Московский государственный технический университет им. Н.Э. Баумана

Факультет «Информатика и системы управления»
Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Основы информатики»

Отчет по лабораторной работе №5 «Сортировка одномерного числового массива»

Выполнил:
студент группы ИУ5-14Б
Микулин Михаил

Подпись и дата:

Проверил:
преподаватель каф. ИУ5
Аксенова Мария Владимировна
Подпись и дата:

Москва, 2023 г.

Постановка задачи

Отсортировать числовой массив методом выбора максимального (минимального) элемента и методом пузырькового всплытия. По окончании сортировки вывести отсортированный массив и количество сделанных сравнений и перестановок элементов.

Разработка алгоритма

struct struct_return{}; - структура, нужная для возврата указатель на отсортированный массив, количества сравнений и перестановок
int struct_return.changes - количество перестановок
int* struct_return.mass - указатель на отсортированный массив
int struct_return.comper - количество сравнений

struct_return min_max_elem_sort_from_max_to_min(int* mass, int len); - получает: указатель на массив, длину этого массива; возвращает: массив отсортированный методом выбора максимального (минимального) элемента (от максимального к минимальному), количество сравнений, перестановок

struct_return bubble_sort_from_max_to_min(int* mass, int len); - получает: указатель на массив, длину этого массива; возвращает: массив отсортированный методом пузырькового всплытия (от максимального к минимальному), количество сравнений, перестановок

char* razdel(); - выводит линию-разделитель

void output(int* mass, int changes, int comper, int len); - выводит отсортированный массив, количество перестановок и сравнений; получает: указатель на массив, количество перестановок и сравнений, длину массива

struct_return bubble_sort(int* mass, int len); - получает: указатель на массив, длину этого массива; возвращает: массив отсортированный методом пузырькового всплытия (от минимального к максимальному), количество сравнений, перестановок

struct_return min_max_elem_sort(int* mass, int len); - получает: указатель на массив, длину этого массива; возвращает: массив отсортированный методом выбора максимального (минимального) элемента (от минимального к максимальному), количество сравнений, перестановок

int* rand_int(int len) - функция возвращающая указатель на массив, заполненный случайными числами; на вход получает длину массива, который нужно вернуть

Текст программы

hppcommon.hpp

```
#ifndef hppcommon_hpp
#define hppcommon_hpp

#include <iostream>
#include <stdio.h>
#include <iterator>
#include <vector>
#include <algorithm>
#include <iterator>
#include <stdio.h>

struct struct_return{
   int changes;
   int* mass;
   int comper;
};

struct_return min_max_elem_sort_from_max_to_min(int* mass, int len);
struct_return bubble_sort_from_max_to_min(int* mass, int len);
char* razdel();
void output(int* mass, int changes, int comper, int len);
struct_return bubble_sort(int* mass, int len);
struct_return min_max_elem_sort(int* mass, int len);
int* rand_int(int len);

#endif /* hppcommon_hpp */
```

razdel.cpp

```
#include "hppcommon.hpp"
char* razdel(){
   return
"----------------------------------
";

}
```

rand.cpp

```
#include "hppcommon.hpp"
using namespace std;
int iRandom( int a, int b ){
   return (a + ( b - a +1) * rand())%100
;
}

int* rand_int(int len){

   const int n = len;
   int i;
   int* return_rand = new int[n];

   srand( (unsigned int) time( NULL )
);
   rand( );
   for( i = 0; i < n; i++ ){
      return_rand[i] = iRandom( 1, 9 );
   }

   return return_rand;
```

output.cpp

```
#include "hppcommon.hpp"
void output(int* mass, int changes, int comper, int len){
   char y_or_n = 'n';
   std::cout << "вывести отсортированный массив? y/n ";
   std::cin >> y_or_n;

   if (y_or_n == 'y'){
      std::copy ( mass, mass + len,
std::ostream_iterator<int>(std::cout, ", ") );
      std::cout << '\n';
   }


   std::cout << "сравнения: "  <<  comper << std::endl;
   std::cout << "перестановки: "  << changes << std::endl;
}
```

```
}
```

