МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе №4

по дисциплине «Программирование»

Тема: Обзор стандартной библиотеки

Студент гр. 8303 _____ Гришин К. И.

Преподаватель _____ Чайка К. В.

Санкт-Петербург

2018

Цель работы

Изучить функционал таких стандартных библиотек как "stdlib.h" и "time.h", научиться использовать как собственные алгоритмы сортировки, так и функцию сортировки стандартной библиотеки, а так же измерять время работы той или иной функции.

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.

Ход работы

Подключаются стандартные библиотеки:

- *«stdio.h»* — для ввода и вывода.
- *«stdlib.h»* — для использования функции *«qsort»*.
- *«time.h»* — для использования функции *«clock()»*.

Определяется идентификатор *«SZ»*, обозначающий количество входных данных. Данный идентификатор нужен для изменения количества входных данных по надобности, например для тестирования программы при малом количестве вводимых данных.

Объявляются функции *«bubbleSort»* – для сортировки переданного ей массива методом *„пузырька“*, а также подсчета времени за которое эта сортировка произошла; функции *«quickSort»* – для сортировки переданного ей массива методом быстрой сортировки, а также подсчета времени, за которое эта сортировка произошла; функции *«comp»* – которая используется как компоратор для сравнения двух чисел.

В теле функции *«main»* создается два челочисленных массива, размера *«SZ»*, *«nums1»* и *«nums2»*, затем происходит считывание каждого элемента массива. Далее путем использования функций *«bubbleSort»* и *«quickSort»* происходит сортировка массива а так же подсчет времени, за которое была совершена сортировка, времена записаны в переменные *«bubbleTime»* и *«quickTime»* соответственно. Затем вывод отсортированного массива а также времен за которое были совершены сортировки.

Определение функции *«bubbleSort»* – создается цикл, который определяет область сортировки с каждой итеррацией, уменьшая ее постоянно на одно значение справа. Далее идет цикл отвечающий за „всплывание“ самых больших чисел на текущей итеррации. После того, как область сортировки станет равна нулю, цикл завершается и на выход подается отсортированный массив. Также используется функция *«clock()»* до и после цикла для подсчета времени.

Определение функции *«quickSort»* – данная функция использует функцию *«qsort»* стандартной библиотеки, а также считает время за которое была совершена сортировка с помощью функции *«clock()»*.

Выводы.

В ходе выполнения работы были изучены стандартные библиотеки «stdlib.h» и *«time.h»*, а также написана программа на языке С, которая сортирует введенные числа, а также выводит время за которое отработала каждая из сортировок. Исходя из вывода программы можно заметить, что функция «qsort» зачастую быстрее, а на большом наборе данных гораздо быстрее, чем написанная сортировка *„пузырьком“*, из чего следует вывод, что использование сортировки стандартной библиотеки более рационально, чем написание стандартной функции.

ПРИЛОЖЕНИЕ

КОД ПРОГРАММЫ

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

#define SZ 1000

float bubbleSort(int* nums, int size);
float quickSort(int* nums, int size);
int comp(const void* int1, const void* int2);

int main(){
    int nums1[SZ];
    int nums2[SZ];
    for(size_t i = 0; i < SZ; i++){ scanf("%d", &nums1[i]); nums2[i] = nums1[i];}
    float bubbleTime = bubbleSort(nums1, SZ);
    float quickTime = quickSort(nums2, SZ);
    for(size_t i = 0; i < SZ; i++) printf("%d ", nums1[i]);
    printf("\n%f\n", bubbleTime);
    printf("%f\n", quickTime);
    return 0;
}


float bubbleSort(int* nums, int size){
    clock_t start = clock();

    for(int i = size - 1; i >= 0; i--)
        for(int j = 0; j < i; j++)
            if (nums[j] > nums[j+1]){
                int t = nums[j+1];
                nums[j+1] = nums[j];
                nums[j] = t;
            }

    clock_t end = clock();
    return (float)(end - start) / CLOCKS_PER_SEC;
}

float quickSort(int *nums, int size){
    clock_t start = clock();

    qsort(nums, size, sizeof(int), comp);

    clock_t end = clock();
    return (float)(end - start) / CLOCKS_PER_SEC;
}

int comp(const void* int1,const void* int2){
    return *(int*)int1 - *(int*)int2;
}
```