Министерство образования Республики Беларусь

Учреждение образования

БелорусскиЙ государственный университет

информатики и радиоэлектроники

Факультет компьютерных систем и сетей

Кафедра программного обеспечения информационных технологий

|  |
| --- |
|  |
|  |

# оТЧЕТ

по лабораторной работе

на тему:

Работа со списками

|  |  |  |  |
| --- | --- | --- | --- |
| Выполнил  Студент гр. 351001 |  | С. П. Орсик | |
| Проверил |  | Асс. Е.Е. Фадеева | |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

Минск, 2024

1. Теоретические сведения по теме лабораторной работы

**Алгоритм** – система правил, четко описывающая последовательность действий, которые необходимо выполнить для решения определенной задачи.

**Свойства правильного алгоритма**:

1. Детерминированность (определенность) - предполагает получение однозначного результата вычислительного процесса при заданных исходных данных.
2. Результативность - указывает на наличие таких исходных данных, для которых реализуемый по заданному алгоритму вычислительный процесс должен через конечное число шагов остановиться и выдать искомый результат.
3. Массовость - предполагает, что алгоритм должен быть пригоден для решения всех задач данного типа.
4. Дискретность - означает расчлененность определяемого алгоритмом вычислительного процесса на отдельные этапы, возможность выполнения которых исполнителем (компьютером) не вызывает сомнений.

**Способы описания алгоритма:**

1. Словесное описание - запись на естественном языке
2. Графическое описание - изображение в виде схемы
3. Написание кода - запись на алгоритмическом языке

**Типы алгоритмов:**

1. Линейные
2. Разветвляющиеся
3. Циклические

**Цикл** – конструкция, предназначенная для многократного повторения участка схемы.

**Сложный цикл** – цикл, содержащий в себе один или более вложенных циклов.

**Итерационный цикл** - циклический процесс, в котором количество повторений заранее неизвестно и зависит от получающихся в ходе вычислений результатов.

**Массив** – упорядоченная совокупность однотипных элементов, имеющих общее имя.

**Строка** – это последовательность любых символов кода обмена информацией.

**Классификация циклов:**

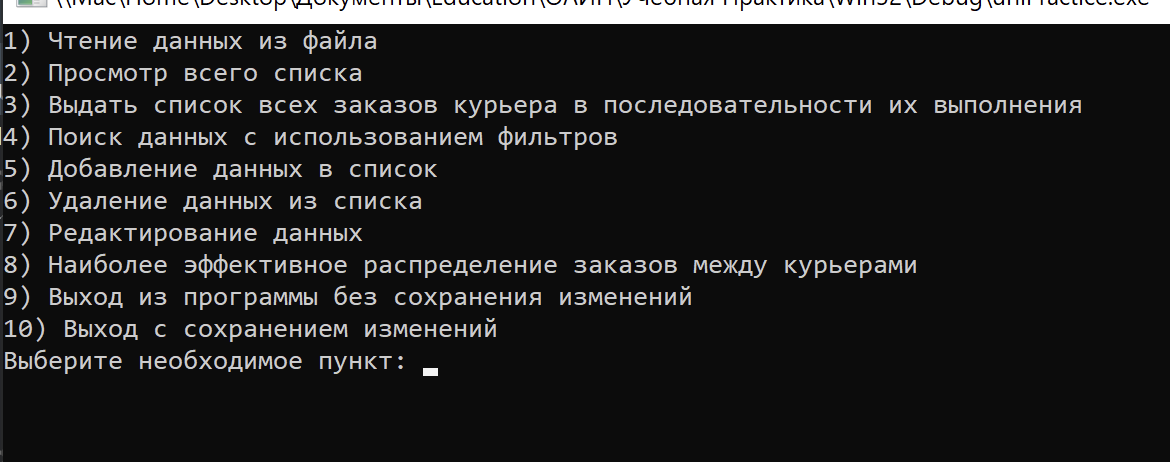
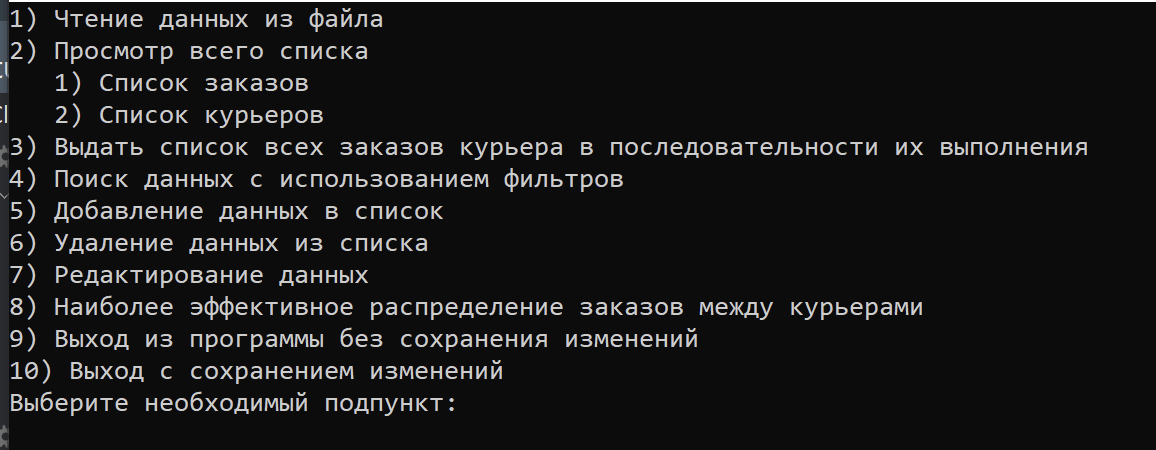
1. По взаимному расположению:
   1. Простые
   2. Сложные
   3. Внутренние
   4. Внешние
2. По местоположению условия выполнения цикла:
   1. С предусловием
   2. С постусловием
3. По виду условия выполнения:
   1. С параметром
   2. Итерационные

