Министерство образования Республики Беларусь

Учреждение образования

“Брестский государственный технический университет”

**Лабораторная работа №2**

**По дисциплине АиСД за 2 семестр**  
**Тема: «Алгоритмы сортировки»**

**Вариант 4**

**Выполнил:**

Студент группы ПО-6(1)  
 1-го курса

Мартынович Даниил

**Проверила:**

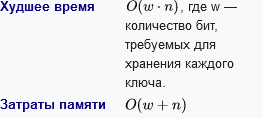
Гирель Т. Н.

Задание: В соответствии с номером в журнале необходимо реализовать 3 алгоритма сортировки и провести экспериментальное исследование их эффективности.

Выполнение программ выполнялось на процессоре Intel Core I7, объем памяти 8ГБ, OC – Windows 10.

**Алгоритмы, не использующие операцию сравнения**.

Поразрядная сортировка .



#include<iostream>

#include<cstdlib>

#include<chrono>

using namespace std;

void RandomArray(int\* array, int size) {

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

array[i] = rand() % 100000;

}

}

int getMax(int array[], int size) { //Функция для подсчёта максимального значения в массиве

int max = array[0];

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

if (array[i] > max) {

max = array[i];

}

}

return max;

}

void countSort(int array[], int size, int exp) { //Функция для подсчёта сортировки array[] в соответствии с цифрой, представленной exp

int\* output = new int[size]; //Выходной массив

int i, count[10] = { 0 };

for (i = 0; i < size; i++) { //Сохраняем количество вхождений в count

count[(array[i] / exp) % 10]++;

}

for (i = 1; i < 10; i++) { //Изменить count[i] так, чтобы count[i] содержал фактическое положение этой цифры во входных данных

count[i] += count[i - 1];

}

for (i = size - 1; i >= 0; i--) { //Создаём выходной массив

output[count[(array[i] / exp) % 10] - 1] = array[i];

count[(array[i] / exp) % 10]--;

}

for (i = 0; i < size; i++) { //Копируем выходной массив array[] так что теперь array[] содержит отсортированные номера по текущей цифре

array[i] = output[i];

}

}

void RadixSort(int array[], int size) { //Основная фукнция для сортировки array[] размера size

int max = getMax(array, size); //Узнаём максимальное число, чтобы узнать количество цифр

for (int exp = 1; max / exp > 0; exp \*= 10) {

countSort(array, size, exp); //Делаем подсчёт для каждой цифры

}

}

void ProsmotrMassive(int array[], int size)

{

for (size\_t i = 0; i < size; i++)

{

cout << array[i] << "\t";

}

cout << endl;

}

int main() {

for (int size = 50000; size <= 1000000; size += 50000) {

int\* array = new int[size];

RandomArray(array, size);

cout << "Radixsort: ";

RadixSort(array, size);

cout << "\nSize = " << size;

//ProsmotrMassive(array, size);

cout << endl << endl;

delete[]array;

}

system("pause");

return 0;

}

|  |  |  |
| --- | --- | --- |
| # | **Количество элементов в массиве** | **Время выполнения алгоритма, с** |
| 1 | 50 000 | 19.327 |
| 2 | 100 000 | 37.924 |
| 3 | 150 000 | 57.039 |
| 4 | 200 000 | 77.772 |
| 5 | 250 000 | 97.089 |

**Квадратичные алгоритмы сортировки.**

Сортировка чет-нечет

. 

#include <iostream>

using namespace std;

int main()

{

setlocale(LC\_ALL, "RU");

const int N = 50000;

int array[N];

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

{

array[i] = 0 + rand() % 1000;

}

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

{

cout << array[i] << " ";

}

cout << endl;

cout << endl;

cout << endl;

for (size\_t i = 0; i < N; i++) {

// (i % 2) ? 0 : 1 возвращает 1, если i четное, 0, если i не четное

for (size\_t j = (i % 2) ? 0 : 1; j + 1 < N; j += 2) {

if (array[j] > array[j + 1]) {

swap(array[j], array[j + 1]);

}

}

}

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

{

cout << array[i] << " ";

