МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования

«ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ
И РАДИОЭЛЕКТРОНИКИ» (ТУСУР)

Кафедра автоматизации обработки информации (АОИ)

УТВЕРЖДАЮ
Заведующий кафедрой АОИ
д-р техн. наук, проф.
_____Ю.П. Ехлаков
«___»_____2016 г.

Информатика и программирование

методические указания к курсовому проекту для студентов направления 09.03.04 – «Программная инженерия» заочной формы обучения

Разработчик
ст. преподаватель каф. АОИ
_____Н.В. Пермякова
«___»_____2016 г.

Содержание

1. Введение

Курсовой проект по дисциплине «Информатика и программирование» имеет целью: получение навыков самостоятельной разработки программного продукта в соответствии с принципами структурного программирования, рассмотренными в процессе изучения дисциплины.

Для выполнения курсового проекта студент получает индивидуальное задание – описание задачи, для решения которой разрабатывается и реализуется программный продукт. Описания задач приведены в приложении В настоящих методических указаний.

2. Общие требования к выполнению курсового проекта

Вариант индивидуального задания выбирается по следующему алгоритму: два последних номера зачетной книжки делятся на 10. Полученный остаток от деления принимается в качестве номера варианта.

Студент самостоятельно разрабатывает структуру программы – определяет типы входных и выходных данных, разрабатывает алгоритмы решения предложенных задач, определяет функциональные блоки будущего программного продукта и реализует предложенную задачу на языке Си/Си++.

Программный продукт должен иметь дружественный интерфейс. Такой интерфейс может быть реализован с помощью системы меню. Пример меню дан в приложении Б. В приложении 1 описаны функции для работы с текстовым экраном.

Студент должен провести полное тестирование своей программы и описать в пояснительной записке тестовые данные и результат тестирования.

Требования к оформлению пояснительной записки приведены в пункте 4.

3. Содержание курсового проекта

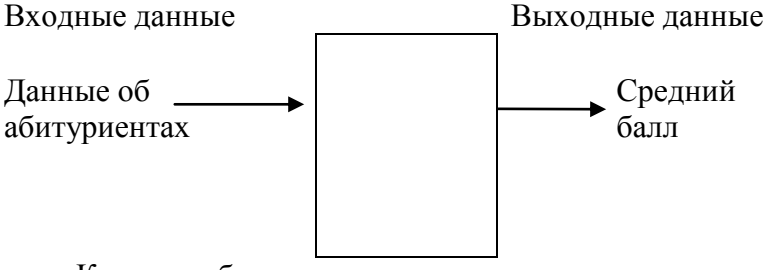
В основной части проекта излагаются проектные решения, соответствующие следующим этапам разработки:

1. Разработка проекта.

1.1. Описание структуры входных и выходных данных.

По своему варианту задания выделите данные, являющиеся входными для задачи, и данные, являющиеся выходными. Укажите взаимосвязи между входными и выходными данными.

Рассмотрим описание структуры данных на примере функции, вычисляющей средний балл абитуриентов ВУЗа.



Каждого абитуриента в этом случае характеризуют данные:

- Фамилия
- Имя
- Отчество

- Балл_математика
- Балл_физика
- Балл_русский_язык

1.2. Разработка алгоритма решения задачи.

Описание алгоритма (или алгоритмов) решения задачи. Описание может быть выполнено в словесной форме, или выполнено в виде блок-схемы.

Приведем пример описания алгоритма получения ведомости.

Шаг 1. *n* – количество абитуриентов.

Шаг 2. Прочитать данные об абитуриентах в массив *Abiturient.*

Шаг 3. *S* – сумма баллов всех абитуриентов

Шаг 4. *Mball – S/(3*n)*

Шаг 5. Вывести информацию об абитуриентах и среднем балле в текстовый файл.

1.3. Определение формы представления входных и выходных данных.

На этом этапе определяют типы для входных и выходных данных и форму их хранения.

Например:

Данные об абитуриенте – структура, имеющая 6 полей:

- Фамилия, Имя, Отчество – строковые переменные,
- Балл_математика, Балл_физика, Балл_русский_ язык – целочисленные переменные.

Данные хранятся в текстовом файле.

Средний балл – вещественная переменная, вычисляется при выполнении соответствующей функции.

1.4. Разработка структуры программы.

На этом этапе определяют функциональную структуру программы, описывают будущие функции программы и их взаимосвязь.

