Министерство образования и науки Российской Федерации

Новосибирский национальный исследовательский государственный университет

ЭВМ и периферийные устройства

Отчет по лабораторной работе № 1

«ОПРЕДЕЛЕНИЕ ВРЕМЕНИ РАБОТЫ ПРИКЛАДНЫХ

ПРОГРАММ»

Студент: Трусов Тимофей Алексеевич, 24205

Преподаватель: Мичуров Михаил Антонович

Новосибирск, 2025 г.

## Цель работы

1. Изучение методики измерения времени работы подпрограммы.

2. Изучение приемов повышения точности измерения времени работы

подпрограммы.

3. Изучение способов измерения времени работы подпрограммы.

4. Измерение времени работы подпрограммы в прикладной программе.

## Задание

Написать алгоритм сортировки методом пузырька и измерить его скорость работы при заданных N с относительной погрешностью не более 1%.

## Описание работы

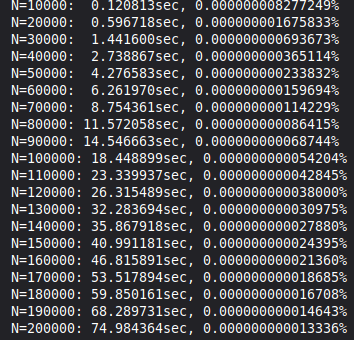
Был написан алгоритм сортировки метдом пузырька на языке С и проверен на корректность.

Была выбрана библиотечная функция «clock\_gettime» для измерения времени работы программы, поскольку она позволяет замерять время на выбранном участке кода, поэтому инициализация программы и генерация массива случайных чисел не учитывается во время работы алгоритма.

Был написан bash скрипт для тестирования алгоритма с разными N, были проведены тесты

Далее приведены результаты из терминала, где первая колонка — размер отсортированного массива, вторая — время в секундах, затраченное на сортировку, третья — относительная погрешность, учитывая, что абсолютная на Linux равняется 1нс.

А также приведена диаграмма, показывающая зависимость времени от количества элементов в массиве.



## 4. Приложения

Код:

#include <stdlib.h>

#include <stdio.h>

#include <time.h>

void swap(int\* a, int\* b) {

int tmp = \*a;

\*a = \*b;

\*b = tmp;

}

void bubbleSort(int\* arr, size\_t len) {

for(size\_t i = 0; i < len - 1; ++i) {

for(int j = 0; j < len - i - 1; ++j) {

if(arr[j] > arr[j + 1]) swap(&arr[j], &arr[j + 1]);

}

}

}

int main(int argc, char\* argv[]) {

if(argc != 2) {

printf("Не один аргумент\n");

return 1;

}

int number = atoi(argv[1]);

if(number <= 0) {

printf("Аргумент должен быть натуральным числом\n");

return 1;

}

int\* array = (int\*)malloc(number \* sizeof(int));

if(array == NULL) {

printf("Не удалось выделить память\n");

return 1;

}

for(int i = 0; i < number; ++i) {

array[i] = rand();

}

struct timespec start, end;

clock\_gettime(CLOCK\_MONOTONIC\_RAW, &start);

bubbleSort(array, (size\_t)number);

clock\_gettime(CLOCK\_MONOTONIC\_RAW, &end);

double delta = (end.tv\_sec - start.tv\_sec + 0.000000001 \* (end.tv\_nsec - start.tv\_nsec));

printf("%9lfsec, %.15lf%%\n",delta,(1 / (delta \* 1000000000)));

free(array);

return 0;

}

Bash-скрипт запуска:

#!/bin/bash

gcc main.c -o BubbleSort -lrt

step=10000

for (( i=1; i<=20; i++))

do

N=$((i \* step))

echo -n " N=$N: "

./BubbleSort $N

done

echo "готово"

Репозиторий: https://github.com/TrusovTimofey/24205-Trusov-ECM

5. Вывод

Ознакомились с методами измерения скорости программы на практике, поняли их отличия, основываясь на которых, узнали, где какой метод подходит больше.