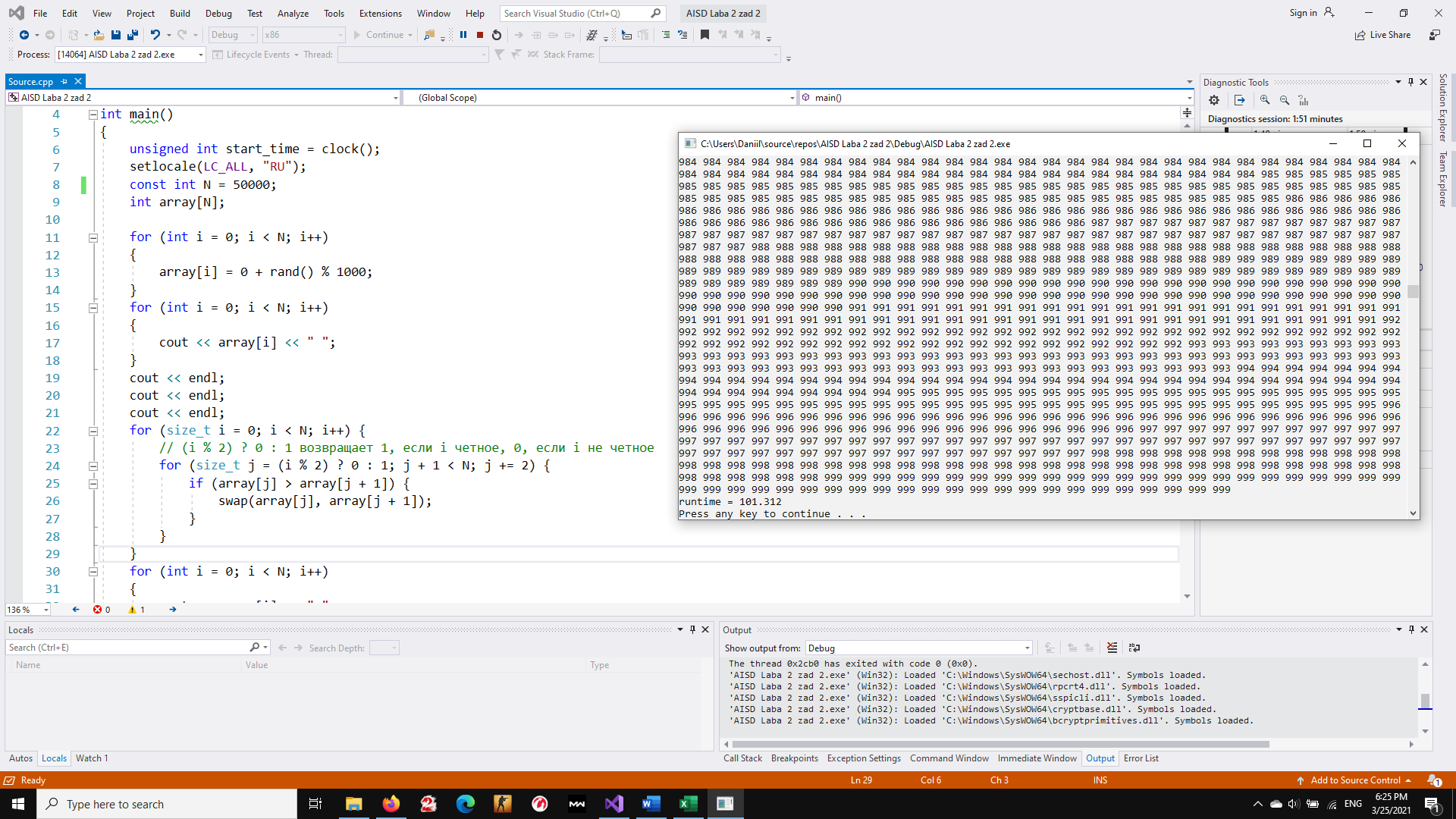
}

cout << endl;

system("pause");

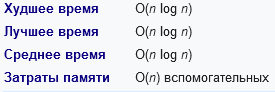
}

|  |  |  |
| --- | --- | --- |
| # | **Количество элементов в массиве** | **Время выполнения алгоритма, с** |
| 1 | 50 000 | 96.33 |
| 2 | 100 000 | 303.78 |
| 3 | 150 000 | 510.23 |
| 4 | 200 000 | 718.098 |
| 5 | 250 000 | 925.07 |



**Быстрые алгоритмы сортировки.**

Сортировка слиянием.



#include <iostream>

#include <stdio.h>

#include <stdlib.h>

using namespace std;

#define SIZE 50

void mergeSort(int\* a, int l, int r)

{

if (l == r) return;

int mid = (l + r) / 2;

mergeSort(a, l, mid);

mergeSort(a, mid + 1, r);

int i = l;

int j = mid + 1;

int\* tmp = (int\*)malloc(r \* sizeof(int));

for (int step = 0; step < r - l + 1; step++)

{

if ((j > r) || ((i <= mid) && (a[i] < a[j])))

{

tmp[step] = a[i];

i++;

}

else

{

tmp[step] = a[j];

j++;

}

}

for (int step = 0; step < r - l + 1; step++)

a[l + step] = tmp[step];

}

int main()

{

int a[SIZE];

