Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное   
учреждение высшего образования

Национальный исследовательский Нижегородский государственный университет им. Н.И. Лобачевского

Институт информационных технологий, математики и механики

**Отчет по лабораторной работе**

**«Сортировки»**

**Выполнил**:

студент 3822Б1ПМ1

Кубышкин В. В.

**Проверил**:

преподаватель каф. МОСТ,

Волокитин В.Д.

Нижний Новгород

2022

**Содержание**

[Постановка задачи 3](#_Toc26962562)

[Метод решения 4](#_Toc26962563)

[Руководство пользователя 5](#_Toc26962564)

[Описание программной реализации 6](#_Toc26962565)

[Подтверждение корректности 7](#_Toc26962566)

[Результаты экспериментов 8](#_Toc26962567)

[Заключение 9](#_Toc26962568)

[Приложение 10](#_Toc26962569)

**Постановка задачи**

1. Реализовать 2 вида сортировок: медленные(квадратичные) и быстрые. (Пузырьком с остановкой, вставками, слиянием и расческой).
2. Измерить время работы каждой сортировки.
3. Сравнить работу сортировок с библиотечной сортировкой qsort.
4. Написать отчет о проделанной работе.

# Метод решения

1. Сортировка пузырьком с остановкой: массив обходится от начала и до конца, попутно сравнивая соседние элементы, если соседние элементы находятся не в правильном порядке, то они меняются местами при помощи функции swap и осуществляется выход из цикла и далее сравниваются другие соседние элементы. Если соседние элементы стоят на своих местах, то массив будет обходиться до элементов, стоящих не на своих местах или до конца массива. Таким образом за первую проходу “всплывает” максимальный элемент и так далее.
2. Сортировка вставками: берется 2 элемента массива и сравнивается с 1, если 2 элемент будет меньше, то они меняются местами при помощи функции swap и осуществляется выход из цикла. Далее берется 3 элемент массива и сравнивается с 1 элементом (который сейчас стоит на 2 позиции), если 3 элемент меньше, то 1 и 3 меняются местами, потом 3 сравнивается со 2 элементом (который сейчас стоит на 1 позиции) и если 3 будет меньше, то они меняются местами, а если будет больше, то мы выходим из цикла и берем элемент с 4 позиции и так далее.
3. Сортировка расческой: выбирается размер шага, который будет равен размеру массива, таким образом в первую очередь сравнивают первый и последний элементы массивы, если первый больше, то они меняются местами при помощи функции swap. Затем берется шаг = предыдущий шаг/ фактор (= 1.247). Таким образом проходят другие элементы и если один элемент будет больше, чем элемент с его номером + шаг, то они меняются местами. И так делается до тех пор, пока шаг не будет равен 0. \*Фактор – фактор уменьшения, который вычисляется по формуле: 1 / (1-e^(-φ)
4. Сортировка слияния: имеющийся массив делится пополам до тех пор, пока каждый подмассив не будет состоять из 1-2 элементов, в зависимости от их числа. Эти подмассивы сортируются, после чего они сходятся в 3х - 4х – элементный подмассив, где элементы тоже сортируются и так до тех пор, пока подмассивы не сольются в один массив. Так же на каждом этапе, если один из подмассив закончится, то оставшиеся элементы второго автоматически перейдут в конец.

# Руководство пользователя

1. При запуске программы высвечивается надпись “Размер массива:”, где пользователь должен ввести число элементов массива и нажать enter.

2. Далее высвечивается надпись “Левая граница:”, где пользователь должен ввести минимальное число для того, чтобы программа создала рандомное число и нажать enter.

3. Далее высвечивается надпись “Правая граница:”, где пользователь должен ввести максимальное число для того, чтобы программа создала рандомное число и нажать enter.

4. После данных действий программа создает и заполняет массив с заданными пользователем данными.

5. Создаются 2 пустых массива и в оба копируются числа из раннее созданного массива.

6. Затем на высвечиваются надписи “Выберите сортировку: 1. Пузырек 2. Вставками 3. Расческой 4. Слиянием” и “Выбор:”, после чего пользователю следует выбрать вид сортировки, написать ее номер и нажать enter.

