ДСОГЛАСОВАНО

Научный руководитель,  
доцент департамента программной инженерии факультета компьютерных, канд. техн. наук

|  |  |
| --- | --- |
| Подп. и дата |  |
| Инв. № дубл. |  |
| Взам. инв. № |  |
| Подп. и дата |  |
| Инв. № подл |  |

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А.И. Легалов

«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2020 г.

УТВЕРЖДАЮ

**Вариант 12**

**Пояснительная записка**

**ЛИСТ УТВЕРЖДЕНИЯ**

**RU.17701729.04.01-01 81 01-1 ЛУ**

Исполнитель

cтудент группы БПИ197  
\_\_\_\_\_\_\_\_\_\_\_\_\_/ П. О. Кулешова /

«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2020 г.

Академический руководитель образовательной программы «Программная инженерия» профессор департамента программной инженерии, канд. техн. наук

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_В. В. Шилов

«\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2020 г.

|  |  |
| --- | --- |
| Подп. и дата |  |
| Инв. № дубл. |  |
| Взам. инв. № |  |
| Подп. и дата |  |
| Инв. № подл |  |

**Вариант 12**

**Пояснительная записка**

**RU.17701729.04.01-01 81 01-1**

**Листов 13**

сОДЕРЖАНИЕ

[ВВЕДЕНИЕ 3](#_Toc57747952)

[1.1. Наименование программы 3](#_Toc57747953)

[1.2. Документ, на основе которого ведется разработка 3](#_Toc57747954)

[1. НАЗНАЧЕНИЕ И ОБЛАСТЬ ПРИМЕНЕНИЯ 4](#_Toc57747955)

[1.1 Назначение разработки 4](#_Toc57747956)

[1.2 Краткая характеристика области применения 4](#_Toc57747957)

[2. Описание программы 5](#_Toc57747958)

[Условие задачи 5](#_Toc57747959)

[Метод решения 5](#_Toc57747960)

[Алгоритм решения 5](#_Toc57747961)

[Замечание 6](#_Toc57747962)

[ИСТОЧНИКИ, ИСПОЛЬЗОВАННЫЕ ПРИ РАЗРАБОТКЕ 8](#_Toc57747963)

[ПРИЛОЖЕНИЕ 1 Код программы 9](#_Toc57747964)

[ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ 14](#_Toc57747965)

# ВВЕДЕНИЕ

## Наименование программы

Наименование программы: Многопоточное программирование.

Взаимодействие потоков. Микропроект.

Краткое наименование программы: Микропроект

## Документ, на основе которого ведется разработка

http://www.softcraft.ru/edu/comparch/tasks/mp02/

# НАЗНАЧЕНИЕ И ОБЛАСТЬ ПРИМЕНЕНИЯ

## Назначение разработки

Создать многопоточное приложение, моделирующее рабочий день клиники.

## Краткая характеристика области применения

Научно-исследовательская область.

# Описание программы

## Условие задачи

Задача о больнице. В больнице два врача принимают пациентов, выслушивают их жалобы и отправляют их или к стоматологу или к хирургу или к терапевту. Стоматолог, хирург и терапевт лечат пациента. Каждый врач может принять только одного пациента за раз. Пациенты стоят в очереди к врачам и никогда их не покидают. Создать многопоточное приложение, моделирующее рабочий день клиники.

## Метод решения

Задача решалась с использованием мьютексов и семафоров. Так как одновременно принимают два врача, то был использован семафор. Все специалисты принимают по одному, поэтому использовался мьютекс.

## Алгоритм решения

1. Создаём потоки-пациенты
2. На семафоре пускаем подвое для перенаправления к специалисту
3. Далее, на мьютексе, задерживаем потоки, чтобы к каждому специалисту мог попасть только один поток
4. Ожидаем завершения всех потоков

## Замечание

При выводе каждый специалист находится в отдельной колонке. Считалось, что после похода к врачу, пациент сразу идёт к специалисту. Считалось, что у нас небольшая больница и пациентов до 30 человек в день. **3. Формат входных данных**

**Входные данные командной строки**

Входные данные не предусмотрены.

