МИНОБРНАУКИ РОССИИ

Санкт-Петербургский государственный

электротехнический университет

«ЛЭТИ» им. В.И. Ульянова (Ленина)

Кафедра МОЭВМ

отчет

по лабораторной работе №1

по дисциплине «Программирование»

Тема: Обзор стандартной библиотеки.

|  |  |  |
| --- | --- | --- |
| Студентка гр. 6303 |  | Горбунова А.П. |
| Преподаватель |  | Берленко Т.А. |

Санкт-Петербург

2017

**Цель**: Написать программу, на вход которой подается массив целых чисел длины **1000**.

Программа должна совершать следующие действия:

* отсортировать массив с помощью алгоритма "сортировка пузырьком"
* посчитать время, за которое будет совершена сортировка, используя при этом **функцию стандартной библиотеки**
* отсортировать массив с помощью алгоритма "быстрая сортировка" (quick sort), используя при этом **функцию стандартной библиотеки**
* посчитать время, за которое будет совершена сортировка, используя при этом **функцию стандартной библиотеки**
* вывести отсортированный массив (элементы массива должны быть разделены пробелом)
* вывести время, за которое была совершена сортировка пузырьком
* вывести время, за которое была совершена быстрая сортировка

*(Отсортированный массив, время сортировки пузырьком, время быстрой сортировки должны быть выведены с новой строки, при этом элементы массива должны быть разделены пробелами.)*

**Ход работы:**

* Для выполнения данной работы необходимо подключить три библиотеки: «*stdio.h*», «*time.h*» (в которой содержится функция clock) и «stdlib.h» (в которой содержится функция qsort).

|  |
| --- |
| #include <stdio.h>  #include <time.h>  #include<stdlib.h> |

* Далее, до функции *main*, объявляется функция сравнения *cmp,* необходимая для работы функции сортировки *qsort*:

|  |
| --- |
| int cmp(const void \*a, const void \*b) //функция сравнения  {  return \*(int\*)a - \*(int\*)b;  } |

Функция сравнения получает в качества аргумента адреса двух блоков, которые нужно сравнить и возвращает 1, 0 или -1:

положительное значение, если a > b

0, если a == b

отрицательное значение, если a < b

Необходимо привести данные указатели типа (const void\*) к типу (int \*) и осуществляется это с помощью дописывания перед указателем выражения «(const int\*)». Затем нужно получить значение переменной типа int, которая лежит по этому адресу. Это делается с помощью дописывания спереди звездочки.

* В функции *main* создаются два массива, затем входные данные записываются в один из массивов, после чего данные из одного массива копируются в другой. Это нужно для реализации двух различных способов сортировки данных массива.

int main() {

int i, j;

int N=1000;

int arr1[N], arr2[N];

//для массива arr1 будем использовать метод bubble sort, для arr2 - функцию qsort

double timebsort, timeqsort;

//выбираем тип данных double чтобы вывести время в долях секунд

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

{

scanf("%d", &arr1[i]);

}

//считываем элементы массива arr1

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

{

arr2[i]=arr1[i];

}

//копируем массив arr1 в масссив arr2

Для того, чтобы посчитать время, за которое будет совершена сортировка, необходимо воспользоваться функцией стандартной библиотеки. Функция clock() возвращает количество временных тактов, прошедших с начала запуска программы. Чтобы перевести его в секунды, разделим возвращаемое функцией значение на СLOCKS\_PER\_SEC (константа которая представляет количество единиц процессорного времени в секунде), значение которой хранится в <time.h>. Следовательно, в результате деления разницы между двумя возвращаемыми значениями функции clock () на константу CLOCKS\_PER\_SEC получается количество секунд, прошедшее между двумя вызовами функции. Приведение значений к типу double до операции деления позволит получать результат в долях секунды.

Далее реализуется «сортировка пузырьком». После «сортировки пузырьком» с помощью функции «clock()» вычисляется время, за которое была произведена сортировка.

timebsort = clock();

//записываем в переменную timebsort время до начала сортировки

for(i = 0 ; i < N - 1; i++)

{

// сравниваем два соседних элемента.

for(j = 0 ; j <N - i - 1 ; j++)

{

if(arr1[j] > arr1[j+1])

{

// если они идут в неправильном порядке, то меняем их местами.

int tmp = arr1[j];

arr1[j] = arr1[j+1] ;

arr1[j+1] = tmp;

}

}

}

timebsort = (clock()-timebsort)/CLOCKS\_PER\_SEC;

Далее необходимо вызвать функцию быстрой сортировки *qsort()* стандартной библиотеки и засечь время ее выполнения с помощью функции *clock* (см.предыдущий пункт)

timeqsort = clock();

//записываем текущее время в переменную timeqsort

qsort(arr2, N, sizeof(int), cmp ); // вызываем функцию сортировки

timeqsort = (clock()-timeqsort)/CLOCKS\_PER\_SEC;

//отнимаем от текущего времени время из timeqsort, переводим его в секунды

* Затем выводится один из отсортированных массивов, а также время «сортировки пузырьком» и время быстрой сортировки.

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

{

//выводим отсортированный массив

printf("%d ", arr2[i]);

}

printf("\n%.5f seconds\n", timebsort);

printf("%.5f seconds\n", timeqsort);

return;

}

ИСХОДНЫЙ КОД

#include <stdio.h>

#include <time.h>

#include <stdlib.h>

int cmp(const void \*a, const void \*b)

{

return \*(int\*)a - \*(int\*)b;

}

int main() {

int i, j;

int N=1000;

int arr1[N], arr2[N];

//для массива arr1 будем использовать метод bubble sort, для arr2 - функцию qsort

double timebsort, timeqsort;

//выбираем тип данных double чтобы вывести время в долях секунд

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

{

scanf("%d", &arr1[i]);

}

//считываем элементы массива arr1

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

{

arr2[i]=arr1[i];

}

//копируем массив arr1 в масссив arr2

timebsort = clock();

//записываем в переменную timebsort время до начала сортировки

for(i = 0 ; i < N - 1; i++)

{

// сравниваем два соседних элемента.

for(j = 0 ; j <N - i - 1 ; j++)

{

if(arr1[j] > arr1[j+1])

{

// если они идут в неправильном порядке, то меняем их местами.

int tmp = arr1[j];

arr1[j] = arr1[j+1] ;

arr1[j+1] = tmp;

}

}

}

timebsort = (clock()-timebsort)/CLOCKS\_PER\_SEC;

//отнимаем от текущего времени время из time1, переводим его в секунды

//СLOCKS\_PER\_SEC - константа которая представляет количество единиц процессорного времени в секунде

timeqsort = clock();

//записываем текущее время в переменную timeqsort

qsort(arr2, N, sizeof(int), cmp ); // вызываем функцию сортировки

timeqsort = (clock()-timeqsort)/CLOCKS\_PER\_SEC;

//отнимаем от текущего времени время из timeqsort, переводим его в секунды

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

{

//выводим отсортированный массив

printf("%d ", arr2[i]);

}

printf("\n%.5f seconds\n", timebsort);

printf("%.5f seconds\n", timeqsort);

}

**Пример работы программы**

**Input:**

-20 -20 -20 -19 -18 -18 -18 -18 -17 -17 -16 -16 -16 -16 -15 -15 -15 -14 -10 -9 -7 -7 -5 -5 -5 -4 -3 -3 -2 0 1 2 3 3 4 4 4 4 5 5 8 8 8 9 9 10 10 10 10 12 13 14 15 17 18 18 18 22 22 22 23 24 24 25 27 27 28 29 29 29 31 32 33 33 36 36 36 36 37 37 37 38 39 39 40 40 40 41 42 42 43 43 45 47 47 48 48 48 48 49 -20 -20 -20 -19 -18 -18 -18 -18 -17 -17 -16 -16 -16 -16 -15 -15 -15 -14 -10 -9 -7 -7 -5 -5 -5 -4 -3 -3 -2 0 1 2 3 3 4 4 4 4 5 5 8 8 8 9 9 10 10 10 10 12 13 14 15 17 18 18 18 22 22 22 23 24 24 25 27 27 28 29 29 29 31 32 33 33 36 36 36 36 37 37 37 38 39 39 40 40 40 41 42 42 43 43 45 47 47 48 48 48 48 49 -20 -20 -20 -19 -18 -18 -18 -18 -17 -17 -16 -16 -16 -16 -15 -15 -15 -14 -10 -9 -7 -7 -5 -5 -5 -4 -3 -3 -2 0 1 2 3 3 4 4 4 4 5 5 8 8 8 9 9 10 10 10 10 12 13 14 15 17 18 18 18 22 22 22 23 24 24 25 27 27 28 29 29 29 31 32 33 33 36 36 36 36 37 37 37 38 39 39 40 40 40 41 42 42 43 43 45 47 47 48 48 48 48 49 -20 -20 -20 -19 -18 -18 -18 -18 -17 -17 -16 -16 -16 -16 -15 -15 -15 -14 -10 -9 -7 -7 -5 -5 -5 -4 -3 -3 -2 0 1 2 3 3 4 4 4 4 5 5 8 8 8 9 9 10 10 10 10 12 13 14 15 17 18 18 18 22 22 22 23 24 24 25 27 27 28 29 29 29 31 32 33 33 36 36 36 36 37 37 37 38 39 39 40 40 40 41 42 42 43 43 45 47 47 48 48 48 48 49 -20 -20 -20 -19 -18 -18 -18 -18 -17 -17 -16 -16 -16 -16 -15 -15 -15 -14 -10 -9 -7 -7 -5 -5 -5 -4 -3 -3 -2 0 1 2 3 3 4 4 4 4 5 5 8 8 8 9 9 10 10 10 10 12 13 14 15 17 18 18 18 22 22 22 23 24 24 25 27 27 28 29 29 29 31 32 33 33 36 36 36 36 37 37 37 38 39 39 40 40 40 41 42 42 43 43 45 47 47 48 48 48 48 49 -20 -20 -20 -19 -18 -18 -18 -18 -17 -17 -16 -16 -16 -16 -15 -15 -15 -14 -10 -9 -7 -7 -5 -5 -5 -4 -3 -3 -2 0 1 2 3 3 4 4 4 4 5 5 8 8 8 9 9 10 10 10 10 12 13 14 15 17 18 18 18 22 22 22 23 24 24 25 27 27 28 29 29 29 31 32 33 33 36 36 36 36 37 37 37 38 39 39 40 40 40 41 42 42 43 43 45 47 47 48 48 48 48 49 -20 -20 -20 -19 -18 -18 -18 -18 -17 -17 -16 -16 -16 -16 -15 -15 -15 -14 -10 -9 -7 -7 -5 -5 -5 -4 -3 -3 -2 0 1 2 3 3 4 4 4 4 5 5 8 8 8 9 9 10 10 10 10 12 13 14 15 17 18 18 18 22 22 22 23 24 24 25 27 27 28 29 29 29 31 32 33 33 36 36 36 36 37 37 37 38 39 39 40 40 40 41 42 42 43 43 45 47 47 48 48 48 48 49 -20 -20 -20 -19 -18 -18 -18 -18 -17 -17 -16 -16 -16 -16 -15 -15 -15 -14 -10 -9 -7 -7 -5 -5 -5 -4 -3 -3 -2 0 1 2 3 3 4 4 4 4 5 5 8 8 8 9 9 10 10 10 10 12 13 14 15 17 18 18 18 22 22 22 23 24 24 25 27 27 28 29 29 29 31 32 33 33 36 36 36 36 37 37 37 38 39 39 40 40 40 41 42 42 43 43 45 47 47 48 48 48 48 49 -20 -20 -20 -19 -18 -18 -18 -18 -17 -17 -16 -16 -16 -16 -15 -15 -15 -14 -10 -9 -7 -7 -5 -5 -5 -4 -3 -3 -2 0 1 2 3 3 4 4 4 4 5 5 8 8 8 9 9 10 10 10 10 12 13 14 15 17 18 18 18 22 22 22 23 24 24 25 27 27 28 29 29 29 31 32 33 33 36 36 36 36 37 37 37 38 39 39 40 40 40 41 42 42 43 43 45 47 47 48 48 48 48 49 -20 -20 -20 -19 -18 -18 -18 -18 -17 -17 -16 -16 -16 -16 -15 -15 -15 -14 -10 -9 -7 -7 -5 -5 -5 -4 -3 -3 -2 0 1 2 3 3 4 4 4 4 5 5 8 8 8 9 9 10 10 10 10 12 13 14 15 17 18 18 18 22 22 22 23 24 24 25 27 27 28 29 29 29 31 32 33 33 36 36 36 36 37 37 37 38 39 39 40 40 40 41 42 42 43 43 45 47 47 48 48 48 48 49