7. Программа работает с условием, заданным пользователем и использует сортировку, которую выбрали, сортирую первый копированный массив, а также начинает отсчет времени ее работы.

8. Производится расчет времени работы сортировки.

9. Происходит встроенная сортировка второго копированного массива.

10. Сравнение сортировок первого и второго копированных массивов.

11. Если есть числа, которые не совпали, то высветится надпись “Не совпало - \*число не совпавших элементов\*”, в другом случае “Не совпало – 0”.

12. В конце выведет время работы сортировки, которую выбирал пользователь “Время: …”

# Описание программной реализации

Для реализации данной программы мне пришлось воспользоваться такими директивам, как:

#include "stdio.h" – для использования основных функций языка C;

#include "stdlib.h" – для использования функции malloc и другими из стандартной библиотеке;

#include "time.h" – для измерения времени работы сортировок;

#include "locale.h" – для использования русского языка в коде.

Так же для сравнения работы сортировок, я воспользовался сортировкой qsort из библиотеки, для это мне нужно было написать компаратор для qsort:

Изображение выглядит как текст

Автоматически созданное описание

Рис.1 компаратор для qsort.

# Подтверждение корректности

Для подтверждения корректности в программе я воспользовался библиотечной сортировкой qsort и сравнивал ее работу с работой моих сортировок. Для это я создавал два пустых массива (array1\_sort[i] и array2\_sort[i]) и копировал в них массив с рандомными числами, а затем array1\_sort[i] я сортировал при помощи одной из моих сортировок, а array2\_sort[i] я сортировал при помощи qsort. В конце программы я написал цикл с условием, которые сравнивают отсортированные массивы и если есть элементы, которые не совпали, то выводится их количество:

Изображение выглядит как текст

Автоматически созданное описание

Рис.2 Сравнение array1\_sort[i] и array2\_sort[i]

# Результаты экспериментов

Я решил замерить время работы каждой сортировки, для этого я взял диапазон чисел от -100 до 100 и разные размеры массива от 1000 до … и результаты занес в таблицу:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Пузырек | Вставки | Расческа | Слиянием |
| 1000 эл | 0,007000 сек | 0,004000 сек | 0,000000 сек | 0,001000 сек |
| 10000 эл | 0,694000 сек | 0,299000 сек | 0,003000 сек | 0,005000 сек |
| 100000 эл | 70,471000 сек | 30,523000 сек | 0,026000 сек | 0,058000 сек |

Табл.1 Результаты эксперимента

По результатам тестирования с задачей лучше всего справилась сортировка расческой, которая превосходит сортировку слиянием примерно в 2 раза. Не зря эти сортировки названы быстрыми. Хуже всего справилась сортировка пузырьком, которая сортировала числа примерно в 2 раза медленнее, чем сортировка вставками.

Таким образом можно рассчитать время работы каждой сортировки:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Пузырек | Вставки | Расческа | Слиянием |
| Лучший | N | N | NlogN | NlogN |
| Средний | N(N-1)/2 | N(N-1)/2 | NlogN | NlogN |
| Худший | N^2 | N^2 | N^2 | NlogN |

Табл.2 Время работы сортировок

# Заключение

Итак, поставленная в начале работы задача выполнена: написана программа для исполнения заданных сортировок (код см. в Приложении), описаны взаимодействие пользователя с программой, структура проекта, способ проверки корректности алгоритмов, приведены результаты экспериментов, согласующиеся с теоретическими данными.