# ИСТОЧНИКИ, ИСПОЛЬЗОВАННЫЕ ПРИ РАЗРАБОТКЕ

1. ГОСТ 19.404-79 Пояснительная записка. Требования к содержанию и оформлению// Единая система программной документации. – М.: ИПК Стандартинформ, 2010.
2. Сайт «SoftCraft». URL: http://www.softcraft.ru/edu/comparch/practice/thread/02-sync/ [http://www.softcraft.ru] Просмотрено: 09.12.2020
3. Сайт «Pthreads: Потоки в русле POSIX». URL: <https://habr.com/ru/post/326138/> Просмотрено: 09.12.2020
4. Сайт «[C++] часть 2: МЬЮТЕКС. Пишем наш первый код для многопоточной среды». URL: <https://medium.com/nuances-of-programming/c-%D0%BC%D1%8C%D1%8E%D1%82%D0%B5%D0%BA%D1%81-%D0%BF%D0%B8%D1%88%D0%B5%D0%BC-%D0%BD%D0%B0%D1%88-%D0%BF%D0%B5%D1%80%D0%B2%D1%8B%D0%B9-%D0%BA%D0%BE%D0%B4-%D0%B4%D0%BB%D1%8F-%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE%D0%BF%D0%BE%D1%82%D0%BE%D1%87%D0%BD%D0%BE%D0%B9-%D1%81%D1%80%D0%B5%D0%B4%D1%8B-543a3d60ef30> Просмотрено: 01.12.2020
5. Сайт «SO Documentation» URL: <https://sodocumentation.net/ru/cplusplus/topic/9785/%D1%81%D0%B5%D0%BC%D0%B0%D1%84%D0%BE%D1%80> Просмотрено: 09.12.2020
6. Сайт « Семафоры: введение». URL: <https://learnc.info/c/pthreads_semaphores.html> Просмотрено: 09.12.2020
7. Сайт «RIP Tutorial». URL: <https://riptutorial.com/ru/cplusplus/example/30148/%D1%81%D0%B5%D0%BC%D0%B0%D1%84%D0%BE%D1%80-%D0%B2-%D0%B4%D0%B5%D0%B9%D1%81%D1%82%D0%B2%D0%B8%D0%B8> Просмотрено: 09.12.2020
8. Сайт «cplusplus.com». URL: <http://www.cplusplus.com/reference/chrono/microseconds/> Просмотрено: 09.12.2020

# ПРИЛОЖЕНИЕ 1 **Код программы**

#include <pthread.h>

#include <semaphore.h>

#include <stdlib.h>

#include <stdio.h>

#include <iostream>

#include <vector>

#include <string>

#include<ctime>

#ifdef \_WIN32

#include <Windows.h>

#else

#include <unistd.h>

#endif

using namespace std;

const int bufSize = 2; //количество врачей

int rear = 0; //индекс для записи в буфер

int front = 0; //индекс для чтения из буфера

sem\_t emptySem; //семафор, отображающий насколько буфер пуст

sem\_t full; //семафор, отображающий насколько полон буфер

pthread\_mutex\_t mutexDentist; //мутекс для дантиста

pthread\_mutex\_t mutexOculist; //мутекс для окулиста

pthread\_mutex\_t mutexTherapist; //мутекс для терапевт

vector<pthread\_t> threadC; //потоки к врачам

vector<pthread\_t> threadQ; //потоки к специалистам

unsigned int start\_time;

/// <summary>

/// Структура - пациент

/// </summary>

struct Patient

{

int id; //id

int doctor; //специалист, к которому направили

pthread\_t thread; //поток, в котором он сущесивуеи

};

/// <summary>

/// Дантист

/// </summary>

/// <param name="param">пациент</param>

/// <returns></returns>

void\* Dantist(void\* param);

/// <summary>

/// Окулист

/// </summary>

/// <param name="param">пациент</param>

/// <returns></returns>

void\* Oculist(void\* param);

/// <summary>

/// Терапевт

/// </summary>

/// <param name="param">пациент</param>

/// <returns></returns>

void\* Therapist(void\* param);

/// <summary>

/// Доктор, который направляет пациента

/// </summary>

/// <param name="param">пациент</param>

/// <returns></returns>

void\* Doctor(void\* param) {

Patient\* patient = (Patient\*)param;//извлечь пациента из буфера

sem\_wait(&emptySem); //количество занятых ячеек увеличить на единицу

srand(GetCurrentThreadId());

int doctor = rand() % 3; //к какому специалисту будет направлен

rear = (rear + 1) % bufSize; //номер принимающего врача

patient->doctor = doctor;//доктор

unsigned int stop\_time = clock();

sem\_post(&full); //количество свободных ячеек уменьшить на единицу

//обработать полученный элемент

printf("%sВремя: %d Пациент %d:\tбыл направлен к %s от %s\n", patient->doctor == 0 ?

