Министерство образования и науки РФ

федеральное государственное бюджетное образовательное учреждение высшего образования
«Омский государственный технический университет»

Факультет (институт) Факультет информационных технологий и компьютерных систем

Кафедра Информатики и вычислительной техники

КУРСОВОЙ ПРОЕКТ (РАБОТА)

по дисциплине Операционные системы

на тему Многопоточная Windows программная модель ямской связи между 11 почтовыми станциями в старой Российской империи

Пояснительная записка

Шифр проекта (работы) _____

Студента (ки) Леммер Вадим Евгеньевич
фамилия, имя, отчество полностью

Курс 2 **Группа** ПИН-161

Направление (специальность) _____

09.03.04 Программная инженерия
код, наименование

Руководитель доцент, к.т.н
ученая степень, звание

Флоренсов Александр Николаевич
фамилия, инициалы

Выполнил (а) 02.06.2018
дата, подпись студента (ки)

К защите 2.06.2018
дата, подпись руководителя

Выполнение и подготовка к защите КП (КР)	Защита КП (КР)	Итоговый рейтинг
53	38	91

Проект (работа) защищен (а) с оценкой отл

Омск 2018

Реферат

Пояснительная записка по курсовому проекту 17 с., 1 ч., 1 рис., 3 источ, 1 прил.

C, WINDOWS, ЯМСКАЯ СВЯЗЬ, GCC, МНОГОПОТОЧНОЕ ПРИЛОЖЕНИЕ.

Объектом исследования является алгоритм взаимодействия нескольких потоков в операционной системе Windows при работе в графическом окне.

Цель работы – разработка многопоточной программы, имитирующей работу ямской связи в операционной системе Windows.

В ходе работы реализован алгоритм взаимодействия нескольких потоков при работе с графическим окном в операционной системе Windows.

В результате была получена программа, которая демонстрирует модель перемещение пассажиров в разные города с помощью коней.

Содержание

Введение

Курсовой проект по дисциплине «Операционные системы», 2 курс. В проекте использовался язык программирования C.

Задача: Задача: Многопоточная Windows программа модель ямской связи между 11 почтовыми станциями в старой Российской империи. Разработать программную модель ямской почтовой связи, имевшейся в Российской империи. Модель представляет 11 почтовых станций, между которыми перемещаются почтовые подводы. За каждой почтовой стацией закреплено некоторое число лошадей. Это количество может быть не полностью готовым для перевозок (лошади отдыхают после гоньбы), промоделировать случайными величинами. На пяти ведущих станциях, моделирующих города, случайным образом формируются лица, едущие по казенной надобности, случайным образом в один из городов, которые представляются станциями. Изображение городов – в центре окна и по его углам, дороги по периметру окна и диагоналям. Подобрать характеристики программных генераторов ездоков, чтобы на станциях возникали ситуации ожидания освоившихся или отдохнувших лошадей. Для упрощения модели в подводу для путешественника впрягается одна лошадь. Поведение каждого пассажира и подводы должно реализоваться отдельной нитью. Для правильного взаимодействия использовать семафоры или мьютексы.

Проект состоит из пяти разделов:

- Введение в проблематику разработки многопоточных приложений
- Декомпозиция разрабатываемой программы снизу-вверх с формированием основных процедур ее функционирования и описанием их функционального назначения
- Описание глобальных информационных объектов программы: глобальных переменных, средств синхронизации потоков и используемых структур данных в случае их применения
- Детальное текстовое описание на основе сочетания естественного языка и программных конструкций алгоритмов всех процедур

1 Введение в проблематику разработки многопоточных приложений

В современных операционных системах широко используются нити (thread), называемые несколько неточно в русском переводе также потоками. Это понятие возникло как результат развития понятия абстрактного процесса. Оказалось, что иногда целесообразно процесс – как некоторую общность программного выполнения, имеющую прикладную цель, – разбить на части, которые выполнялись бы параллельно, конкурируя за главный ресурс – процессор, но в остальном выполняли бы общую работу. Можно подойти к осознанию понятия процесса, с другой стороны. Абстрактные процессы теоретически разделяют на конкурирующие и кооперативные. Конкурирующие процессы, по существу, мешают друг другу, но в совокупности выполняют много работ одновременно. Кооперативные процессы выполняют по частям общую работу совместно. Организационно-техническое объединение (под одной "крышей" обобщенного процесса) аналогов кооперативных процессов и составляет существо объединения нитей в одном процессе.

При разработке многопоточных приложений следует иметь в виду необходимость явного указания для системы разработки, что данное приложение будет использовать более одной нити. Для таких приложений в процессе построения исполняемого файла подключаются специальные библиотеки подпрограмм.

При разработке многопоточных программ для Windows следует указывать соответствующую библиотеку поддержки.

```
#include <windows.h>
```

При компиляции программы используется атрибут –mwindows для работы в оконном режиме.

```
gcc kurs.c -mwindows
```

2 Декомпозиция разрабатываемой программы снизу-вверх с формированием основных процедур ее функционирования и описанием их функционального назначения

Основным элементом функционирования являются графические примитивы и буквенные символы, используемые для обозначения городов, коней и дорог. Графические примитивы имеют две координаты, начинающиеся с верхнего левого угла, и стандартный цвет. Для графических примитивов, представляющих коней, существует процедура, отвечающая за их положение в окне, а также за их поведение в программе. Графические примитивы городов статичны, однако для них также была создана процедура, которая вызывалась один раз. Для лошадей есть отдельная процедура, цель которой – изменять положение коней от одного города до другого.

Совокупность процедур изменения координат графических примитивов в зависимости от текущего состояния лошадей, а также процедур изменения состояния городов, составляют основу анимации модели имитации работы ямской связи.

Процедуры изменения координат примитивов являются отдельными нитями для каждой графической фигуры. Аналогично и для процедур, отвечающие за изменение состояния структуры каждого отдельной лошади. Каждая нить отвечает за вывод результата своей работы в данный момент времени. В главной процедуре приложения происходит предварительная подготовка и инициализация всех необходимых переменных и структур, а также запуск процедур, предназначенных для моделирования.

3 Описание глобальных информационных объектов программы: глобальных переменных, средств синхронизации потоков и используемых структур данных в случае их применения

Для работоспособности программы были созданы следующие структуры:

1. Horse – это структура, описывающая лошадей. В этой структуре представлены следующие члены структуры:
    a. int x – значение текущего нахождения лошади по оси x;
    b. int y – значение текущего нахождения лошади по оси y;
    c. int fromPointX – значение точки отправления лошади по оси x;
    d. int fromPointY – значение точки отправления лошади по оси y;

Помимо указанной выше структуры, в программе используются следующие глобальные переменные:

1. HWND hwnd – идентификатор окна;
2. HBRUSH horseBrush, cityBrush – идентификатор кисти заливки;

3. HPEN blackPen, whitePen – идентификатор кисти для обводки;
4. Horse horses[HORSE_COUNT] – массив лошадей;
5. DWORD hid1, hid2, hid3, hid4, hid5, did – идентификатор нити для лошадей:
6. HANDLE Semaphore1, Semaphore2, Semaphore3, Semaphore4, Semaphore5, Semaphore6, Semaphore7, Semaphore8, Semaphore9, Semaphore10, Semaphore11 – идентификатор семафоров для лошадей;
7. int citisPosX[] – массив координаты городов по оси x;
8. int citisPosY[] – массив координаты городов по оси y;
9. char* citiesNames[] – массив названий городов;

4 Детальное текстовое описание на основе сочетания естественного языка и программных конструкций алгоритмов всех процедур

Программа работает за счёт изменения координат x и y с заданной задержкой руководствуясь условиями, определяющими текущее состояние объекта, а также в зависимости от наличия или отсутствия блокировки на необходимый участок одноколейного движения. В результате этих действий координаты графических примитивов изменяются с заданной частотой. Изменение координат происходит за счёт простого инкремента или декремента переменных, отвечающих за соответствующие координаты у объекта графического примитива.

Вывод графической информации в текущий момент времени производится главной нитью приложения, а также нитями самих графических примитивов, таких как лошадь. Они работают в бесконечном цикле обновляя данные в окно на основе данных, которые сохранены в массиве структур и глобальных переменных, с которыми работают остальные нити.

Отдельными процедурами являются процедуры для рисования лошадей и городов. Опираясь на данные, получаемые из структуры лошадей и глобальных переменных, нити, помимо выполнения своих главных обязанностей, занимаются отображением объекта в окне. Для рисования городов существует отдельная процедура, которая вызывается один раз в программе.

Также, как уже было замечено, за изменения положения лошади и его отображение в окне используется отдельная нить. Её основная задача заключаются инкрементировании и декрементировании переменной, хранящей текущее положение лошади на этаже, значение которой будет использоваться нитью пассажира.

В теле главной нити происходит процесс получения с консольного окна значения количества лошадей, которые будут участвовать в данной программе. Также в ней происходит инициализация семафора и нити лифта.

Схема алгоритма главной процедуры представлена на рисунке 1.

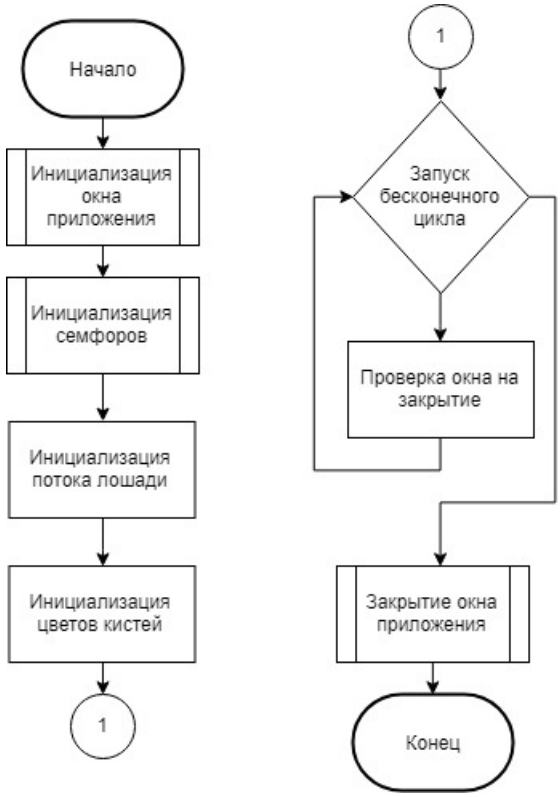

Рисунок 1 – Схема алгоритма основной процедуры

Скриншот работы программы представлен на рисунке 2.

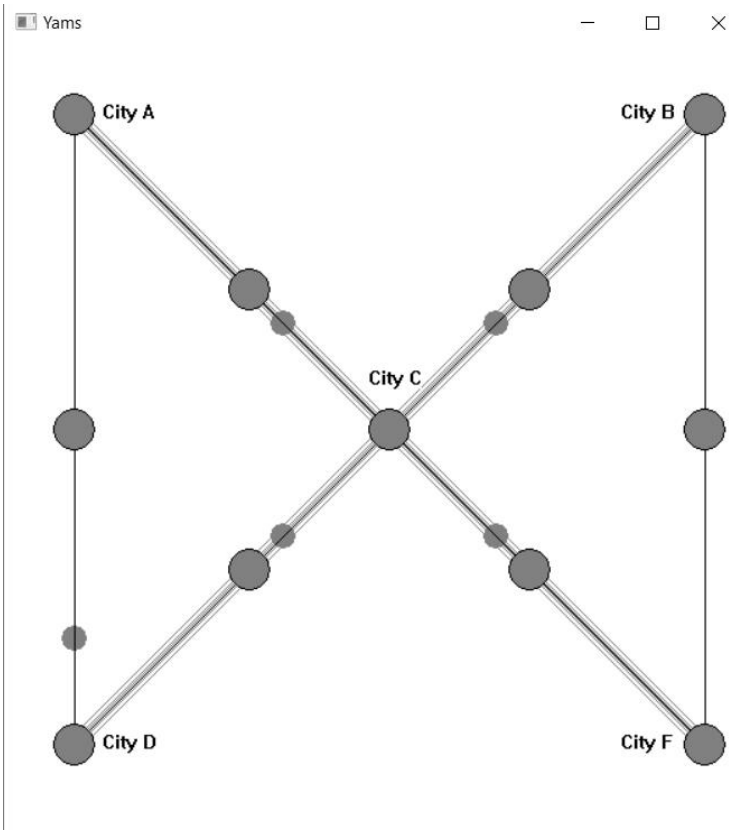

Рисунок 2 – Пример работы программы

Заключение

В результате работы над проектом был реализован алгоритм взаимодействия нескольких потоков при работе с графическим окном в операционной системе Windows в программе, написанной на языке программирования C. Программа демонстрирует модель ямской связи, перемещающихся лошадей по пяти городам, замена лошадей на промежуточных станциях и на самих городах с использованием семафоров, используя графические примитивы и текстовые символы.

Список использованных источников

1. Флоренсов, А.Н. Операционные системы для программиста. Омск. ОмГТУ, 2005.
2. Гордеев, А.В. Операционные системы / А.В. Гордеев. – 2-е изд. – СПб.: Питер, 2007
3. ГОСТ 7.1-2003 Библиографическая запись. Библиографическое описание. Общие требования и правила составления.

Приложение

Исходный код программы

```
/*
   Использую  WinAPI
*/

#include <windows.h>

#define WINDOW_WIDTH 550 //Ширина окна
#define WINDOW_HEIGHT 600 //Высота окна

#define CITY_RADIUS 15 //Радиус кружка города
#define HORSE_RADIUS 10 //Радиус кружка лошади

#define HORSE_COUNT 5 //Количество лошадей

/*Структура лошади*/
typedef struct {
   int x;
   int y;
   int fromPointX;
   int fromPointY;
} Horse;

HWND hwnd;
HDC hdc;

HBRUSH horseBrush, cityBrush; //Заливка
HPEN blackPen, whitePen; //Обводка

Horse horses[HORSE_COUNT];
DWORD hid1, hid2, hid3, hid4, hid5, did; //Идентификаторы потоков
HANDLE Semaphore1, Semaphore2, Semaphore3, Semaphore4, Semaphore5, Semaphore6, Semaphore7, Semaphore8,
Semaphore9, Semaphore10, Semaphore11;

int citisPosX[] = {50, 500}; //Координаты X городов
int citisPosY[] = {50, 500}; //Координаты Y городов
char* citiesNames[] = {"City A", "City B", "City C", "City D", "City F"}; //Название городов

/*Создание цветов для карты и лошадей*/
void createColors() {
   //Заливка для лошадей и городов
   horseBrush = CreateSolidBrush(RGB(0, 255, 0)); //зеленый цвет
   cityBrush = CreateSolidBrush(RGB(255, 0, 0)); //красный цвет
   //Обводка для лошадей и городов
   blackPen = CreatePen(PS_SOLID, 1, RGB(0, 0, 0)); //черный цвет
   whitePen = CreatePen(PS_SOLID, 1, RGB(255, 255, 255)); //белый цвет
}

/*Отрисовка города*/
void drawCity(int x, int y, int index) {
   SelectObject(hdc, cityBrush); //выбор кисти
   SelectObject(hdc, blackPen); //выбор обводки
   Ellipse(hdc, x - CITY_RADIUS, y - CITY_RADIUS, x + CITY_RADIUS, y + CITY_RADIUS);
}

/*Отрисовка карты*/
void drawMap() {
   /*Отрисовка линий*/
   SelectObject(hdc, blackPen);
```

```
    /*Вертикальные*/
    /*Лево*/
    MoveToEx(hdc, 50, 50, NULL);
    LineTo(hdc, 50, 500);
    /*Право*/
    MoveToEx(hdc, 500, 50, NULL);
    LineTo(hdc, 500, 500);
    /*Диагональ*/
    MoveToEx(hdc, 50, 50, NULL);
    LineTo(hdc, 500, 500);
    MoveToEx(hdc, 500, 50, NULL);
    LineTo(hdc, 50, 500);

    /*Отрисовка городов*/
    /*Высь*/
    drawCity(50, 50, 0); //левый город
    drawCity(500, 50, 2); //правый город
    /*Седедина*/
    drawCity(50, 275, 3); //левый город
    drawCity(275, 275, 4); //цетральный город
    drawCity(500, 275, 5); //правый город
    /*Низ*/
    drawCity(50, 500, 6); //левый город
    drawCity(500, 500, 8); //правый город
    /*Диагональ*/
    drawCity(175, 175, 9);
    drawCity(375, 375, 10);
    drawCity(375, 175, 10);
    drawCity(175, 375, 10);

    /*Название городов*/
    TextOut(hdc, 70, 40, citiesNames[0], strlen(citiesNames[0])); //Город A
    TextOut(hdc, 440, 40, citiesNames[1], strlen(citiesNames[1])); //Город B
    TextOut(hdc, 260, 230, citiesNames[2], strlen(citiesNames[2]));//Город C
    TextOut(hdc, 70, 490, citiesNames[3], strlen(citiesNames[3])); //Город D
    TextOut(hdc, 440, 490, citiesNames[4], strlen(citiesNames[4])); //Город F
}

/*Отрисовка лошади*/
void drawHorse(int index) {
    Horse horse = horses[index];
    SelectObject(hdc, horseBrush); //выбор кисти
    SelectObject(hdc, whitePen);  //выбор обводки
    Ellipse(hdc, horse.x - HORSE_RADIUS, horse.y - HORSE_RADIUS, horse.x + HORSE_RADIUS, horse.y +
HORSE_RADIUS); //рисование круга
}

/*Создание лошади*/
void createHorse(int xPos, int yPos, int index) {
    Horse horse = {
        .x = xPos,
        .y = yPos,
        .fromPointX = xPos,
        .fromPointY = yPos,
    };
    horses[index] = horse;
}

/*Движение лошади*/
void doStep(Horse *horse, int random, int random2) {
    int fromPointX = horse -> fromPointX;
    int fromPointY = horse -> fromPointY;
    /*Из города A*/
```

```
    if(fromPointX == 50 && fromPointY == 50) {
        if(random == 0) {
            //движение вниз
            horse -> y++;
        }
        else {
            //движение по диагонали
            horse -> x++;
            horse -> y++;
        }
    }
    /*Из города B*/
    if(fromPointX == 500 && fromPointY == 50) {
        if(random == 1) {
            //движение вниз
            horse -> y++;
        }
        else {
            //движение по диагонали
            horse -> x--;
            horse -> y++;
        }
    }
    /*Из города D*/
    if(fromPointX == 50 && fromPointY == 500) {
        if(random == 0) {
            //движение вверх
            horse -> y--;
        }
        else {
            //движение по диагонали
            horse -> x++;
            horse -> y--;
        }
    }
    /*Из города F*/
    if(fromPointX == 500 && fromPointY == 500) {
        if(random == 1) {
            //движение по диагонали
            horse -> x--;
            horse -> y--;
        }
        else {
            //движение вверх
            horse -> y--;
        }
    }
}

/*Отрисовка всех элементов*/
DWORD WINAPI drawAllThread(void* arg) {
    while (1) {
    drawMap(); //отрисовки карты
        for (int i = 0; i < HORSE_COUNT; i++) {
            drawHorse(i); // отрисовки лошади
        }
    }
}

/*Нить лошади*/
DWORD WINAPI horseThread(void* arg) {
    //определение номера лошади по номеру потока
    int index = (int)arg;
```

```
//случайное задание идентификатора потопка
srand(GetCurrentThreadId());
//случайные, начальные координаты лошади
int x = citisPosX[rand() % 2];
int y = citisPosY[rand() % 2];
int random = rand() % 2;
int random2 = rand() % 4;
//создание лошади
createHorse(x, y, index);
do{
   //вызов метода шага лошади
   doStep(&horses[index], random, random2);
   //проверка лошади в точке A
   if (horses[index].x == 50 && horses[index].y == 50){
      WaitForSingleObject(Semaphore1, 5000+rand() % 1000); // вызов семаформа точки A
      //задание новой случайно позиции появления лошади
      int x = citisPosX[rand() % 2];
      int y = citisPosY[rand() % 2];
      int random = rand() % 2;
      createHorse(x, y, index);
   }
   //проверка лошади в точке B
   if (horses[index].x == 50 && horses[index].y == 500) {
      WaitForSingleObject(Semaphore2, 5000+rand() % 1000); // вызов семаформа точки B
      //задание новой случайно позиции появления лошади
      int x = citisPosX[rand() % 2];
      int y = citisPosY[rand() % 2];
      int random = rand() % 2;
      createHorse(x, y, index);
   }
   //проверка лошади в точке C
   if (horses[index].x == 500 && horses[index].y == 50){
      WaitForSingleObject(Semaphore3, 5000+rand() % 1000); // вызов семаформа точки C
      //задание новой случайно позиции появления лошади
      int x = citisPosX[rand() % 2];
      int y = citisPosY[rand() % 2];
      int random = rand() % 2;
      createHorse(x, y, index);
   }
   //проверка лошади в точке D
   if (horses[index].x == 500 && horses[index].y == 500) {
      WaitForSingleObject(Semaphore4, 5000+rand() % 1000); // вызов семаформа точки D
      //задание новой случайно позиции появления лошади
      int x = citisPosX[rand() % 2];
      int y = citisPosY[rand() % 2];
      int random = rand() % 2;
      createHorse(x, y, index);
   }
   //проверка лошади в точке F
   if (horses[index].x == 275 && horses[index].y == 275) {
      WaitForSingleObject(Semaphore5, 5000+rand() % 1000); // вызов семаформа точки F
      //задание новой случайно позиции появления лошади
      int x = citisPosX[rand() % 2];
      int y = citisPosY[rand() % 2];
      int random = rand() % 2;
      createHorse(x, y, index);
   }
   //временные пункты, точки замены лошадей
   if (horses[index].x == 50 && horses[index].y == 275) {
      WaitForSingleObject(Semaphore6, 5000+rand() % 1000); //вызов семафора точки 1
   }
   if (horses[index].x == 175 && horses[index].y == 175) {
      WaitForSingleObject(Semaphore7, 5000+rand() % 1000); //вызов семафора точки 2
```

```
        }
        if (horses[index].x == 175 && horses[index].y == 375) {
            WaitForSingleObject(Semaphore8, 5000+rand() % 1000); //вызов семафора точки 3
        }
        if (horses[index].x == 375 && horses[index].y == 175) {
            WaitForSingleObject(Semaphore9, 5000+rand() % 1000); //вызов семафора точки 4
        }
        if (horses[index].x == 375 && horses[index].y == 375) {
            WaitForSingleObject(Semaphore10, 5000+rand() % 1000); //вызов семафора точки 5
        }
        if (horses[index].x == 500 && horses[index].y == 275) {
            WaitForSingleObject(Semaphore11, 5000+rand() % 1000); //вызов семафора точки 6
        }
        Sleep(10);
    } while(1);
}

/*Создание семафоров*/
void CreateSemaphores() {
    Semaphore1 = CreateSemaphore(NULL, 5, 5, NULL);
    Semaphore2 = CreateSemaphore(NULL, 5, 5, NULL);
    Semaphore3 = CreateSemaphore(NULL, 5, 5, NULL);
    Semaphore4 = CreateSemaphore(NULL, 5, 5, NULL);
    Semaphore5 = CreateSemaphore(NULL, 5, 5, NULL);
    Semaphore6 = CreateSemaphore(NULL, 5, 5, NULL);
    Semaphore7 = CreateSemaphore(NULL, 5, 5, NULL);
    Semaphore8 = CreateSemaphore(NULL, 5, 5, NULL);
    Semaphore9 = CreateSemaphore(NULL, 5, 5, NULL);
    Semaphore10 = CreateSemaphore(NULL, 5, 5, NULL);
    Semaphore11 = CreateSemaphore(NULL, 5, 5, NULL);
}

/*Создание потоков*/
void CreateThreads() {
    HANDLE horse1 = CreateThread(NULL, 0, horseThread, (void*)0,  0, &hid1);
    HANDLE horse2 = CreateThread(NULL, 0, horseThread, (void*)1,  0, &hid2);
    HANDLE horse3 = CreateThread(NULL, 0, horseThread, (void*)2,  0, &hid3);
    HANDLE horse4 = CreateThread(NULL, 0, horseThread, (void*)3,  0, &hid4);
    HANDLE horse5 = CreateThread(NULL, 0, horseThread, (void*)4,  0, &hid5);
    HANDLE draw = CreateThread(NULL, 0, drawAllThread, (void*)5,  0, &did);
}

/*Обработка событий окна*/
LRESULT WINAPI WinProc(HWND hwnd, UINT tmsg, WPARAM wParam, LPARAM lParam) {
    switch(tmsg) {
        case WM_DESTROY: {
            PostQuitMessage(0);
            return 0;
        }
        case WM_CLOSE: {
            PostQuitMessage(0);
            return 0;
        }
    }
    return DefWindowProc(hwnd, tmsg, wParam, lParam);
}

/*Создание окна и запуск всех объектов*/
int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpszCmdLine, int nCmdShow) {
    MSG msg;
    WNDCLASS w;

    memset(&w,0,sizeof(WNDCLASS)); //выделение памяти классу
```

```
    w.lpszClassName = "Yams"; //определение имени класса
    w.lpfnWndProc = WinProc; //процесс
    w.hInstance = hInstance;
    w.hCursor=LoadCursor(NULL, IDC_ARROW); //фиксация значка курсора
    w.hbrBackground = (HBRUSH)(WHITE_BRUSH); //фон окна белый
    RegisterClass(&w); //регистрация класса

    //создание окна с именем Yams
    hwnd = CreateWindow("Yams", "Yams", WS_OVERLAPPEDWINDOW, CW_USEDEFAULT, CW_USEDEFAULT,
WINDOW_WIDTH, WINDOW_HEIGHT , NULL, NULL, hInstance, NULL);
    ShowWindow(hwnd,nCmdShow); //показать окно

    hdc = GetDC(hwnd);
    createColors(); //создание цветов
    CreateSemaphores(); //создание семафоров
    CreateThreads(); //создание потоков

    while(GetMessage(&msg,NULL,0,0)) DispatchMessage(&msg);
    return msg.wParam;
}
```

Министерство образования и науки РФ
федеральное государственное бюджетное образовательное учреждение высшего образования
«Омский государственный технический университет»

ОТЗЫВ
на курсовой проект (работу)

Факультет (институт) Факультет информационных технологий и компьютерных систем

Кафедра Информатики и вычислительной техники

Дисциплина Операционные системы

Тема Многопоточная Windows программная модель ямской связи между 11 почтовыми станциями в старой Российской империи

Студент (ка) Леммер Вадим Евгеньевич
фамилия, имя, отчество полностью

Курс 2 Группа ПИН-161

Руководитель доцент, к.т.н, Флоренсов Александр Николаевич
ученая степень, звание, ФИО

Содержание отзыва

Задание выполнено полностью. Работа доведена до нормального функционирования. Освоены средства построения многопоточных приложений, средства синхронизации взаимодействий потоков. Сформированы навыки программирования динамики графических окон.

Рейтинговые баллы за выполнение и подготовку к защите курсового проекта (работы)	53
Заключение о допуске к защите	допущено

Руководитель [подпись]
подпись
Дата 2.06.2018 **г.**

Министерство образования и науки РФ
федеральное государственное бюджетное образовательное учреждение высшего образования
«Омский государственный технический университет»

Факультет (институт) Факультет информационных технологий и компьютерных систем

Кафедра Информатики и вычислительной техники

Дисциплина Операционные системы

ЗАДАНИЕ
на выполнение курсового проекта (работы)

Студенту (ке) Леммеру Вадиму Евгеньевичу Группа ПИН-161
фамилия, имя, отчество полностью

Направление (специальность) 09.03.03 Программная инженерия
код, наименование

Тема проекта (работы) Многопоточная Windows программная модель ямской связи между 11 почтовыми станциями

Срок сдачи проекта (работы) на кафедру « 29 » мая 201 8 г.

Исходные данные к проекту (работе)

Разработать программную модель ямской почтовой связи, имевшейся в Российской империи. Модель представляет 11 почтовых станций, между которыми перемещаются почтовые подводы. За каждой почтовой станцией закреплено некоторое число лошадей. (Для модели выбрать число в пределах 5 — 7). Это количество может быть не полностью готовым для перевозок (лошади отдыхают после гоньбы), промоделировать случайными величинами. На пяти ведущих станциях, моделирующих города, случайным образом формируются лица, едущие по казенной надобности, случайным образом в один из городов, которые представляются станциями. Изображения городов — в центре окна и по его углам, дороги по периметру окна и диагоналям. Подобрать характеристики программных генераторов ездоков, чтобы на станциях возникали ситуации ожидания освоившихся или отдохнувших лошадей. Для упрощения модели в подводу для путешественника впрягается одна лошадь. Разработку провести в ОС типа Windows как многопоточную программную имитацию. Поведение каждого пассажира и подводы должно реализоваться отдельной нитью. Для правильного взаимодействия использовать семафоры или мьютексы.

Содержание пояснительной записки (перечень подлежащих разработке вопросов)

1. Введение в проблематику разработки многопоточных приложений
2. Декомпозиция разрабатываемой программы снизу-вверх с формирование основных процедур ее функционирования и описанием их функционального назначения
3. Описание глобальных информационных объектов программы: глобальных переменных, средств синхронизации потоков и используемых структур данных в случае их применения
3. Детальное текстовое описание на основе сочетания естественного языка и программных конструкций алгоритмов всех процедур.
4. Приложение должно содержать программу на языке С, решающую поставленную задачу, и командный файл для ее трансляции на всем изучаемом семействе ОС, поддерживаемых в настоящее время, и использующий транслятор gcc.

Перечень графического материала с указанием основных чертежей и (или) иллюстративного материала

1. Блок-схема алгоритма главной процедуры
2. Скриншоты консольного окна операционной системы, в котором запущена основная программа

Графические изображения скриншотов должны отображать не менее пяти различных состояний во взаимодействиях потоков, составляющих процесс работы приложения

Методическая литература и иные информационные источники

1. Флоренсов, А.Н. Операционные системы для программиста. Омск. ОмГТУ, 2005.
2. Гордеев, А.В. Операционные системы / А. В. Гордеев. - 2-е изд. - СПб.: Питер, 2007.
3. ГОСТ 7.1-2003 Библиографическая запись. Библиографическое описание. Общие требования и правила составления
4. Рихтер Дж. Windows для профессионалов: создание эффективных Win32-приложений с учетом специфики 64-разр. версии Windows. - СПб:. Питер; М.: Русская редакция, 2001. Мэтью, Н. Основы программирования в Linux. СПб.: БХВ-Петербург, 2009.

Дата выдачи задания «__» _____ 20 __г.

Руководитель _____ к.т.н., доцент Флоренсов А.Н. 15.02.18
подпись — ученая степень, звание, ФИО — дата

Зав. кафедрой _____ д.т.н., профессор Потапов В.И. 16.02.18
подпись — ученая степень, звание, ФИО — дата

Задание принял к исполнению студент (ка) _____ «16» февраля 20 18 г.
подпись