main.cpp

```
#include <iostream>
#include "hppcommon.hpp"
int main() {

    int len;
    std::cin >> len;
    int *rand_gen_mass;
    int *mass_bubble = new int[len];
    int *mass_min_max_elem = new int[len];
    rand_gen_mass = rand_int(len);
    for (int i=0; i < len; i++){
      mass_bubble[i] = rand_gen_mass[i];
      mass_min_max_elem[i] = rand_gen_mass[i];
    }
    std::cout<< razdel() << '\n';
    struct_return min_max_elem_struct_return = min_max_elem_sort(mass_min_max_elem, len);
    int* min_max_elem_mass = min_max_elem_struct_return.mass;
    int min_max_elem_changes = min_max_elem_struct_return.changes;
    int min_max_elem_comper = min_max_elem_struct_return.comper;
    std::cout<< "метод выбора максимального (минимального) элемента I раз" << '\n';
    output(min_max_elem_mass, min_max_elem_changes, min_max_elem_comper, len);
    std::cout<< razdel() << '\n';
    struct_return bubble_struct_return = bubble_sort(mass_bubble, len);
    int* bubble_mass = bubble_struct_return.mass;
    int bubble_changes = bubble_struct_return.changes;
    int bubble_comper = bubble_struct_return.comper;
    std::cout<< "метод пузырькового всплытия I раз" << '\n';
    output(bubble_mass, bubble_changes, bubble_comper, len);
    std::cout<< razdel() << '\n';

    struct_return min_max_elem_struct_return_1 = min_max_elem_sort(min_max_elem_mass, len);
    int* min_max_elem_mass_1 = min_max_elem_struct_return_1.mass;
    int min_max_elem_changes_1 = min_max_elem_struct_return_1.changes;
    int min_max_elem_comper_1 = min_max_elem_struct_return_1.comper;

    std::cout<< "метод выбора максимального (минимального) элемента II раз" << '\n';

    output(min_max_elem_mass_1, min_max_elem_changes_1, min_max_elem_comper_1, len);

    std::cout<< razdel() << '\n';

    struct_return bubble_struct_return_1 = bubble_sort(bubble_mass, len);
    int* buuble_mass_1 = bubble_struct_return_1.mass;
    int bubble_changes_1 = bubble_struct_return_1.changes;
    int bubble_comper_1 = bubble_struct_return_1.comper;

    std::cout<< "метод пузырькового всплытия II раз" << '\n';

    output(buuble_mass_1, bubble_changes_1, bubble_comper_1, len);

    std::cout<< razdel() << '\n';

    struct_return min_max_elem_struct_return_from_max_to_min = min_max_elem_sort_from_max_to_min(min_max_elem_mass, len);
    int* min_max_elem_mass_from_max_to_min = min_max_elem_struct_return_from_max_to_min.mass;
    int min_max_elem_changes_from_max_to_min = min_max_elem_struct_return_from_max_to_min.changes;
    int min_max_elem_comper_from_max_to_min = min_max_elem_struct_return_from_max_to_min.comper;

    std::cout<< "метод выбора максимального (минимального) элемента от максимального к минимальному" << '\n';

    output(min_max_elem_mass_from_max_to_min, min_max_elem_changes_from_max_to_min, min_max_elem_comper_from_max_to_min, len);

    std::cout<< razdel() << '\n';

    struct_return bubble_struct_return_from_max_to_min = bubble_sort_from_max_to_min(bubble_mass, len);
    int* buuble_mass_from_max_to_min = bubble_struct_return_from_max_to_min.mass;
    int bubble_changes_from_max_to_min = bubble_struct_return_from_max_to_min.changes;
    int bubble_comper_from_max_to_min = bubble_struct_return_from_max_to_min.comper;

    std::cout<< "метод пузырькового всплытия от максимального к минимальному" << '\n';

    output(buuble_mass_from_max_to_min, bubble_changes_from_max_to_min, bubble_comper_from_max_to_min, len);

    std::cout<< razdel() << '\n';

    delete[]mass_bubble;
```

```
        delete[]mass_min_max_elem;
        delete[]rand_gen_mass;

        return 0;
    }
```

bubble.cpp

```
#include "hppcommon.hpp"
struct_return bubble_sort(int* mass, int len){
  struct_return bubble_struct_return;
  int k = 0, n = 0;
  for (int i = len; i > 0; i--){
    for (int j = 0; j < len - 1; j++){
      int old = mass[j];

      if (mass[j] > mass[j+1]){
        mass[j] =mass[j+1];
        mass[j+1] = old;
        k++;
      }
      ++n;

    }
  }
  bubble_struct_return.mass = mass;
  bubble_struct_return.changes = k;
  bubble_struct_return.comper = n;

  return bubble_struct_return;
}
```

bubble-max->min.cpp

```
#include "hppcommon.hpp"
struct_return bubble_sort_from_max_to_min(int* mass, int len){
  struct_return bubble_struct_return;
  int k = 0, n = 0;
  for (int i = len; i > 0; i--){
    for (int j = 0; j < len - 1; j++){
      int old = mass[j+1];

      if (mass[j] < mass[j+1]){
        mass[j+1] =mass[j];
        mass[j] = old;
        k++;
      }
      n++;

    }
  }
  bubble_struct_return.mass = mass;
  bubble_struct_return.changes = k;
  bubble_struct_return.comper = n;

  return bubble_struct_return;
}
```

min-max-elem.cpp

```
#include "hppcommon.hpp"
struct_return min_max_elem_sort(int* mass, int len){
  struct_return min_max_elem_struct_return;
  int min_j = 0, old = 0, k = 0, n = 0;
  for (int i = 0; i < len-1; i ++){
    old = mass[i];
    int  min = 100;
    for (int j = i; j < len; j++){
      if (mass[j] < min){
        min = mass[j];
        min_j = j;
      }
      n ++;
    }
    mass[i] = mass[min_j];
    mass[min_j] = old;
    if (min_j != i){
      k++;
    }
  }
  min_max_elem_struct_return.changes = k;
  min_max_elem_struct_return.comper = n;
  min_max_elem_struct_return.mass = mass;
  return min_max_elem_struct_return;
```

min-max-elem-max->min.cpp

```
#include "hppcommon.hpp"
struct_return min_max_elem_sort_from_max_to_min(int* mass, int len){
  struct_return min_max_elem_struct_return;
  int max_j = 0, old = 0, k = 0, n = 0;
  for (int i = 0; i < len-1; i ++){
    old = mass[i];
    int  max = -100;
    for (int j = i; j < len; j++){
      if (mass[j] > max){
        max = mass[j];
        max_j = j;
      }
      n ++;
    }
    mass[i] = mass[max_j];
    mass[max_j] = old;
    if (max_j != i){
      k++;
    }
  }
  min_max_elem_struct_return.changes = k;
  min_max_elem_struct_return.comper = n;
  min_max_elem_struct_return.mass = mass;
```

```
}
```

```
return min_max_elem_struct_return;}
```

Тестирование