**1)Сортировка Шелла:**

#include <ctime>

#include <iomanip>

#include <iostream>

using namespace std;

const long long COUNT = 10;

void shell\_sort(int\* arr, int size)

{

for (int s = size / 2; s > 0; s /= 2)

{

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

{

for (int j = i - s; j >= 0 and arr[j] > arr[j + s]; j -= s)

{

int temp = arr[j];

arr[j] = arr[j + s];

arr[j + s] = temp;

}

}

}

}

int main()

{

int experiments = 10;

double sumTime = 0;

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

{

int \*mass = new int[COUNT];

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

{

mass[i] = rand() % 100;

}

clock\_t start = clock();

shell\_sort(mass, COUNT);

delete[] mass;

clock\_t end = clock();

double time = ((double)(end - start)) / CLOCKS\_PER\_SEC;

sumTime += time;

cout << fixed << setprecision(6) << time << endl;

}

double averageTime = sumTime / experiments;

cout << fixed << setprecision(6) << averageTime << endl;

return 0;

}

**2)Сортировка подсчётом:**

#include <iostream>

#include <ctime>

#include <climits>

#include <iomanip>

using namespace std;

const long long COUNT = 10;

void sort\_count (int mass[], int len)

{

int minNum = INT\_MAX;

int maxNum = INT\_MIN;

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

{

minNum = min(mass[i], minNum);

maxNum = max(mass[i], maxNum);

}

int subLen = maxNum - minNum + 1;

int subMass [subLen];

fill(subMass, subMass + subLen, 0);

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

{

subMass[mass[i] - minNum] += 1;

}

int pos = 0;

for (int num = 0; num < subLen; ++num)

{

for (int i = 0; i < subMass[num]; i++)

{

mass[pos] = num + minNum;

pos++;

}

}

}

int main()

{

int experiments = 10;

int mass;

double sumTime = 0;

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

{

int \*mass = new int[COUNT];

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

{

mass[j] = rand() % 100;

}

clock\_t start = clock();

sort\_count( mass,COUNT);

delete[] mass;

clock\_t end = clock();

double time = ((double)(end - start)) / CLOCKS\_PER\_SEC;

sumTime += time;

cout << fixed << setprecision(6) << time << endl;

}

double averageTime = sumTime / experiments;

cout << fixed << setprecision(6) << averageTime << endl;

return 0;

}