# 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 Linux графическая модель транспортных перевозок дилижансами по двум круговым маршрутам между четырьмя городами. |

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 работы бензозаправочной станции с тремя колонками | | | | | | | | | | | | | | | |  |
|  | |  | | | | | | | | | | | | | | |  |
| 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 24 с., 1 ч., 7 рис., 3 источ, 1 прил.

C, WINDOWS, 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 работы бензозаправочной станции с тремя колонками.

В результате работы разработан алгоритм взаимодействия нескольких нитей между собой и отображения результатов их работы в графическом окне в ОС Windows. После чего была разработана сама программа, имитирующая работу бензозаправочной станции с тремя колонками.

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 модель следующей задачи.На станцию со случайным интервалом заезжают N автомашин и заправляются из любой незанятой в этот момент колонки. После заправки водитель направляется в кассу и, подождав, если у кассы наоходятся люди и он не может сразу же притиснуться к ней, оплачивает бензин. Затем возвращается к автомобилю и уезжает со станции, освобождая место. Модель должна иметь минимальное отображение ситуации с условными изображениями колонок и кассы. Сами машины отображаются в окне одной из латинских букв. Перемещение условного изображения машины к колонке желательно, перемещение водителя между кассой и автомобилем не требуется, но следует как то отображать наличие толпящихся в кассу. Ожидание событий следует организовать на снове семафоров и функции ожидания события. Программа должна перед началом работы вводить значение числа N. Отображение динамики модели должно осуществляться в графическом окне с помощью условных схем, значков и текстов, размер окна не менее 600 на 800 пикселей. Поведение каждого водителя 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 водителей пользующихся заправкой.

-Алгоритм отображения динамики модели в графическом окне операционной системы.

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

Алгоритм поведения водителей находится в бесконечном цикле в функции voditelFunction. Он предназначен для симуляции каждого отдельного водителя и его автомобиля. При этом данная функция выполняется многими запущенными процессами, каждый из которых по отдельности симулирует поведение одного пассажира.

Графическое отображение производится в функции paintFunction, 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 пассажиры, создаются потоки, в которые передается функция управления водителям, и аргумент, указывающий, где именно в общем массиве всех водителей 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. hwnd — дескриптор окна
4. hdc — дескриптор для отображения графики в окне
5. msg — переменная храянящая последнее переданное окном сообщение
6. voditeliChisloC — массив из четырех символов, куда записывается вводимое число водителей
7. voditeliChislo — в эту переменную сохранятеся число водителей преобразованное из строки из предыдущей переменной
8. curDigit — номер текущей вводимой цифры
9. pen — перменная содержащая информацию о том, как нужно рисовать линии
10. kolonka1Pos — kolonka4Pos — позиции городов
11. kolonka1 – kolonka 3 – состояние колонки(занято/не занято)
12. oblastForma, gorodForma — многоугольники всех городов с разными формами
13. struct voditel — структура хранящая всю необходимую информацию о водителе
14. voditeli — массив со всеми водителями
15. hmtx — мьютекс для ограничения доступа пассажиров к самолету в необходимый момент, чтобы в самолете не оказалось более 4 пассажиров.

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

Функция содержащая алгоритм водителей работает по принципу изменения состояний. Каждый водитель может находиться в одном из нескольких возможных состояний. Например он может ехать к колонке, идти к кассе, ждать в очереди или ехать с колонки. В функции происходит бесконечный цикл, который при каждом своем проходе проверяет состояние водителя, 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 arg. 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 водители обновляют информацию об очереди в кассу. Весь процесс посадки помещен между вызовами WaitForSingleObject и ReleaseMutex с параметром hmtx что гарантирует, что в один момент времени изменить данные об очереди в кассу сможет только один водитель, а остальные будут ждать до тех пор, пока не получат доступ к данному мьютексу hmtx.

Отрисовка расположена в отдельной функции paintFunction. Она вызывается в бесконечном цикле в main. Данная функция с помощью цикла for получает данные из массива с водителями по ним рисует прямоугольники на экране. Так же она берт координаты колонок и координаты кассы. Каждый раз отрисовывая все эти объекты. Для создания иллюзии движения перед всеми этими функциями выполняется заполнение всего окна монотонным 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 мьютекса. Опрос пользователя о количесве создаваемых водителей. Создание водителей, создание потоков, и передача в потоки функции управления водителями и числа указывающего на ячейку в массиве хранящую информацию о водителе который будет управляться данным

Схема алгоритма главной процедуры представлена на рисунке 1.

