***Цель работы:***

***Разработать программу, осуществляющую нахождение пути от одного населённого пункта к другому.***

**Введение**

В настоящее время индустрия производства компьютеров и программного обеспечения для них является одной из наиболее важных сфер экономики развитых стран. Ежегодно в мире продаются десятки миллионов компьютеров. Только в США объем продаж компьютеров составляет десятки миллионов долларов и постоянно продолжает расти.

В чем же причины такого стремительного роста индустрии персональных компьютеров и их сравнительная выгодность для многих деловых применений?

1. Простота использования, обеспеченная с помощью диалогового способа взаимодействия с компьютером.
2. Относительно высокие возможности по переработке информации, наличие программного обеспечения, а так же мощных систем для разработки нового программного обеспечения.

Использованная в отчёте программа может использоваться для решения задач, связанных с проложением маршрута дороги любого типа.

# Определение достижимости населённых пунктов.

**1.1 Анализ требований.**

В списке задаются города (населённые пункты), а также дороги между ними (есть или нет), необходимо разработать программу с использованием модульного программирования, осуществляющую нахождение кратчайшего пути между населёнными пунктами, задаваемыми пользователем в процессе работы программы.

Решение поставленной задачи осуществляется следующим методом:

Cтроится граф, вершины которого - населённые пункты, а ребра - дороги между ними.

В процессе работы программы в данном графе с помощью рекуррентной процедуры находятся пути из одной вершины в другую. Данная процедура в качестве параметров получает массив пройденных вершин, текущую вершину и количество уже пройденных вершин. На каждом этапе процедура проверяет все, не пройденные достигнутые вершины, и либо находит заданный путь, если достигнута конечная вершина, либо вызывает саму себя для всех, не пройденных вершин.

Для организации данного алгоритма используется две процедуры: процедура нахождения всего пути и рекурсивная процедура поиска единичного маршрута.

Процедура нахождения всего пути осуществляет перебор всех населённых пунктов и вызов рекурсивной процедуры, которая осуществляет поиск маршрута между этими населёнными пунктами.

Средства решения задачи: используются средства логического программирования языка Turbo Pascal 7.0.

**1.2 Проектирование.**

Для реализации поставленной задачи программа должна выполнять следующие функции:

1. Ввод данных пользователем с клавиатуры - вводятся названия населённых пунктов и дороги, соединяющие их;
2. Вывод данных - вывод на экран списка населённых пунктов и дорог, соединяющий их.
3. Запись в файл - запись в файл, имя которого указывает пользователь в диалоговом режиме, названия населённых пунктов и существующих дорог между ними в виде текстовой информации;
4. Считывание файла с диска, с именем, которое указывает пользователь в диалоговом режиме
5. Вывод результата - пользователь задаёт начальный и конечный населённый пункт, между которыми необходимо проложить путь, на экране появляется маршрут, либо сообщение: "маршрут не найден".

Данная программа реализована с использованием принципа модульного программирования, главным преимуществом которого является простота использования, возможность подключения программой разных модулей, которые могли быть разработаны ранее, быстрое нахождение основного текста программы, а также исправление и отладка процедур при использовании другой программы или специальной программы-отладчика, которая подключает к себе данный модуль.

Все процедуры, используемые данной программой, находятся в unit-модуле ph.tpu и предназначены для использования основной программой, которая находится в файле path.pas.

Основная программа осуществляет вывод меню на экран, опрос клавиатуры и вызов процедуры, соответствующей нажатой клавише.

Для реализации ввода данных используется процедура InputData, которая осуществляет ввод имён городов с клавиатуры, если вместо названия города был нажат ввод, то процедура выводит список городов на экран и пользователь, передвигая курсор и, нажимая ввод, составляет список дорог, соединяющие эти города между собой, при нажатии клавиши Esc процедура прекращает свою работу и выходит в главное меню.

Для реализации вывода данных на экран используется процедура OutputData, которая осуществляет чтение в цикле массива городов и вывод его на экран, а также массива дорог, соединяющие эти города и выводит из на экран.

Для реализации запроса имени файла и записи данных в файл используется процедура Save, которая сначала выводит запрос на экран, осуществляет ввод имени файла, открывает файл, имя которого вводится пользователем с клавиатуры в текущем каталоге, в цикле из массива городов записывает на диск список городов, затем также список дорог, соединяющих их.

 Для реализации запроса имени файла и чтения данных из файла в массив используется процедура load, которая сначала выводит запрос имени файла на экран, осуществляет ввод имени файла, открывает файл, имя которого вводится пользователем, считывает данные в массив городов, затем в массив дорог.

