Сортировка Шелла на случайном наборе.

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <stdlib.h>;

#include <time.h>;

void shell\_sort(int\* array, int size) {

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

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

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

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

int temp = array[j];

array[j] = array[i];

array[i] = temp;

}

}

}

}

}

main() {

int\* a;

int N;

printf("Введите размер массива: ");

scanf("%d", &N);

a = (int\*)malloc(N \* sizeof(int));

srand(time(NULL));

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

{

a[i] = rand() % 1000;

}

printf("\n");

clock\_t start = clock();

shell\_sort(a, N);

clock\_t end = clock();

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

printf("Сортировка случайного набора %f", time\_spent);

}

Сортировка Шелла на возрастающей последовательности

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <stdlib.h>;

#include <time.h>;

void shell\_sort(int\* array, int size) {

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

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

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

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

int temp = array[j];

array[j] = array[i];

array[i] = temp;

}

}

}

}

}

void DecreasingSort(int n, int mass[])

{

int new, location;

for (int i = 1; i < n; i++)

{

new = mass[i];

location = i - 1;

while (location >= 0 && mass[location] < new)

{

mass[location + 1] = mass[location];

location = location - 1;

}

mass[location + 1] = new;

}

}

void AscendingSort(int n, int mass[])

{

int new, location;

for (int i = 1; i < n; i++)

{

new = mass[i];

location = i - 1;

while (location >= 0 && mass[location] > new)

{

mass[location + 1] = mass[location];

location = location - 1;

}

mass[location + 1] = new;

}

}

main() {

int\* a;

int N,k;

printf("Введите размер массива: ");

scanf("%d", &N);

a = (int\*)malloc(N \* sizeof(int));

srand(time(NULL));

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

{

a[i] = rand() % 10000;

}

AscendingSort(N, a);

printf("\n");

clock\_t start = clock();

shell\_sort(a, N);

clock\_t end = clock();

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

printf("Сортировка случайного набора %f", time\_spent);

}

Сортировка Шелла, половина –возрастает, половина-убывает

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <stdlib.h>;

#include <time.h>;

void shell\_sort(int\* array, int size) {

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

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

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

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

int temp = array[j];

array[j] = array[i];

array[i] = temp;

}

}

}

}

}

void DecreasingSort(int n, int mass[])

{

int new, location;

for (int i = 1; i < n; i++)

{

new = mass[i];

location = i - 1;

while (location >= 0 && mass[location] < new)

{

mass[location + 1] = mass[location];

location = location - 1;

}

mass[location + 1] = new;

}

}

void AscendingSort(int n, int mass[])

{

int new, location;

for (int i = 1; i < n; i++)

{

new = mass[i];

location = i - 1;

while (location >= 0 && mass[location] > new)

{

mass[location + 1] = mass[location];

location = location - 1;

}

mass[location + 1] = new;

}

}

main() {

int\* a;

int N,k;

printf("Введите размер массива: ");

scanf("%d", &N);

a = (int\*)malloc(N \* sizeof(int));

srand(time(NULL));

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

{

a[i] = rand() % 10000;

}

DecreasingSort(N, a);

AscendingSort(N/2, a);

printf("\n");

clock\_t start = clock();

shell\_sort(a, N);

clock\_t end = clock();

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

printf("Сортировка случайного набора %f", time\_spent);

}

**Qsort**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <stdlib.h>;

#include <time.h>;

void qs(int\* items, int left, int right) //вызов функции: qs(items, 0, count-1);

{

int i, j;

int x, y;

i = left; j = right;

/\* выбор компаранда \*/

x = items[(left + right) / 2];

do {

while ((items[i] < x) && (i < right)) i++;

while ((x < items[j]) && (j > left)) j--;

if (i <= j) {

y = items[i];

items[i] = items[j];

items[j] = y;

i++; j--;

}

} while (i <= j);

if (left < j) qs(items, left, j);

if (i < right) qs(items, i, right);

}

void shell\_sort(int\* array, int size) {

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

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

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

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

int temp = array[j];

array[j] = array[i];

array[i] = temp;

}

}

}

}

}

void DecreasingSort(int n, int mass[])

{

int new, location;

for (int i = 1; i < n; i++)

{

new = mass[i];

location = i - 1;

while (location >= 0 && mass[location] < new)

{

mass[location + 1] = mass[location];

location = location - 1;

}

mass[location + 1] = new;

}

}

void AscendingSort(int n, int mass[])

{

int new, location;

for (int i = 1; i < n; i++)

{

new = mass[i];

location = i - 1;

while (location >= 0 && mass[location] > new)

{

mass[location + 1] = mass[location];

location = location - 1;

}

mass[location + 1] = new;

}

}

int comp(const void\* x1, const void\* x2) // функция сравнения элементов массива

{

return (\*(int\*)x1 - \*(int\*)x2); // если результат вычитания равен 0, то числа равны, < 0: x1 < x2; > 0: x1 > x2

}

main() {

int\* a;

int N,k;

printf("Введите размер массива: ");

scanf("%d", &N);

a = (int\*)malloc(N \* sizeof(int));

srand(time(NULL));

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

{

a[i] = rand() % 10000;

}

DecreasingSort(N, a);

AscendingSort(N / 2, a);

printf("\n");

clock\_t start = clock();

qsort(a, N, sizeof(int), (int(\*) (const void\*, const void\*)) comp);

clock\_t end = clock();

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

printf("Сортировка случайного набора %f", time\_spent);

}