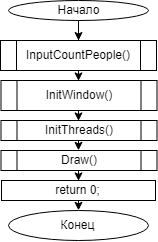


Рисунок 1 – схема алгоритма основной процедуры

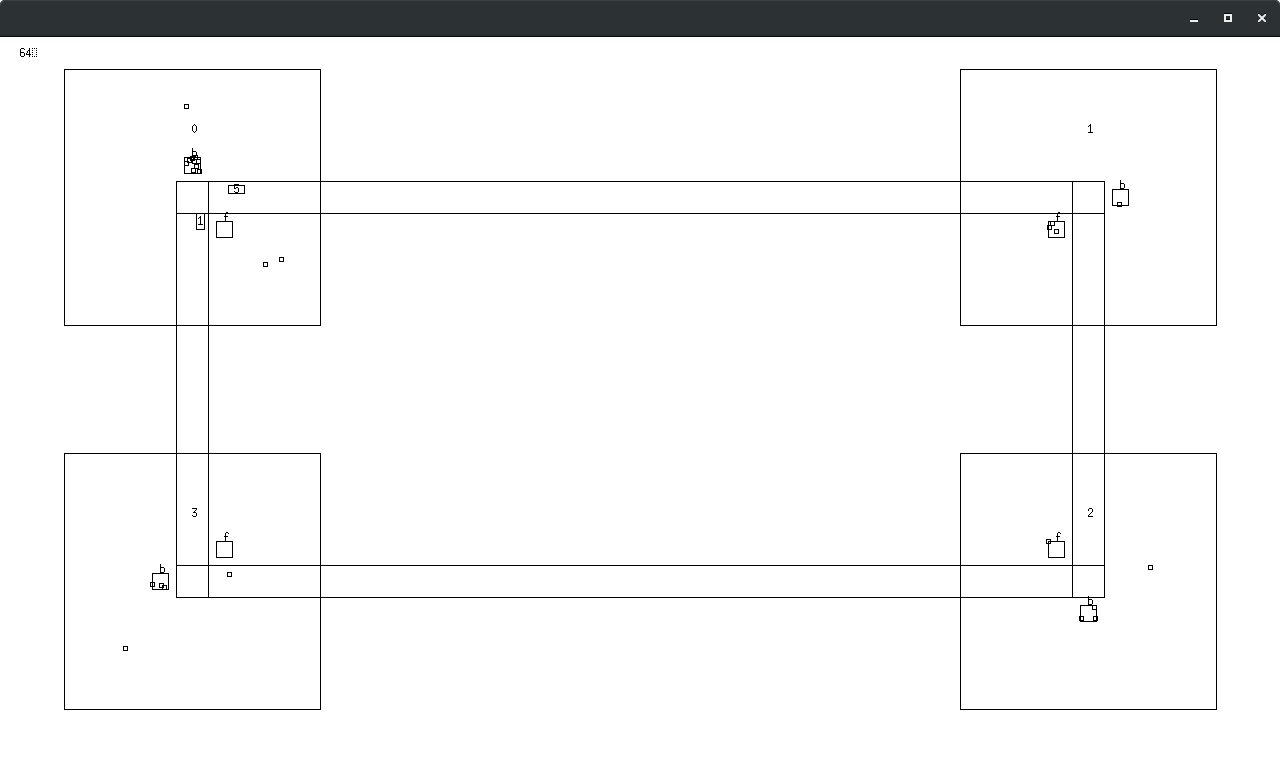
Скриншоты работы программы представлен на рисунках 2-6.

Рисунок 2 – пример работы программы

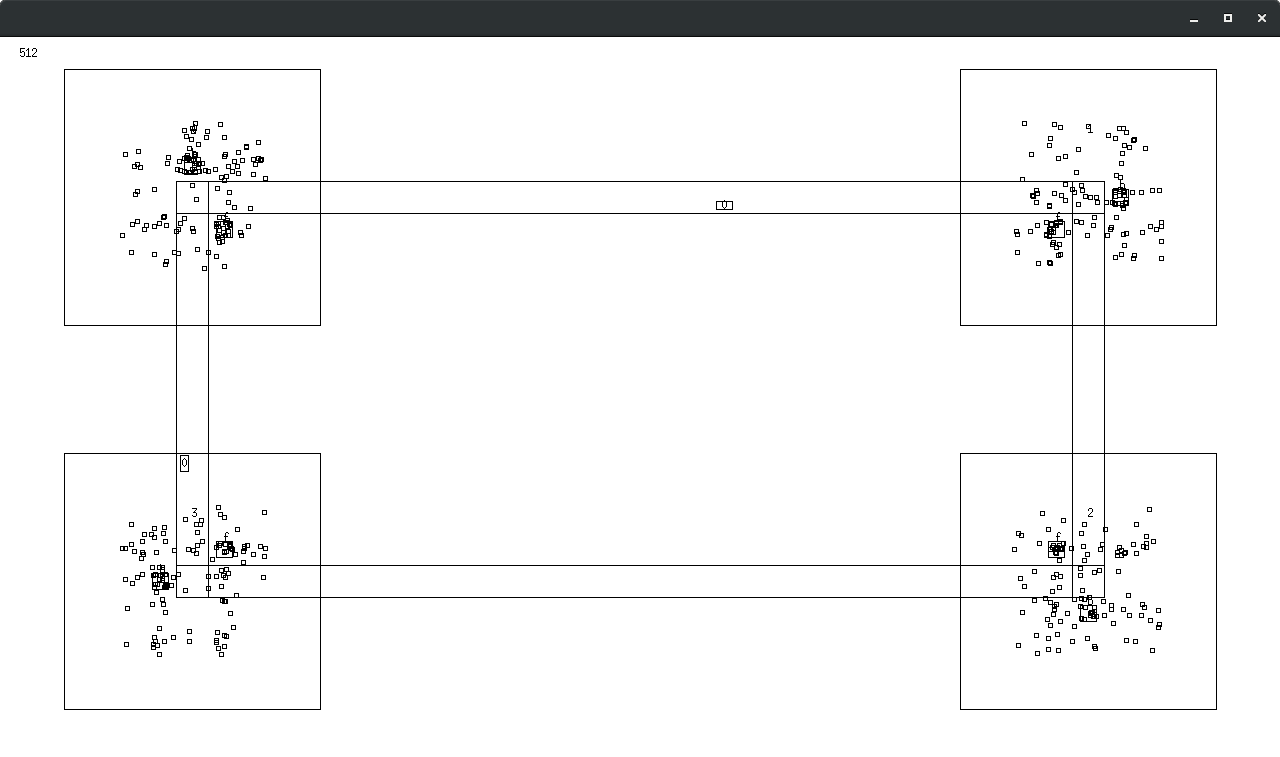


Рисунок 3 – пример работы программы

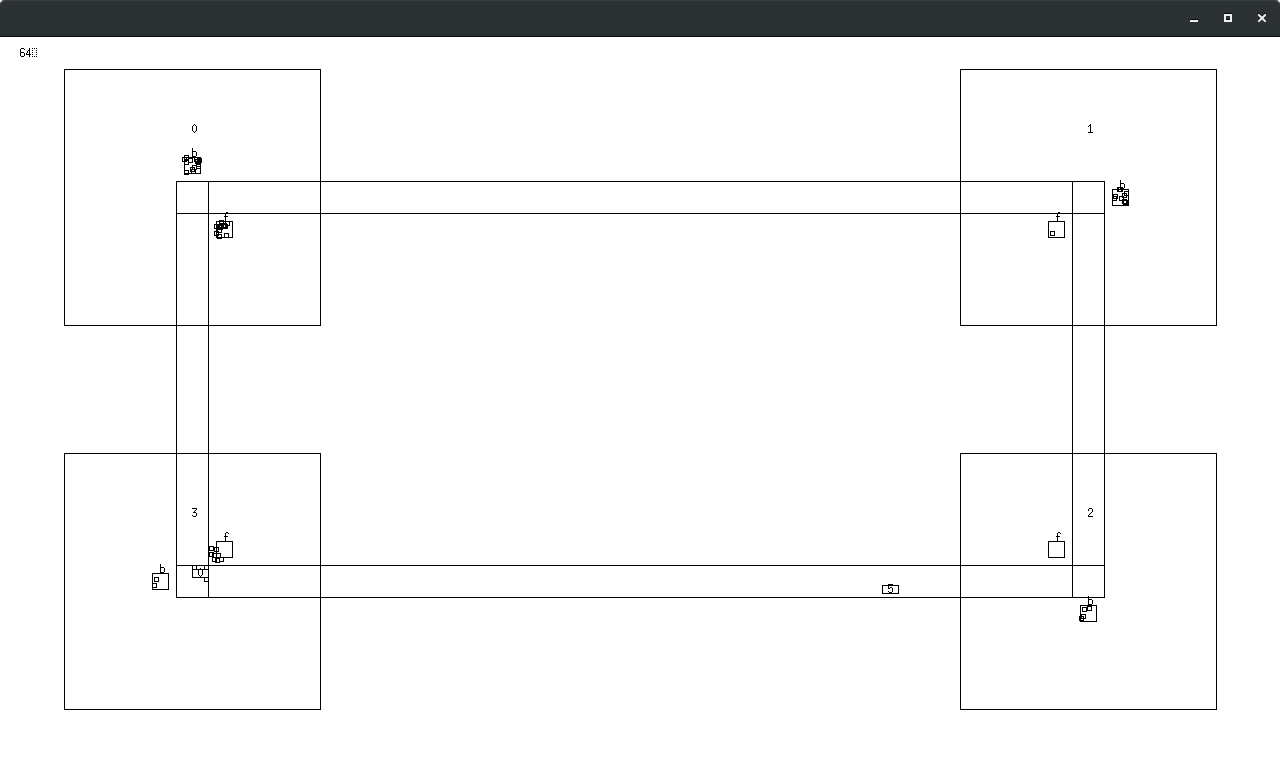


