**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

Лабораторная работа № 7 по дисциплине «Информатика»

Сортировка и поиск

Группа: АВТ-615

Студент: Кузенков В.

Преподаватель: Гейнц О.М.

НОВОСИБИРСК

2016

# Задание

6.     Сортировка c предварительным выбором. Из массива путем однократного просмотра выбирается последовательность элементов, находящихся в порядке возрастания и переносится очередную строку двумерного массива с заменой во входном       на **-1**.  Этот процесс повторяется, пока в массиве не останется возрастающих последовательностей. После этого в массиве остается последовательность в порядке убывания, которая также заносится в строку массива «задом наперед». Затем для строк производится однократное слияние. Если строки в двумерном массиве окончатся раньше, после слияния оставшиеся элементы добавляются в выходной массив при помощи погружения.

# Алгоритм решения

1. Для тестирования используем два вида массивов: со случайными числами и заданные вводом. В теле main организован ввод и вывод, а также вызов функции сортировки.
2. Перед сортировкой, мы переносим все возрастающие последовательности в строки двумерного массива, пока они существуют, а во входном заменяем на -1. Для этого:
   1. Выделяем память, под двумерный массив, в первой строке будем хранить количество строк, а также индекс последнего наименьшего элемента. Создаем массив индексов элементов возрастающей последовательности. Создаем переменную temp, равную первому элементу сортируемого массива. А также сразу создадим переменную счетчик i.
   2. Создаем цикл, проверяющий, есть ли возрастающие последовательности в массиве, для этой цели заводим соответствующую функцию, в которой:
      1. Ищем первый элемент не равный -1, с помощью функции поиска по всем элементам. Приравниваем его переменной noMOne.
      2. Пробегаемся циклом по массиву, и если находим элемент больше, чем noMOne
      3. Если нашли, то значит есть последовательности и возвращаем 1, если нет, то 0.
   3. Пока существуют возрастающие последовательности, выполняется:
      1. Цикл, в котором ищется первый элемент последовательности, хранится в переменной temp, размер в growSz, в цикл индексов добавляется индекс первого элемента возрастающей последовательности.
      2. Если growSz больше 1, то переносим последовательность в строку двумерного массива. В 0 элементе храним размер строки.
   4. Теперь, нам осталось перенести все, что осталось в последнюю строку массива задом наперед.
   5. В 0 строке, в 0 элементе двумерного массива храним кол-во строк. И в этой же строке, проставляем единички, означающие минимальный элемент последовательностей в строках, в дальнейшем он будет использоваться для навигации при слиянии.
3. Затем мы сливаем в одномерный массив все строки двумерного, слиянием. Сортировка погружением не требуется, так как никаких элементов не остается. Слияние выполняется следующим образом:
   1. По списку индексов минимальных элементов в строках ищется наименьший элемент среди них.
   2. Увеличиваем индекс для строки, с которой взяли число.

# Ограничения

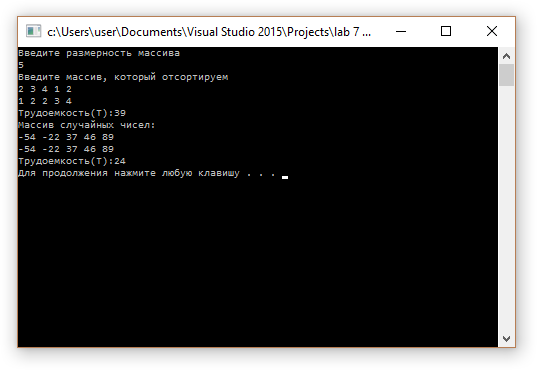
На вводе не стоит вводить минус единицу(-1), тк она используется как флаг и не будет отсортирована.

Количество вводимых элементов ограничено размерностью типа **int**.

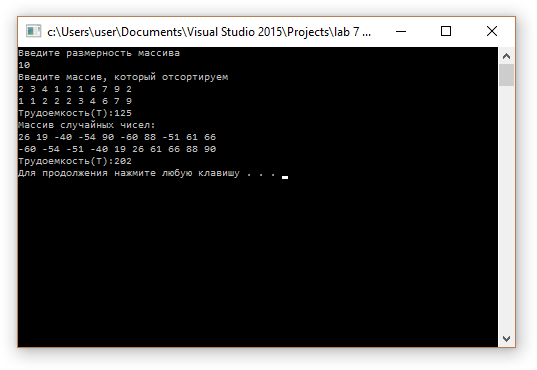
Значение каждого элемента ограничено размерностью типа **int**.