Например:

Функция *Abiturient* Read(const char*, int*)* – функция возвращает массив данных об абитуриентах, параметры функции – имя текстового файла, содержащего данные, адрес переменной, по которому записывается размерность считанного массива. Текстовый файл организован следующим образом: на каждой строке файла через пробел указаны Имя, Фамилия, Отчество и баллы, полученные на экзаменах по математике, физике и русскому языку.

Функция *void Print (Abiturient* , int, char*)* – функция формирования ведомости, параметры функции – массив данных об абитуриентах, размерность этого массива, имя файла для сохранения ведомости.

2. Разработка программы.

2.1. Программирование и отладка.

2.2. Формирование тестовых данных.

Этап формирования тестовых данных для проверки работоспособности проектируемого программного обеспечения.

Например: создание файлов с данными об абитуриентах. При этом могут быть созданы файлы, использование которых приведет к исключительным ситуациям – неверно заполненные файлы, пустые файлы и тому подобное.

2.3. Тестирование программы.

Тестовые данные для проверяемой системы должны быть сохранены в текстовом файле. Если программная система обрабатывает какие-либо массивы данных, массив должен состоять не менее, чем из 15 элементов.

3. Разработка программной документации.

3.1. Описание структуры программы.

На этом этапе описывается взаимодействие ранее описанных функций программы.

На рис. 3.1. представлена схема взаимодействия функций *Read* и *Print.*

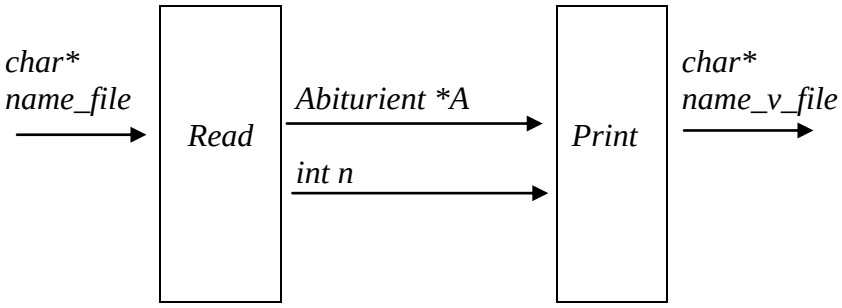


Рис. 3.1. Взаимодействие функций системы

3.2. Руководство пользователя.

Описание правил пользования программным средством, правил формирования входных данных.

4. Оформление курсового проекта

Оформление пояснительной записки должно соответствовать Образовательному стандарту ОС ТУСУР 6.1.-97.

Рекомендуется следующее содержание отчета:

- титульный лист,
- задание на проектирование,
- введение,
- основная часть,
- заключение,
- список использованных источников.

Пример оформления титульного листа приведен в приложении В.

Введение должно содержать цель работы, ее значение.

Основная часть пояснительной записки должна отражать выполнение всех этапов работы, составляющих содержание курсового проекта, описанных в пункте 3.

Заключение должно содержать краткие выводы по результатам выполненной работы. Список использованных источников оформляется согласно стандарту.

5. Список литературы

При работе над проектом возможно использование следующей дополнительной литературы:

1. Павловская Т. А. C/C++. Программирование на языке высокого уровня для магистров и бакалавров : учебник для вузов / Т. А. Павловская. - СПб. : ПИТЕР, 2012. - 461 с. (Экземпляры всего: 3).

2. Хабибуллин И. Ш. Программирование на языке высокого уровня C/C++ : учебное пособие для вузов / И. Ш. Хабибуллин. - СПб. : БХВ-Петербург, 2006. - 485[13] с. : (Экземпляры всего: 6).

3. Иванов Б. Н. Дискретная математика. Алгоритмы и программы : Учебное пособие для вузов / Б. Н. Иванов. - М. : Лаборатория Базовых Знаний, 2003. - 288 с. (Экземпляры всего: 50).

4. Новиков Ф. А. Дискретная математика для программистов : Учебное пособие для вузов / Ф. А. Новиков. - 2-е изд. - СПб. ; М. ; Нижний Новгород : Питер, 2007. - 363[5] с. (Экземпляры всего: 80).

5. Вирт Н. Алгоритмы и структуры данных (с примерами на Паскале) : пер. с англ. / Н. Вирт ; пер. Д. Б. Подшивалов. - 2-е изд., испр. . - СПб. : Невский диалект, 2007. - 351[1] с. (Экземпляры всего: 1).

Приложение А

Описание функций для работы с текстовым экраном (заголовочный файл *conio.h*)

*char *cgets(char *str)*– читает строку с консоли в *str*.

void clreol(void) – удаляет все символы от позиции курсора до конца строки в текущем текстовом окне без передвижения курсора.

void clrscr()– очищает текущее текстовое окно и перемещает курсор в верхний левый угол (1,1).

*int cprintf(const char *format [,argument,...])* – форматированный вывод на экран. Каждый *argument* выводится в соответствии со спецификатором формата, содержащимся в строке, *format*. Вывод осуществляется в текущее текстовое окно. Число аргументов и спецификаторов должны совпадать.

*int cputs(const char *str)* – пишет строку *str*, завершенную нулем, в текущее текстовое окно.

*int cscanf(char *format [,address, ...])* – форматный ввод с консоли. Посимвольно читает последовательность входных полей с консоли. Каждое поле форматируется в соответствии со спецификацией формата, передаваемой *cscanf* в строке формата, на которую указывает *format*. Записывает форматированный ввод по адресам, переданным в качестве аргументов, следующих за *format*. Ввод отображается на экране. Количество спецификаций формата и адресов должно совпадать с количеством входных полей.

void delline(void) – удаляет строку, в которой находится курсор и сдвигает все строки, находящиеся ниже нее, на строку вверх. Работает с текущим активным окном.

int getch(void) – читает один символ непосредственно с клавиатуры без отображения на экран. Результат работы функции - считанный символ

int getche(void) – читает один символ непосредственно с клавиатуры и отображает его в текущем текстовом окне, используя прямой вывод в видеопамять или сервис BIOS. Результат работы функции - считанный символ

*char *getpass(const char *prompt)* – выводит на системную консоль подсказку *prompt* (строку символов), отключает вывод на экран и читает пароль. Возвращает указатель на строку, завершаемую нуль-символом и состоящую не более чем из восьми символов.

*int gettext(int left, int top, int right, int bottom, void *destin)* –
сохраняет текстовое содержимое прямоугольной области экрана, определенной аргументами *left, top, right u bottom* в области памяти, на которую указывает *destin*. Все экранные координаты являются абсолютными (не зависящими от текущего окна). Координаты верхнего левого угла (1,1).Функция читает содержимое области экрана в память последовательно, слева направо и сверху вниз. Каждая позиция на экране занимает 2 байта в памяти: первый байт содержит символ, второй - его видеоатрибут. Объем памяти, требуемый для прямоугольной области шириной *w* колонок и высотой *h* рядов, определяется следующим образом:

*число байтов = (h рядов) * (w столбцов) * 2*

Если операция прошла успешно, возвращается 1. В случае ошибки (например, если заданы недопустимые для текущего экранного режима координаты) возвращается 0.

*void gettextinfo(struct text_info *r)* – помещает информацию о текущем текстовом изображении в структуру типа *text_info*; *r* – указатель на эту структуру. Структура *text_info* определена в *conio.h* следующим образом:

```
struct text_info {
unsigned char winleft;     /*левый угол окна*/
unsigned char wintop;    /*верхний угол окна*/
unsigned char  winright;  /*правый угол окна*/
unsigned char winbottom;  /*нижний угол окна*/
unsigned char attribute; /*текстовый атрибут*/
unsigned char normattr; /*нормальный атрибут*/
```

unsigned char currmode; /*BW40, BW80, C40,
C80, или C4350*/
unsigned char screenheight; /*высота экрана*/
unsigned char screenwidth; /*ширина экрана*/
unsigned char curx; /*x-координата
в текущем окне */
unsigned char cury; /*y-координата
в текущем окне */ };

void gotoxy(int x,int y) – помещает курсор в заданную позицию в текущем текстовом окне. Если задаются недопустимые значения координат (например, вызывается *gotoxy(40,30),* когда нижний правый угол окна находится в позиции (35,25)), вызов *gotoxy* игнорируется.

void highvideo(void) – выбирает повышенную яркость символов путем установки бита повышенной яркости для текущего цвета переднего плана.

void insline(void) – вставляет пустую строку в текстовом ок не в позиции курсора, используя текущий цвет фона для текста. Все строки, которые находятся ниже, сдвигаются на одну строку вниз, а последняя строка сдвигается за пределы окна. Эта функция используется в текстовом режиме.

