Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

**Лабораторная работа №3 по курсу**

**«Операционные системы»**

Студент: Казацнев Данила Игоревич

Группа: М8О-207Б-21

Вариант: 1

Преподаватель: Миронов Евгений Сергеевич

Оценка: \_\_\_\_\_\_\_\_\_\_\_

Дата: \_\_\_\_\_\_\_\_\_\_\_

Подпись: \_\_\_\_\_\_\_\_\_\_\_

Москва, 2022

**Содержание**

1. Репозиторий
2. Постановка задачи
3. Общий метод и алгоритм решения
4. Исходный код
5. Демонстрация работы программы
6. Выводы

**Репозиторий**

https://github.com/thgdanilaya/mai\_os\_labs/lab2

**Постановка задачи**

**Цель работы**

Целью является приобретение практических навыков в:

Управление потоками в ОС

Обеспечение синхронизации между потоками

**Задание**

Отсортировать массив целых чисел при помощи битонической сортировки

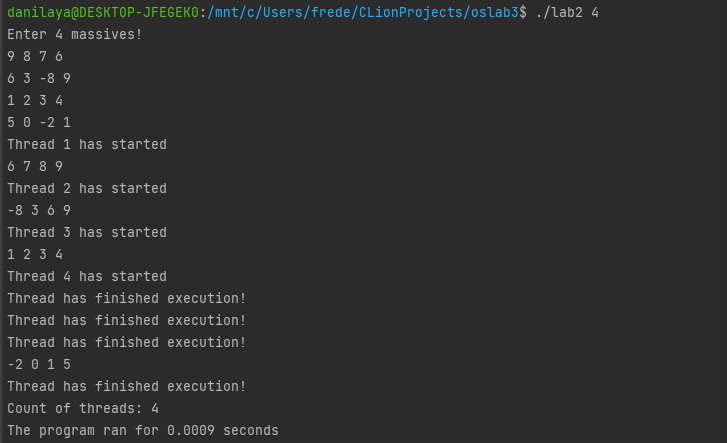
**Общий метод и алгоритм решения**

На вход подается кол-во потоков, потом такое же кол-во массивов чисел(строго степень двойки). Дальше сортируется методом битоннической сортировки

**Исходный код**

#include "time.h"  
#include "stdio.h"  
#include "stdlib.h"  
#include "pthread.h"  
  
#define MAX 10  
  
int up = 1;  
int down = 0;  
  
void sswap(int \*num1, int \*num2) {  
 int temp;  
  
 temp = \*num1;  
 \*num1 = \*num2;  
 \*num2 = temp;  
}  
  
void compare(int \*arr, int i, int j, int dir) {  
 if (dir == (arr[i] > arr[j])) {  
 sswap(&arr[i], &arr[j]);  
 }  
}  
  
void bitonicmerge(int \*arr, int low, int c, int dir) {  
 int k;  
 int i;  
  
 if (c > 1) {  
 k = c / 2;  
 i = low;  
 while (i < low + k) {  
 compare(arr, i, i + k, dir);  
 i++;  
 }  
  
 bitonicmerge(arr, low, k, dir);  
 bitonicmerge(arr, low + k, k, dir);  
 }  
}  
  
void recbitonic(int \*arr, int low, int v, int dir) {  
 int k;  
  
 if (v > 1) {  
 k = v / 2;  
 recbitonic(arr, low, k, up);  
 recbitonic(arr, low + k, k, down);  
 bitonicmerge(arr, low, v, dir);  
 }  
}  
  
  
void \*routine(void \*arg) {  
 int \*mass = (int \*) arg;  
 recbitonic(mass, 0, MAX, up);  
 for (int i = 0; i < MAX; i++)  
 printf("%d ", mass[i]);  
 printf("\n");  
 free(arg);  
 return NULL;  
}  
  
int main(int argc, char \*argv[]) {  
 int n = atoi(argv[1]);  
 pthread\_t pid[n];  
  
 int data[n][MAX];  
 printf("Enter %d massives!\n", n);  
 for (int i = 0; i < n; i++) {  
 for (int u = 0; u < MAX; u++)  
 scanf("%d", data[i] + u);  
 }  
  
 double start = clock();;  
  
 if (argc == 2) {  
 for (int i = 0; i < n; i++) {  
 int \*data2 = malloc(sizeof(int) \* MAX);  
 for (int u = 0; u < MAX; u++) {  
 data2[u] = data[i][u];  
 }  
 if (pthread\_create(&pid[i], NULL, &routine, data2) != 0) {  
 perror("Couldn't create a thread\n");  
 return 1;  
 }  
 printf("Thread %d has started\n", i + 1);  
 }  
 for (int i = 0; i < n; i++) {  
 if (pthread\_join(pid[i], NULL) != 0) {  
 return 2;  
 }  
 printf("Thread has finished execution!\n");  
 }  
 } else {  
 printf("Please enter an appropriate program key !\n");  
 }  
 printf("Count of threads: %d\n", n);  
 printf("The program ran for %.4lf seconds\n", (clock() - start) / (CLOCKS\_PER\_SEC));  
 return 0;  
}

**Демонстрация работы программы**



**Выводы**

Я приобрел навыки в управлении потоками и обеспечении синхронизации между потоками.