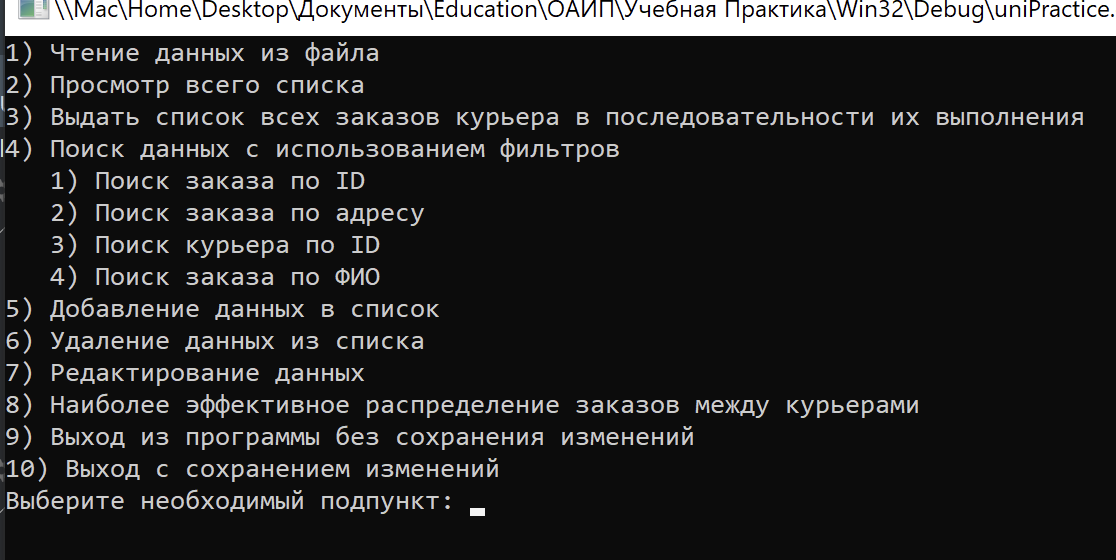
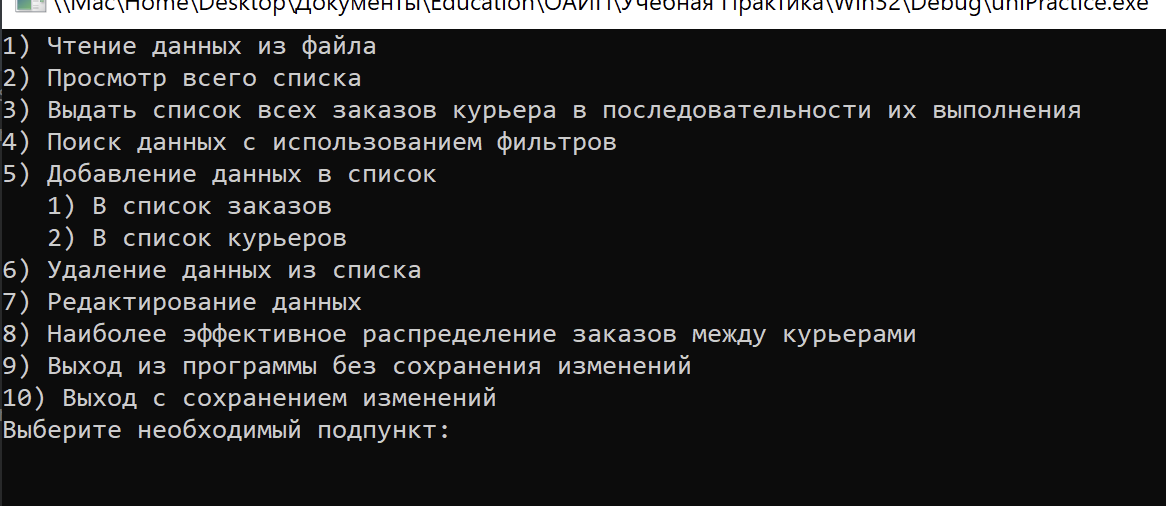
**Подпрограмма** – поименованная, логически законченная группа операторов языка, которую можно вызвать для выполнения по имени любое количество раз из различных мест программы.

1. Задание на лабораторную работу и эскиз работы программы

Необходимо разработать пользовательское меню и реализовать следующие пункты меню: просмотр списка/списков, добавление записи в список, удаление записи из списка, поиск записи в списке

Пользовательское меню:





Выполнение

* 1. Разработка алгоритма по ГОСТ 19.701-90

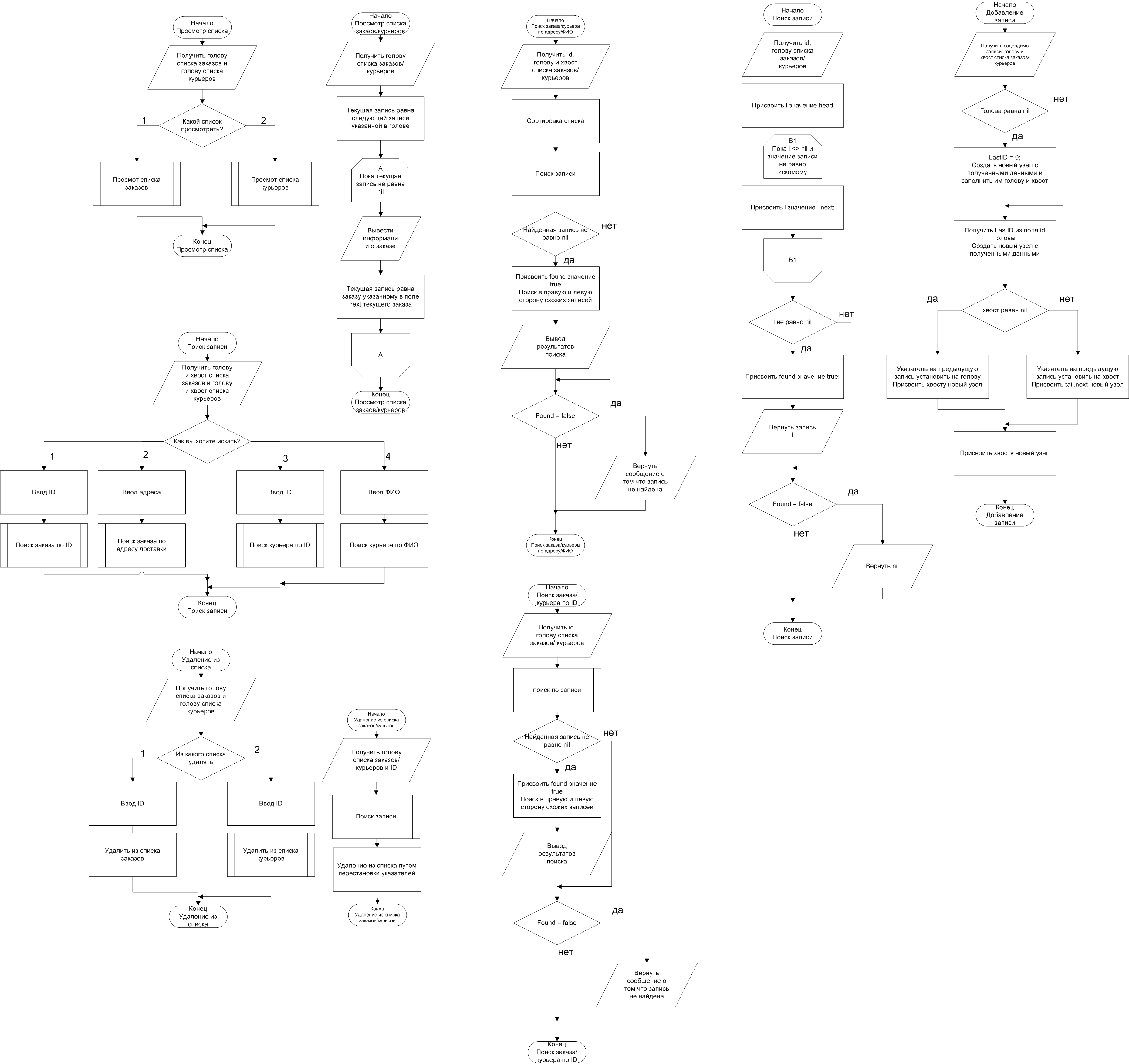


Рисунок 3.1 – Схема работы программы

3.2 Определение подпрограмм и их описание

Таблица 3.1 используемые идентификаторы

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя подпрограммы | Назначение подпрограммы | Заголовок подпрограммы | Имя параметра | Назначение параметра |
| PrintOrders | Вывод списка заказов | procedure PrintOrders(Head: POrder); | Head | Голова списка |
|
|
| AppendOrder | Добавление записи в список | procedure AppendOrder(var Head, Tail: POrder; Item: TOrderData); | Head | Голова списка |
| Tail | Хвост списка |
| Item | Содержимое новой записи |
|
|
| OrderByID | Поиск по ID | procedure OrderByID(var Head: POrder; argID: Integer;argAddr:String; cmp:TCompare); | Head | Голова списка |
| argID | Искомы ID |
| argAddr | Строковая переменнаая |
| cmp | компаратор |
| OrderByAddr | Поиск по адресу | procedure OrderByAddr(var Head: POrder; argID: Integer;argAddr:String; cmp:TCompare); | Head | Голова списка |
| argID | Численная переменная |
| argAddr | Искомый адрес |
| cmp | компаратор |

* 1. Выделение основных структур данных

Таблица 3.1 используемые идентификаторы

|  |  |  |
| --- | --- | --- |
| Имя идентификатора структуры | Назначение структуры | Тип структуры |
| TOrder | Список заказов | TOrderData = record  id: Integer;  addr: string[100];  time\_in: TTime;  time\_out: TTime;  volume: Double;  weight: Double;  end;  TOrder = record  data: TOrderData;  next: POrder;  prev: POrder;  end; |
| TPostman | Список курьеров | TPostmanData = record  id: Integer;  fio: string[100];  time\_in: TTime;  time\_out: TTime;  volume: Double;  weight: Double;  end;  TPostman = record  data: TPostmanData;  next: PPostman;  prev: PPostman;  end; |

3.4 Пример описание структур в виде таблиц

Структура\_1: mas Таблица 3.2 пример заполнения реальными данными

3.5 Тестирование и отладка программы

Таблица 3.2 Прохождение тестов программы

|  |  |  |  |
| --- | --- | --- | --- |
| Номер теста | Вводимые данные | Ожидаемый результат | Полученный результат |
| 1 | - | 1) Чтение данных из файла  2) Просмотр всего списка  1) Список заказов  2) Список курьеров  3) Выдать список всех заказов курьера в последовательности их выполнения  4) Поиск данных с использованием фильтров  5) Добавление данных в список  6) Удаление данных из списка  7) Редактирование данных  8) Наиболее эффективное распределение заказов между курьерами  9) Выход из программы без сохранения изменений  10) Выход с сохранением изменений  Выберите необходимый подпункт: 1  Список всех заказов:  ID: 1  ФИО: Дзержинск  Время прибытия: 10:00  Время убытия: 12:00  Объем: 2.50  Вес: 10.00  ID: 2  ФИО: Минск  Время прибытия: 11:00  Время убытия: 13:00  Объем: 1.80  Вес: 8.00  ID: 3  ФИО: Осиповичи  Время прибытия: 10:00  Время убытия: 12:00  Объем: 2.20  Вес: 10.00  ID: 4  ФИО: Липень  Время прибытия: 11:00  Время убытия: 13:00  Объем: 1.30  Вес: 8.00  ID: 5  ФИО: Осиповичи  Время прибытия: 10:00  Время убытия: 12:00  Объем: 0.50  Вес: 10.00  ID: 6  ФИО: Минск  Время прибытия: 11:00  Время убытия: 13:00  Объем: 1.00  Вес: 8.60  Нажмите Enter, чтобы продолжить. | Тест пройден успешно |
| 2 | - | 1) Чтение данных из файла  2) Просмотр всего списка  1) Список заказов  2) Список курьеров  3) Выдать список всех заказов курьера в последовательности их выполнения  4) Поиск данных с использованием фильтров  5) Добавление данных в список  6) Удаление данных из списка  7) Редактирование данных  8) Наиболее эффективное распределение заказов между курьерами  9) Выход из программы без сохранения изменений  10) Выход с сохранением изменений  Выберите необходимый подпункт: 2  Список всех курьеров:  ID: 1  ФИО: Орсик  Время прибытия: 10:00  Время убытия: 22:00  Объем: 2.50  Вес: 10.00  ID: 2  ФИО: Кузьменко  Время прибытия: 09:00  Время убытия: 23:00  Объем: 1.80  Вес: 8.00  ID: 3  ФИО: Иваненко  Время прибытия: 14:00  Время убытия: 12:00  Объем: 2.50  Вес: 10.00  ID: 4  ФИО: Федорако  Время прибытия: 10:00  Время убытия: 13:00  Объем: 1.80  Вес: 8.00  ID: 5  ФИО: Бурчук  Время прибытия: 19:00  Время убытия: 20:00  Объем: 2.50  Вес: 10.00  ID: 6  ФИО: Бобченок  Время прибытия: 11:00  Время убытия: 13:00  Объем: 1.80  Вес: 8.00  ID: 7  ФИО: Орсик  Время прибытия: 10:00  Время убытия: 22:00  Объем: 2.50  Вес: 10.00  ID: 8  ФИО: Кузьменко  Время прибытия: 09:00  Время убытия: 23:00  Объем: 1.80  Вес: 8.00  ID: 9  ФИО: Иваненко  Время прибытия: 14:00  Время убытия: 12:00  Объем: 2.50  Вес: 10.00  ID: 10  ФИО: Федорако  Время прибытия: 10:00  Время убытия: 13:00  Объем: 1.80  Вес: 8.00  ID: 11  ФИО: Бурчук  Время прибытия: 19:00  Время убытия: 20:00  Объем: 2.50  Вес: 10.00  ID: 12  ФИО: Бобченок  Время прибытия: 11:00  Время убытия: 13:00  Объем: 1.80  Вес: 8.00  Нажмите Enter, чтобы продолжить. | Тест пройден успешно |
| 3 | 2 | 1) Чтение данных из файла  2) Просмотр всего списка  3) Выдать список всех заказов курьера в последовательности их выполнения  4) Поиск данных с использованием фильтров  1) Поиск заказа по ID  2) Поиск заказа по адресу  3) Поиск курьера по ID  4) Поиск заказа по ФИО  5) Добавление данных в список  6) Удаление данных из списка  7) Редактирование данных  8) Наиболее эффективное распределение заказов между курьерами  9) Выход из программы без сохранения изменений  10) Выход с сохранением изменений  Выберите необходимый подпункт: 1  Введите необходимый ID: 2  ID: 2  Адрес: Минск  Время прибытия: 11:00  Время убытия: 13:00  Объем: 1.80  Вес: 8.00  Нажмите Enter, чтобы продолжить. | Тест пройден успешно |
| 4 | Минск | 1) Чтение данных из файла  2) Просмотр всего списка  3) Выдать список всех заказов курьера в последовательности их выполнения  4) Поиск данных с использованием фильтров  1) Поиск заказа по ID  2) Поиск заказа по адресу  3) Поиск курьера по ID  4) Поиск заказа по ФИО  5) Добавление данных в список  6) Удаление данных из списка  7) Редактирование данных  8) Наиболее эффективное распределение заказов между курьерами  9) Выход из программы без сохранения изменений  10) Выход с сохранением изменений  Выберите необходимый подпункт: 2  Введите необходимый адрес: Минск  Найден(ы) заказ(ы):  ID: 2  Адрес: Минск  Время прибытия: 11:00  Время убытия: 13:00  Объем: 1.80  Вес: 8.00  ID: 6  Адрес: Минск  Время прибытия: 11:00  Время убытия: 13:00  Объем: 1.00  Вес: 8.60  Нажмите Enter, чтобы продолжить. | Тест пройден успешно |
| 5 | 10 | 1) Чтение данных из файла  2) Просмотр всего списка  3) Выдать список всех заказов курьера в последовательности их выполнения  4) Поиск данных с использованием фильтров  1) Поиск заказа по ID  2) Поиск заказа по адресу  3) Поиск курьера по ID  4) Поиск заказа по ФИО  5) Добавление данных в список  6) Удаление данных из списка  7) Редактирование данных  8) Наиболее эффективное распределение заказов между курьерами  9) Выход из программы без сохранения изменений  10) Выход с сохранением изменений  Выберите необходимый подпункт: 3  Введите необходимый ID: 10  ID: 10  ФИО: Федорако  Время прибытия: 10:00  Время убытия: 13:00  Объем: 1.80  Вес: 8.00  Нажмите Enter, чтобы продолжить. | Тест пройден успешно |
| 6 | 1) Чтение данных из файла  2) Просмотр всего списка  3) Выдать список всех заказов курьера в последовательности их выполнения  4) Поиск данных с использованием фильтров  5) Добавление данных в список  1) В список заказов  2) В список курьеров  6) Удаление данных из списка  7) Редактирование данных  8) Наиболее эффективное распределение заказов между курьерами  9) Выход из программы без сохранения изменений  10) Выход с сохранением изменений  Выберите необходимый подпункт: 1  Введите адрес доставки: Новоельня  Введите время с которого можно доставить: 12:23  Введите время до которого можно доставить: 23:01  Введите объем(м^3) заказа: 3  Введите вес(кг) заказа: 20  Добавить еще один заказ(Да/нет): нет  Нажмите Enter, чтобы продолжить. | …  Список всех заказов:  ID: 1  ФИО: Дзержинск  Время прибытия: 10:00  Время убытия: 12:00  Объем: 2.50  Вес: 10.00  ID: 4  ФИО: Липень  Время прибытия: 11:00  Время убытия: 13:00  Объем: 1.30  Вес: 8.00  ID: 2  ФИО: Минск  Время прибытия: 11:00  Время убытия: 13:00  Объем: 1.80  Вес: 8.00  ID: 6  ФИО: Минск  Время прибытия: 11:00  Время убытия: 13:00  Объем: 1.00  Вес: 8.60  ID: 3  ФИО: Осиповичи  Время прибытия: 10:00  Время убытия: 12:00  Объем: 2.20  Вес: 10.00  ID: 5  ФИО: Осиповичи  Время прибытия: 10:00  Время убытия: 12:00  Объем: 0.50  Вес: 10.00  ID: 7  ФИО: Новоельня  Время прибытия: 10:00  Время убытия: 12:00  Объем: 2.50  Вес: 10.00  Нажмите Enter, чтобы продолжить. | Тест пройден успешно |
| 7 | 1) Чтение данных из файла  2) Просмотр всего списка  3) Выдать список всех заказов курьера в последовательности их выполнения  4) Поиск данных с использованием фильтров  5) Добавление данных в список  6) Удаление данных из списка  1) Из списка заказов  2) Из списка курьеров  7) Редактирование данных  8) Наиболее эффективное распределение заказов между курьерами  9) Выход из программы без сохранения изменений  10) Выход с сохранением изменений  Выберите необходимый подпункт: 1  Введите ID заказа, который необходимо удалить: 5  Заказ удален  Нажмите Enter, чтобы продолжить. | 1) Чтение данных из файла  2) Просмотр всего списка  1) Список заказов  2) Список курьеров  3) Выдать список всех заказов курьера в последовательности их выполнения  4) Поиск данных с использованием фильтров  5) Добавление данных в список  6) Удаление данных из списка  7) Редактирование данных  8) Наиболее эффективное распределение заказов между курьерами  9) Выход из программы без сохранения изменений  10) Выход с сохранением изменений  Выберите необходимый подпункт: 1  Список всех заказов:  ID: 1  ФИО: Дзержинск  Время прибытия: 10:00  Время убытия: 12:00  Объем: 2.50  Вес: 10.00  ID: 4  ФИО: Липень  Время прибытия: 11:00  Время убытия: 13:00  Объем: 1.30  Вес: 8.00  ID: 2  ФИО: Минск  Время прибытия: 11:00  Время убытия: 13:00  Объем: 1.80  Вес: 8.00  ID: 6  ФИО: Минск  Время прибытия: 11:00  Время убытия: 13:00  Объем: 1.00  Вес: 8.60  ID: 3  ФИО: Осиповичи  Время прибытия: 10:00  Время убытия: 12:00  Объем: 2.20  Вес: 10.00  ID: 7  ФИО: Новоельня  Время прибытия: 10:00  Время убытия: 12:00  Объем: 2.50  Вес: 10.00  Нажмите Enter, чтобы продолжить. | Тест пройден успешно |

Анализ прохождения тестов, причины возникших ошибок, внесение изменений в алгоритм и текст программы

3.6 Итоговый текст программы с комментария

Файл uniPractice

program uniPractice;

{$APPTYPE CONSOLE}

{$R \*.res}

uses

System.SysUtils,

Windows,

ownFunctions in 'ownFunctions.pas',

orderModule in 'orderModule.pas',

postmanModule in 'postmanModule.pas',

rawData in 'rawData.pas';

var

Head, Tail: POrder;

HeadP, TailP: PPostman;

flag: boolean = true;

isMenu: boolean = true;

menuVal, menuVal2:integer;

begin

Head := nil;

Tail := nil;

HeadP := nil;

TailP:= nil;

generateOrders(Head, Tail);

generatePostmans(HeadP, TailP);

while isMenu do

begin

clrscr();

writeln('1) Чтение данных из файла');

writeln('2) Просмотр всего списка');

writeln('3) Выдать список всех заказов курьера в последовательности их выполнения');

writeln('4) Поиск данных с использованием фильтров');

writeln('5) Добавление данных в список');

writeln('6) Удаление данных из списка');

writeln('7) Редактирование данных');

writeln('8) Наиболее эффективное распределение заказов между курьерами');

writeln('9) Выход из программы без сохранения изменений');

writeln('10) Выход с сохранением изменений');

write('Выберите необходимое пункт: ');

readln(menuVal);

case menuVal of

1:

begin

Writeln('Нажмите Enter, чтобы продолжить.');

Readln;

end;

2:

begin

LookMenu(Head, HeadP);

Writeln('Нажмите Enter, чтобы продолжить.');

Readln;

end;

3:

begin

Writeln('Нажмите Enter, чтобы продолжить.');

Readln;

end;

4:

begin

SearhMenu(Head, Tail, HeadP, TailP);

Writeln('Нажмите Enter, чтобы продолжить.');

Readln;

end;

5:

begin

AddMenu(Head, Tail, HeadP, TailP);

Writeln('Нажмите Enter, чтобы продолжить.');

Readln;

end;

6:

begin

DeleteMenu(Head, Tail, HeadP, TailP);

Writeln('Нажмите Enter, чтобы продолжить.');

Readln;

end;

7:

begin

EditMenu(Head, Tail, HeadP, TailP);

Writeln('Нажмите Enter, чтобы продолжить.');

Readln;

end;

8:

begin

Writeln('Нажмите Enter, чтобы продолжить.');

Readln;

end;

9:

begin

Writeln('Нажмите Enter, чтобы продолжить.');

Readln;

end;

10:

begin

Writeln('Нажмите Enter, чтобы продолжить.');

Readln;

end;

end;

end;

end.

Файл ownFunctions

unit ownFunctions;

interface

uses

System.SysUtils,

Windows,

orderModule,

postmanModule;

Type

time = array [0 .. 1] of Integer;

// Structure of order

order = ^orderT;

orderT = record

data: record

id: Integer;

addr: string[100];

time\_in: time;

time\_out: time;

volume: double;

weight: double;

end;

next: order;

prev: order;

end;

// Structure of postmans

postman = ^postmanT;

postmanT = record

data: record

id: Integer;

name: string[100];

time\_from: time;

time\_to: time;

weight\_lim: double;

car\_volume: double;

end;

next: postman;

prev: postman;

end;

function StrToTime24(const TimeStr: string): TTime;

procedure LookMenu(HeadOrder: POrder; HeadPostman: PPostman);

procedure SearhMenu(HeadOrder, TailOrder: POrder;

HeadPostman, TailPostman: PPostman);

procedure clrscr;

procedure AddMenu(HeadOrder, TailOrder: POrder;

HeadPostman, TailPostman: PPostman);

procedure DeleteMenu(HeadOrder, TailOrder: POrder;

HeadPostman, TailPostman: PPostman);

procedure EditMenu(HeadOrder, TailOrder: POrder;

HeadPostman, TailPostman: PPostman);

implementation

function StrToTime24(const TimeStr: string): TTime;

var

Hour, Min, Sec, MSec: Word;

begin

Hour := StrToInt(Copy(TimeStr, 1, 2));

Min := StrToInt(Copy(TimeStr, 4, 2));

// Если строка содержит секунды и миллисекунды, их также можно извлечь и использовать

if Length(TimeStr) >= 8 then

begin

Sec := StrToInt(Copy(TimeStr, 7, 2));

MSec := StrToInt(Copy(TimeStr, 10, 3));

end

else

begin

Sec := 0;

MSec := 0;

end;

Result := EncodeTime(Hour, Min, Sec, MSec);

end;

procedure clrscr;

var

cursor: COORD;

r: cardinal;

begin

r := 300;

cursor.X := 0;

cursor.Y := 0;

FillConsoleOutputCharacter(GetStdHandle(STD\_OUTPUT\_HANDLE), ' ', 80 \* r,

cursor, r);

SetConsoleCursorPosition(GetStdHandle(STD\_OUTPUT\_HANDLE), cursor);

end;

procedure LookMenu(HeadOrder: POrder; HeadPostman: PPostman);

var

menuVal: Integer;

begin

clrscr();

writeln('1) Чтение данных из файла');

writeln('2) Просмотр всего списка');

writeln(' 1) Список заказов');

writeln(' 2) Список курьеров');

writeln('3) Выдать список всех заказов курьера в последовательности их выполнения');

writeln('4) Поиск данных с использованием фильтров');

writeln('5) Добавление данных в список');

writeln('6) Удаление данных из списка');

writeln('7) Редактирование данных');

writeln('8) Наиболее эффективное распределение заказов между курьерами');

writeln('9) Выход из программы без сохранения изменений');

writeln('10) Выход с сохранением изменений');

write('Выберите необходимый подпункт: ');

readln(menuVal);

case menuVal of

1:

begin

PrintOrders(HeadOrder);

end;

2:

begin

PrintPostmans(HeadPostman);

end;

end;

end;

procedure SearhMenu(HeadOrder, TailOrder: POrder;

HeadPostman, TailPostman: PPostman);

var

menuVal: Integer;

searchStr: string;

searchID: Integer;

begin

clrscr();

searchID:=0;

searchStr:='';

writeln('1) Чтение данных из файла');

writeln('2) Просмотр всего списка');

writeln('3) Выдать список всех заказов курьера в последовательности их выполнения');

writeln('4) Поиск данных с использованием фильтров');

writeln(' 1) Поиск заказа по ID');

writeln(' 2) Поиск заказа по адресу');

writeln(' 3) Поиск курьера по ID');

writeln(' 4) Поиск заказа по ФИО');

writeln('5) Добавление данных в список');

writeln('6) Удаление данных из списка');

writeln('7) Редактирование данных');

writeln('8) Наиболее эффективное распределение заказов между курьерами');

writeln('9) Выход из программы без сохранения изменений');

writeln('10) Выход с сохранением изменений');

write('Выберите необходимый подпункт: ');

readln(menuVal);

case menuVal of

1:

begin

Write('Введите необходимый ID: ');

readln(searchID);

OrderByID(HeadOrder, searchID, searchStr, CompareOrderID);

end;

2:

begin

Write('Введите необходимый адрес: ');

readln(searchStr);

OrdersByAddr(HeadOrder, TailOrder, searchID,searchStr, CompareOrderAddr);

end;

3:

begin

Write('Введите необходимый ID: ');

readln(searchID);

PostmanByID(HeadPostman, searchID, searchStr, ComparePostmanID);

end;

4:

begin

Write('Введите искомое ФИО: ');

readln(searchStr);

PostmanByFio(HeadPostman, TailPostman, searchID, searchStr, ComparePostmanFio);

end;

end;

end;

procedure AddMenu(HeadOrder, TailOrder: POrder;

HeadPostman, TailPostman: PPostman);

var

menuVal: Integer;

str:string;

isWriting:boolean;

srtParam:string;

exitStr: String[5];

doubleParam:double;

OrderData:TOrderData;

PostmanData:TPostmanData;

begin

isWriting:=True;

clrscr();

writeln('1) Чтение данных из файла');

writeln('2) Просмотр всего списка');

writeln('3) Выдать список всех заказов курьера в последовательности их выполнения');

writeln('4) Поиск данных с использованием фильтров');

writeln('5) Добавление данных в список');

writeln(' 1) В список заказов');

writeln(' 2) В список курьеров');

writeln('6) Удаление данных из списка');

writeln('7) Редактирование данных');

writeln('8) Наиболее эффективное распределение заказов между курьерами');

writeln('9) Выход из программы без сохранения изменений');

writeln('10) Выход с сохранением изменений');

write('Выберите необходимый подпункт: ');

readln(menuVal);

case menuVal of

1:

begin

while isWriting do

begin

Write('Введите адрес доставки: ');

Readln(srtParam);

OrderData.addr := srtParam;

Write('Введите время с которого можно доставить: ');

Readln(srtParam);

OrderData.time\_in := StrToTime('10:00');

Write('Введите время до которого можно доставить: ');

Readln(srtParam);

OrderData.time\_out := StrToTime('12:00');

Write('Введите объем(м^3) заказа: ');

Readln(doubleParam);

OrderData.volume := 2.5;

Write('Введите вес(кг) заказа: ');

Readln(doubleParam);

OrderData.weight := 10;

AppendOrder(HeadOrder, TailOrder, OrderData);

Write('Добавить еще один заказ(Да/нет): ');

Readln(exitStr);

while not ((exitStr='Нет') or (exitStr='нет') or (exitStr='Да') or (exitStr='да')) do

begin

Write('Неверный ввод, повторите попытку: ');

Readln(exitStr);

end;

if (exitStr='Нет') or (exitStr='нет') then

isWriting:=False;

end;

end;

2:

begin

PrintPostmans(HeadPostman);

end;

end;

end;

procedure DeleteMenu(HeadOrder, TailOrder: POrder;

HeadPostman, TailPostman: PPostman);

var

menuVal: Integer;

str:string;

isWriting:boolean;

ID:integer;

OrderData:TOrderData;

PostmanData:TPostmanData;

begin

isWriting:=True;

clrscr();

writeln('1) Чтение данных из файла');

writeln('2) Просмотр всего списка');

writeln('3) Выдать список всех заказов курьера в последовательности их выполнения');

writeln('4) Поиск данных с использованием фильтров');

writeln('5) Добавление данных в список');

writeln('6) Удаление данных из списка');

writeln(' 1) Из списка заказов');

writeln(' 2) Из списка курьеров');

writeln('7) Редактирование данных');

writeln('8) Наиболее эффективное распределение заказов между курьерами');

writeln('9) Выход из программы без сохранения изменений');

writeln('10) Выход с сохранением изменений');

write('Выберите необходимый подпункт: ');

readln(menuVal);

case menuVal of

1:

begin

Write('Введите ID заказа, который необходимо удалить: ');

Readln(ID);

DeleteOrder(HeadOrder, ID);

Writeln('Заказ удален');

end;

2:

begin

Write('Введите ID курьера, которого необходимо удалить: ');

Readln(ID);

DeletePostman(HeadPostman, ID);

Writeln('Курьер удален');

end;

end;

end;

procedure EditMenu(HeadOrder, TailOrder: POrder;

HeadPostman, TailPostman: PPostman);

var

menuVal: Integer;

str:string;

isWriting:boolean;

ID:integer;

OrderData:TOrderData;

PostmanData:TPostmanData;

begin

isWriting:=True;

clrscr();

writeln('1) Чтение данных из файла');

writeln('2) Просмотр всего списка');

writeln('3) Выдать список всех заказов курьера в последовательности их выполнения');

writeln('4) Поиск данных с использованием фильтров');

writeln('5) Добавление данных в список');

writeln('6) Удаление данных из списка');

writeln('7) Редактирование данных');

writeln(' 1) Из списка заказов');

writeln(' 2) Из списка курьеров');

writeln('8) Наиболее эффективное распределение заказов между курьерами');

writeln('9) Выход из программы без сохранения изменений');

writeln('10) Выход с сохранением изменений');

write('Выберите необходимый подпункт: ');

readln(menuVal);

case menuVal of

1:

begin

Write('Введите ID заказа, который необходимо отредактировать: ');

Readln(ID);

EditOrder(HeadOrder, TailOrder, ID);

Writeln('Заказ изменен');

end;

2:

begin

Write('Введите ID курьера, которого необходимо отредактировать: ');

Readln(ID);

EditPostman(HeadPostman, TailPostman, ID);

Writeln('Курьер изменен');

end;

end;

end;

end.

Файл postmanModule

unit postmanModule;

interface

type

PPostman = ^TPostman;

TPostmanData = record

id: Integer;

fio: string[100];

time\_in: TTime;

time\_out: TTime;

volume: Double;

weight: Double;

end;

TPostman = record

data: TPostmanData;

next: PPostman;

prev: PPostman;

end;

TCompare=function(const zap: PPostman; argID: Integer; argFio:string):integer;

procedure PrintPostmans(Head: PPostman);

procedure AppendPostman(var Head, Tail: PPostman; Item: TPostmanData);

procedure PostmanByID(var Head: PPostman; argID: Integer; argAddr:string; cmp:TCompare);

procedure PostmanByFio(Head, Tail: PPostman; argID: Integer; argFio:string; cmp:TCompare);

procedure DeletePostman(Head:PPostman; const ID: integer);

procedure EditPostman(Head, Tail: PPostman; const id: Integer);

function ComparePostmanID(const zap: PPostman; argID: Integer; argFio:string):integer;

function ComparePostmanFio(const zap: PPostman; argID: Integer; argFio:string):integer;

implementation

uses

System.SysUtils;

procedure AppendPostman(var Head, Tail: PPostman; Item: TPostmanData);

var

NewNode: PPostman;

LastID: Integer;

begin

// Проверяем, если головной узел пустой, то устанавливаем начальное значение ID как 0

if Head = nil then

begin

LastID := 0;

New(NewNode);

NewNode^.data := Item;

NewNode^.prev := nil;

NewNode^.next := Tail;

Head := NewNode;

Tail := NewNode;

end

else

// Иначе берем ID последнего узла

LastID := Head^.data.id;

Inc(LastID); // Автоинкремент для нового ID

Item.id := LastID; // Установка нового ID для добавляемой записи

Head^.data.id:=LastID;

New(NewNode);

NewNode^.data := Item;

NewNode^.next := nil;

if Tail = nil then

begin

NewNode^.prev := Head;

Tail:= NewNode;

end

else

begin

NewNode^.prev := Tail;

Tail^.next := NewNode;

end;

Tail := NewNode;

end;

procedure PrintPostmans(Head: PPostman);

var

Current: PPostman;

Index: Integer;

begin

Writeln('Список всех курьеров:');

Current := Head^.next ;

while Current <> nil do

begin

Writeln('ID: ', Current^.data.id);

Writeln('ФИО: ', Current^.data.fio);

Writeln('Время прибытия: ', FormatDateTime('HH:NN', Current^.data.time\_in));

Writeln('Время убытия: ', FormatDateTime('HH:NN', Current^.data.time\_out));

Writeln('Объем: ', Current^.data.volume:4:2);

Writeln('Вес: ', Current^.data.weight:4:2);

Writeln;

Current := Current^.next;

end;

end;

function ComparePostmanID(const zap: PPostman; argID: Integer; argFio:string):integer;

begin

if zap^.data.id = argID then

Result:=1

else if zap^.data.id < argID then

Result:=0

else

Result:=2;

end;

function ComparePostmanFio(const zap: PPostman; argID: Integer; argFio:string):integer;

begin

if zap^.data.fio = argFio then

Result:=1

else if zap^.data.fio < argFio then

Result:=0

else

Result:=2;

end;

// Поиск по ID

function FindPostamn(var Head: PPostman; argID: Integer; argFio:string; cmp:TCompare): PPostman;

var

l, r, m: PPostman;

found: Boolean;

res:integer;

begin

l := Head;

found := False;

while (l <> nil) and (cmp(l,argID,argFio) <> 1) do

l := l^.next;

if l<>nil then

begin

Result:=l;

found:=True;

end;

if not found then

Result:=nil;

end;

procedure PostmanByID(var Head: PPostman; argID: Integer; argAddr:string; cmp:TCompare);

var

order: PPostman;

found: Boolean;

begin

found:= false;

order := FindPostamn(Head, argID, argAddr, cmp);

if order<>nil then

begin

found:=true;

Writeln('ID: ', order^.data.id);

Writeln('ФИО: ', order^.data.fio);

Writeln('Время прибытия: ', FormatDateTime('HH:NN', order^.data.time\_in));

Writeln('Время убытия: ', FormatDateTime('HH:NN', order^.data.time\_out));

Writeln('Объем: ', order^.data.volume:4:2);

Writeln('Вес: ', order^.data.weight:4:2);

// Вывод остальных данных

Writeln;

end;

if not Found then

Writeln('Запись с таким ID не найдена');

end;

// Поиск по полую "ФИО"

procedure SwapOrders(order1, order2: PPostman);

var

tempData: TPostmanData;

begin

tempData := order1^.data;

order1^.data := order2^.data;

order2^.data := tempData;

end;

function Partition(start, stop: PPostman): PPostman;

var

pivot, Current: PPostman;

begin

pivot := start;

Current := pivot^.next;

while Current <> stop do

begin

if CompareText(Current^.data.fio, pivot^.data.fio) < 0 then

// Сравниваем строки addr

begin

pivot := pivot^.next;

SwapOrders(pivot, Current);

end;

Current := Current^.next;

end;

SwapOrders(start, pivot);

Result := pivot;

end;

procedure QuickSort(start, stop: PPostman);

var

pivot: PPostman;

begin

if (start <> nil) and (stop <> nil) and (start <> stop) and

(start <> stop^.next) then

begin

pivot := Partition(start, stop^.next);

QuickSort(start, pivot);

QuickSort(pivot^.next, stop);

end;

end;

procedure Sort(var Head, Tail: PPostman);

begin

QuickSort(Head, Tail);

end;

procedure PostmanByFio(Head, Tail: PPostman; argID: Integer; argFio:string; cmp:TCompare);

var

CurrentL, CurrentR: PPostman;

found, nextStep: Boolean;

begin

// Сначала сортируем список в алфавитном порядке

Sort(Head, Tail);

found := False;