# Приложение

#include "stdio.h"

#include "stdlib.h"

#include "time.h"

#include "locale.h"

void swap(float\* a, float\* b) // функция смены позиции

{

float temp = \*a;

\*a = \*b;

\*b = temp;

}

void bubble(float\* arr, int size) // пузырек с остановкой

{

int flag;

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

{

flag = 0;

for (int j = 0; j < size - 1; j++)

{

if (arr[j] > arr[j + 1])

{

swap(&arr[j], &arr[j + 1]);

}

flag = 1;

if (flag == 0)

{

return;

}

}

}

}

void Insert(float\* arr, int size) // вставками

{

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

{

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

{

if (arr[j] < arr[j - 1])

{

swap(&arr[j], &arr[j - 1]);

}

else

{

break;

}

}

}

}

void comb(float\* arr, int size) // расческой

{

float factor = 1.247; // число - фактор уменьшения

int step = size - 1;

while (step != 0)

{

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

{

if (arr[i] > arr[i + step])

{

swap(&arr[i], &arr[i + step]);

}

}

step /= factor;

}

}

void merge(float\* arr, int left, int mid, int right)

{

int i = 0, i1 = 0, i2 = 0;

float\* b = (float\*)malloc(sizeof(float) \* (right - left));

while ((left + i1 < mid) && (mid + i2 < right))

{

if (arr[left + i1] < arr[mid + i2])

{

b[i1 + i2] = arr[left + i1];

i1++;

}

else

{

b[i1 + i2] = arr[mid + i2];

i2++;

}

}

while (left + i1 < mid)

{

b[i1 + i2] = arr[left + i1];

i1++;

}

while (mid + i2 < right)

{

b[i1 + i2] = arr[mid + i2];

i2++;

}

for (i = 0; i < (i1 + i2); i++)

{

arr[left + i] = b[i];

}

free(b);

}

void mergeSort(float\* arr, int left, int right) // Слиянием

{

int mid;

if (left + 1 >= right)

return;

mid = (left + right) / 2;

mergeSort(arr, left, mid);

mergeSort(arr, mid, right);

merge(arr, left, mid, right);

}

float\* full\_mass(int size, int a, int b) //заполнение массива

{

float\* mass;

mass = (float\*)malloc(size \* sizeof(float));

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

{

mass[i] = (rand() / (float)(RAND\_MAX))\*(b-a)+a;

}

return mass;

}

float\* null\_mass(int size) //создание пустого массива

{

float\* mas;

mas = (float\*)malloc(size \* sizeof(float));

return mas;

}

void mass\_copy(float\* orig, float\* copy, int size)

{

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

{

copy[i] = orig[i];

}

}

int compare(const void\* a, const void\* b)//компаратор для qsort

{

float arg1 = \*(const float\*)a;

float arg2 = \*(const float\*)b;

if (arg1 < arg2)

{

return -1;

}

if (arg1 > arg2)

{

return 1;

}

return 0;

}

int main()

{

float\* mass = 0, \* array1\_sort = 0, \* array2\_sort = 0;

int size, count = 0;

double time;

clock\_t start, end;

int a, b;

int c;

setlocale(LC\_ALL, "Rus");

printf("Размер массива:\t");

scanf\_s("%d", &size);

printf("Левая граница:\t");

scanf\_s("%d", &a);

printf("Правая граница:\t");

scanf\_s("%d", &b);

mass = full\_mass(size, a, b);

array1\_sort = null\_mass(size);

array2\_sort = null\_mass(size);

mass\_copy(mass, array1\_sort, size);

mass\_copy(mass, array2\_sort, size);

printf("Выберите сортировку:\n1.Пузырек\n2.Вставками\n3.Расческой\n4.Слиянием\n");

printf("Выбор:");

scanf\_s("%d", &c);

if (c == 1)

{

start = clock();

bubble(array1\_sort, size);

end = clock();

}

else if (c == 2)

{

start = clock();

Insert(array1\_sort, size);

end = clock();

}

else if (c == 3)

{

start = clock();

comb(array1\_sort, size);

end = clock();

}

else if (c == 4)

{

start = clock();

mergeSort(array1\_sort, 0, size);

end = clock();

}

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

qsort(array2\_sort, size, sizeof(float), compare);

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

{

if (array1\_sort[i] != array2\_sort[i])

{

count++;//счётчик не схожих элементов

printf("%f - %f\n", array1\_sort[i], array2\_sort[i]);

}

}

printf("Не совпало - %d\n", count);

printf("Время: %lf", time);

return 0;

}