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

{

a[i] = (rand() % 1000);

cout << a[i] << " ";

}

mergeSort(a, 0, SIZE - 1);

cout << endl;

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

{

cout << a[i] << " ";

}

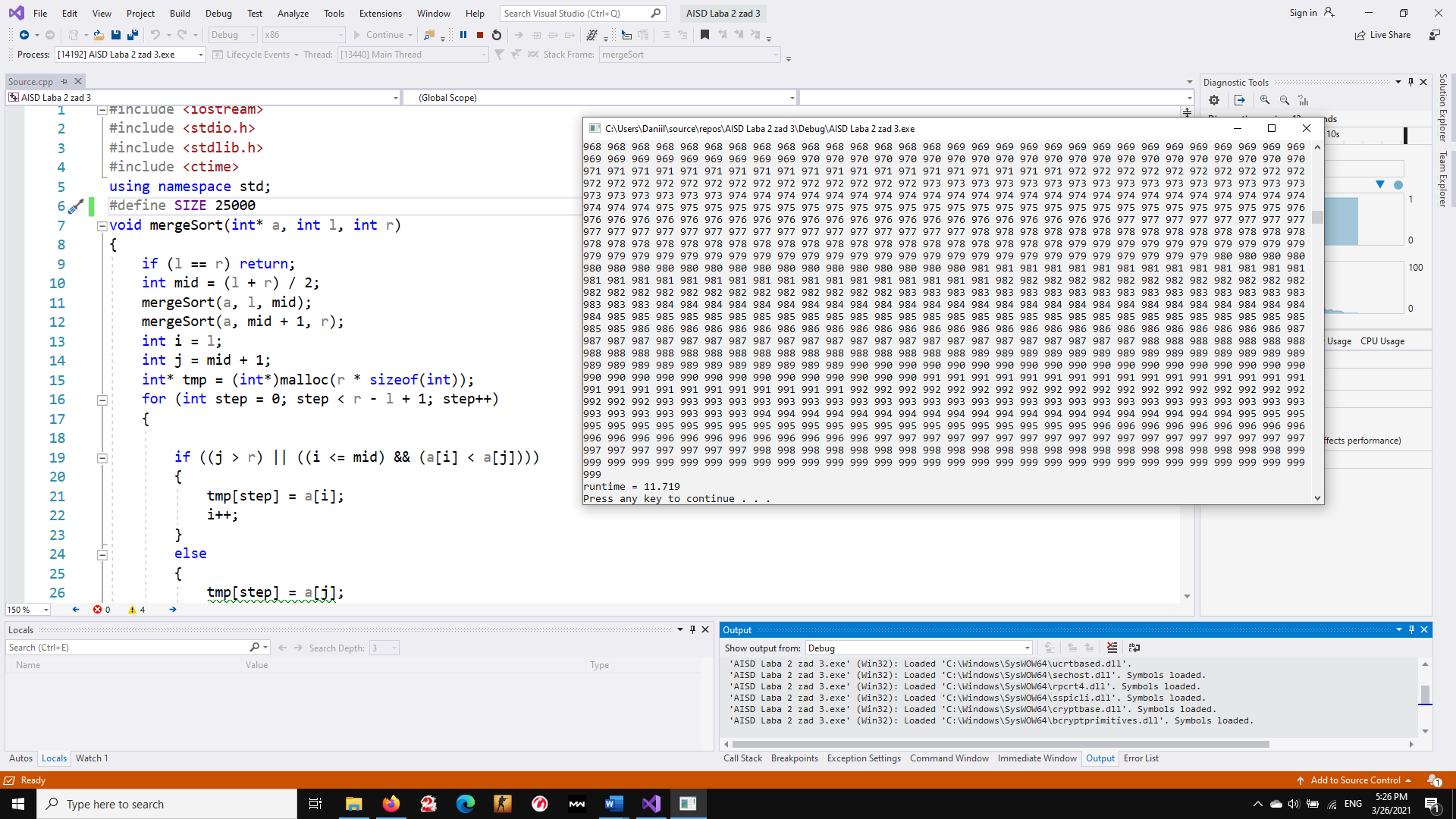
cout << endl;

system("pause");

return 0;

}

|  |  |  |
| --- | --- | --- |
| # | **Количество элементов в массиве** | **Время выполнения алгоритма, с** |
| 1 | 50 00 | 2.481 |
| 2 | 100 00 | 4.719 |
| 3 | 150 00 | 6.936 |
| 4 | 200 00 | 9.152 |
| 5 | 250 00 | 11.334 |



Вывод: реализовал 3 алгоритма сортировки и провел экспериментальное исследование их эффективности. Самым эффективным алгоритмом сортировки оказалась поразрядная сортировка (Radixsort).