Для поиска пути между городами используется процедура FindPath, которая осуществляет вывод списка городов на экран, опрос клавиатуры, при этом пользователь может выбрать курсором начальный и конечный населённый пункт в своём пути, процедура FindPath вызывает с параметрами рекурсивную процедуру, которая производит поиск оптимального маршрута между выбранными городами.

Для поиска маршрута используется рекурсивная процедура findnext, которой при её вызове передаются следующие параметры:

a(vec) - вектор, каждому городу соответствует номер в маршруте или ноль, если города нет в маршруте;

tv(integer) - город, следующий в маршруте;

nv(integer) - город, в который необходимо добраться;

lv(integer) - количество пройденных городов.

На каждом этапе процедура проверяет все, не пройденные достигнутые вершины, и либо находит заданный путь, если достигнута конечная вершина, либо вызывает саму себя для всех, не пройденных вершин.

**1.3 Кодирование**

Краткая функциональная спецификация.

Процедура InputData

Назначение: Осуществляет ввод исходных данных пользователем с клавиатуры.

Входные данные: нет.

Выходные данные: нет.

Не вызывает никаких процедур.

Вызывается из основной программы.

Процедура OutputData

Назначение: Осуществляет вывод данных на экран.

Входные данные: нет.

Выходные данные: нет.

Не вызывает никаких процедур.

Вызывается из основной программы.

Процедура Load

Назначение: Осуществляет запрос имени, чтение файла данных с этим именем в массив городов и в массив дорог.

Входные данные: нет.

Выходные данные: нет.

Не вызывает никаких процедур.

Вызывается из основной программы.

Процедура Save

Назначение: Осуществляет запрос имени, запись в файл данных с этим именем массива городов и в массива дорог.

Входные данные: нет.

Выходные данные: нет.

Не вызывает никаких процедур.

Вызывается из основной программы.

Процедура FindPath

Назначение: Осуществляет поиск пути между городами.

Входные данные: нет.

Выходные данные: нет.

Вызывает findnext.

Вызывается из основной программы.

Процедура FindNext

Назначение: Осуществляет поиск маршрута.

Входные данные:

 a(vec) - вектор, каждому городу соответствует номер в

маршруте или ноль, если города нет в маршруте;

tv(integer) - город, следующий в маршруте;

nv(integer) - город, в который необходимо добраться;

lv(integer) - количество пройденных городов.

Выходные данные: нет.

Вызывает findnext.

Вызывается из FindPath.

Основная программа

Осуществляет оформление экрана, вывод и обработку меню, опрос клавиатуры, вызов процедуры, соответствующей выбранному пункту меню.

**1.4 Тестирование**

Разработанное программное средство было протестировано методом функционального тестирования.

Введённые в программу данные показали, что результаты работы совпадают с вычисленными вручную.

**Программы разработки.**

Программа path

program path;

uses crt,ph;

var

 t:town; {Данные о городах}

 nt:integer; {Число городов}

 r:road; {Данные о дорогах}

 nr:integer; {Число дорог}

 sl:integer; {Выбранный пункт меню}

 c:char; {Нажатый символ}

 i:integer; {Счетчик}

 fv:vec; {Вектор пройденных городов}

 nfv:integer; {Количество городов}

Const

 KItems = 6; {Количество пунктов меню}

 mas: array[1..KItems] of string =

 {Инициализация пунктов меню}

 ('¦ Ввод данных ¦',

 '¦ Вывод данных ¦',

 '¦ Запись в файл ¦',

 '¦ Считывание файла ¦',

 '¦ Вывод результата ¦',

 'L------ Выход -------');

{Основная программа}

begin

 sl:=1;

{Городов и дорог нет}

 nt:=0;

 nr:=0;

 repeat

 textattr:=7; {Основной цвет меню}

 clrscr;

 for i:=1 to KItems do begin

 gotoxy (25,i+3);

 write (mas[i]); {Вывод пунктов меню}

 end;

 textattr:= 77; {Цвет активного пункта}

 gotoxy (25,sl+3);

 write (mas[sl]); {Вывод активного пункта}

 c:=readkey; {Ввод символа с клавиатуры}

 textattr:=7;

 case c of {Определить код нажатой клавиши}

 #13: case sl of {Клавиша Enter}

 1: InputData;

 2: OutputData;

 3: Save;

 4: Load;

 5: FindPath;

 end;

 #0: begin {Анализ функциональных клавиш}

 c:=readkey;

 case c of

 #80: if sl<KItems then sl:=sl+1 else sl:=1;

 #72: if sl>1 then sl:=sl-1 else sl:=KItems;

 end

 end

 end;

 until ((c=#13) and (sl=6) or (c=#27));

 textattr:=7;

 clrscr;

end.