Рисунок 4 – пример работы программы

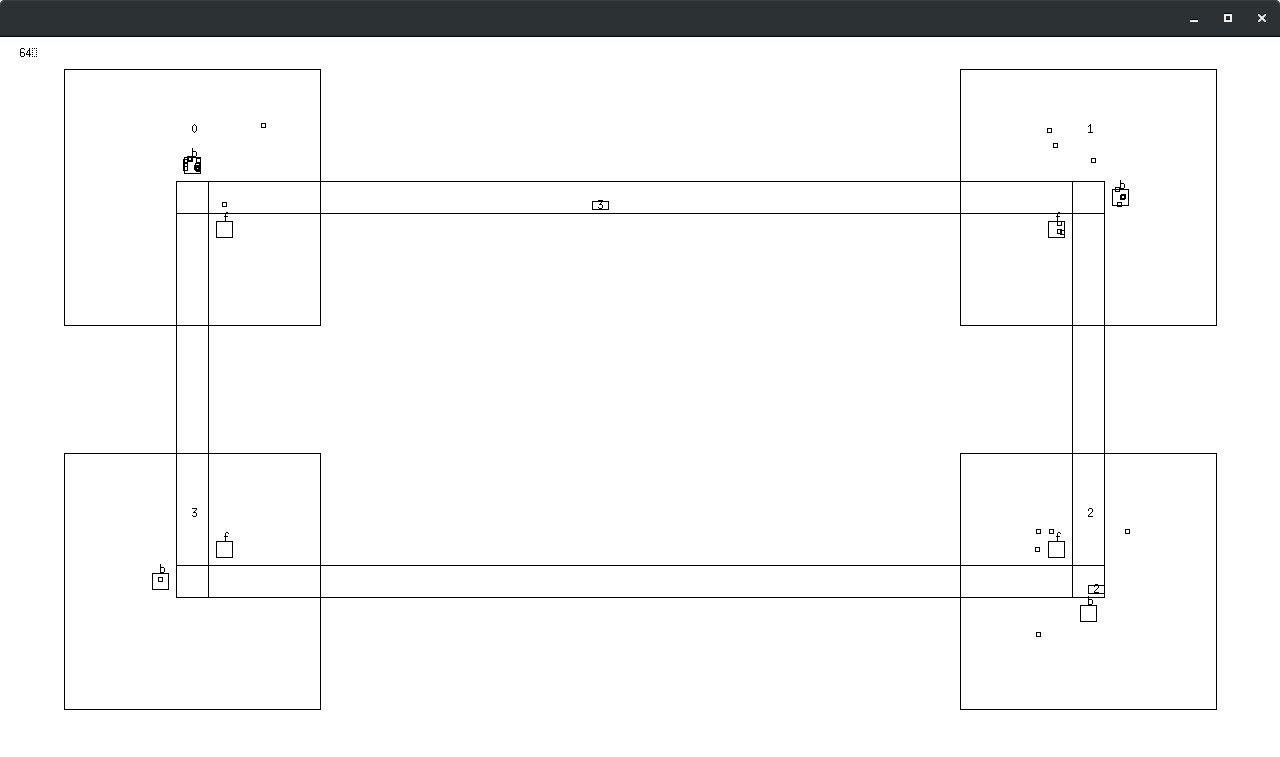


Рисунок 5 – пример работы программы

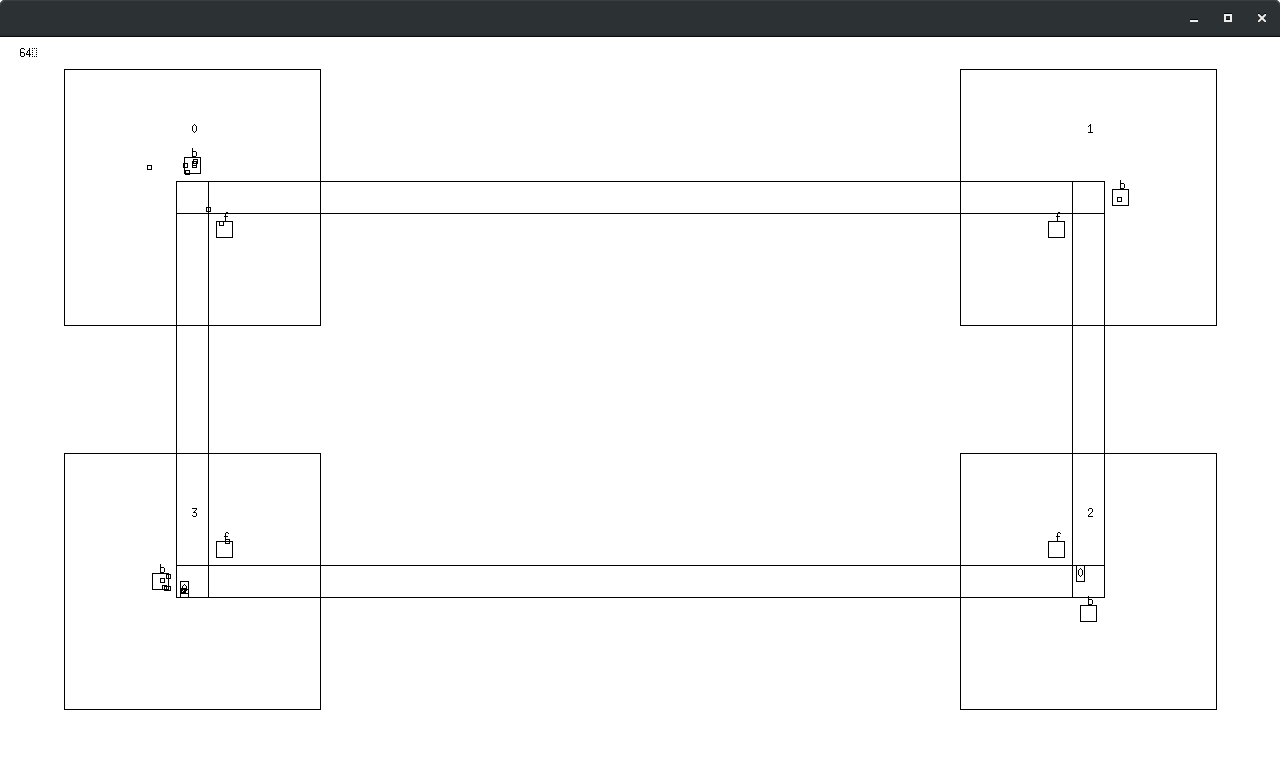


Рисунок 6 – пример работы программы

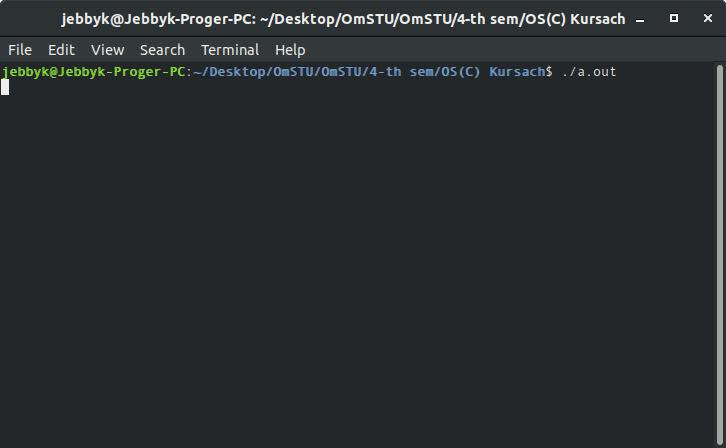
Скриншот файла для трансляции программы представлен на рисунке 7.

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 бензозаправочной станции с тремя колонками, а так же симулирующая поведение людей пользующихся этой станцией. 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

char voditeliChisloC[4] = "";//строка куда вводится число пассажиров

int voditeliChislo = 1;//число в котором преобразуется число водятлов из строки в цифру

int curDigit = 0;//для вода числа водятлов (тукущий вводимый разряд числа)

HPEN pen;//карандаш для рисования линий

int tScorost = 60;//скорость и плавность анимации (если мигает - делать ниже)

char text[12] = "";

POINT kolonka1Pos = {128, 256};

POINT kolonka2Pos = {256, 256};

POINT kolonka3Pos = {384, 256};

int kolonka1;//состояния колонок (заняты/не заняты)

int kolonka2;

int kolonka3;

POINT kassaPos = {512,256};

int kassaLudi = 0;

HANDLE hmtx;

typedef struct {//структура с данными об отдельном водителе

float x;//точные координаты (тоже для перемещения)

float y;

float mx;//координаты машины

float my;

int cx;//целевая координата

int cy;

int walkKassa;//к кассе

int walkMachina;//к машине

int machina;//в машине (уезжает с колонки)

int ojidanie;// ждет

int son;//спит

int driveKolonka;//к колонке

float skorost;

int curKolonka;//текущая колонка (к которой едем и у которой стоим и тд.)

}voditel;

voditel voditeli[1024];//массив с пассажирами

void voditelFunction(void \*arg)//функция пассажира (с ней тред запускается)

{

srand(arg);

int i = (int)arg;//передаем аргумент, по которому определяем какой именно элемент массива модифицеровать

voditeli[i].curKolonka = rand()%3;

if(voditeli[i].curKolonka==0)

{

voditeli[i].cx = kolonka1Pos.x; //указываем начальные целевые координаты

voditeli[i].cy = kolonka1Pos.y;

}

if(voditeli[i].curKolonka==1)

{

voditeli[i].cx = kolonka2Pos.x; //указываем начальные целевые координаты

voditeli[i].cy = kolonka2Pos.y;

}

if(voditeli[i].curKolonka==2)

{

voditeli[i].cx = kolonka3Pos.x; //указываем начальные целевые координаты

voditeli[i].cy = kolonka3Pos.y;

}

voditeli[i].x = voditeli[i].cx;//устанавливаем водителя с его машиной в определенную позицию

voditeli[i].y = 800 + rand()%800;

voditeli[i].driveKolonka = 1;//едем к колонке

while(1)

{

if(voditeli[i].driveKolonka == 1)

{

float dx;

float dy;

dx = voditeli[i].cx - voditeli[i].x;

dy = voditeli[i].cy - voditeli[i].y;

float l = sqrt(dx\*dx + dy\*dy);

dx /= l;

dy /= l;//перемещение по вектору

if(voditeli[i].skorost < l/10)

{

voditeli[i].skorost+=0.1;//ускорение до опр предела

}else{

voditeli[i].skorost = l/10;

}

if(voditeli[i].skorost > 5)

{

voditeli[i].skorost = 5;//торможение если близко к цели

}

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

{

float dy;

dy = voditeli[i].y - voditeli[n].y;

if(n!=i && dy <= 64 && dy > 0 && voditeli[i].curKolonka == voditeli[n].curKolonka)

{

voditeli[i].skorost = ((dy-42)/10)-0.1;

}

}

voditeli[i].x += dx\*voditeli[i].skorost;

voditeli[i].y += dy\*voditeli[i].skorost;

voditeli[i].mx = voditeli[i].x;

voditeli[i].my = voditeli[i].y;

if(voditeli[i].x > voditeli[i].cx -4 && voditeli[i].x < voditeli[i].cx + 4 && voditeli[i].y > voditeli[i].cy -4 && voditeli[i].y < voditeli[i].cy + 4)

