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

Студент гр. 2384 _____ Кузьминых Е.М

Преподаватель _____ Гаврилов А.В.

Санкт-Петербург
2023

Цель работы.

Изучить работу функций из стандартной библиотеки языка программирования Си.

Задачи.

Вариант №3

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.

Выполнение работы.

В начале программы мы инициализируем массив *arr* чисел размером 1000, так-как знаем количество чисел, которое должна считать программа. С помощью цикла *for* и функции *scanf* мы сохраняем каждое число в массив *arr*. После создадим копию этого массива (инициализируем массив чисел *arr_qsort*

и с помощью функции *memcpy* скопируем в него все числа, хранящиеся в массиве *arr*).

Измерять время выполнения сортировок мы будем при помощи функций стандартной библиотеки *time.h,* создадим переменную *time_spend1* типа *double,* которая будет хранить в себе время работы цикла, выполняющего сортировку пузырьком и зададим ей стартовое значение 0.0, а также создадим переменную *start1* типа *clock_t,* и сохраним в нее значение вызова функции *clock(),* т.е. сохраним время начала работы для сортировки пузырьком массива arr.

После этого напишем сортировку пузырьком с помощью двух циклов, перебирающих все возможные пары элементов в массиве. Если предыдущий элемент (*arr[i]*) больше следующего (*arr[j]*), то мы меняем их местами, путем создания переменной c, в которую мы сохраняем значение *arr[j]*, *arr[j]* делаем равным *arr[i]*, а *в arr[i*] записываем переменную c. В результате этих действий на месте i-ого элемента массива находится j-ый элемент, а на месте j-ого элемента – i-ый элемент.

После цикла сортировки пузырьком создадим переменную *end1* типа *clock_t,* в которую сохраним вызов функции *clock(),* сохранив время окончания работы цикла. Вычислим время работы цикла по формуле: *(end1-start1)/CLOCK_PER_SEC*. С помощью нее мы получим время работы в секундах, значение сохраняем в *time_spend1*.

После делаем аналогичные действия для вычисления работы функции *qsort(),* создаем переменные *time_spend2* типа *double*, *start2* типа *clock_t,* в которую сохраняем результат работы функции *clock()* – начало работы функции *qsort().*

Для работы функции *qsort()* напишем свой компаратор для сравнения элементов массива. Создадим функцию comp, принимающую на вход два аргумента *s1* и *s2* типа *const void**, в функции мы приводим их к типу *int* и возвращаем их разницу. Если *s1>s2,* то возвращаемое значение будет больше нуля, при равенстве будет возвращен 0, если *s1<s2* будет возвращено отрицательное значение. После вызовем функцию *qsort(),* передав ей массив

для сортировки (*arr_qsort*), количество элементов в массиве (1000), размер типа данных, которые хранятся в массиве (*sizeof(int)*) и саму функцию компаратор (*comp*). После вызова функции создадим переменную clock2, в которой сохраним время окончания работы функции *qsort()* и найдем ее время работы также, как и для сортировки пузырьком.

Затем остается вывести отсортированные элементы массива с помощью цикла *for* и значения времени, за которое произошла сортировка массива пузырьком и с помощью алгоритма быстрой сортировки (функции *qsort()* из стандартной библиотеки).

Тестирование.

№	Входные данные	Выходные данные	Комментарий
1	1000 псевдо-случайных чисел от 0 до 1000	*Отсортированный массив из 1000 чисел* 0.003394 0.000095	Ответ верный
2	50 псевдо-случайных чисел от 0 до 1000	1 36 104 159 183 184 198 219 220 241 287 294 304 356 369 378 380 387 394 395 418 464 491 516 521 525 582 583 623 641 648 653 667 692 702 703 724 729 743 765 796 802 887 897 901 909 969 971 979 992 0.000012 0.000005	Ответ верный

Вывод.

В ходе выполнения лабораторной работы была написана программа, считывающая массив чисел длиной 1000 и вычисляющая время, за которое произойдет сортировка массива с помощью алгоритма быстрой сортировки (функции *qsort()*) и сортировка пузырьком.

Приложение А.

Исходный код программы.

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <string.h>
#define N 1000
int comp(const void* s1,const void*s2){
	return *(int*)s1-*(int*)s2;
}



int main(){
	int arr[N];
	for (int i = 0; i < N; i++) {
		scanf("%d",&arr[i]);
	}
	int arr_qsort[N];
	memcpy(arr_qsort,arr,sizeof (arr));
	double time_spend1=0.0;
	clock_t start1 = clock();
	for (int i = 0; i < N; i++) {
		for (int j = 0; j < N; j++) {
			if(arr[i]>arr[j]){
				int c=arr[j];
				arr[j]=arr[i];
				arr[i]=c;
			}
		}
	}

	clock_t end1 = clock();
	time_spend1+=(double )(end1-start1)/CLOCKS_PER_SEC;
	double time_spend2=0.0;
	clock_t start2=clock();
	qsort(arr_qsort,N,sizeof(int),comp);
```

```
        clock_t end2=clock();
        time_spend2+=(double )(end2-start2)/CLOCKS_PER_SEC;
        for (int i = 0; i < N; i++) {
             printf("%d ",arr_qsort[i]);
        }
        printf("\n%lf\n",time_spend1);
        printf("%lf",time_spend2);
        return 0;

}
```