int kbhit(void) – позволяет определить, была ли нажата клавиша. Любая нажатая клавиша может быть получена с помощью функций *getch* или *getche*. Если какая-либо клавиша была нажата, возвращается ненулевое значение, в противном случае возвращается нуль.

void lowvideo(void) – выбирает пониженную яркость символов путем установки бита пониженной яркости для текущего цвета переднего плана.

int movetext(int left, int right, int bottom, int destleft, int desttop) – копирует содержимое окна определенного размера (*left,top,right* и *bottom)* в новое окно тех же размеров. *destleft* и *desttop* – позиция левого верхнего угла нового окна. Все координаты - это абсолютные экранные координаты. Если прямоугольные области перекрываются, копирование будет происходить корректно. Возвращает не ноль, если операция успешна. Если неудача (например, если заданные координаты выходят за пределы текущего режима экрана), *movetext* возвращает 0.

void normvideo(void) – выбирает нормальную яркость символов путем установки бита нормальной яркости для текущего цвета переднего плана.

int putch(int c) – выводит символ с в текущее текстовое окно. Эта функция работает в текстовом режиме и осуществляет прямой вывод на консоль. При успешном завершении *putch* возвращает напечатанный символ *c*. В случае ошибки возвращается *EOF*.

*int puttext(int left, int top, int right, int bottom, void *source)* –
выводит содержимое области памяти в прямоугольную область экрана, определенную параметрами *left, top, right* и *bottom. source* - указатель на исходную область. Все координаты являются абсолютными экранными координатами, не зависящими от текущего окна. Верхний левый угол находится в точке (1,1). *puttext* помещает содержимое области памяти в указанный прямоугольник последовательно слева направо и сверху вниз. Функция возвращает ненулевое значение, если операция завершилась успешно; в случае неудачи возвращается 0 (например, если заданные значения координат не допустимы для текущего режима экрана).

void _setcursortype(int cur_t) – устанавливает тип курсора в один из следующих:

_NOCURSOR – выключает курсор
_SOLIDCURSOR – блок-курсор
_NORMALCURSOR – обычный курсор.

void textattr(int newattr) – позволяет установить как основной цвет, так и цвет фона посредством одного вызова. Обычно для этого вызывают функции *textcolor* и *textbackground.*

Эта функция не оказывает влияния на символы, уже находящиеся на экране. Информация о цвете закодирована в параметре *newattr* следующим образом:

7	6	5	4	3	2	1	0
B	*b*	*b*	*b*	*f*	*f*	*f*	*f*

Параметр *newattr* (8 бит)*:*

- *ffff* – четырех-битовый основной цвет (от 0 до 15).
- *bbb* – трех-битовый фоновый цвет (от 0 до 7).
- *B* – бит мерцания.

void textbackground(int newcolor) – выбирает фоновый цвет. Эта функция предназначена для тех функций, которые осу ществляют прямой вывод на экран в текстовом режиме. В *newcolor* указывается новый цвет фона. Можно задать *newcolor* как целое число от 0 до 7 или как символьную константу, определенную в *conio.h.* После вызова, *textbackground,* все последующие функции, осуществляющие прямой вывод на экран (например, *cprintf*), будут использовать *newcolor.*

В следующей таблице описаны символьные константы и числовые значения допустимых цветов:

Символьная константа	Числовое значение
BLACK (черный)	0
BLUE (синий)	1
GREEN (зеленый)	2
CYAN (бирюзовый)	3
RED (красный)	4
MAGENTA (малиновый)	5
BROWN (коричневый)	6
LIGHTGRAY (светло-серый)	7

void textcolor(int newcolor) – выбирает основной цвет (цвет символов). Эта функция предназначена для тех функций, которые осуществляют прямой вывод на экран в текстовом режиме. В *newcolor* указывается новый основной цвет. Можно задать *newcolor* как целое число или как символьную константу, определенную в *conio.h*. Дополнительно к цветам, определенным для *textbackground* можно использовать следующие цвета:

Символьная константа	Числовое значение
DARKGRAY (темно-серый)	8
LIGHTBLUE (светло-синий)	9
LIGHTGREEN (светло-зеленый)	10
LIGHTCYAN (светло-бирюзовый)	11
LIGHTRED (светло-красный)	12
LIGHTMAGENTA (светло-малиновый)	13
YELLOW (желтый)	14
WHITE (белый)	15
BLINK (мерцание)	128

Можно сделать символы мерцающими, добавив 128 к основному цвету. Для этой цели существует предопределенная константа *BLINK*, например: *textcolor(CYAN + BLINK);*

int wherex(void) – возвращает *x*-координату текущей позиции курсора (внутри текущего текстового окна).