{

WaitForSingleObject(hmtx,INFINITE);

if(voditeli[i].curKolonka == 0 ) kolonka1 = 1;

if(voditeli[i].curKolonka == 1 ) kolonka2 = 1;

if(voditeli[i].curKolonka == 2 ) kolonka3 = 1;

kassaLudi++;

voditeli[i].cx = kassaPos.x;

voditeli[i].cy = kassaPos.y + 4 + kassaLudi \* 4;

voditeli[i].walkKassa = 1;//меняем его состояние на ожидание, если он очень близко к целевой точке

voditeli[i].driveKolonka = 0;

voditeli[i].skorost = 0;

ReleaseMutex(hmtx);

}

}

if(voditeli[i].walkKassa == 1)//если пассажир сейчас идет в касу

{

float dx;

float dy;

dx = voditeli[i].cx - voditeli[i].x;//берем вектор по направлению к целевой точке

dy = voditeli[i].cy - voditeli[i].y;

float l = sqrt(dx\*dx + dy\*dy);// узнаем рассатояние

if(l > 1.0f)

{

dx /= l;//нормализуем вектор

dy /= l;

}

voditeli[i].x += dx;//перемещаем человека

voditeli[i].y += dy;

if(voditeli[i].x > voditeli[i].cx -1 && voditeli[i].x < voditeli[i].cx + 1 && voditeli[i].y > voditeli[i].cy -1 && voditeli[i].y < voditeli[i].cy + 1)

{

voditeli[i].walkKassa = 0;//меняем его состояние на ожидание, если он очень близко к целевой точке

voditeli[i].ojidanie = 1;

}

}

if(voditeli[i].ojidanie == 1)//если сейчас ожидание

{

float dx;

float dy;

dx = voditeli[i].cx - voditeli[i].x;//берем вектор по направлению к целевой точке

dy = voditeli[i].cy - voditeli[i].y;

float l = sqrt(dx\*dx + dy\*dy);// узнаем рассатояние

if(l > 1.0f)

{

dx /= l;//нормализуем вектор

dy /= l;

}

voditeli[i].x += dx;//перемещаем человека

voditeli[i].y += dy;

dx = kassaPos.x - voditeli[i].x;

dy = kassaPos.y - voditeli[i].y;

l = sqrt(dx\*dx + dy\*dy);//так же беерм вектор до касы узнаем расстояние

if(l < 10 )//если оно очень маленькое то "типа платим в кассу в течении 2 секунд"

{

voditeli[i].ojidanie = 0;

voditeli[i].son = 1;

Sleep(2000);

WaitForSingleObject(hmtx,INFINITE);

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

{

if(voditeli[n].ojidanie == 1)

{

voditeli[n].cy -= 4;

}

if(voditeli[n].walkKassa == 1)

{

voditeli[n].cy -= 4;

}

}

kassaLudi--;

voditeli[i].son = 0;

voditeli[i].walkMachina = 1;

voditeli[i].cx = voditeli[i].mx;

voditeli[i].cx = voditeli[i].mx;

ReleaseMutex(hmtx);

}

}

if(voditeli[i].walkMachina == 1)//если идем к машине

{

float dx;

float dy;

dx = voditeli[i].cx - voditeli[i].x;//такое же перемещение по вектору до точки

dy = voditeli[i].cy - voditeli[i].y;

float l = sqrt(dx\*dx + dy\*dy);

if(l > 1.0f)

{

dx /= l;

dy /= l;

}

voditeli[i].x += dx;

voditeli[i].y += dy;

if(voditeli[i].x > voditeli[i].cx -1 && voditeli[i].x < voditeli[i].cx + 1 && voditeli[i].y > voditeli[i].cy -1 && voditeli[i].y < voditeli[i].cy + 1)

{

if(voditeli[i].curKolonka == 0 ) kolonka1 = 0;

if(voditeli[i].curKolonka == 1 ) kolonka2 = 0;

if(voditeli[i].curKolonka == 2 ) kolonka3 = 0;

voditeli[i].walkMachina = 0;

voditeli[i].machina = 1;

voditeli[i].cx = voditeli[i].x;

voditeli[i].cy = -100;

}

}

if(voditeli[i].machina == 1)//если уезжаем с колонки

{

float dx;

float dy;

dx = voditeli[i].cx - voditeli[i].x;

dy = voditeli[i].cy - voditeli[i].y;

float l = sqrt(dx\*dx + dy\*dy);

dx /= l;

dy /= l;//перемещение по вектору

if(voditeli[i].skorost < l/10)

