ГУАП

КАФЕДРА № 43

ОТЧЕТ   
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Старший преподаватель |  |  |  | Рогачев С. А. |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

|  |
| --- |
| ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №5 |
| «АЛГОРИТМЫ СОРТИРОВКИ» |
| по курсу: Структуры и алгоритмы обработки данных |
|  |
|  |

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ ГР. № | М011 |  |  |  | Борисов С. И. |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург 2021

**Цель работы**

Целью работы является изучение алгоритмов внутренней сортировки и получение практических навыков их использования, и анализа их сложности. **Задание на лабораторную**

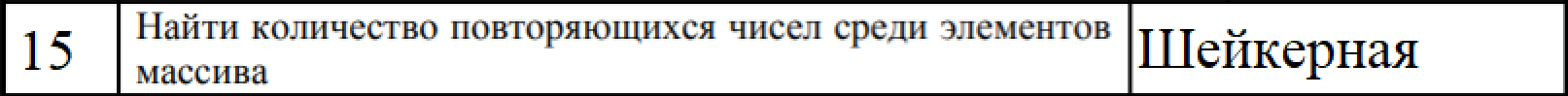
Использовать неупорядоченный массив A, содержащий n целочисленных элементов. Величина n определяется по согласованию с преподавателем. Дополнительно в программе должны быть реализованы следующие функции:

1) Поиск элемента либо по его порядковой позиции, либо по его содержимому;

2) Добавление/удаление элемента с последующей пересортировкой последовательности;

3) В программе должен быть реализован подсчет количества сравнений и перестановок, при осуществлении сортировки.

Вариант 15



**Листинг программы, реализующей алгоритм**

// Борисов С.И. ЛБ-5 Вариант 15.

#include<iostream>

#include<ctime>

using namespace std;

void add\_to\_array(int\* A, int value, int n)

{

int\* arr = new int[n + 1];

arr = A;

arr[n] = value;

A = arr;

}

//ф-ция "шейкер"-сортировки

void ShakerSort(int\* A, int& n)

{

int numOfCompare = 0; //кол-во сравнений

int numOfReshuffle = 0; // кол-во перестановок

bool isSorted = false;

while (!isSorted)

{

isSorted = true;

int leftMark = 1;

int rightMark = n - 1;

while (leftMark <= rightMark)

{

for (int i = rightMark; i >=leftMark; i--)

{

numOfCompare++;

if (A[i - 1] > A[i])

{

swap(A[i - 1], A[i]);

//

numOfReshuffle++;

}

}

leftMark++;

for (int i = leftMark; i <= rightMark; i++)

{

numOfCompare++;

if (A[i - 1] > A[i])

{

swap(A[i - 1], A[i]);

//

numOfReshuffle++;

}

}

rightMark--;

}

}

cout << "Кол-во операций сравнения: " << numOfCompare << "\n";

cout << "Кол-во операций перестановки: " << numOfReshuffle << "\n";

}

void delete\_by\_index(int\*& A, int& n, int index)

{

int\* arr = new int[n - 1];

if (index == (n - 1))

{

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

{

arr[i] = A[i];

}

delete[] A;

A = arr;

}

else

{

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

{

A[i] = A[i + 1];

}

}

}

void print\_array(int\* A, int& n)

{

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

{

cout << " " << A[i] << "[" << i << "]";

}

cout << endl;

}

void count\_repeated(int\* A, int& n)

{

ShakerSort(A, n);

print\_array(A, n);

int k = 0;

int ret = 0;

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

{

if (A[i] == A[i + 1])

ret++;

else

{

if (ret >= 1)

k++;

ret = 0;

}

}

cout << "Кол-во повторяющихся чисел: " << k << "\n";

}

int main()

{

setlocale(LC\_ALL, "russian");

srand(time(0));

int n;

cout << "Введите размер массива: ";

cin >> n;

int\* A = new int[n];

cout << "Исходный массив:\n";

/\*

for (int k = 0; k < n; k++)///////////////////////

{

A[k] = n - k; // запись значений по убыванию

cout << A[k] << "[" << k << "]" "\t";

}

cout << endl;

//\*/

///\*

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

{

A[i] = rand() % 10;

cout << " " << A[i] << "[" << i << "]";

}

cout << endl;

//\*/

int Key\_board = 0;

char x[100];

while (Key\_board != 7)

{

cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;

cout << "|Нажмите 1 для добавления элемента. |\n";

cout << "|Нажмите 2 для удаления элемента. |\n";

cout << "|Нажмите 3 чтобы вывести количество повторяющихся чисел|\n";

cout << "|Нажмите 4 чтобы найти элемент по индексу |\n";

cout << "|Нажмите 5 чтобы найти элемент по значению |\n";

cout << "|Нажмите 6 для сортировки |\n";

cout << "|Нажмите 7 для выхода из программы |\n";

cout << "|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|" << endl;

do

{

cout << "Выберите пункт: ";

cin >> x;

