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.

|  |  |  |
| --- | --- | --- |
| Студент гр. 6303 |  | Жахин А. А. |
| Преподаватель |  | Чайка К. В. |

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 две библиотеки: «*stdio.h*» и «*time.h*». А также, для удобства, число 1000 заменяется буквой «*N*»

|  |
| --- |
| #include <stdio.h>  #include <time.h>  #define N 1000 |

* Далее, до функции *main*, объявляется функция сравнения *cmp*:

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

* 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 arr\_puz[N]; // массив для метода пузырька  int arr\_qsort[N]; // массив для быстрой сортировки  int i = 0;  int j = 0;  for (i = 0; i<1000; i++)  scanf("%d", &arr\_puz[i]); //запишем входные данные в один из массивов  for (i = 0; i<1000; i++)  {  arr\_qsort[i] = arr\_puz[i]; //скопируем один массив в другой  } |

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

|  |
| --- |
| clock\_t t\_puz;  t\_puz = clock(); //засекаем начальное время  for(i = 0 ; i < N - 1; i++)  {  // сравниваем два соседних элемента.  for(j = 0 ; j < N - i - 1 ; j++)  {  if(arr\_puz[j] > arr\_puz[j+1])  {  // если они идут в неправильном порядке, то меняем их местами.  int tmp = arr\_puz[j];  arr\_puz[j] = arr\_puz[j+1];  arr\_puz[j+1] = tmp;  }  }  }  t\_puz = clock() - t\_puz; //вычисляем время сортировки пузырьком |

* Далее необходимо вызвать функцию быстрой сортировки *qsort()* стандартной библиотеки и засечь время ее выполнения с помощью функции *clock*, описанной в предыдущем пункте.

|  |
| --- |
| clock\_t t\_qsort;  t\_qsort = clock(); // засекаем начальное время  qsort(arr\_qsort, N, sizeof(int), cmp ); //выполняем быструю сортировку  t\_qsort = clock() - t\_qsort; // вычисляем время быстрой сортировки |

* Затем выводится один из отсортированных массив с элементами, разделенными пробелами, а также выводится время «сортировки пузырьком» и время быстрой сортировки.

|  |
| --- |
| for(i = 0 ; i < N; i++) // выводим отсортированный массив  {  printf("%d ", arr\_qsort[i]);  }  printf("\n");  printf("%f\n", t\_puz);  printf("%f\n", t\_qsort);  return 0; |

**Вывод:** в ходе лабораторной работы получены навыки использования различных функций стандартных библиотек.