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 основных функций, оптимизирующих преобразований и уровней оптимизации. А также анализ их влияния на время исполнения программы.
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 от 30 до 60 секунд.
4. Скомпилировать программу с использованием разных уровней оптимизации под архитектуру x86.
5. Для каждого варианта компиляции измерить время работы программы при нескольких значениях N.
6. 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. Использовался таймер времени процесса. Время измерялось перед началом и концом работы функции перестановки. Разность конца и начала давала общее время исполнения подпрограммы. Был подобран аргумент N так, чтобы время работы было от 30 до 60 секунд.

Далее, программа была скомпилирована с разными уровнями оптимизации (используя ключи компилятора). Этот процесс был автоматизирован с помощью bash-скрипта [2]. Результаты работы программы были внесены в таблицу, по таблице был сделан график, для визуализации данных [3].

ЗАКЛЮЧЕНИЕ

В ходе работы я ознакомился на практике с процедурой измерения времени работы программы, а также с разными уровнями оптимизации, которые на это время влияют

Приложение 1. *Исходный код прикладной программы на “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 long N = atol(argv[1]);

    struct tms start, end;
    long clocks_per_sec = sysconf(_SC_CLK_TCK); // ticks
per sec
    long clocks;

    srand(0);

    int *arr = (int*)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;

    printf("Time taken = %lfs\n", clocks_to_sec);

    free(arr);


    return EXIT_SUCCESS;

   }
```

Приложение 2. *Bash-скрипт*

```
#/bin/bash

LEVELS_OPTIMIZATION=("-O0" "-O1" "-O2" "-O3" "-Os" "-Ofast" "-Og")

SOURCE_F="bubble.c"

N_LIST=("75000" "100000" "140000")

OUTPUT_F="results.csv"
echo "Lv. of optimization, N value, Time" > $OUTPUT_F

for OPT_LEVEL in "${LEVELS_OPTIMIZATION[@]}"; do
   gcc $OPT_LEVEL $SOURCE_F -o bubble${OPT_LEVEL}
done

for OPT_LEVEL in "${LEVELS_OPTIMIZATION[@]}"; do
   for N_VALUE in "${N_LIST[@]}"; do
       EXE_TIME=$(./bubble${OPT_LEVEL} $N_VALUE | awk '{print $NF}')
       echo "$OPT_LEVEL, $N_VALUE, $EXE_TIME" >> $OUTPUT_F
   done
```

```
done

for OPT_LEVEL in "${LEVELS_OPTIMIZATION[@]}"; do
   rm bubble${OPT_LEVEL}
done

echo "End of script execution. Results in $OUTPUT_F"
exit
```

Приложение 3. *График времени работы подпрограммы в зависимости от исходных данных и уровня оптимизации.*