МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

НОВОСИБИРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Факультет информационных технологий

Кафедра параллельных вычислений

ОТЧЕТ

О ВЫПОЛНЕНИИ ЛАБОРАТОРНОЙ РАБОТЫ

«ОПРЕДЕЛЕНИЕ ВРЕМЕНИ РАБОТЫ ПРИКЛАДНЫХ ПРОГРАММ»

студента 2 курса, 23203 группы

Князькова Кирилла Вячеславовича

Направление 09.03.01 – «Информатика и вычислительная техника»

Преподаватель:
А.Ю. Власенко

Новосибирск 2024

СОДЕРЖАНИЕ

ЦЕЛИ

1. Изучение методов и способов измерения времени работы подпрограммы.
2. Изучение приемов уменьшения влияния посторонних факторов на время выполнения подпрограммы для повышения точности измерений
3. Практическое применение перечисленных выше навыков в прикладной программе.

ЗАДАНИЕ

1. Написать программу на языке C или C++, которая реализует выбранный алгоритм из задания.
2. Проверить правильность работы программы на нескольких тестовых наборах входных данных
3. Выбрать значение параметра N таким, чтобы время работы программы было порядка 15 секунд.
4. По приведенной методике определить время работы подпрограммы тестовой программы с относительной погрешностью не более 1%.
5. Составить отчет по лабораторной работе.

ОПИСАНИЕ РАБОТЫ

В начале работы был реализован алгоритм сортировки методом пузырька [1]. Также была проведена проверка на корректность программы, реализующей данный алгоритм, с помощью разных тестовых входных данных.

Следующим этапом, был добавлен функционал замера времени исполнения получившейся подпрограммы. Для этого был использован таймер времени процесса. Время измерялось 5 раз, результат - среднее и минимальное время.

```
krillin@t14s:~/code/nsu/evm$ ./bubble 88000
#1 time taken = 22.410000s
#2 time taken = 15.480000s
#3 time taken = 14.840000s
#4 time taken = 15.220000s
#5 time taken = 15.590000s
average time = 16.708000
minimal time = 14.840000
```

Последним шагом я измерил время работы всего приложения с помощью утилиты time.

```
krillin@t14s:~/code/nsu/evm$ time ./bubble 88000

real    0m22.619s
user    0m22.459s
sys     0m0.008s
```

ЗАКЛЮЧЕНИЕ

В ходе работы я ознакомился на практике с процедурой измерения времени работы программы, а также с приемами уменьшения влияния посторонних факторов на время выполнения прикладной программы.

Приложение 1. *Исходный код прикладной программы на “C”*

```c
#include <stdio.h>     // for printf
#include <stdlib.h>    // for rand
#include <sys/times.h> // for times
#include <unistd.h>    // for sync and sysconf

int randomGen(int lower, int upper) {
 return (rand() % (upper + 1 - lower)) + lower;
}

void fillRandArray(long n, int *arr) {
 for (long i = 0; i < n; ++i) {
   arr[i] = randomGen(1, 10000);
 }
}

void swap(int *a, int *b) {
 int temp = *a;
 *a = *b;
 *b = temp;
}

void bubbleSort(int arr[], long n) {
 char isSwapped;
 for (long i = 0; i < n; i++) {
   isSwapped = 0;
   for (int j = 0; j < n - i - 1; j++) {
     if (arr[j] > arr[j + 1]) {
       swap(&arr[j], &arr[j + 1]);
```

```
            isSwapped = 1;
        }
      }
      if (!isSwapped)
        break;
    }
  }

  int main(int argc, char **argv) {
    sync();
    long N = atol(argv[1]);
    int iterations = 5;
    double total_time = 0.0;
    double min_time = 100.0;

    struct tms start, end;
    long clocks_per_sec = sysconf(_SC_CLK_TCK);
    long clocks;

    srand(0);

    for (int i = 1; i <= iterations; i++) {
      int *arr = malloc(sizeof(int) * N);
      fillRandArray(N, arr);

      times(&start);
      bubbleSort(arr, N);
      times(&end);
```

```
        clocks = end.tms_utime - start.tms_utime;

        double clocks_to_sec = (double)clocks / clocks_per_sec;

        total_time += clocks_to_sec;

        min_time = (clocks_to_sec < min_time) ? clocks_to_sec :
min_time;

        printf("#%d time taken = %lfs\n", i, clocks_to_sec);

        free(arr);

      }


     printf("average time = %f\n", total_time / iterations);

     printf("minimal time = %f\n", min_time);


     return EXIT_SUCCESS;

    }
```