Функция *wherex* возвращает целое число в диапазоне от 1 до 80.

int wherey(void) – возвращает *x*-координату текущей позиции курсора (внутри текущего текстового окна).

Функция *wherey* возвращает целое число в диапазоне от 1 до 25.

void window(int left,int top, int right, int bottom) –определяет текстовое окно на экране. Если задаются неправильные координаты, вызов функции *window* игнорируется. *left* и *top* - экранные координаты левого верхнего угла окна. *right* и *bottom* - экранные координаты правого нижнего угла окна. Минимальный размер текстового окна: 1 столбец и 1 строка. Размер текстового окна, принятый по умолчанию, соответствует всему экрану с координатами: 1,1,80,25.

Приложение Б.

Пример программы, реализующий оконное меню

```
#include <conio.h>
#include <stdio.h>
#include <dos.h>
#include <bios.h>
#include <string.h>
#include <stdlib.h>
// условие передвижения по пунктам меню
#define STOP  if (kod<0) kod=0;if (kod>=kodmax) kod=kodmax-1;
#define VIDEO 0x10 // определение текстового режима
//Шаблон структуры Окно
typedef struct {
    int LtRow, LtCol;//координаты верхнего левого угла
    int Width;      //ширина окна
    int Height;     //высота окна
    int TypteBorder; //тип рамки
    char *Titl;     //заголовок
    int SimvolColor; //цвет символов
    int Background;  //цвет фона
    int n;          //количество тем меню
    char **Text;    //текстовое содержание окна
    } Menu_W;
// пункты меню
char *mnu1[]={ "Тема 1  \n",
            "Тема 2  \n",
            "Тема 3  \n"};
// описание прототипов функций
//-------------------------------------------------------
  int ShowWindow(Menu_W Menu);
//-------------------------------------------------------
  int Select_tema(Menu_W Menu,int kod);
//-------------------------------------------------------
  int Navig(Menu_W Menu);
//-------------------------------------------------------

 main()
  {
  union REGS regs;
  regs.h.ah = 1;
  regs.h.ch = 32;
  int86(VIDEO, &regs, &regs); //Курсор невидим


  Menu_W Menu= {1,1,79,24,0,"",RED,BLUE,0,NULL},
    Menu1= {20,5,16,5,0,"Заголовок1",BLACK,LIGHTGRAY,3,mnu1},
    Menu2= {1,25,79,0,0,"Подсказка",RED,LIGHTGRAY,0,NULL};

  ShowWindow(Menu);
  ShowWindow(Menu2);
//   gotoxy(2,25);
```

```
  // printf("Подсказка");
   ShowWindow(Menu1);
   Navig(Menu1);
 }
//-----------------------------------------------------
// Нарисовать окно с меню
  int ShowWindow(Menu_W Menu)
{ window(Menu.LtRow,Menu.LtCol,Menu.LtRow+Menu.Width,
  Menu.LtCol+Menu.Height);
  textcolor(Menu.SimvolColor);
  textbackground(Menu.Background);
  clrscr();
  gotoxy((1+Menu.Width/2-strlen(Menu.Titl)/2),1);
  cprintf("%s",Menu.Titl);
  int i;
  if(Menu.Text!=NULL)
    {gotoxy(2,2);
    for(i=0;i<Menu.n;i++)
      { cprintf("%s",Menu.Text[i]);
        gotoxy(2,2+i+1);
       }
      }
   return 0;
 }
//-----------------------------------------------------
// Отобразить активный пункт меню
int Select_tema(Menu_W Menu,int kod)
  {gotoxy(2,kod+2);
   textcolor(abs(Menu.SimvolColor-15));
   textbackground(abs(Menu.Background-15));
   cprintf("%s",Menu.Text[kod]);
   return 0;
  }
//-----------------------------------------------------
// Отобразить неактивный пункт меню
  int tema(Menu_W Menu,int kod)
  {gotoxy(2,kod+2);
   textcolor(Menu.SimvolColor);
   textbackground(Menu.Background);
   cprintf("%s",Menu.Text[kod]);
   return 0;
  }
//-----------------------------------------------------------
// передвижение по меню
int Navig(Menu_W Menu)
{
/* установить номер активного пункта (kod), максимально возможный номер пункта
(kodmax) */
 int kod=0,kodmax=Menu.n;
// отобразить активный пункт меню
 Select_tema(Menu,kod);
 union KEY {int i;
```

```
          char ch[2];
         }press;
int flag=1;
do {
press.i=bioskey(0);  // прием с ожиданием
if (press.ch[0])      // нажата ASCII-клавиша
  { switch (press.ch[0])
    { case 27: flag=0;break;
      case 13: /*вызов выбранной функции*/;break;
    }
  }
else      //Нажата специальная клавиша
  { switch (press.ch[1])
    {// передвижение в верх по меню
      case 72: {tema(Menu,kod);
               kod-=1;STOP;
               Select_tema(Menu,kod);;
               break;
              }
// передвижение вниз по меню
      case 80:{tema(Menu,kod);
               kod+=1; STOP;
               Select_tema(Menu,kod);;
               break;
              }
    }
  }

} while (flag==1);
 return 0;
}
```

