МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет: Информационных технологий

Кафедра: Программной инженерии

Выполнила: студентка 1 курса 5 группы

специальности ПОИТ Дзивнель М.А.

Проверил: преподаватель

Белодед Николай Иванович

**РЕФЕРАТ**

По дисциплине “Основы алгоритмизации и программирования”

На тему “Сортировки”

Минск

2022

СОДЕРЖАНИЕ

ВВЕДЕНИЕ ……………………………………………………………………….2

Сортировка пузырьком……….……………………………………………2

Шейкерная сортировка……………………………………………………..3

Сортировка выбором……………………………………………………….3

Сортировка вставками……………………………………………………...4

Быстрая сортировка………………………………………………………...5

Тестирование сортировок………………………………………………….6

ВЫВОД …………………………………………………………………………..11

**ВВЕДЕНИЕ**

При работе со структурами данных очень часто возникает потребность с их сортировкой. Например, сортировка нужна для облегчения поиска данных, для более удобного чтения их, для работы с ними. Существует множество сортировок, и основные из них будут приведены ниже.

**СОРТИРОВКА ПУЗЫРЬКОМ**

Сортировка пузырьком - это самый простой алгоритм сортировки, который работает путем многократной замены соседних элементов, если они находятся в неправильном порядке.

**Реализация на C/C++**

int\* bubble\_sort(int\* arr,int n\_arr) {

int temp = 0; //Временная переменная

for (int i = 0; i < n\_arr - 1; i++) {

for (int j = i + 1; j < n\_arr; j++) {

if (arr[i] > arr[j]) {

temp = arr[i];

arr[i] = arr[j];

arr[j] = temp;

}

}

}

return arr;

}

**ШЕЙКЕРНАЯ СОРТИРОВКА (СОРТИРОВКА ПЕРЕМЕШИВАНИЯ)**

Разновидность пузырька. На первом проходе как обычно — задвигаем максимум в конец. Потом резко разворачиваемся и толкаем минимум в начало. Отсортированные крайние области массива увеличиваются в размерах после каждой итерации.

**Реализация на C/C++**

int\* shaker\_sort(int\* arr,int n\_arr) {

int left, right, i;

left = 0;

right = n\_arr - 1;

while (left <= right) {

for (i = right; i >= left; i--) {

if (arr[i - 1] > arr[i]) {

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

}

}

left++;

for (i = left; i <= right; i++) {

if (arr[i - 1] > arr[i]) {

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

}

}

right--;

}

return arr;

}

**СОРТИРОВКА ВЫБОРОМ**

В чём идея сортировок выбором?

1. В неотсортированном подмассиве ищется локальный максимум (минимум).
2. Найденный максимум (минимум) меняется местами с последним (первым) элементом в подмассиве.
3. Если в массиве остались неотсортированные подмассивы — смотри пункт 1.

**Реализация на C/C++**

int\* selection\_sort(int\* arr,int n\_arr) {

int temp; //Временная переменная

int max;

for (int i = n\_arr; i > 0; i--) {

max = i;

for (int j = i - 1; j > 0; j--)

max = (arr[j] > arr[max]) ? j : max;

if (i != max) {

temp = arr[i];

arr[i] = arr[max];

arr[max] = temp;

}

}

return arr;

}

**СОРТИРОВКА ВСТАВКАМИ**

Сортировка вставками — алгоритм сортировки, в котором элементы входной последовательности просматриваются по одному, и каждый новый поступивший элемент размещается в подходящее место среди ранее упорядоченных элементов.

**Реализация на C/C++**

int\* insertion\_sort(int\* arr,int n\_arr) {

int temp,i,j;

for (i = 1; i < n\_arr; i++) {

temp = arr[i];

for (j = i - 1; j >= 0 && arr[j] > temp; j--)

arr[j + 1] = arr[j];

arr[j + 1] = temp;

}

return arr;

}

**БЫСТРАЯ СОРТИРОВКА**

Быстрая сортировка - в целом это один из самых быстрых алгоритмов сортировки массивов. Является примером принципа «разделяй и властвуй». Идея алгоритма заключается в том, что выбирается опорный элемент, относительно которого будет происходит сортировка. Равные и бОльшие элементы помещаются справа, меньшие – слева. Затем к полученным подмассивам рекурсивно применяются два первых пункта.