// Ищем первую запись с совпадающим адресом

currentL:=FindPostamn(Head, argID, argFio, cmp);

currentR:=currentL;

// Если запись не найдена, выходим из процедуры

if (currentL <> nil) and (currentR<>nil) then

begin

// Выводим найденную запись

Writeln('Найден(ы) заказ(ы):');

Writeln('ID: ', currentL^.data.id);

Writeln('ФИО: ', currentL^.data.fio);

Writeln('Время прибытия: ', FormatDateTime('HH:NN', currentL^.data.time\_in));

Writeln('Время убытия: ', FormatDateTime('HH:NN', currentL^.data.time\_out));

Writeln('Объем: ', currentL^.data.volume:4:2);

Writeln('Вес: ', currentL^.data.weight:4:2);

// Вывод остальных данных

Writeln;

found := True;

// Перемещаемся к предыдущим записям с совпадающим адресом

while (currentL^.prev <> nil) and (currentL^.prev^.data.fio = argFio) do

begin

currentL := currentL^.prev;

// Выводим найденную запись

Writeln('ID: ', currentL^.data.id);

Writeln('ФИО: ', currentL^.data.fio);

Writeln('Время прибытия: ', FormatDateTime('HH:NN',

currentL^.data.time\_in));

Writeln('Время убытия: ', FormatDateTime('HH:NN',

currentL^.data.time\_out));

Writeln('Объем: ', currentL^.data.volume:4:2);

Writeln('Вес: ', currentL^.data.weight:4:2);

// Вывод остальных данных

Writeln;

found := True;

end;

// Перемещаемся к следующим записям с совпадающим адресом

// Перемещаемся к следующим записям с совпадающим адресом

while (currentR^.next <> nil) and (currentR^.next^.data.fio = argFio) do

begin

currentR := currentR^.next;

// Выводим найденную запись

Writeln('ID: ', currentR^.data.id);

Writeln('ФИО: ', currentR^.data.fio);

Writeln('Время прибытия: ', FormatDateTime('HH:NN',

currentR^.data.time\_in));

Writeln('Время убытия: ', FormatDateTime('HH:NN',

currentR^.data.time\_out));

Writeln('Объем: ', currentR^.data.volume:4:2);

Writeln('Вес: ', currentR^.data.weight:4:2);

// Вывод остальных данных

Writeln;

found := True;

end;

end;

// Если не было найдено ни одной записи с совпадающим адресом, выведем сообщение об этом

if not found then

Writeln('Записи с курьером "', argFio, '" не найдены.');

end;

procedure DeletePostman(Head:PPostman; const ID: integer);

var

order:PPostman;

str:string;

begin

str:='';

order := FindPostamn(Head, ID,str, ComparePostmanID);

if order^.prev<>nil then

order^.prev^.next:=order^.next;

if order^.next<>nil then

order^.next^.prev:=order^.prev;

end;

procedure EditPostman(Head, Tail: PPostman; const id: Integer);

var

order: PPostman;

isWriting: Boolean;

strParam, exitStr: string;

doubleParam: Double;

menuVal:integer;

begin

strParam:='';

order := FindPostamn(Head, id,strParam,ComparePostmanID);

Writeln('Запись, которую необходимо отредактировать');

Writeln('ID: ', order^.data.id);

Writeln('1) ФИО: ', order^.data.fio);

Writeln('2) Время прибытия: ', FormatDateTime('HH:NN', order^.data.time\_in));

Writeln('3) Время убытия: ', FormatDateTime('HH:NN', order^.data.time\_out));

Writeln('4) Объем: ', order^.data.volume:4:2);

Writeln('5) Вес: ', order^.data.weight:4:2);

Writeln('6) Выйти из режима редактирования');

while isWriting do

begin

Write('Выберите поле, которое необходимо отредактировать (6 - выйти): ');

readln(menuVal);

case menuVal of

1:

begin

Write('Введите ФИО: ');

Readln(strParam);

order^.data.fio := strParam;

end;

2:

begin

Write('Введите время с которого можно доставить: ');

Readln(strParam);

order^.data.time\_in := StrToTime(strParam);

end;

3:

begin

Write('Введите время до которого можно доставить: ');

Readln(strParam);

order^.data.time\_out := StrToTime(strParam);

end;

4:

begin

Write('Введите объем(м^3) заказа: ');

Readln(doubleParam);

order^.data.volume := doubleParam;

end;

5:

begin

Write('Введите вес(кг) заказа: ');

Readln(doubleParam);

order^.data.weight := doubleParam;

end;

6:

begin

isWriting:=False;

end;

end;

end;

end;

end.

Файл orderModule

unit orderModule;

interface

type

POrder = ^TOrder;

TOrderData = record

id: Integer;

addr: string[100];

time\_in: TTime;

time\_out: TTime;

volume: Double;

weight: Double;

end;

TOrder = record

data: TOrderData;

next: POrder;

prev: POrder;

end;

TCompare=function(const zap: POrder; argID: Integer; argAddr:string):integer;

procedure PrintOrders(Head: POrder);

procedure AppendOrder(var Head, Tail: POrder; Item: TOrderData);

procedure OrderByID(var Head: POrder; argID: Integer;argAddr:String; cmp:TCompare);

procedure OrdersByAddr(Head, Tail: POrder; argID: Integer; argAddr:string; cmp:TCompare);

procedure DeleteOrder(Head: POrder; const id: Integer);

procedure EditOrder(Head, Tail: POrder; const id: Integer);

function CompareOrderID(const zap: POrder; argID: Integer; argAddr:string):integer;

function CompareOrderAddr(const zap: POrder; argID: Integer; argAddr:string):integer;

implementation

uses

System.SysUtils;

// Добавить в список

procedure AppendOrder(var Head, Tail: POrder; Item: TOrderData);

var

NewNode: POrder;

LastID: Integer;

begin

// Проверяем, если головной узел пустой, то устанавливаем начальное значение ID как 0

if Head = nil then

begin

LastID := 0;

New(NewNode);

NewNode^.data := Item;

NewNode^.prev := nil;

NewNode^.next := Tail;

Head := NewNode;

Tail := NewNode;

end

else

// Иначе берем ID последнего узла

LastID := Head^.data.id;

Inc(LastID); // Автоинкремент для нового ID

Item.id := LastID; // Установка нового ID для добавляемой записи

Head^.data.id := LastID;

New(NewNode);

NewNode^.data := Item;

NewNode^.next := nil;

if Tail = nil then

begin

NewNode^.prev := Head;

Tail := NewNode;

end

else

begin

NewNode^.prev := Tail;

Tail^.next := NewNode;

end;

Tail := NewNode;

end;

// Вывод списка

procedure PrintOrders(Head: POrder);

var

Current: POrder;

Index: Integer;

begin

Writeln('Список всех заказов:');

Current := Head^.next;

while Current <> nil do

begin

Writeln('ID: ', Current^.data.id);

Writeln('ФИО: ', Current^.data.addr);

Writeln('Время прибытия: ', FormatDateTime('HH:NN', Current^.data.time\_in));

Writeln('Время убытия: ', FormatDateTime('HH:NN', Current^.data.time\_out));

Writeln('Объем: ', Current^.data.volume:4:2);

Writeln('Вес: ', Current^.data.weight:4:2);

Writeln;