Приложение В

Варианты заданий на курсовой проект

1. Система **Платежи. Клиент** имеет **Счет** в банке и **Кредитную карту (КК). Клиент** может оплатить **Заказ,** сделать платеж на другой **Счет,** заблокировать **КК** и аннулировать **Счет. Администратор** может заблокировать **КК** за превышение кредита.
2. Система **Больница. Пациенту** назначается лечащий **Врач. Врач** может сделать назначение **Пациенту(**процедуры, лекарства, операции**). Медсестра** или другой **Врач** выполняют назначения. **Пациент** может быть выписан из **Больницы** по окончании лечения, при нарушении режима или иных обстоятельствах.
3. Система **Вступительные экзамены. Абитуриент** регистрируется на **Факультет,** сдает **Экзамены. Преподаватель** выставляет **Оценку.** Система подсчитывает средний балл и определяет **Абитуриентов,** зачисленных в учебное заведение.
4. Система **Библиотека. Читатель** оформляет **Заказ** на **Книгу.** Система осуществляет поиск в **Каталоге. Библиотекарь** выдает **Книгу Читателю** на абонемент или в читальный зал. При невозвращении **Книги Читателем** в срок он может быть занесен **Администратором** в «черный список».
5. Система **Конструкторское бюро.** Заказчик предоставляет **Техническое задание (ТЗ)** на проектирование многоэтажного **Дома**. **Конструктор** регистрирует **ТЗ**, определяет стоимость проектирования и строительства, выставляет **Заказчику Счет** за проектирование и создает **Бригаду Конструкторов** для выполнения проекта.
6. Система **Телефонная станция**. **Абонент** оплачивает **Счет** за разговоры и **Услуги**, может попросить **Администратора** сменить номер и отказаться от **Услуг. Администратор** изменяет номер, **Услуги** и временно отключает **Абонента** за неуплату.
7. Система **Автобаза. Диспетчер** распределяет заявки на **Рейсы** между **Водителями** и назначает для этого **Автомобиль. Водитель** может сделать заявку на ремонт. **Диспетчер** может отстранить **Водителя** от работы. **Водитель** делает отметку о выполнении **Рейса** и состоянии **Автомобиля.**
8. Система **Интернет-магазин**. **Администратор** добавляет информацию о **Товаре**. **Клиент** делает и оплачивает **Заказ** на **Товары**. **Администратор** регистрирует **Продажу** и может занести неплательщиков в «черный список».
9. Система **«Железнодорожная касса»**. **Пассажир** делает **Заявку** на станцию назначения, время и дату поездки. Система регистрирует **Заявку** и осуществляет поиск подходящего **Поезда. Пассажир** делает выбор **Поезда** и получает **Счет** на оплату. **Администратор** вводит номера **Поездов**, промежуточные и конечные станции, цены.
10. Система **Городской транспорт**. На **Маршрут** назначают **Автобус**, **Троллейбус** или **Трамвай**. Транспортные средства должны двигаться с определенным для каждого **Маршрута** интервалом. При поломке на **Маршрут** должен выходить резервный транспорт или увеличиваться интервал движения.