**Реализация на C/C++**

int\* quick\_sort(int b, int e, int\* arr) //b-первый элемент массива,а е-последний

{

int l = b, r = e;

int piv = arr[(l + r) / 2]; // Опорным элементом для примера возьмём средний

while (l <= r)

{

while (arr[l] < piv)

l++;

while (arr[r] > piv)

r--;

if (l <= r)

swap(arr[l++], arr[r--]);

}

if (b < r)

quick\_sort(b, r, arr);

if (e > l)

quick\_sort(l, e, arr);

return arr;

}

//вызов осуществляется следующим образом quick\_sort(0, n-1, arr);

**КОД ДЛЯ ТЕСТИРОВАНИЯ**

Для того, чтобы определить самую эффективную сортировку, необходимо протестировать каждую из сортировок в лучшей и случайной ситуациях. Было создано два массива на 200 000 элементов. Первый заполнен случайными числами, а второй упорядоченными (первый элемент со вторым поменяны местами) для создания лучшей ситуации.

#include <iostream>

#define ARR\_N 1000000

using namespace std;

int\* bubble\_sort(int\* arr, int n\_arr);

int\* shaker\_sort(int\* arr, int n\_arr);

int\* selection\_sort(int\* arr, int n\_arr);

int\* insertion\_sort(int\* arr, int n\_arr);

int\* quick\_sort(int b, int e, int\* arr);

int main() {

setlocale(LC\_ALL, "Russian");

system("cls");

srand(unsigned(time(0))\*200\*100);

//Создание массивов

static int arrA[ARR\_N];

static int arrB[ARR\_N];

//Заполнение массива A случайными числами ( случайная ситуация )

for (int i = 0; i < ARR\_N; i++) {

arrA[i] = rand();

}

//Заполнение массива B упорядоченными числами ( лучшая ситуация )

for (int i = 0; i < ARR\_N; i++) {

arrB[i] = i;

}

// создание лучшей ситуации в массиве arrB

int temp = arrB[1];

arrB[1] = arrB[2];

arrB[2] = temp;

//Время

unsigned int start\_time, end\_time;

//----------------------------------------случайная ситуация-----------------------------------

//Пузырьковская сортировка

start\_time = clock();

bubble\_sort(arrA, ARR\_N);

end\_time = clock();

unsigned int bubble\_sort\_time = end\_time - start\_time;

// Обнуление времени

start\_time = 0;

end\_time = 0;

// Шейкерная сортировка

start\_time = clock();

shaker\_sort(arrA, ARR\_N);

end\_time = clock();

unsigned int shaker\_sort\_time = end\_time - start\_time;

// Обнуление времени

start\_time = 0;

end\_time = 0;

// Сортировка выбором

start\_time = clock();

selection\_sort(arrA, ARR\_N);

end\_time = clock();

unsigned int selection\_sort\_time = end\_time - start\_time;

// Обнуление времени

start\_time = 0;

end\_time = 0;

// Сортировка вставками

start\_time = clock();

insertion\_sort(arrA, ARR\_N);

end\_time = clock();

unsigned int insertion\_sort\_time = end\_time - start\_time;

// Обнуление времени

start\_time = 0;

end\_time = 0;

// Быстрая сортировка

start\_time = clock();

quick\_sort(0, ARR\_N-1, arrA);

end\_time = clock();

unsigned int quick\_sort\_time = end\_time - start\_time;

// Обнуление времени

start\_time = 0;

end\_time = 0;

//Вывод времени

cout << "--------------случайная ситуация-----------------\n";

cout << "\nВремя сортировки пузырьковой:" << bubble\_sort\_time << "мс.\n";

cout << "\nВремя сортировки Шейкером:" << shaker\_sort\_time << "мс.\n";

cout << "\nВремя сортировки выбором:" << selection\_sort\_time << "мс.\n";

cout << "\nВремя сортировки вставкой:" << insertion\_sort\_time << "мс.\n";

cout << "\nВремя сортировки быстрой:" << quick\_sort\_time << "мс.\n\n";

//----------------------------------------лучшая ситуация-----------------------------------

//Пузырьковская сортировка

start\_time = clock();

bubble\_sort(arrB, ARR\_N);

end\_time = clock();

bubble\_sort\_time = end\_time - start\_time;

// Обнуление времени

start\_time = 0;

end\_time = 0;

// Шейкерная сортировка

start\_time = clock();

shaker\_sort(arrB, ARR\_N);

end\_time = clock();

shaker\_sort\_time = end\_time - start\_time;

// Обнуление времени

start\_time = 0;

end\_time = 0;

// Сортировка выбором

start\_time = clock();

selection\_sort(arrB, ARR\_N);

end\_time = clock();

selection\_sort\_time = end\_time - start\_time;