} while (!isdigit(x[0]));

Key\_board = atof(x);

switch (Key\_board)

{

case 1:

{

int value;

cout << "Введите значение элемента: ";

cin >> value;

add\_to\_array(A, value, n);

n++;

ShakerSort(A, n);

cout << "Отсортированный массив: ";

print\_array(A, n);

break;

}

case 2:

{

int index;

cout << "Введите индекс элемента: ";

cin >> index;

delete\_by\_index(A, n, index);

n--;

ShakerSort(A, n);

cout << "Отсортированный массив: ";

print\_array(A, n);

break;

}

case 3:

{

count\_repeated(A, n);

break;

}

case 4:

{

int index;

cout << "Введите индекс элемента: ";

cin >> index;

cout << "Найденный элемент: " << A[index] << ": [" << index << "]" << endl;

break;

}

case 5:

{

int value;

cout << "Введите значение элемента: ";

cin >> value;

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

{

if (A[i] == value)

{

cout << "Найденный элемент: " << A[i] << ": [" << i << "]" << endl;

}

}

break;

}

case 6:

{

ShakerSort(A, n);

cout << "Отсортированный массив: ";

print\_array(A, n);

break;

}

case 7: { return 0; break; }

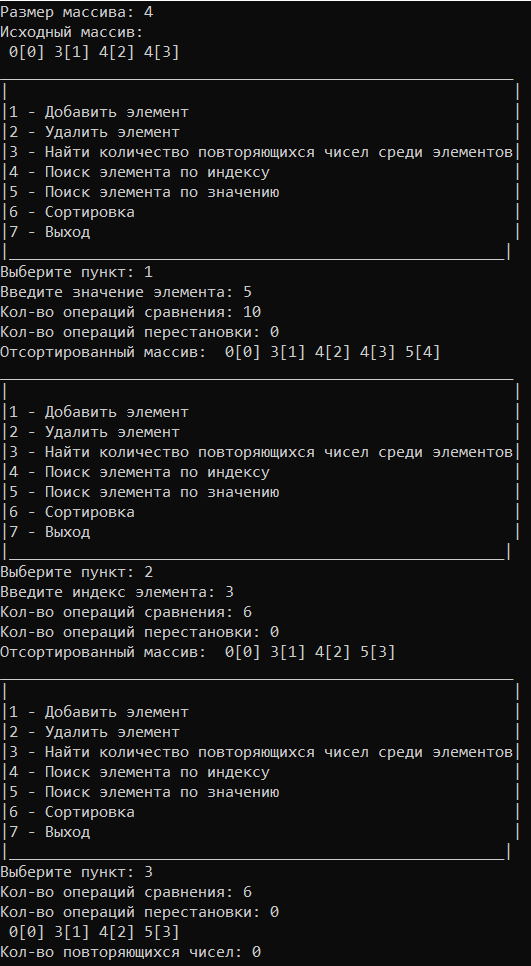
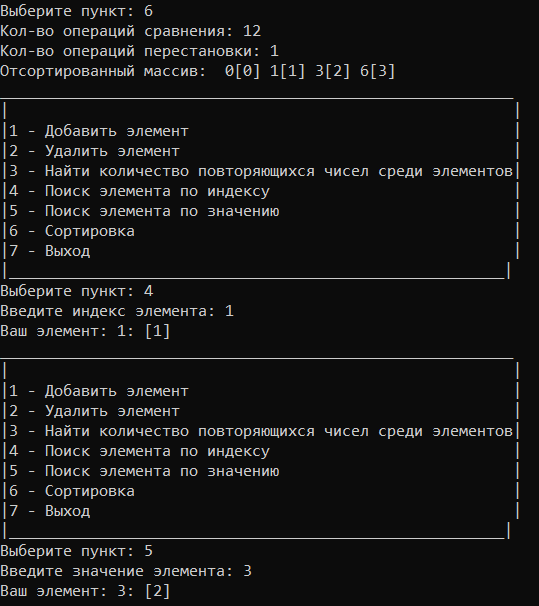
default: {cout << "Ошибка ввода.\n"; break; }

}

}

}

**Параметры алгоритма:**

**Расчет сложности алгоритма:**

Пространственная сложность алгоритма

v = n\*Cint + 2\*Cint + Cbool

Теоретическая пространственная сложность алгоритма:

𝑉=𝑂(𝑣)=𝑂(max(𝑂(𝑛∗ Cint),𝑂(2∗Cint),O(Cbool))=𝑂(max(𝑂(𝑛),𝑂(1),O(1))=𝑂(𝑛)

Теоретическая временная сложность:

𝑇=𝑂(max(𝑂(𝑛\*(𝐾35+K36+K37)),O(𝑛\*(𝐾47+K48+K49)))=O(max(O(n),O(n))=𝑂(n)

**Выводы**

На основе этих расчетов можно сделать вывод, что был разработан алгоритм, характеристики которого соответствуют поставленному заданию.