Current := Current^.next;

end;

end;

function CompareOrderID(const zap: POrder; argID: Integer; argAddr:string):integer;

begin

if zap^.data.id = argID then

Result:=1

else if zap^.data.id < argID then

Result:=0

else

Result:=2;

end;

function CompareOrderAddr(const zap: POrder; argID: Integer; argAddr:string):integer;

begin

if zap^.data.addr = argAddr then

Result:=1

else if zap^.data.addr < argAddr then

Result:=0

else

Result:=2;

end;

// Поиск по ID

function FindOrder(var Head: POrder; argID: Integer; argAddr:string; cmp:TCompare): POrder;

var

l, r, m: POrder;

found: Boolean;

res:integer;

begin

l := Head;

found := False;

while (l <> nil) and (cmp(l,argID,argAddr) <> 1) do

l := l^.next;

if l<>nil then

begin

Result:=l;

found:=True;

end;

if not found then

Result:=nil;

end;

procedure OrderByID(var Head: POrder; argID: Integer; argAddr:string; cmp:TCompare);

var

order: POrder;

found: Boolean;

begin

found:= false;

order := FindOrder(Head, argID, argAddr, cmp);

if order<>nil then

begin

found:=true;

Writeln('ID: ', order^.data.id);

Writeln('Адрес: ', order^.data.addr);

Writeln('Время прибытия: ', FormatDateTime('HH:NN', order^.data.time\_in));

Writeln('Время убытия: ', FormatDateTime('HH:NN', order^.data.time\_out));

Writeln('Объем: ', order^.data.volume:4:2);

Writeln('Вес: ', order^.data.weight:4:2);

// Вывод остальных данных

Writeln;

end;

if not Found then

Writeln('Запись с таким ID не найдена');

end;

// Поиск по полую "Адрес"

procedure SwapOrders(order1, order2: POrder);

var

tempData: TOrderData;

begin

tempData := order1^.data;

order1^.data := order2^.data;

order2^.data := tempData;

end;

function Partition(start, stop: POrder): POrder;

var

pivot, Current: POrder;

begin

pivot := start;

Current := pivot^.next;

while Current <> stop do

begin

if CompareText(Current^.data.addr, pivot^.data.addr) < 0 then

// Сравниваем строки addr

begin

pivot := pivot^.next;

SwapOrders(pivot, Current);

end;

Current := Current^.next;

end;

SwapOrders(start, pivot);

Result := pivot;

end;

procedure QuickSort(start, stop: POrder);

var

pivot: POrder;

begin

if (start <> nil) and (stop <> nil) and (start <> stop) and

(start <> stop^.next) then

begin

pivot := Partition(start, stop^.next);

QuickSort(start, pivot);

QuickSort(pivot^.next, stop);

end;

end;

procedure Sort(var Head, Tail: POrder);

begin

QuickSort(Head, Tail);

end;

procedure OrdersByAddr(Head, Tail: POrder; argID: Integer; argAddr:string; cmp:TCompare);

var

CurrentL, CurrentR: POrder;

found, nextStep: Boolean;

begin

// Сначала сортируем список в алфавитном порядке

Sort(Head, Tail);

found := False;

// Ищем первую запись с совпадающим адресом

currentL:=FindOrder(Head, argID, argAddr, cmp);

currentR:=currentL;

// Если запись не найдена, выходим из процедуры

if (currentL <> nil) and (currentR<>nil) then

begin

// Выводим найденную запись

Writeln('Найден(ы) заказ(ы):');

Writeln('ID: ', currentL^.data.id);

Writeln('Адрес: ', currentL^.data.addr);

Writeln('Время прибытия: ', FormatDateTime('HH:NN', currentL^.data.time\_in));

Writeln('Время убытия: ', FormatDateTime('HH:NN', currentL^.data.time\_out));

Writeln('Объем: ', currentL^.data.volume:4:2);

Writeln('Вес: ', currentL^.data.weight:4:2);

// Вывод остальных данных

Writeln;

found := True;

// Перемещаемся к предыдущим записям с совпадающим адресом

while (currentL^.prev <> nil) and (currentL^.prev^.data.addr = argAddr) do

begin

currentL := currentL^.prev;

// Выводим найденную запись

Writeln('ID: ', currentL^.data.id);

Writeln('Адрес: ', currentL^.data.addr);

Writeln('Время прибытия: ', FormatDateTime('HH:NN',

currentL^.data.time\_in));

Writeln('Время убытия: ', FormatDateTime('HH:NN',

currentL^.data.time\_out));

Writeln('Объем: ', currentL^.data.volume:4:2);

Writeln('Вес: ', currentL^.data.weight:4:2);

// Вывод остальных данных

Writeln;

found := True;

end;

// Перемещаемся к следующим записям с совпадающим адресом

// Перемещаемся к следующим записям с совпадающим адресом

while (currentR^.next <> nil) and (currentR^.next^.data.addr = argAddr) do

begin

currentR := currentR^.next;

// Выводим найденную запись

Writeln('ID: ', currentR^.data.id);

Writeln('Адрес: ', currentR^.data.addr);

Writeln('Время прибытия: ', FormatDateTime('HH:NN',

currentR^.data.time\_in));

Writeln('Время убытия: ', FormatDateTime('HH:NN',

currentR^.data.time\_out));

Writeln('Объем: ', currentR^.data.volume:4:2);

Writeln('Вес: ', currentR^.data.weight:4:2);

// Вывод остальных данных

Writeln;

found := True;

end;

end;

// Если не было найдено ни одной записи с совпадающим адресом, выведем сообщение об этом

if not found then

Writeln('Записи с адресом "', argAddr, '" не найдены.');

end;

// Удaление из списка

procedure DeleteOrder(Head: POrder; const id: Integer);

var

order: POrder;

str:string;

begin

str:='';

order := FindOrder(Head, id,str, CompareOrderID);

if order^.prev <> nil then

order^.prev^.next := order^.next;

if order^.next <> nil then

order^.next^.prev := order^.prev;

end;

procedure EditOrder(Head, Tail: POrder; const id: Integer);

var

order: POrder;

isWriting: Boolean;

strParam, exitStr,str: string;

doubleParam: Double;

menuVal:integer;

begin

str:='';

order := FindOrder(Head, id,str, CompareOrderID);

Writeln('Запись, которую необходимо отредактировать');

Writeln('ID: ', order^.data.id);

Writeln('1) Адрес: ', order^.data.addr);

Writeln('2) Время прибытия: ', FormatDateTime('HH:NN', order^.data.time\_in));

Writeln('3) Время убытия: ', FormatDateTime('HH:NN', order^.data.time\_out));

Writeln('4) Объем: ', order^.data.volume:4:2);

Writeln('5) Вес: ', order^.data.weight:4:2);

Writeln('6) Выйти из режима редактирования');

while isWriting do

begin

Write('Выберите поле, которое необходимо отредактировать (6 - выйти): ');

readln(menuVal);

case menuVal of

1:

begin

Write('Введите адрес доставки: ');

Readln(strParam);

order^.data.addr := strParam;

end;

2:

begin

Write('Введите время с которого можно доставить: ');

Readln(strParam);

order^.data.time\_in := StrToTime(strParam);

end;

3:

begin

Write('Введите время до которого можно доставить: ');

Readln(strParam);

order^.data.time\_out := StrToTime(strParam);

end;

4:

begin

Write('Введите объем(м^3) заказа: ');

Readln(doubleParam);

order^.data.volume := doubleParam;

end;

5:

begin

Write('Введите вес(кг) заказа: ');

Readln(doubleParam);

order^.data.weight := doubleParam;

end;

6:

begin

isWriting:=False;

end;

end;

end;

end;

end.