// Обнуление времени

start\_time = 0;

end\_time = 0;

// Сортировка вставками

start\_time = clock();

insertion\_sort(arrB, ARR\_N);

end\_time = clock();

insertion\_sort\_time = end\_time - start\_time;

// Обнуление времени

start\_time = 0;

end\_time = 0;

// Быстрая сортировка

start\_time = clock();

quick\_sort(0, ARR\_N - 1, arrB);

end\_time = clock();

quick\_sort\_time = end\_time - start\_time;

// Обнуление времени

start\_time = 0;

end\_time = 0;

//Вывод времени

cout << "--------------лучшая ситуация-----------------\n";

cout << "\nВремя сортировки пузырьковой:" << bubble\_sort\_time << "мс.\n";

cout << "\nВремя сортировки Шейкером:" << shaker\_sort\_time << "мс.\n";

cout << "\nВремя сортировки выбором:" << selection\_sort\_time << "мс.\n";

cout << "\nВремя сортировки вставкой:" << insertion\_sort\_time << "мс.\n";

cout << "\nВремя сортировки быстрой:" << quick\_sort\_time << "мс.\n\n";

return 0;

}

int\* bubble\_sort(int\* arr, int n\_arr) {

int temp = 0; //Временная переменная

for (int i = 0; i < n\_arr - 1; i++) {

for (int j = i + 1; j < n\_arr; j++) {

if (arr[i] > arr[j]) {

temp = arr[i];

arr[i] = arr[j];

arr[j] = temp;

}

}

}

return arr;

}

int\* shaker\_sort(int\* arr, int n\_arr) {

int left, right, i;

left = 0;

right = n\_arr - 1;

while (left <= right) {

for (i = right; i >= left; i--) {

if (arr[i - 1] > arr[i]) {

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

}

}

left++;

for (i = left; i <= right; i++) {

if (arr[i - 1] > arr[i]) {

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

}

}

right--;

}

return arr;

}

int\* selection\_sort(int\* arr, int n\_arr) {

int temp; //Временная переменная

int max;

for (int i = n\_arr; i > 0; i--) {

max = i;

for (int j = i - 1; j > 0; j--)

max = (arr[j] > arr[max]) ? j : max;

if (i != max) {

temp = arr[i];

arr[i] = arr[max];

arr[max] = temp;

}

}

return arr;

}

int\* insertion\_sort(int\* arr, int n\_arr) {

int temp, i, j;

for (i = 1; i < n\_arr; i++) {

temp = arr[i];

for (j = i - 1; j >= 0 && arr[j] > temp; j--)

arr[j + 1] = arr[j];

arr[j + 1] = temp;

}

return arr;

}

int\* quick\_sort(int b, int e, int\* arr) //b-первый элемент массива,а е-последний

{

int l = b, r = e;

int piv = arr[(l + r) / 2]; // Опорным элементом для примера возьмём средний

while (l <= r)

{

while (arr[l] < piv)

l++;

while (arr[r] > piv)

r--;

if (l <= r)

swap(arr[l++], arr[r--]);

}

if (b < r)

quick\_sort(b, r, arr);

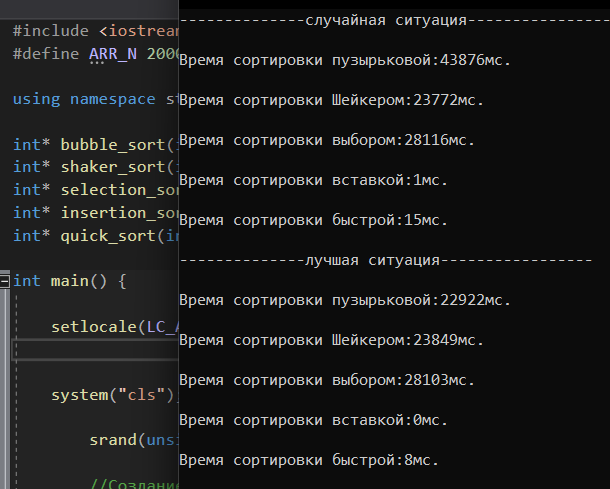
if (e > l)

quick\_sort(l, e, arr);

return arr;

}

Результат выполнения кода:



**ВЫВОД**

Лучший результат показала сортировка вставкой в 2 случаях. Наихудшей в сортировке случайной ситуации себя показала пузырьковая сортировка, при лучшей ситуации наихудшая – сортировка выбором. Очень неплохой результат имеет быстрая сортировка.