Московский Авиационный Институт

(Национальный Исследовательский Университет)

Кафедра 806 «Вычислительная информатика и программирование»

Факультет: «Информационные технологии и прикладная математика»

Лабораторная работа

Дисциплина: «Операционные системы»

I I семестр

Задание 3

|  |  |
| --- | --- |
| Группа: | М8О-208Б-18, №9 |
| Студент: | Игитова Александра Андреевна |
| Преподаватель: | Миронов Евгений Сергеевич |
| Оценка: |  |
| Дата: | 29.11.2019 |

Москва, 2019

1. **Задание**

(*вариант № 8*): 8. Произвести перемножение 2-ух матриц, содержащих комплексные числа.

1. **Код программы на С**

main.c

#include <pthread.h>

#include <stdlib.h>

#include <stdio.h>

#include <unistd.h>

#include "rw.h"

pthread\_mutex\_t mutex = PTHREAD\_MUTEX\_INITIALIZER;

//структура для данных потока

typedef struct{

int len; //length of line

struct Complex \*\*A, \*\*B, \*\*Res; //pointers of matrixs

int line, column;

} pthrData;

struct Complex multiplyComplex(struct Complex a, struct Complex b){

struct Complex c;

c.re = a.re\*b.re - a.im\*b.im;

c.im = a.im\*b.re + b.im\*a.re;

return c;

}

void\* threadFunc(void\* thread\_data){

pthrData \*data = (pthrData \*) thread\_data;

data->Res[data->line][data->column].re = 0;

for(int i = 0; i < data->len; i++){

data->Res[data->line][data->column].re += multiplyComplex(data->A[data->line][i], data->B[i][data->column]).re;

data->Res[data->line][data->column].im += multiplyComplex(data->A[data->line][i], data->B[i][data->column]).im;

}

return (void \*)data;

}

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

int m, n, p, q; //размерности матриц

writeString("Добро пожаловать в программу по перемножению 2-х матриц с комплексными элементами.\n");

writeString("Введите размерность (m x n) 1-й матрицы.\n");

writeString("m = ");

m = readInt(); if(m <= 0) error("m > 0", -1);

writeString("n = ");

n = readInt(); if(n <= 0) error("n > 0", -2);

writeString("Введите размерность (p x q) 2-й матрицы.\n");

writeString("p = ");

p = readInt(); if(p <= 0) error("p > 0", -3);

if(p != n) error("Матрицы не сопряжены. Перемножение невозможно.", -4);

writeString("q = ");

q = readInt(); if(q <= 0) error("q > 0", -5);

int maxCountThread = 0;

if (argc >= 2){

maxCountThread = atoi(argv[1]);

}

struct Complex \*\*A = (struct Complex\*\*) malloc(m \* sizeof(struct Complex\*));

for (int i = 0; i < m; ++i)

A[i] = (struct Complex\*) malloc(n \* sizeof(struct Complex));

struct Complex \*\*B = (struct Complex\*\*) malloc(p \* sizeof(struct Complex\*));

for (int i = 0; i < p; ++i)

B[i] = (struct Complex\*) malloc(q \* sizeof(struct Complex));

struct Complex \*\*Res = (struct Complex\*\*) malloc(m \* sizeof(struct Complex\*));

for (int i = 0; i < m; ++i)

Res[i] = (struct Complex\*) malloc(q \* sizeof(struct Complex));

writeString("Введите элементы 1-й матрицы:\n");

for(int i = 0; i < m; i++)

for(int j = 0; j < n; j++)

A[i][j] = readComplex();

writeString("Введите элементы 2-й матрицы:\n");

for(int i = 0; i < p; i++)

for(int j = 0; j < q; j++){

B[i][j] = readComplex();

}

//\* вывод матриц

writeString("Введёная мaтрица1:\n");

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

for (int j = 0; j < n; j++) { writeComplex(A[i][j]); writeChar(' '); }

writeChar('\n');

}

writeString("Введёная мaтрица2:\n");

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

for (int j = 0; j < q; j++) { writeComplex(B[i][j]); writeChar(' '); }

writeChar('\n');

}

//\*/

//выделяю память под массив идентификаторов потоков

pthread\_t\* threads = (pthread\_t\*) malloc(m\*q \* sizeof(pthread\_t));

//сколько потоков - столько и структур с потоковых данных

pthrData\* threadData = (pthrData\*) malloc(m\*q \* sizeof(pthrData));

//запуск потоков

int status, jj = -1;

