

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ”

Факультет прикладної математики

Кафедра системного програмування і спеціальних комп’ютерних систем

**Лабораторна робота №4**

з дисципліни “Паралельні та розподілені обчислення”

Виконав:

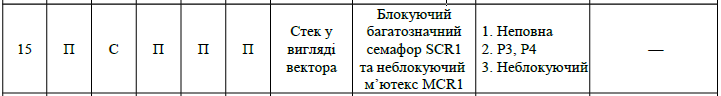
студент 2 курсу групи КВ-41:

Мелащенко Олександр

**Перевірено:**

Київ 2016

**Завдання**



**Реалізація**

#include <stdio.h>

#include <malloc.h>

#include <unistd.h>

#include <semaphore.h>

#include <pthread.h>

#include "stack.h"

pthread\_t thread1;

pthread\_t thread2;

pthread\_t thread3;

pthread\_t thread4;

pthread\_t thread5;

pthread\_mutex\_t mut\_q = PTHREAD\_MUTEX\_INITIALIZER;

sem\_t sem\_q;

sem\_t sem\_tr34;

sem\_t completedProducers;

void\* thread\_producer(void\* arg)

{

sem\_wait(&completedProducers);

int num = \*(int\*)arg,

sem\_value;

while (1)

{

if (length == MAX\_LENGTH)

break;

if((num == 3 || num == 4) && sem\_trywait(&sem\_tr34) == 0)

{

sem\_getvalue(&sem\_q, &sem\_value);

if (pthread\_mutex\_trylock(&mut\_q) == 0) {

push(num);

sem\_getvalue(&sem\_q, &sem\_value);

printf("Producer thread%d: semaphore=%d; length=%d CREATED; \n",

num, sem\_value, length);

pthread\_mutex\_unlock(&mut\_q);

}

else {

printf("Producer thread%d does some useful work\n", num);

}

sem\_post(&sem\_q);

sem\_post(&sem\_tr34);

}

}

sem\_post(&completedProducers);

printf("Producer thread%d stopped !!!\n",num);

return NULL;

}

void\* thread\_consumer (void\* arg)

{

int num = \*(int\*)arg,

curr\_elem,

sem\_value;

while (1)

{

sem\_getvalue(&sem\_q, &completedProducers);

if(completedProducers == 4 && length == 0)

break;

sem\_wait(&sem\_q);

if(pthread\_mutex\_trylock(&mut\_q) == 0)

{

curr\_elem = pop();

sem\_getvalue(&sem\_q, &sem\_value);

printf("Consumer thread%d: semaphore=%d; element %d TAKEN; \n",

num, sem\_value, curr\_elem);

pthread\_mutex\_unlock(&mut\_q);

}

else

{

printf("Consumer thread%d does some useful work\n", num);

}

}

printf("Consumer thread%d stopped !!!\n",num);

return NULL;

}

int main()

{

sem\_init (&sem\_q, 0, 0);

sem\_init (&sem\_tr34, 0, 1);

sem\_init (&completedProducers, 0, 4);

int thread1\_number = 1;

int thread2\_number = 2;

int thread3\_number = 3;

int thread4\_number = 4;

int thread5\_number = 5;

pthread\_create (&thread1,NULL,&thread\_producer,(void\*)&thread1\_number);

pthread\_create (&thread2,NULL,&thread\_consumer,(void\*)&thread2\_number);

pthread\_create (&thread3,NULL,&thread\_producer,(void\*)&thread3\_number);

pthread\_create (&thread4,NULL,&thread\_producer,(void\*)&thread4\_number);

pthread\_create (&thread4,NULL,&thread\_producer,(void\*)&thread5\_number);

pthread\_join(thread1, NULL);

pthread\_join(thread2, NULL);

pthread\_join(thread3, NULL);

pthread\_join(thread4, NULL);

pthread\_join(thread5, NULL);

printf("All threads stopped !!!\n");

return 0;

}