"\t" : patient->doctor == 1 ? "\t\t" : "\t\t\t", stop\_time - start\_time, patient->id, patient->doctor == 0 ?

"дантисту" : patient->doctor == 1 ? "окулисту" : "терапевту", rear-1 == 0 ? "Иванова" : "Сергеевича"); //вывод информации о посещении врача

pthread\_create(&threadQ[patient->id - 1], nullptr, (patient->doctor == 0) ?

Dantist : patient->doctor == 1 ? Oculist : Therapist, (void\*)(patient)); //отправка пациента к специалисту

sem\_wait(&full);//количество занятых ячеек уменьшить на единицу

sem\_post(&emptySem);//количество свободных ячеек увеличить на единицу

return nullptr;

}

/// <summary>

/// Терапевт

/// </summary>

/// <param name="param">пациент</param>

/// <returns></returns>

void\* Therapist(void\* param)

{

pthread\_mutex\_lock(&mutexTherapist);

Patient\* patient = (Patient\*)param;

unsigned int stop\_time = clock();

printf("\t\t\tВремя: %d Пациент %d:\tпосетил %s\n", stop\_time - start\_time, patient->id, "терапевта");

pthread\_mutex\_unlock(&mutexTherapist);

return nullptr;

}

/// <summary>

/// Окулист

/// </summary>

/// <param name="param">пациент</param>

/// <returns></returns>

void\* Oculist(void\* param)

{

pthread\_mutex\_lock(&mutexOculist);

Patient\* patient = (Patient\*)param;

unsigned int stop\_time = clock();

printf("\t\tВремя: %d Пациент %d:\tпосетил %s\n", stop\_time - start\_time, patient->id, "окулиста");

pthread\_mutex\_unlock(&mutexOculist);

return nullptr;

}

/// <summary>

/// Дантист

/// </summary>

/// <param name="param">пациент</param>

/// <returns></returns>

void\* Dantist(void\* param)

{

pthread\_mutex\_lock(&mutexDentist);

Patient\* patient = (Patient\*)param;

unsigned int stop\_time = clock();

printf("\tВремя: %d Пациент %d:\tпосетил %s\n", stop\_time - start\_time, patient->id, "дантиста");

pthread\_mutex\_unlock(&mutexDentist);

return nullptr;

}

int main()

{

srand((unsigned)time(0));

int n = rand() % 20 + 11;

setlocale(LC\_ALL, "Russian");

cout << "Сегодня на приём пришло: " << n << " человек. Они стоят в очереди к двум специалистам (Иванов и Сергеевич).\n"; //инициализация мутексов и семафоров

pthread\_mutex\_init(&mutexDentist, nullptr); //инициализация мьютекса дантиста

pthread\_mutex\_init(&mutexOculist, nullptr);//инициализация мьютекса окулиста

pthread\_mutex\_init(&mutexTherapist, nullptr);//инициализация мьютекса терапист

sem\_init(&emptySem, 0, bufSize); //количество свободных ячеек равно bufSize

sem\_init(&full, 0, 0); //количество занятых ячеек равно 0

threadC= vector<pthread\_t>(n);

threadQ = vector<pthread\_t>(n);

vector<Patient> patients(n);

start\_time = clock();

//пациенты приходят и встают в очередь

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

{

patients[i].id = i+1;

patients[i].thread = threadC[i];

unsigned int stop\_time = clock();

printf("Время: %d Пациент %d занял очередь\n",stop\_time-start\_time, i+1);

pthread\_create(&threadC[i], nullptr, Doctor, (void\*)(&patients[i]));

}

//ожидание завершения всех потоков

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

{

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

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

}

return 0;

}

# ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Изм. | Номера листов (страниц) | | | | Всего листов (страниц) в документе | № документа | Входящий № сопроводительного документа и дата | Подпись | Дата |
| измененных | замененных | новых | аннулированных |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |