# 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 функционирования автодрома из четырех пересекающихся замкнутых дорог |

Пояснительная записка

|  |  |
| --- | --- |
| Шифр проекта (работы) | 020-КП-09.03.01-01-ПЗ |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  | Студента (ки) | | | | Зайцев Владимир Александрович | | | | | | |  |
|  |  |  |  | | | | фамилия, имя, отчество полностью | | | | | | |  |
|  |  |  | Курс | 2 |  | | | Группа | | ИВТ-172 | | |  |  |
|  |  |  |  |  | |  | | |  |  | | |  |  |
|  |  |  | Направление (специальность) | | | | | | |  | | | |  |
|  |  |  | ***09.03.01 (правильный номер)*** | | | | | | | ***Информатика и вычислительная техника*** | | | |  |
|  |  |  | код, наименование | | | | | | | | | | |  |
|  |  |  | Руководитель | | | | доцент, к.т.н | | | | | | |  |
|  |  |  |  | | | | ученая степень, звание | | | | | | |  |
|  |  |  | Флоренсов Александр Николаевич | | | | | | | | | | |  |
|  |  |  | фамилия, инициалы | | | | | | | | | | |  |
|  |  |  | Выполнил (а) | | | |  | | | | | | |  |
|  |  |  |  | | | | дата, подпись студента (ки) | | | | | | |  |
|  |  |  | К защите | | | |  | | | | | | |  |
|  |  |  |  | | | | дата, подпись руководителя | | | | | | |  |
|  |  |  | Выполнение и подготовка к защите КП (КР) | | | | | Защита КП (КР) | | | | Итоговый рейтинг | | |
|  |  |  |  | | | | |  | | | |  | | |
|  |  |  | Проект (работа) защищен (а) с оценкой | | | | | | | |  | | |  |

Омск 2019

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 графическая модель функционирования автодрома из четырех пересекающихся замкнутых дорог | | | | | | | | | | | | | | | |  |
|  | |  | | | | | | | | | | | | | | |  |
| Студент (ка) | | | | Зайцев Владимир Александрович | | | | | | | | | | | | |  |
|  | | | | фамилия, имя, отчество полностью | | | | | | | | | | | | |  |
| Курс | 2 | | | | Группа | | | ИВТ-172 |  | | | | | | | |  |
|  | |  | | |  | | |  |  | | | | | | | |  |
| Руководитель | | | | | Флоренсов Александр Николаевич | | | | | | | | | | | |  |
|  | | | | | ученая степень, звание, ФИО | | | | | | | | | | | |  |
|  | | | | | | | | | | | | | | | | |  |
| **Содержание отзыва** | | | | | | | | | | | | | | | | |  |
|  | | | | | | | | | | | | | | | | |  |
|  | | | | | | | | | | | | | | | | |  |
|  | | | | | | | | | | | | | | | | |  |
|  | | | | | | | | | | | | | | | | |  |
|  | | | | | | | | | | | | | | | | |  |
|  | | | | | | | | | | | | | | | | |  |
|  | | | | | | | | | | | | | | | | |  |
|  | | | | | | | | | | | | | | | | |  |
| **Рейтинговые баллы за выполнение и подготовку к защите курсового проекта (работы)** | | | | | | | | | | | | |  | | | | |
| **Заключение о допуске к защите** | | | | | | | | | |  | | | | | | | |
| **Руководитель** | | | | | |  | | | | | Дата |  | | 20 |  | г. |  |

Реферат

Пояснительная записка по курсовому проекту 20 с., 1 ч., 7 рис., 3 источ, 1 прил.

C, WINDOWS, GDI, GCC, МНОГОПОТОЧНОЕ ПРИЛОЖЕНИЕ.

Объектом исследования является организация взаимодействя нескольких потоков в операционной системе Windows при работе в графическом окне.

Цель работы – разработка многопоточной графической модели имитации рабочих поездок пассажиров по круговому чартерному международному авиамаршруту.

В результате работы разработан алгоритм взаимодействия нескольких нитей между собой и отоборажения результатов их работы в графическом окне в ОС Windows. После чего была разработана сама программа, имитирующая работу одного самолета на круговом маршруте, а так же симулирующая поведение людей пользующихся этим самолетом. Все это отображается в отдельном графическом окне операционной системы Windows.

Содержание

[Введение 4](#_Toc512456690)

[1 Введение в проблематику разработки многопоточных приложений 5](#_Toc512456691)

[2 Декомпозиция разрабатываемой программы снизу-вверх с формированием основных процедур ее функционирования и описанием их функционального назначения 6](#_Toc512456692)

[3 Описание глобальных информационных объектов программы: глобальных переменных, средств синхронизации потоков и используемых структур данных в случае их применения](#_Toc512456693) 7

[4 Детальное текстовое описание на основе сочетания естественного языка и программных конструкций алгоритмов всех процедур](#_Toc512456694) 8

[Заключение](#_Toc512456695) 14

[Список использованных источников](#_Toc512456696) 15

[Приложение](#_Toc512456697) 16

Введение

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

Задача: Разработать для Windows многопоточную программную модель следующей задачи. Имеется автодром из четырех пересекающихся дорог, каждая из которых представляет собой квадрат, отображаемый в графическом окне. Дороги пересекаются с двумя соседними, так, что для каждой существуют 4 перекрестка пересечения. На каждой дороге находится один автомобиль, двигающийся со случайной скоростью. (Практически следует задавать случайную задержку между шагами отображения автомобиля). У перекрестков следует отобразить светофор, который является не самостоятельным, а только отображает занятость перекрестка проезжающими через него автомобилями. Пересечение перекрестков «без аварий» в программе обеспечить с помощью программных семафоров. Отображение динамики модели должно осуществляться в графическом окне с помощью условных схем, значков и текстов, размер окна не менее 600 на 800 пикселов. Поведение каждого автомобиля должно имитироваться с помощью отдельной нити. Для правильного взаимодействия использовать семафоры или мьютексы.

Проект состоит из четырех 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

При разработке многопоточных приложений возникает ряд проблем.

В первую очередь нужно помнить, что в зависимости от системы при компиляции может оказаться необходимым явное указание того, что данное приложение будет использовать более одной нити. Тем не менее в операционной системе Windows этот вопрос не рассматривается, т.к. там все есть по умолчанию.

При компиляции проекта можно подключить библиотеки, необходимые для работы с графикой в Windows. Так, для данного курсового проекта оказалось необходимым явное укзание того, что программа должна работать в отдельном графическом окне. Для этого при написании команды для компиляции понадобилось указание команды –mwindows.

В современных операционных системах широко используются потоки (thread). Именно они применяются и в данном курсовом проекте. Обычно в потоке выполняются действия одной из процедур программы. Хоть любой нити процесса и доступны все части программы этого процесса, реально же работа организуется так, чтобы нити отвечала отдельная процедура. Учитывая, что процедуре для нормальной работы необходимы локальные переменные, становится понятным закрепление области этих переменных за нитью. Объект хранения локальных переменных (вместе со служебной информацией при вызове подпрограмм) называют стеком. Этот стек в действительности является частью оперативной памяти, он используется не только программно, но и аппаратно.

Так же следует уделить внимание организации взаимодействия различных потоков друг с другом, а так же с памятью, к которой эти потоки могут обращаться одновременно. Дело в том, что нередко могут возникать такие ситуации, когда два или более потоков пытаются модифицировать одну и ту же область памяти. В таких случаях возникает так называемая ситуация состязания нитей, когда результаты работы их обоих перемешиваются, что в результате приводит к неопределенному поведению программы, различным ошибкам и прочим нежелательным последствиям. Одним из удобных средств для синхронизации работы нескольких потоков являются мьютексы. Их задача состоит в организации одновременного доступа к области памяти не более одного потока.

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

Поставленная задача была разделена в целях упрощения разработки и в результате были выделены следующие подзадачи:

-Алгоритм симулирования автомобилей.

-Алгоритм отображения динамики модели в графическом окне операционной системы.

-Первоначальная настройка необходимого окружения при запуске программы, настройка окна, создание пассажиров, запуск процессов.

Алгоритм поведения автомобилей находится в функции CarThread. Он предназначен для симуляции каждой отдельной машины. При этом данная функция выполняется многими запущенными процессами, каждый из которых по отдельности симулирует поведение одной машины.

Графическое отображение производится в функции Draw, с помощью графических функций из встроенного в windows.h функционала gdi (с помощью точек, многоугольных фигур и текста).

Первоначальная настройка всех необходимых компонентов для отображения и для реализации многопоточности происходит так же в функции main при старте программы перед входом в бесконечный цикл отрисовки. Там создается окно программы, создаются отдельные автомобили, создаются потоки, в которые передаетсся функция управления автомобилями, и аргумент, указывающий, где именно в общем массиве всех авто находится тот, которым должен управлять конкретный, создаваемый поток.

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

В программе используются следующие глобальные переменные:

1. hstdout — хендл стандартного вывода
2. hInstance — переменная содержащая экзепляр созданного окна
3. si — переменная содержащая информацию о старте окна
4. hwnd — дескриптор окна
5. hdc — дескриптор для отображения графики в окне
6. msg — переменная храянящая последнее переданное окном сообщение
7. brushGreen — brushRed — кисти разных цветов для закрашивания областей
8. pen — перменная содержащая информацию о том, как нужно рисовать линии
9. struct car — структура хранящая всю необходимую информацию о машине
10. struct coord – структура для хранения координат
11. struct wpoint – структура ддля хранения информации о точке следования
12. struct road – структура хранящая информацию об отедльной дорге
13. passajiry — массив со всеми пассажирами
14. mutexes — массив мьютексов для каждого перекрестка, неоходимый для блокирования доступа к перекрестку во избежание аварий.

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

Функция содержащая алгоритм автомобилей работает по принципу изменения состояний. Каждый автомобиль может находиться в одном из двух возможных состояний. Например он может ехать или ждать пока освободится перекресток. В функции происходит бесконечный цикл, который при каждом своем проходе проверяет состояние авто, и в зависимости от этого может изменять его координаты, или его состояние. Эти данные постоянно берутся из ячейки массива номер которой равен переданному при создании потока аргументу i. После произведения действий, они помещаются обратно в массив в ту же ячейку. В данной функции содержится область программы, в которой может произойти так называемая ситуация состязания процессов. Это место, где автомобили пытаются заблокировать проезд на перекрестке. Весь процесс переезда помещен между вызовами WaitForSingleObject и ReleaseMutex с параметром hmtx что гарантирует, что в один момент времени сможет ехать только один автомобиль, а остальные будут ждать до тех пор, пока не получат доступ к данному мьютексу hmtx.

Отрисовка расположена в отдельной функции Draw, вызывается в бесконечном цикле в main. Данная функция с помощью цикла for получает данные из массива с авто и по ним рисует точки на экране. Так же она берт и координаты городов. Каждый раз отрисовывая все эти объекты. Для создания иллюзии движения перед всеми этими функциями выполняется заполнение всего окна монтонным цветом, а сама функция выполняется приблизительно 30 раз в секунду.

Перед тем как войти в бесконечный цикл в функции main выполняется создание окна заданного размера. Создание мьютексов. Создание авто, создание потоков, и передача в потоки функции управления авто и числа указывающего на ячейку в массиве хранящую информацию о авто которое будет управляться данным потоком.

Схема алгоритма главной процедуры представлена на рисунке 1.

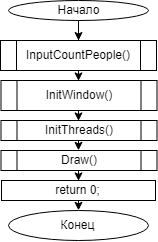


Рисунок 1 – схема алгоритма основной процедуры

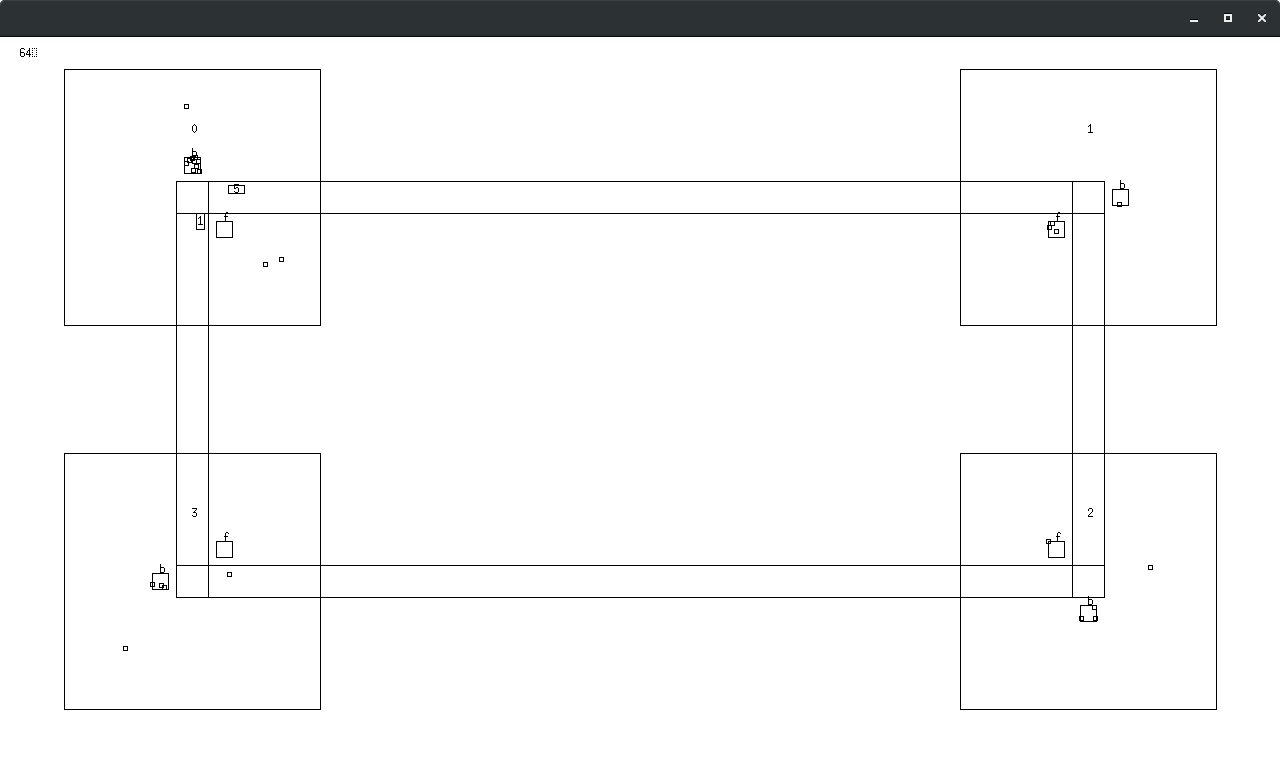
Скриншот работы программы представлен на рисунках 2-6.

Рисунок 2 – пример работы программы

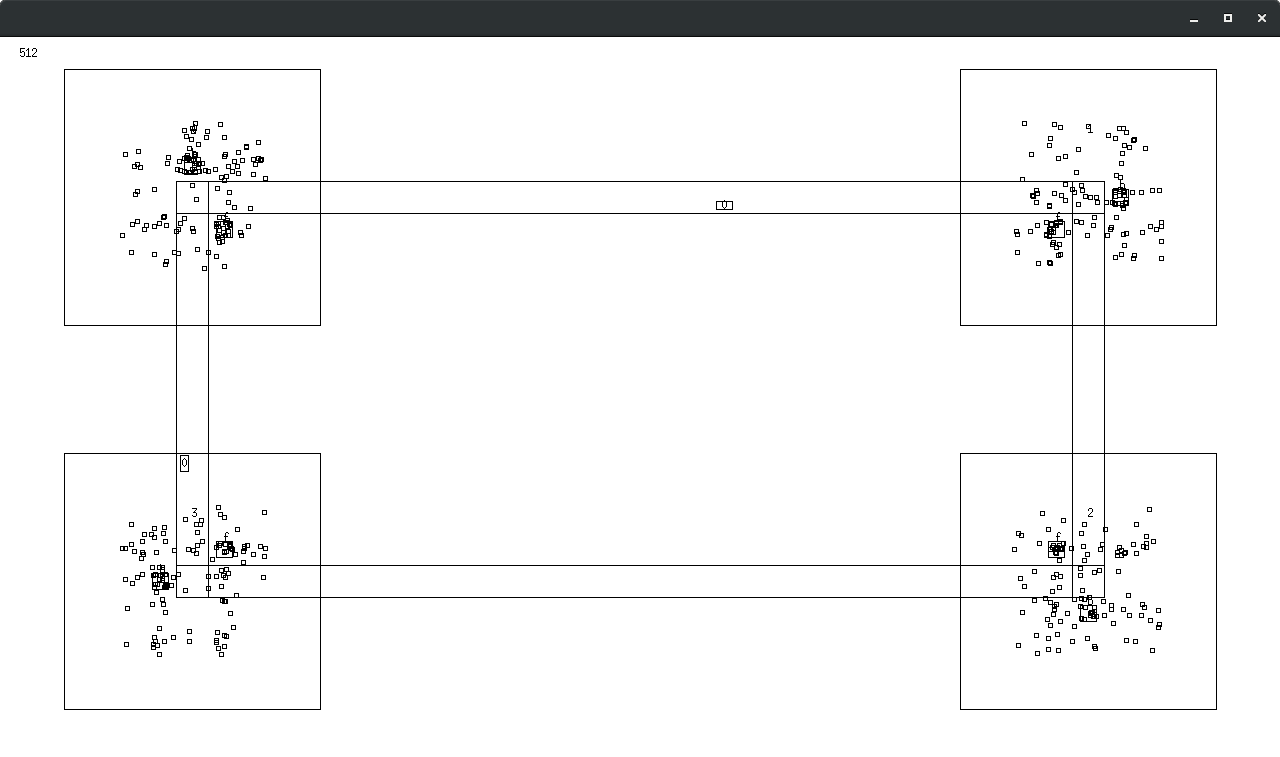


Рисунок 3 – пример работы программы

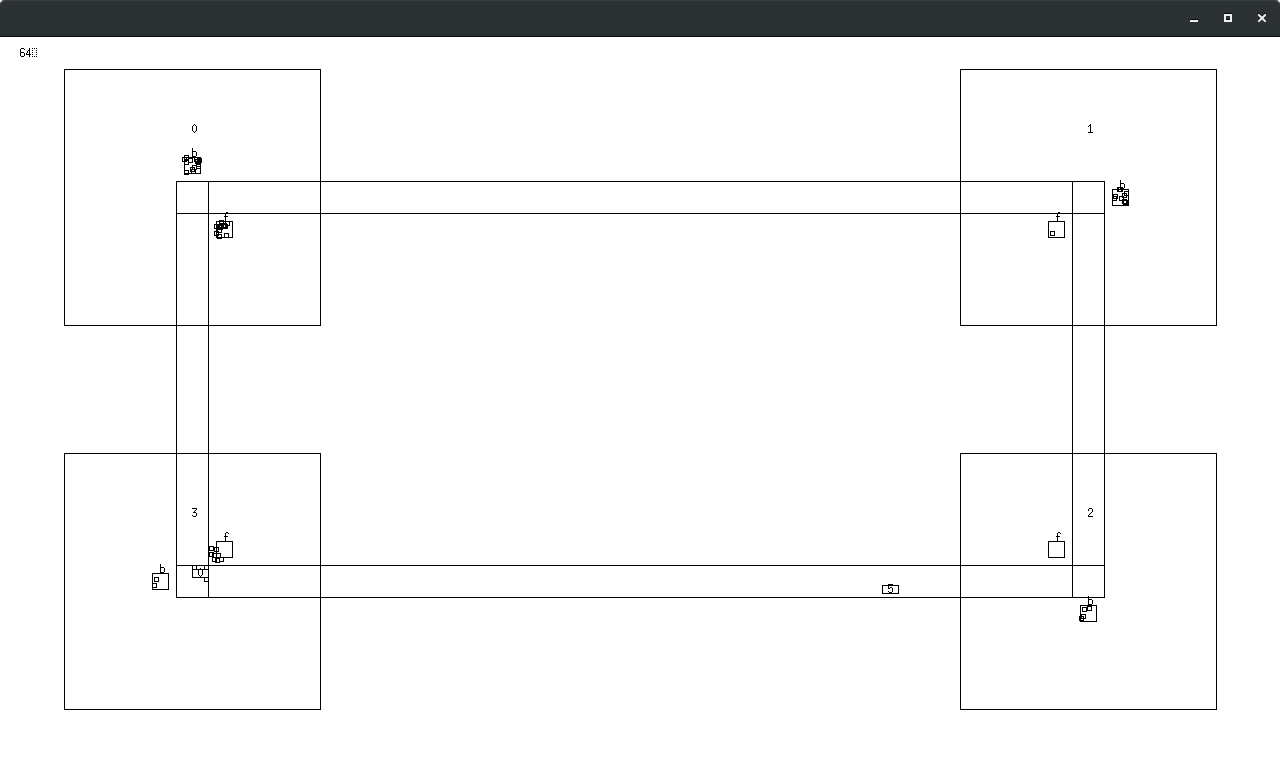


Рисунок 4 – пример работы программы

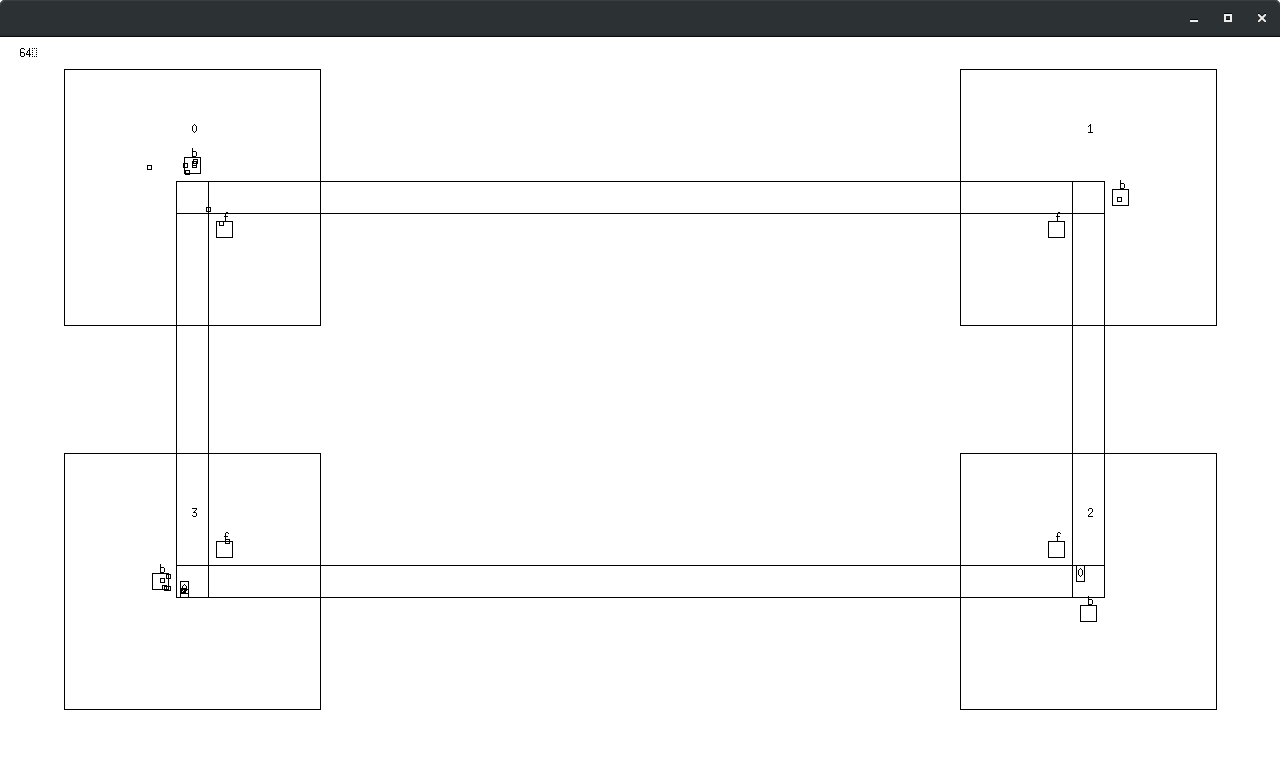
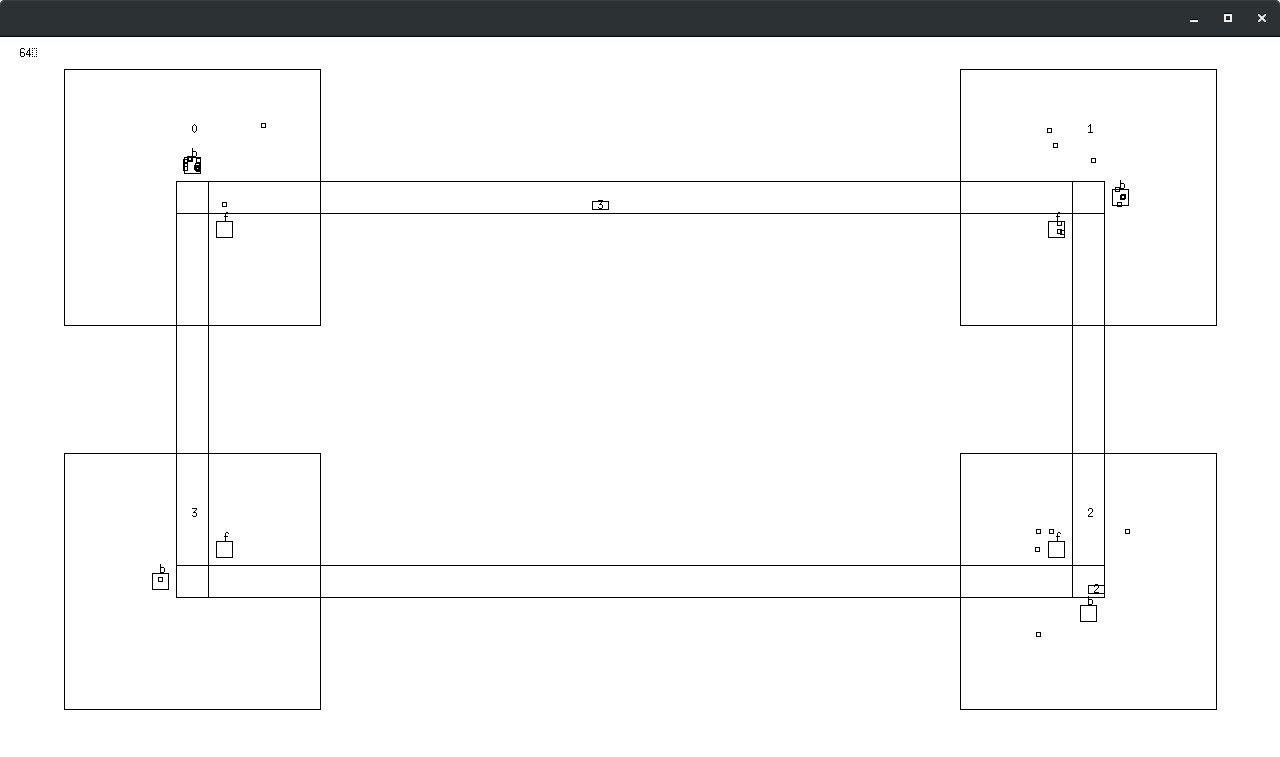
Рисунок 5 – пример работы программы

Рисунок 6 – пример работы программы

Скриншот файла для трансляции программы представлен на рисунке 7.

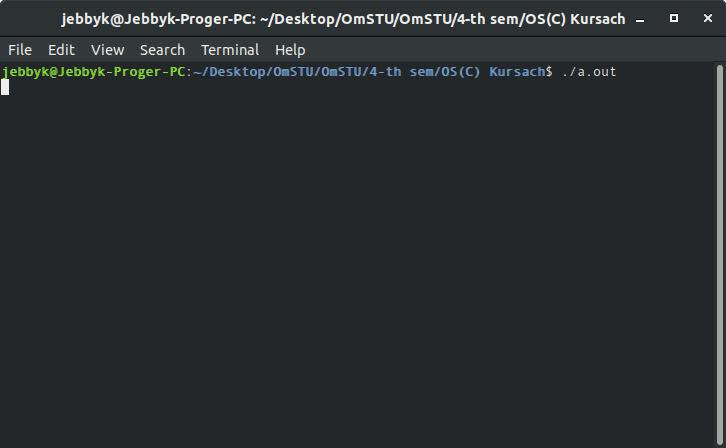


Рисунок 7 – cкриншот файла для трансляции программы

Заключение

В результате работы разработан алгоритм взаимодействия нескольких нитей между собой и отоборажения результатов их работы в графическом окне в ОС Windows. После чего была разработана сама программа, имитирующая работу одного самолета на круговом маршруте, а так же симулирующая поведение людей пользующихся этим самолетом. Все это отображается в отдельном графическом окне операционной системы Windows.

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**

#include <windows.h>

#include <stdio.h>

#include <math.h>

HANDLE hstdout;//хендл стандартного вывода

HINSTANCE hInstance;//для создания окна

STARTUPINFO si;//информация о старте окна

int nCmdShow;

HWND hwnd;//дескриптор окна

HDC hdc;//дескриптор для рисования

MSG msg;//сообщение передаваемое окном

WNDCLASS wc;//для создания окна

HBRUSH brushGreen;//кисти для заливки разными цветами

HBRUSH brushRed;

HBRUSH brushBlack;

HBRUSH brushGray;

HPEN pen;

HANDLE hmtx;

typedef struct{

float x;

float y;

}coord;

typedef struct {

float x;

float y;

short type;//1 - перекресток 0 - поворот

short open;// 0 - закртыо 1 - все открыто

}wpoint;

float s = 0.80;

wpoint wpoints[] =

{

{100, 300, 0, 1},//0

{300, 300, 1, 1},//1

{400, 300, 0, 1},//2

{400, 400, 1, 1},//3

{400, 500, 1, 1},//4

{400, 600, 0, 1},//5

{300, 600, 1, 1},//6

{100, 600, 0, 1},//7

{300, 100, 0, 1},//8

{600, 100, 0, 1},//9

{600, 300, 1, 1},//10

{600, 400, 0, 1},//11

{500, 400, 1, 1},//12

{300, 400, 0, 1},//13

{500, 300, 0, 1},//14

{800, 300, 0, 1},//15

{800, 600, 0, 1},//16

{600, 600, 1, 1},//17

{500, 600, 0, 1},//18

{500, 500, 1, 1},//19

{300, 500, 0, 1},//20

{600, 500, 0, 1},//21

{600, 800, 0, 1},//22

{300, 800, 0, 1}//23

};

HANDLE mutexes[24];

typedef struct{

short wpoints[8];

}road;

road roads[4];

void ConstructRoads()

{

road r1;

r1.wpoints[0] = 0;

r1.wpoints[1] = 1;

r1.wpoints[2] = 2;

r1.wpoints[3] = 3;

r1.wpoints[4] = 4;

r1.wpoints[5] = 5;

r1.wpoints[6] = 6;

r1.wpoints[7] = 7;

road r2;

r2.wpoints[0] = 8;

r2.wpoints[1] = 9;

r2.wpoints[2] = 10;

r2.wpoints[3] = 11;

r2.wpoints[4] = 12;

r2.wpoints[5] = 3;

r2.wpoints[6] = 13;

r2.wpoints[7] = 1;

road r3;

r3.wpoints[0] = 14;

r3.wpoints[1] = 10;

r3.wpoints[2] = 15;

r3.wpoints[3] = 16;

r3.wpoints[4] = 17;

r3.wpoints[5] = 18;

r3.wpoints[6] = 19;

r3.wpoints[7] = 12;

road r4;

r4.wpoints[0] = 20;

r4.wpoints[1] = 4;

r4.wpoints[2] = 19;

r4.wpoints[3] = 21;

r4.wpoints[4] = 17;

r4.wpoints[5] = 22;

r4.wpoints[6] = 23;

r4.wpoints[7] = 6;

roads[0] = r1;

roads[1] = r2;

roads[2] = r3;

roads[3] = r4;

}

typedef struct {

float x;

float y;

float tx;

float ty;

float ox;

float oy;

float spd;

float maxSpd;

short dir;

short state;

short r;

short curP;

}car;

car cars[4];

void CarThread(void\* \_i)

{

short i = \_i;

while(1)

{

if(cars[i].state == 0)

{

coord dir\_vector = {cars[i].tx-cars[i].x, cars[i].ty-cars[i].y};//рассчитываем вектор для движеня персонажа

float l = sqrt(pow(dir\_vector.x,2)+pow(dir\_vector.y, 2));//рассчитываем длину этого вектора

coord normal\_v = dir\_vector;//нормализованный вектор, который должен иметь то же направление, но длинну 1

normal\_v.x /= l;//для этого делим его на длину первого вектора

normal\_v.y /= l;

short nextP = cars[i].curP + 1;

if (nextP == 8) nextP = 0;

if(wpoints[roads[cars[i].r].wpoints[nextP]].open == 0)

{

if(cars[i].spd < cars[i].maxSpd && cars[i].spd < l/40)//если скорость персонажа меньше 0.5 и в принципе достаточно млаа

{

cars[i].spd += cars[i].maxSpd/40;//то постеменно ее увеличиваем

}

if(cars[i].spd > l/40)//если скорость слишком велика по отношению к оставшемуся расстоянию до цели,

{

cars[i].spd = l/40;//то снижаем ее

}

}else{

if(cars[i].spd < cars[i].maxSpd && cars[i].spd < l)//если скорость персонажа меньше 0.5 и в принципе достаточно млаа

{

cars[i].spd += cars[i].maxSpd/40;//то постеменно ее увеличиваем

//if(cars[i].spd > l) cars[i].spd = l;

}

}

cars[i].x += normal\_v.x \* cars[i].spd;//изменяем координаты персонажа на основании его скорости и нормализованного вектора длвижения

cars[i].y += normal\_v.y \* cars[i].spd;

// if(wpoints[roads[cars[i].r].wpoints[cars[i].curP]].type == 1)

//{

coord old\_vector = {cars[i].ox-cars[i].x, cars[i].oy-cars[i].y};//рассчитываем вектор для движеня персонажа

float ol = sqrt(pow(old\_vector.x,2)+pow(old\_vector.y, 2));

if(ol > 64) {

ReleaseMutex(mutexes[roads[cars[i].r].wpoints[cars[i].curP]]);

wpoints[roads[cars[i].r].wpoints[cars[i].curP]].open = 1;

}

// }

if(l < 2)//если персонаж остановился

{

if(wpoints[roads[cars[i].r].wpoints[nextP]].open == 0) cars[i].spd = 0;

cars[i].state = 1;

cars[i].curP++;

if(cars[i].curP == 8) cars[i].curP = 0;

}

}

if(cars[i].state == 1)

{

short nextP = cars[i].curP + 1;

if(nextP == 8) nextP = 0;

// if(wpoints[roads[cars[i].r].wpoints[cars[i].curP]].type == 1)

//{

WaitForSingleObject(mutexes[roads[cars[i].r].wpoints[cars[i].curP]], INFINITE);

wpoints[roads[cars[i].r].wpoints[cars[i].curP]].open = 0;

/// }

coord dir\_vector = {wpoints[roads[cars[i].r].wpoints[nextP]].x-cars[i].x, wpoints[roads[cars[i].r].wpoints[nextP]].y-cars[i].y};//рассчитываем вектор для движеня персонажа

if(dir\_vector.x \* dir\_vector.x > dir\_vector.y \* dir\_vector.y)

{

if(dir\_vector.x > 0) cars[i].dir = 0;

else cars[i].dir = 2;

}

else{

if(dir\_vector.y > 0) cars[i].dir = 1;

else cars[i].dir = 3;

}

switch(cars[i].dir)

{

case 0:

cars[i].tx = wpoints[roads[cars[i].r].wpoints[nextP]].x -32;//точки остановки перед перекрестками

cars[i].ty = wpoints[roads[cars[i].r].wpoints[nextP]].y;

break;

case 1:

cars[i].tx = wpoints[roads[cars[i].r].wpoints[nextP]].x;

cars[i].ty = wpoints[roads[cars[i].r].wpoints[nextP]].y -32;

break;

case 2:

cars[i].tx = wpoints[roads[cars[i].r].wpoints[nextP]].x + 32;

cars[i].ty = wpoints[roads[cars[i].r].wpoints[nextP]].y;

break;

case 3:

cars[i].tx = wpoints[roads[cars[i].r].wpoints[nextP]].x;

cars[i].ty = wpoints[roads[cars[i].r].wpoints[nextP]].y + 32;

break;