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

for(int j = 0; j < q; j++){

threadData[i\*n + j].len = n;

threadData[i\*n + j].A = A;

threadData[i\*n + j].B = B;

threadData[i\*n + j].Res = Res;

threadData[i\*n + j].line = i;

threadData[i\*n + j].column = j;

status = pthread\_create(&(threads[i\*n + j]), NULL, threadFunc, (void \*) &threadData[i\*n + j]);

if(status != 0){

error("ошибка создания потока\n",-5);

}

}

}

//ожидаем выполнение всех потоков

for(int i = 0; i < m\*q; i++)

pthread\_join(threads[i], NULL);

//вывод результата

writeString("Результирующая матрица:\n");

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

for(int j = 0; j < q; j++){

writeComplex(threadData[i\*n + j].Res[i][j]);

writeChar(' ');

}

writeChar('\n');

}

for(int i = 0; i < m; ++i)

free(A[i]);

free(A);

for(int i = 0; i < q; ++i)

free(B[i]);

free(B);

for(int i = 0; i < m; ++i)

free(Res[i]);

free(Res);

free(threads);

free(threadData);

pthread\_mutex\_destroy(&mutex);

return 0;

}

Rw.h

#define BS 256

struct Complex{

int re; //действительная чать

int im; //мнимая

};

void readWord(char \*buf){

int i = 0;

do{

read(STDIN\_FILENO, buf, sizeof(char));

}while((buf[i] == '\0' ||

buf[i] == ' ' ||

buf[i] == '\t' ||

buf[i] == '\n' ) && i < BS - 1);

do{

read(STDIN\_FILENO, buf + ++i, sizeof(char));

}while (buf[i] != '\0' &&

buf[i] != ' ' &&

buf[i] != '\t' &&

buf[i] != '\n' && i < BS - 1);

buf[i] = '\0'; //терминирующее значение

//printf("word = %s\n", buf);

}

int readInt(){

char buf[BS];

readWord(buf);

return atoi(buf);

}

struct Complex readComplex(){

int i = 0;

struct Complex ret = {.re = 0, .im = 0};

int signRe = 1, signIm = 1;

char buf[BS]; for(int i = 0; i < BS; i++) buf[i] = '\0';

readWord(buf);

if(buf[0] == 'i') {ret.im = 1; ret.re = 0; return ret;}

if(buf[0] == '-') {signRe = -1; i++;}

for(; i < BS; i++){

if(buf[i] < '0' || buf[i] > '9') break;

ret.re = 10 \* ret.re + (buf[i] - '0');

}

ret.re \*= signRe;

if(buf[i] == 'i'){

ret.im = ret.re;

ret.re = 0;

//printf("ret = %d %d\n",ret.re,ret.im);

return ret;

}

if(buf[i] == '-') {signIm = -1;}

i++;

if(buf[i] == 'i') {ret.im = 1;}

for(; i < BS; i++){

if(buf[i] < '0' || buf[i] > '9') break;

ret.im = 10 \* ret.im + (buf[i] - '0');

}

ret.im \*= signIm;

return ret;

}

void writeChar(char c){

write(STDOUT\_FILENO, &c, sizeof(char));

}

void writeInt(int num){

if (num < 0) { writeChar('-'); num = -num; }

char buf[BS], t;

int len = 0;

do{

buf[len++] = '0' + num % 10;

num /= 10;

}while (num > 0);

for (int i = 0; i < len / 2; i++){

t = buf[i];

buf[i] = buf[len - i - 1];

buf[len - i - 1] = t;

}

write(STDOUT\_FILENO, buf, len \* sizeof(char));

}

void writeComplex(struct Complex x){

if(x.re) writeInt(x.re);

if(x.re && x.im > 0){

writeChar('+');

}

if(x.im){

writeInt(x.im);

writeChar('i');

}

if(!x.re && !x.im){

writeInt(0);

}

}

void writeString(char \*str){

int len = 0;

while (str[len++])

;

write(STDOUT\_FILENO, str, len \* sizeof(char));

}

void error(char \*str, int code){

int len = 0;

while (str[len++])

;

write(STDERR\_FILENO, "Error: ", 8 \* sizeof(char));

write(STDERR\_FILENO, str, len \* sizeof(char));

writeChar('\n');

exit(code);

}

1. **Вывод**

Я изучила некоторые основы Операционных систем. Например я узнала о потоках, которые увеличивают работоспособность и скорость команды во много раз по сравнению с процессами. Данная тема часто используется в реальной жизни. Также я вспомнила материал первого курса, поработала с матрицами и погрузилась в пучину ностальгии и воспоминаний о молодости. ☺