**Output:**

Успешно time: 0 memory: 10320 signal:0

-20 -20 -20 -20 -20 -20 -20 -20 -20 -20 -20 -20 -20 -20 -20 -20 -20 -20 -20 -20 -20 -20 -20 -20 -20 -20 -20 -20 -20 -20 -19 -19 -19 -19 -19 -19 -19 -19 -19 -19 -18 -18 -18 -18 -18 -18 -18 -18 -18 -18 -18 -18 -18 -18 -18 -18 -18 -18 -18 -18 -18 -18 -18 -18 -18 -18 -18 -18 -18 -18 -18 -18 -18 -18 -18 -18 -18 -18 -18 -18 -17 -17 -17 -17 -17 -17 -17 -17 -17 -17 -17 -17 -17 -17 -17 -17 -17 -17 -17 -17 -16 -16 -16 -16 -16 -16 -16 -16 -16 -16 -16 -16 -16 -16 -16 -16 -16 -16 -16 -16 -16 -16 -16 -16 -16 -16 -16 -16 -16 -16 -16 -16 -16 -16 -16 -16 -16 -16 -16 -16 -15 -15 -15 -15 -15 -15 -15 -15 -15 -15 -15 -15 -15 -15 -15 -15 -15 -15 -15 -15 -15 -15 -15 -15 -15 -15 -15 -15 -15 -15 -14 -14 -14 -14 -14 -14 -14 -14 -14 -14 -10 -10 -10 -10 -10 -10 -10 -10 -10 -10 -9 -9 -9 -9 -9 -9 -9 -9 -9 -9 -7 -7 -7 -7 -7 -7 -7 -7 -7 -7 -7 -7 -7 -7 -7 -7 -7 -7 -7 -7 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -4 -4 -4 -4 -4 -4 -4 -4 -4 -4 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -2 -2 -2 -2 -2 -2 -2 -2 -2 -2 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 12 12 12 12 12 12 12 12 12 12 13 13 13 13 13 13 13 13 13 13 14 14 14 14 14 14 14 14 14 14 15 15 15 15 15 15 15 15 15 15 17 17 17 17 17 17 17 17 17 17 18 18 18 18 18 18 18 18 18 18 18 18 18 18 18 18 18 18 18 18 18 18 18 18 18 18 18 18 18 18 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 23 23 23 23 23 23 23 23 23 23 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 25 25 25 25 25 25 25 25 25 25 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 28 28 28 28 28 28 28 28 28 28 29 29 29 29 29 29 29 29 29 29 29 29 29 29 29 29 29 29 29 29 29 29 29 29 29 29 29 29 29 29 31 31 31 31 31 31 31 31 31 31 32 32 32 32 32 32 32 32 32 32 33 33 33 33 33 33 33 33 33 33 33 33 33 33 33 33 33 33 33 33 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 37 37 37 37 37 37 37 37 37 37 37 37 37 37 37 37 37 37 37 37 37 37 37 37 37 37 37 37 37 37 38 38 38 38 38 38 38 38 38 38 39 39 39 39 39 39 39 39 39 39 39 39 39 39 39 39 39 39 39 39 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 41 41 41 41 41 41 41 41 41 41 42 42 42 42 42 42 42 42 42 42 42 42 42 42 42 42 42 42 42 42 43 43 43 43 43 43 43 43 43 43 43 43 43 43 43 43 43 43 43 43 45 45 45 45 45 45 45 45 45 45 47 47 47 47 47 47 47 47 47 47 47 47 47 47 47 47 47 47 47 47 48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 48 49 49 49 49 49 49 49 49 49 49

0.00120 seconds

0.00007 seconds

**Добавление файлов на Github**

### Добавление файлов на Github было произведено с помощью консоли

### Переход на ветку master и создания + переход в новую ветку Gorbunova\_Alina\_lr1 :

git checkout master && git pull origin master

git checkout -b Gorbunova\_Alina\_lr1

* Cозданиe папки :

mkdir Gorbunova\_Alina\_lr1

* Добавлениe файлов, создание коммита и отправка коммитов на GitHub :

git add Gorbunova\_Alina\_lr1/main.c

git commit -m "Gorbunova LR №1 done"

git push origin Gorbunova\_Alina\_lr1

**Вывод:** в ходе лабораторной работы получены навыки использования различных функций стандартных библиотек, изучены и практически применены способы сортировки, такие как: bubble sort(сортировка пузырьком) и quick sort(быстрая сортировка), а также изучены функции работы со временем (clock), а именно подсчет времени работы программы и ее частей. Таким образом, в результате лабораторной работы можно сделать вывод, что метод быстрой сортировки является наиболее эффективным и быстрым методом.