Модуль ph

unit ph;

interface

uses crt;

type

 town= array [1..20] of string; {Данные о городах}

 road= array [1..200] of record {Данные о дорогах}

 a:integer;

 b:integer;

 end;

 vec=array [1..20] of integer; {Данные о пройденных городах}

var

 t:town; {Данные о городах}

 nt:integer; {Число городов}

 r:road; {Данные о дорогах}

 nr:integer; {Число дорог}

 fv:vec; {Вектор пройденных городов}

 nfv:integer; {Количество городов}

procedure InputData;

procedure OutputData;

procedure Save;

procedure Load;

procedure findnext(a:vec; tv:integer; nv:integer; lv:integer);

procedure FindPath;

implementation

**{Ввод данных}**

procedure InputData;

var

 i:integer; {Счетчик}

 n:integer; {Выбранный начальный город}

 sl:integer; {Выбранный город}

 c:char; {Нажатый символ}

begin

{Считывание данных о городах}

 clrscr;

 nt:=1;

 writeln('Введите название города (Пустая строка - закончить: ');

 repeat

 write(' >');

 readln(t[nt]);

 nt:=nt+1;

 until (t[nt-1]='');

 nt:=nt-2;

{Проверка, вводились ли города}

 if (nt>0) then begin

 {Да, ввод дорог}

 nr:=0;

 n:=0;

 sl:=1;

 repeat

 textattr:=7;

 clrscr;

 for i:=1 to nt do begin

 gotoxy (25,i+3);

 write (t[i]); {Вывод городов}

 end;

 textattr := 77; {Цвет активного города}

 gotoxy (25,sl+3);

 write (t[sl]); {Вывод активного города}

 if (n<>0) then begin

 textattr:=66; {Цвет отмеченного города}

 gotoxy (25,n+3);

 write (t[n]); {Вывод отмеченного города}

 end;

 textattr:=7;

 gotoxy(1,20);

 write('Дороги: ');

 for i:=1 to nr do write(' {',r[i].a,',',r[i].b,'}');

 c:=readkey; {Ввод символа с клавиатуры}

 case c of

 #13: begin {Нажат ENTER}

 {Либо помечается начальный город}

 if n=0 then n:=sl else

 {Либо происходит попытка добавить дорогу}

 if (n=sl) then n:=0 else begin

 nr:=nr+1;

 if (n>sl) then begin

 i:=n;

 n:=sl;

 sl:=i;

 end;

 {Проверяется, нет ли такой?}

 for i:=1 to nr-1 do

 if ((r[i].a=n)and(r[i].b=sl)) then n:=0;

 {Если нет - добавляется}

 if n<>0 then begin

 r[nr].a:=n;

 r[nr].b:=sl;

 end else nr:=nr-1;

 n:=0;

 sl:=1;

 end;

 end;

 #0: begin {Анализ функциональных клавиш}

 c:=readkey;

 case c of

 #80: if sl<nt then sl:=sl+1 else sl:=1;

 #72: if sl>1 then sl:=sl-1 else sl:=nt;

 end

 end

 end;

 until (c=#27);

 end;

end;

**{Вывод данных}**

procedure OutputData;

var

 i:integer; {Счетчик}

begin

{Вывод списка городов}

 clrscr;

 writeln(' Города: ');

 for i:=1 to nt do begin

 gotoxy (10,i);

 write (t[i]); {Вывод городов}

 end;

{Вывод списка дорог}

 gotoxy(1,20);

 write(' Дороги: ');

 for i:=1 to nr do write(' {',r[i].a,',',r[i].b,'}');

 gotoxy(2,24);

 write(' ESC- Выход');

{Ожидание ESC}

 repeat until readkey=#27;

end;

**{ Запись данных в файл}**

procedure Save;

var

 f:TEXT; {Файл}

 name:string; {Имя файла}

 n:integer; {Счетчик}

begin

 clrscr;

 writeln(' Запись данных ');

 write(' Введите имя файла: ');

 readln(name);

 assign(f,name);

 rewrite(f);

 writeln(f,nt);

 for n:=1 to nt do writeln(f,t[n]);

 writeln(f,nr);

 for n:=1 to nr do writeln(f,r[n].a,' ',r[n].b);

 close(f);

end;

**{Чтение из файла}**

procedure Load;

var

 f:TEXT; {Файл}

 name:string; {Имя файла}

 n:integer; {Счетчик}

begin

 clrscr;

 writeln(' Чтение данных ');

 write(' Введите имя файла: ');

 readln(name);

 assign(f,name);

 reset(f);

 readln(f,nt);

 for n:=1 to nt do readln(f,t[n]);

 readln(f,nr);

 for n:=1 to nr do readln(f,r[n].a,r[n].b);

 close(f);

end;

