МИНОБРНАУКИ РОССИИ

САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ

ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА
(ЛЕНИНА)

Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе №1

по дисциплине «Программирование»

Тема: Обзор стандартной библиотеки языка Си.

Студент гр. 0382 Тюленев Т.В.

Преподаватель Берленко Т.А.

Санкт-Петербург

2021

Цель работы.

Изучение возможностей стандартной библиотеки языка Си.

Задание.

Напишите 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.

Основные теоретические положения.

stdio.h: стандартная библиотека ввода-вывода.

stdlib.h:

- Функции для сортировки и поиска.
- qsort:

qsort(&x2, 1000, sizeof(int), sort);

1. Функция принимает указатель на начальный элемент массива
2. Количество элементов
3. Размер одного элемента
4. Указатель на функцию для сравнения двух элементов

time.h: заголовочный файл стандартной библиотеки языка программирования Си, содержащий типы и функции для работы с датой и временем.

Ход работы:

- Заполняем массив целых чисел длины 1000.
- Определяем время начала выполнения первого метода сортировки. С помощью команды *clock()* записываем количество тактов процессора с начала выполнения программы в переменную *t1*.
- Выполняем сортировку методом пузырька.
- Определяем время конца выполнения первого метода сортировки. С помощью команды clock() и вычитания количества тактов на начало выполнения сортировки узнаем за кое время выполнилась сортировка и с помощью команды *CLOCKS_PER_SEC* переводим такты в секунды.
- Определяем время начала выполнения первого метода сортировки. С помощью команды *clock()* записываем количество тактов процессора с начала выполнения программы в переменную *t2*.
- Выполняем быструю сортировку *qsort*.
- Определяем время конца выполнения первого метода сортировки. С помощью команды clock() и вычитания количества тактов на начало выполнения сортировки узнаем за кое время выполнилась сортировка и с помощью команды *CLOCKS_PER_SEC* переводим такты в секунды.
- Выводим получившейся массив и время, которое потребовалось для его сортировки двумя способами.

Тестирование.

Результаты тестирования представлены в табл. 1 для разной длины массива и случайном его заполнении числами со значением до 100.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные	Комментарии
1.	83 86 77 15 93 35 86 92 49 21 62 27 90 59 63 26 40 26 72 36 11 68 67 29 82 30 62 23 67 35 29 2 22 58 69 67 9356 11 42 29 73 21 19 84 37 98 24 15 (50 элементов)	0 2 11 11 15 15 19 21 21 22 23 24 26 26 27 29 29 29 30 35 35 36 37 40 42 49 56 58 59 62 62 63 67 67 67 68 69 72 73 77 82 83 84 86 86 90 92 93 93 0.000007 0.000003	Программа работает верно
2.	83 86 77 15 93 35 86 92 49 21 62 27 90 59 63 26 40 26 72 36 11 68 67 29 82 30 62 23 67 35 29 2 22 58 69 67 93 56 11 42 29 73 21 19 84 37 98 24 15 70 13 26 91 80 56 73 62 70 96 81 5 25 84 27 36 5 46 29 13 57 24 95 82 45 14 67 34 64 43 50 87 8 76 78 88 84 3 51 54 99 32 60 76 68 39 12 26 86 94 39 (100 элементов)	0 2 3 5 5 8 11 11 12 13 13 14 15 15 19 21 21 22 23 24 24 25 26 26 26 26 27 27 29 29 29 29 30 32 34 35 35 36 36 37 39 39 40 42 43 45 46 49 50 51 54 56 56 57 58 59 60 62 62 62 63 64 67 67 67 67 67 68 68 69 70 70 72 73 73 76 76 77 78 80 81 82 82 83 84 84 84 86 86 86 87 88 90 91 92 93 93 94 95 96 98 0.000022 0.000004	Программа работает верно
3.	83 -14 77 15 93 35 86 -8 -51 -79 62 -73 -10 -41 63 26 40 -74 72 36 -89 68 67 -71 82 30 -38 23 -33 35 29 -98 -78 -42 -31 67 93 -44 -89 -58 -71 73 -79 19 84 37 98 24 15 70 -87 26 -9 80 56 -27 -38 70 96 -19 5 25 -16 27 36 5 -54 29 13 -43 24 -5 82 45 -86 67 -66 64 -57 50 -13 -92 -24 78 88 84 -97 -49 54 99 32 -40 -24 68 39 -88 -74 86 -6 39 (100 элементов с двойной точностью)	-98 -97 -92 -89 -89 -88 -87 -86 -79 -79 -78 -74 -74 -73 -71 -71 -66 -58 -57 -54 -51 -49 -44 -43 -42 -41 -40 -38 -38 -33 -31 -27 -24 -24 -19 -16 -14 -13 -10 -9 -8 -6 -5 0 5 5 13 15 15 19 23 24 24 25 26 26 27 29 29 30 32 35 35 36 36 37 39 39 40 45 50 54 56 62 63 64 67 67 67 68 68 70 70 72 73 77 78 80 82 82 83 84 84 86 86 88 93 93 96 98 0.000055 0.000006	Программа работает верно
4.	(1000 элементов с	0,001743	Программа работает
	двойной точностью)	0,000069	верно

Выводы.

Был изучен и освоен функционал стандартной библиотеки языка программирования Си.

Разработана программа, принимающая на вход массив целых чисел и сортирующая его двумя методами: быстрой сортировкой и «методом пузырька». Для обоих случаев было определено время выполнения.

По данным экспериментального анализа можно сделать вывод что быстрая сортировка занимает меньше времени, чем сортировка методом пузырька. Вследствие этого можно сказать, что быстрая сортировка эффективнее, чем сортировка методом пузырька.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.c

```
#include <stdio.h>
#include <time.h>
#include <stdlib.h>

int sort(const void *x1, const void *x2){
return ( *(int*)x1 - *(int*)x2 );}

int main (void){
 int i,j,x1[1000],x2[1000],xx,end;
 float t1,t2;

 // for (i=0;i<1000;i++){x1[i]=rand()%200-100;x2[i]=x1[i];}
 for (i=0;i<1000;i++){scanf("%i",&x1[i]);x2[i]=x1[i];}
 // for (i=0;i<1000;i++){printf("%i ",x1[i]);}printf("\n\n");

 t1=clock();
 for (int i = 1000; i >= 0; i--){
   end = 0;
   for (int j = 0; j < i; j++){
      if (x1[j] > x1[j + 1]){
         xx = x1[j];
         x1[j] = x1[j + 1];
         x1[j + 1] = xx;
         end++;}}
   if (end == 0){break;}}
 t1=(clock()-t1)/CLOCKS_PER_SEC;


 t2=clock();
 qsort(&x2, 1000, sizeof(int), sort);
 t2=(clock()-t2)/CLOCKS_PER_SEC;

 for (i=0;i<1000;i++){printf("%i ",x1[i]);}
 printf("\n%f\n%f",t1,t2);
  return 0;
}
```