені Ігоря Сікорського"
Факультет Електроніки
Кафедра мікроелектроніки

# ЗВІТ
# Про виконання лабораторної роботи №2
# з дисципліни: «Алгоритми та структури даних-2»

## Робота з масивами та списками. Методи сортування та пошуку.

Виконавець:
Студент 3-го курсу _____ А. С. Мнацаканов
(підпис)

Перевірив: _____ Д. Д. Татарчук
(підпис)

2020

**Мета роботи** – навчитись використовувати масиви та списки при розробці програм, вивчити методи сортування.

**Завдання**
Написати програму, що виконує наступні дії:

1) Зчитує дані із файлу отриманого в першій лабораторній роботі та зберігає їх у пам'яті у вигляді структури: **Список з одинарними зв'язками**.

2) Виводить дані на екран у вигляді двох стовпців x f(x), розділених трьома символами пробілу. Стовпці повинні мати заголовки X та Y відповідно.

3) Сортувати дані за зростанням або за спаданням за вибором користувача **методом виключення**, заданим у таблиці 3 відповідно до варіанту.

4) Виводити на екран відсортовану послідовність.

5) Здійснювати бінарний пошук введеної з клавіатури величини та виводити на екран результат пошуку. Величини для пошуку (ключі пошуку) повинні зберігатись у програмі у вигляді черги, розмір якої також вводиться з клавіатури. Результати пошуку повинні відображатись на екрані у порядку введення ключів пошуку.

## Виконання роботи

### Код на C++

```
#include <fstream>
#include <iostream>


class spisok_elem{
public:
  void set_x(double x1){
            x = x1;
  }
  void set_y(double y1){
            y = y1;
  }
  void set(double x1, double y1){
            x = x1;
            y = y1;
  }

  double get_x(){
            return x;
```

```
}
double get_y(){
        return y;
}

void set_next(spisok_elem* elem){
        next = elem;
}

spisok_elem* get_next(){
        return next;
}

spisok_elem* get_last(){
  if(next==NULL){
    return this;
  }else{
    return next->get_last();
  }
}

spisok_elem* search(double x){
  if(this->x == x)
  {
    return this;
  }
  if(!next){
    return NULL;
  }
  return next->search(x);
}

void add_after(spisok_elem* node){
  if(next == NULL){
    next = node;
    node -> next = NULL;
  }

  else
  {
    spisok_elem* temp = next;
    next = node;
    node -> next;
  }
}

spisok_elem* remove(spisok_elem* prev = NULL){
  if (prev == NULL)
    {
      return next;
    }
}
```

```
  int size(int k = 0){
    if(next == NULL)
    {return k+1;}
    return next->size(k+1);
  }

spisok_elem* get(int i)
  {
    if (i == 0)
    {
      return this;
    }
    if (next == NULL)
    {
      return NULL;
    }
    return next -> get(i-1);
  }


spisok_elem* find(double x, int start,  int end)
{
  int s = end - start;
  spisok_elem* ttt = get(start+s/2);
  if (x>(ttt->x+0.001) && x<(ttt->x-0.001))
  {
    return ttt;
  }
  if(x > ttt->x)
  {
    return find(x,s/2+1, end);
  }
return find(x, start, s/2);
}




spisok_elem* add_to_order(spisok_elem* node, bool(*cmp)(double,
   double)){
  if(cmp(x,node->x) && (!next || !cmp(x, next->x))){
    add_after(node);
    return this;
  }
  else if(!cmp(x,node->x))
  {
    node -> next = this;
    return node;
  }
  else
    next -> add_to_order(node, cmp);
  return this;
```

```
}



  private:
  double x;
  double y;
  spisok_elem* next;
};

bool less(double x, double y)
{
  if (x>=y)
  {
    return true;
  }
  return false;
}

bool qwe(double x, double y)
{
  if (x<=y)
  {
    return true;
  }
  return false;
}

int main(){
  spisok_elem* begin = NULL;
  std::ifstream in("DP8205.txt");
  while(!in.eof()){
    double r_x, r_y;
          in >> r_x >> r_y;

        spisok_elem* new_el = new spisok_elem;
        new_el->set_x(r_x);
        new_el->set_y(r_y);
        new_el->set_next(NULL);

        if(begin == NULL){
        begin = new_el;
        }
    else
    {
        begin->get_last()->set_next(new_el);
    }
  }

  std::cout << "X \t\t Y" << std::endl;
  for(spisok_elem* curr = begin;curr!=NULL;curr=curr->get_next()){
    std::cout << curr->get_x() << " \t " << curr->get_y() << std::
       endl;
```

```
    }

    spisok_elem* new_spisok =NULL;

    spisok_elem* temp = begin;
    begin = begin -> remove();
    new_spisok = temp;
    new_spisok->set_next(NULL);
    while(begin)
    {
      temp = begin;
      begin = begin -> remove();
      new_spisok = new_spisok-> add_to_order(temp, less);
    }

    std::cout << "X \t\t Y" << std::endl;
    for(spisok_elem* curr = new_spisok;curr!=NULL;curr=curr->
       get_next()){
      std::cout << curr->get_x() << " \t " << curr->get_y() << std::
         endl;
    }
    return 0;
}
```

також тут я написав 2 окремі функції в залежності як ми хочемо відсортувати чиста викликаємо або функцію less, або qwe.

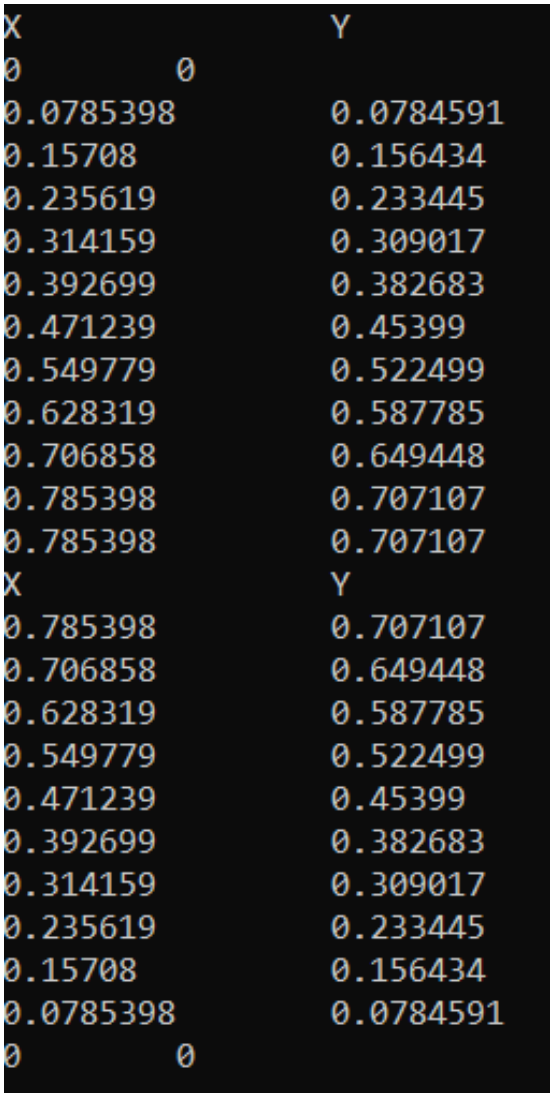


Рис. 1: Резултати.