}

cars[i].state = 0;

cars[i].ox = wpoints[roads[cars[i].r].wpoints[cars[i].curP]].x;

cars[i].oy = wpoints[roads[cars[i].r].wpoints[cars[i].curP]].y;

}

Sleep(1000/120);

}

}

void SetCars(){

for(int i = 0; i < 4; i++)

{

cars[i].r = i;

cars[i].x = wpoints[roads[cars[i].r].wpoints[0]].x;

cars[i].y = wpoints[roads[cars[i].r].wpoints[0]].y;

cars[i].tx = wpoints[roads[cars[i].r].wpoints[1]].x;

cars[i].ty = wpoints[roads[cars[i].r].wpoints[1]].y;

cars[i].state = 0;

cars[i].spd = 0;

int r = rand()%40;

cars[i].maxSpd = 0.1 + r/10.0f;

cars[i].dir = 0;

cars[i].curP = 0;

HANDLE hthr;

hthr = (HANDLE)\_beginthread(CarThread, 0,(void\*)i);

}

}

void Draw()

{

SelectObject(hdc, brushBlack);

Rectangle(hdc, 0,0,800,800);//монотонная заливка всего экрана

for(int i = 0; i < 4; i++)

{

SelectObject(hdc, brushGray);

if(cars[i].dir == 0 || cars[i].dir == 2) Rectangle(hdc, cars[i].x\*s - 16, cars[i].y\*s - 8, cars[i].x\*s + 16, cars[i].y\*s + 8);

if(cars[i].dir == 1 || cars[i].dir == 3) Rectangle(hdc, cars[i].x\*s - 8, cars[i].y\*s - 16, cars[i].x\*s + 8, cars[i].y\*s + 16);

for(int j = 0; j < 8; j++)

{

SelectObject(hdc, pen);

if(j == 0)

{

MoveToEx(hdc, wpoints[roads[i].wpoints[j]].x\*s, wpoints[roads[i].wpoints[j]].y\*s, NULL);

}

else{

LineTo(hdc, wpoints[roads[i].wpoints[j]].x\*s, wpoints[roads[i].wpoints[j]].y\*s);

}

if(j == 7) LineTo(hdc, wpoints[roads[i].wpoints[0]].x\*s, wpoints[roads[i].wpoints[0]].y\*s);

// if(wpoints[roads[i].wpoints[j]].type == 1) {

if(wpoints[roads[i].wpoints[j]].open == 1)

{

SelectObject(hdc, brushGreen);

Rectangle(hdc, wpoints[roads[i].wpoints[j]].x\*s - 8, wpoints[roads[i].wpoints[j]].y\*s - 8, wpoints[roads[i].wpoints[j]].x\*s + 8, wpoints[roads[i].wpoints[j]].y\*s + 8);

}else{

SelectObject(hdc, brushRed);

Rectangle(hdc, wpoints[roads[i].wpoints[j]].x\*s - 8, wpoints[roads[i].wpoints[j]].y\*s - 8, wpoints[roads[i].wpoints[j]].x\*s + 8, wpoints[roads[i].wpoints[j]].y\*s + 8);

}

//}

}

}

}

void main()

{

for(int i = 0; i < 24; i++)

{

mutexes[i] = CreateMutex(NULL, FALSE, NULL);//мьютеексы для всех перекрестков

}

LRESULT WINAPI WinProc(HWND hwnd, UINT tmsg, WPARAM wParam, LPARAM lParam)//создание окна и считывание его событий

{

switch(tmsg)

{

case WM\_DESTROY:

{

PostQuitMessage(0);

exit(0);

return 0;

}

case WM\_KEYDOWN:

{

return 0;

}

}

return DefWindowProc(hwnd, tmsg, wParam, lParam);

}

GetStartupInfo(&si);

if(si.dwFlags & STARTF\_USESHOWWINDOW)

{

nCmdShow = si.wShowWindow;

}else{

nCmdShow = SW\_SHOWDEFAULT;

}

hInstance = GetModuleHandle(NULL);

memset(&wc, 0, sizeof(wc));

wc.lpszClassName = "MyClass";

wc.lpfnWndProc = WinProc;

wc.hCursor=LoadCursor(NULL, IDC\_ARROW);

wc.hbrBackground=(HBRUSH)(COLOR\_WINDOW+1);

wc.hInstance = hInstance;

if(!RegisterClass(&wc)) return;

hwnd = CreateWindow("MyClass", "OurWindow", WS\_OVERLAPPEDWINDOW, 0, 0, 800, 800, 0, 0, hInstance, NULL);

hdc = GetDC(hwnd);

ShowWindow(hwnd, nCmdShow);

brushGreen = CreateSolidBrush(RGB(0,255, 0));

brushRed = CreateSolidBrush(RGB(255,0, 0));

brushBlack = CreateSolidBrush(RGB(32,32, 32));

brushGray = CreateSolidBrush(RGB(128,128,128));

pen = CreatePen(PS\_SOLID, 1, RGB(255, 255, 255));

ConstructRoads();

SetCars();

while(1)

{

Draw();

if(PeekMessage(&msg, hwnd, 0, 0, PM\_REMOVE))//мониторим события окна

{

DispatchMessage(&msg);

}

Sleep(1000/30);

}

}