# Тестирование

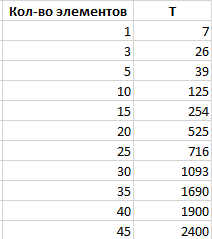
5 элементов:

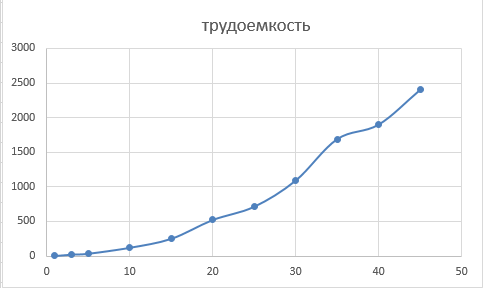


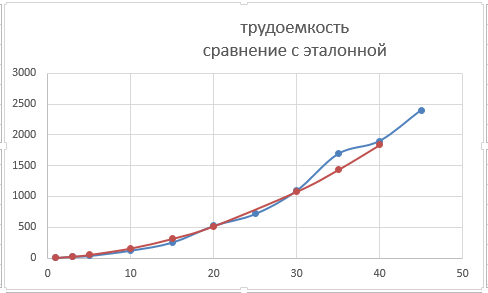
10 элементов:



# Результаты тестирования







Трудоемкость зависит от количества возрастающих последовательностей, а также требуемых проходов проверки, для повторного нахождения возрастающих последовательностей в оставшихся. То есть, очень трудоемкая и чувствительная к данным.

# Приложение

#include <iostream>

#include <fstream>

#include <algorithm>

#include <ctime>

using namespace std;

const int N = 50;//кол-во элементов в массиве со случайными

int T = 0; //сложность сортировки

int FnoMOne(int arr[], int size) //первый не равный -1

{

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

{

if (arr[i] != -1)

{

return arr[i];

T++; //sort

}

}

return -1;

}

bool haveGrow(int arr[], int size)

{

int noMOne = FnoMOne(arr, size); //элемент не равный -1

if (noMOne == -1)

return 0;

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

{

T++; //sort

if (arr[i] == -1)

{

continue;

}

if (noMOne < arr[i])

return 1;

noMOne = arr[i];

}

return 0;

}

int \*\*create2dArr(int arr[], int size)

{

int \*\*input = new int\*[size / 2]; //выделение памяти под двумерный массив

int masStr = 1; //кол-во строк двумерного массива

input[0] = new int[1];//в первой строке хранится кол-во строк

int growSz = 0; //размерность возрастающей п-ти

int \*indexM = new int[size]; //массив индексов элементов возрастающей последовательности

int temp = arr[0];

int i = 0;

/\*пока есть возрастающие последовательности\*/

while (haveGrow(arr, size))

{

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

{

T++;

if (arr[i] == -1)

{

continue;

}

if (arr[i] >= temp) {

growSz++;

temp = arr[i];

indexM[k++] = i;

}

else

{

break;

}

}

if (growSz > 1)

{

/\*переносим последовательность в строку двумерного массива\*/

input[masStr] = new int[growSz + 1];

input[masStr][0] = growSz;

for (int j = 0; j<growSz; ++j)

{

input[masStr][j + 1] = arr[indexM[j]];

arr[indexM[j]] = -1;

}

masStr++;

temp = FnoMOne(arr, size);

i = 0;

}

else

{

temp = arr[i];

}

growSz = 0;

}

if (FnoMOne(arr, size) != -1)

{

input[masStr] = new int[size]; //выделяем максимум памяти под оставшеесе

int size\_t = size - 1;

growSz = 0;

for (int k = 1; size\_t >= 0; --size\_t)

{

T++;//sort

if (arr[size\_t] != -1)

{

input[masStr][k++] = arr[size\_t];

growSz++;

arr[size\_t] = -1;

}

}

input[masStr][0] = growSz;

masStr++;

}

input[0][0] = masStr;

for (int j = 1; j < masStr; ++j)

{

input[0][j] = 1;

}//для индексов строк

return input;

}

int minIndex(int \*\*arr)

{

int minElIndex;

/\*ищем минимальный элемент\*/

for (int j = 1; j < arr[0][0]; ++j)

{

T++;

if (arr[0][j] > arr[j][0]) continue; //если вся строка уже вошла

else

{

minElIndex = j;

break;

}

}

for (int j = 1; j < arr[0][0]; ++j)

{

T++;

if (arr[0][j]>arr[j][0]) continue; //если вся строка уже вошла

if (arr[minElIndex][arr[0][minElIndex]] > arr[j][arr[0][j]])

{

minElIndex = j;

}

}

return minElIndex;

}

int sort(int arr[], int size)

{

int \*\*input = create2dArr(arr, size);

/\*сортировочка слиянием\*/

/\*остатки погружением\*/

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

{

T++;

arr[i] = input[minIndex(input)][input[0][minIndex(input)]];

input[0][minIndex(input)]++;

}

delete[] \* input;

return \*arr;

}

int main()

{

setlocale(LC\_ALL, "Rus");

srand(time(NULL));

int n = 0;

cout << "Введите размерность массива\n";

cin >> n;

int \*arr = new int[n];

cout << "Введите массив, который отсортируем\n";

for (int i = 0; i < n; ++i)

{

cin >> arr[i];

}

sort(arr, n);

for (int i = 0; i < n; ++i)

{

cout << arr[i] << ' ';

}

cout << endl;

delete[] arr;

cout << "Трудоемкость(T):" << T << endl;

cout << "Массив случайных чисел:" << endl;

T = 0;

int l[N];

for (int i = 0; i < N; i++)

{

l[i] = rand() % 100 \* pow(-1, rand() % 2);

}

for (int i = 0; i < N; ++i)

{

cout << l[i] << ' ';

}

cout << endl;

sort(l, N);

for (int i = 0; i < N; ++i)

{

cout << l[i] << ' ';

}

cout << endl;

cout << "Трудоемкость(T):" << T << endl;

system("pause");

}