{

voditeli[i].skorost+=0.1;//ускорение до опр предела

}else{

voditeli[i].skorost = l/10;

}

if(voditeli[i].skorost > 5)

{

voditeli[i].skorost = 5;//торможение если близко к цели

}

voditeli[i].x += dx\*voditeli[i].skorost;

voditeli[i].y += dy\*voditeli[i].skorost;

voditeli[i].mx = voditeli[i].x;

voditeli[i].my = voditeli[i].y;

if(voditeli[i].x > voditeli[i].cx -1 && voditeli[i].x < voditeli[i].cx + 1 && voditeli[i].y > voditeli[i].cy -1 && voditeli[i].y < voditeli[i].cy + 1)

{

Sleep(2000);

voditeli[i].curKolonka = rand()%3;

if(voditeli[i].curKolonka==0)

{

voditeli[i].cx = kolonka1Pos.x; //указываем начальные целевые координаты

voditeli[i].cy = kolonka1Pos.y;

}

if(voditeli[i].curKolonka==1)

{

voditeli[i].cx = kolonka2Pos.x; //указываем начальные целевые координаты

voditeli[i].cy = kolonka2Pos.y;

}

if(voditeli[i].curKolonka==2)

{

voditeli[i].cx = kolonka3Pos.x; //указываем начальные целевые координаты

voditeli[i].cy = kolonka3Pos.y;

}

voditeli[i].machina = 0;

voditeli[i].driveKolonka = 1;

voditeli[i].x = voditeli[i].cx;

voditeli[i].y = 800 + rand()%800;

}

}

Sleep(1000/tScorost);//задержка между отедльными кадрами анимации

}

}

void paintFunction()//фнукция рисования

{

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

Rectangle(hdc, kolonka1Pos.x + 16, kolonka1Pos.y - 8, kolonka1Pos.x + 32, kolonka1Pos.y + 8 );

Rectangle(hdc, kolonka2Pos.x + 16, kolonka2Pos.y - 8, kolonka2Pos.x + 32, kolonka2Pos.y + 8 );

Rectangle(hdc, kolonka3Pos.x + 16, kolonka3Pos.y - 8, kolonka3Pos.x + 32, kolonka3Pos.y + 8 );

Rectangle(hdc, kassaPos.x + 8, kassaPos.y - 64, kassaPos.x + 64, kassaPos.y + 64 );

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

{

Rectangle(hdc, voditeli[i].mx-8, voditeli[i].my-16, voditeli[i].mx+8, voditeli[i].my+16);//рисуем машины

Rectangle(hdc, voditeli[i].x-2, voditeli[i].y-2, voditeli[i].x+2, voditeli[i].y+2);//рисуем маленьких человечков

}

}

void main()

{

hmtx = CreateMutex(NULL, 0, NULL);//главный и единственный мьютекс

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:

{

hdc = GetDC(hwnd);

char key[1];

key[0] = (char)wParam;

if(curDigit == -1)

{

exit(0);

}

if(key[0] >= '0' && key[0] <= '9')

{

TextOut(hdc, 128+64+16+curDigit\*8, 32, key, 1);

voditeliChisloC[curDigit] = key[0];

curDigit++;

}else{

curDigit = -1;

}

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);//тут заканчивается создание окна

pen = CreatePen(PS\_SOLID, 1, RGB(0,0,0));

char text1[] = "Vvedite chislo voditelov";//ввод числа человечков

TextOut(hdc, 32, 32, text1, sizeof(text1));

while(GetMessage(&msg, 0, 0, 0) && curDigit > -1)

{

DispatchMessage(&msg);

}

sscanf(voditeliChisloC, "%d", &voditeliChislo);//преобразование полученной строки в цифру

for(int i = 0; i < voditeliChislo; i++)//создание потоков с человечками

{

voditel vod1;//создаем водятла, настраиваем его параметры

voditeli[i] = vod1;//ложим его в масссива

HANDLE hthr;

hthr = (HANDLE)\_beginthread(voditelFunction, 0,(void\*)i);//непосредественнно запуск потока и передаем номер ячейки (i) массива с этим водитлом

}

while(1)

{

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

{

DispatchMessage(&msg);

}

paintFunction();//рисуем все

Sleep(1000/tScorost);//задержка между кадрами

}

}