{**Рекурсивная процедура поиска маршрута.**

 Входные параметры:

 a:vec - Вектор, каждому городу соответствует номер в маршруте

 или ноль, если города нет в маршруте

 tv:integer - Город, следующий в маршруте

 nv:integer - Город, в который необходимо добраться

 lv:integer - Количество пройденных городов}

procedure findnext(a:vec;tv:integer;nv:integer;lv:integer);

var

 i:integer; {Счетчик}

begin

 a[tv]:=lv; {Устанавливается в векторе

 флаг, что город tv пройден}

 if (tv=nv) then begin

 {Если достигнут необходимый город}

 if ((lv+1)<nfv)or(nfv=0) then begin

{Если найден первый либо более короткий маршрут - он становится найденным}

 nfv:=lv+1;

 fv:=a;

 end;

 end else begin

{Иначе - просмотр всех городов, в которые можно добраться из данного}

 for i:=1 to nr do

{Если город еще не учитывался - запуск для него той же самой функции}

 if ((r[i].a=tv)and(a[r[i].b]=0)) then findnext(a,r[i].b,nv,lv+1);

{Просмотр, но для дорог, где текущий город учитывался как второй}

 for i:=1 to nr do

{Если город еще не учитывался - запуск для него той же самой функции}

 if ((r[i].b=tv)and(a[r[i].a]=0)) then findnext(a,r[i].a,nv,lv+1);

 end;

end;

**{Нахождение пути}**

procedure FindPath;

var

 i:integer; {Счетчик}

 j:integer; {Счетчик}

 n:integer; {Исходный город}

 sl:integer; {Выбираемый город}

 c:char; {Считанный с клавиатуры символ}

 v:vec; {Вектор для начала рекурсии}

begin

{Изначально первый город не выбран}

 n:=0;

 sl:=1;

 nfv:=0; {Маршрут не найден}

{Цикл запроса городов и вывода результатов}

 repeat

 textattr:=7;

 clrscr;

 {Вывод поясняющей надписи}

 gotoxy(2,20);

 if (n=0) then write(' Выберите начальный пункт')

 else writeln(' Выберите конечный пункт ');

 {Вывод списка городов}

 for i:=1 to nt do begin

 gotoxy (25,i+3);

 write (t[i]);

 end;

 textattr:= 77;

 gotoxy (25,sl+3);

 write (t[sl]);

 if (n<>0) then begin

 textattr:=66;

 gotoxy (25,n+3);

 write (t[n]); {Вывод отмеченного города}

 end;

 textattr:=7;

 {Вывод найденного маршрута либо надписи о его отсутствии}

 gotoxy(60,1);

 if (nfv>0) then begin

 write(' Найденный маршрут: ');

 for j:=1 to nfv do

 for i:=1 to nt do if fv[i]=j then begin

 gotoxy(60,j+2);

 write(t[i]);

 end;

 end else write(' Маршрут не найден ');

 c:=readkey; {Ввод символа с клавиатуры}

 case c of

 #13: begin

 {Либо фиксируется начальный город}

 if n=0 then n:=sl else begin

 {Либо убирается ошибочно выбранный город}

 if (n=sl) then n:=0 else begin

 {Либо происходит поиск нового маршрута}

 nfv:=0; {Маршрута нет}

 for i:=1 to 20 do v[i]:=0; {Ни одного пройденного

 города}

 findnext(v,n,sl,1);{Вызывается первый раз рекурсивная

 процедура}

 end;

 n:=0;

 sl:=1;

 end;

 end;

 #0: begin {Анализ функциональных клавиш}

 c:=readkey;

 case c of

 #80: if sl<nt then sl:=sl+1 else sl:=1;

 #72: if sl>1 then sl:=sl-1 else sl:=nt;

 end

 end

 end;

 until (c=#27);

end;

end.

**Результаты выполнения программы.**

 ¦ Ввод данных ¦

 ¦ Вывод данных ¦

 ¦ Запись в файл ¦

 ¦ Считывание файла ¦

 ¦ Вывод результата ¦

 +------ Выход ------+

Ввод данных:

Введите название города (Пустая строка - закончить):

 >Город 1

 >Город 2

 >Город 3

 >Город 4

 >Город 5

 >

 Дороги: {1,3} {3,4} {2,5} {1,4} {2,4} {2,3}

Вывод результата:

Найденный маршрут: Город 1 Город 1

Город 3 Город 2

Город 2 Город 3

Город 5 Город 4

 Город 5

**Список используемых источников**

1. Белецкий Я. Турбо Паскаль с графикой для персональных компьютеров /перевод с польского Д. И. Юренкова. - М. :Машиностроение, 1991.

2. Сергиевский М. В., Шалашов А. В. Турбо Паскаль 7.0; язык, среда программирования. - М: Машиностроение, 1994.

3. Справочник по процедурам и функциям Borland Pascal With Objects 7.0. - Киев: Диалектика, 1993.