Лабораторная работа №2.

**Разработка форматов хранения данных программы. Структурирование.**

Цель:1. Изучить возможность работы с файлами на языке Pascal. Получить навыки разработки формата хранения данных на примере решения задач аналитической геометрии.

2. Освоить приёмы структурирования программ.

Задание:

1. Изучить процедуры и функции языка Pascal для работы с файлами прямого и последовательно доступа.
2. Согласно заданному варианту разработать формат и структуру в файлах прямого доступа.
3. Реализовать возможность ввода данных с клавиатуры либо из файла.
4. Выходные данные вывести на экран и сохранить в текстовый файл.
5. При написании программы использовать контроль вводимых данных, используя функцию IOResult.
6. Условие задачи с реквизитами исполнителя должны быть предоставлены на экране при запуске программы.
7. Структурировать программу с использованием процедур и функций.

Условие задачи аналитической геометрии:

Найти количество окружностей на плоскости имеющих пересечение с прямоугольником стороны которого параллельны осям координат.

Содержание

1. Теоретическая часть

2. Алгоритм программы

3. Листинг программы

4. Экранные формы

**1. Теоретическая часть**

1. По методу доступа к компонентам файлы разделяются на последовательные и прямые. В последовательном файле чтение и запись компонент может быть только последовательной, начиная с первой компоненты. В прямом файле можно прочесть или записать любую компоненту, указав ее номер в файле. При этом перебор предыдущих компонент не требуется. По направлению передачи данных файлы разделяются на входные и выходные.

Также существуют три типа файлов:

1) типизированные, компоненты которых имеют жесткую структуру;

2) текстовые, состоящие из символов и строк;

3) нетипизированные, рассматриваемые как последовательность байтов.

В Турбо-Паскале можно открыть файл только для чтения, только для записи, а также для чтения и записи одновременно. Для этого используются процедуры **Reset**, **Rewrite**, **Append**.

**Reset(F)** - открытие существующего файла. Если это текстовый файл, то допускается только чтение из файла; для типизированного файла допустимы как чтение, так и запись информации. В последнем случае можно легко реализовать обновление компонент файла. Если файл F уже был открыт, то при выполнении процедуры **Reset** этот файл сначала закрывается, а затем открывается. При этом указатель файла устанавливается в начальную позицию файла.

**Rewrite(F)** - открытие нового файла. Если это текстовый файл, то допускается только запись в файл; для типизированного файла допустимы как чтение, так и запись информации. Если файл F уже существует, то при срабатывании процедуры **Rewrite** файл F уничтожается.

**Append(F)** - открытие текстового файла для добавления новых компонент в конец файла. Если файл уже был открыт ранее с помощью процедур **Reset** или **Rewrite**, то процедура **Append** вначале закрывает данный файл, а затем его открывает для добавления новых компонент.

Существуют процедуры и функции для файлов любого типа:

Процедура **Close(F)**. Выполняется закрытие файла F. Если буфер вывода заполнен не полностью, то его содержимое переносится в файл F, после чего из оперативной памяти удаляется служебная информация о файле F.

Процедура **Rename(F, newname)**. Внешнему файлу, связанному с файловой переменной F, присваивается новое имя, заданное в строке.

Процедура **Erase(F)**. Файл F уничтожается. Процедура **Erase** применима только для закрытых файлов.

Функция **Eof(F):boolean**. Функция возвращает значение true, если достигнут конец файла; в противном случае выходное значение равно false.

Функция **IOResult:word** (Input-Output-Result). Возвращает условный признак последней операции ввода-вывода. Результат равен нулю, если операция ввода-вывода завершилась успешно.

Процедура **Truncate(F)**. Файл усекается по текущей позиции указателя. В эту позицию записывается признак eof.

Для выполнения операций ввода-вывода в текстовых файлах используются процедуры **Read**,**Readln**,**Write**,**Writeln**.

Также для текстовых файлов могут использоваться три предописанные функции, неприменимые для файлов другого типа: **Eoln**, **SeekEoln** и **SeekEof**.

Функция **Eoln(F):boolean** возвращает значение true, если во входном потоке достигнут маркер конца строки eol.

Функция **SeekEoln(F):boolean** пропускает все пробелы и знаки табуляции до маркера конца строки eol или до первого значащего символа и возвращает значение true, если такой маркер обнаружен.

Функция **SeekEof(F):boolean** пропускает все пробелы, знаки табуляции и маркеры конца строки вплоть до маркера конца файла eof или до первого значащего символа и возвращает значение true, если такой маркер обнаружен.

Все компоненты типизированного файла, в отличие от текстового файла, имеют одну и ту же длину. Это позволяет программе определить местоположение любой компоненты файла по ее номеру и осуществить прямой доступ к этой компоненте. Компоненты типизированного файла нумеруются натуральным рядом чисел 0,1,2,3,...При открытии типизированного файла его указатель устанавливается на нулевую компоненту. После выполнения каждой операциичтения или записи указатель сдвигается на следующую компоненту.

Для операций чтения и записи в типизированном файле используются процедуры **Read** и **Write** (но не Readln,Writeln).

Для типизированных файлов определены также предописанные процедура **Seek** и функции **FileSize**,**FilePos**.

Процедура **Seek(F,k)** перемещает указатель файла F на компоненту с номером k. Переменная k должна иметь тип longint.

Функция **FileSize(F):longint** возвращает текущий размер файла F (количество компонент файла).

Функция **FilePos(F):longint** возвращает номер текущей позиции файла F.

Нетипизированные файлы применяются главным образом для высокоскоростного обмена данными между диском и памятью. Для этих файлов вместо процедур **Read** и **Write** используются процедуры **BlockRead** и **BlockWrite**.

2. Исходя из условия задачи входные данные для программы:

а) Координаты точки центра окружности;

b) Длина радиуса окружности;

c) Координаты левой верхней и нижней правой точки прямоугольника, однозначно задающие его на плоскости.

Так как количество и тип данных чётко определён в качестве файлов для ввода исходных данных будем использовать типизированные файлы, компоненты которых имеют жесткую структуру и к ним возможен прямой доступ.

Для хранения параметров окружности создадим типизированный файл circle.dat, который будет состоять из массива составных компонент имеющих три параметра: координаты x и y центра окружности и R - радиус окружности. Тип компоненты определим как запись, потому что в записи можно объединить компоненты различных типов:

**CircleType = record**

**x : integer;**

**y : integer;**

**R : word**

**end;**

Координаты центра окружности x и y будут определены как целые числа, которые могут принимать положительное и отрицательное значение, поэтому для них выбран тип integer. Радиус окружности R не может быть отрицательным для него выбран тип word - целое без знака, пределы изменения 0 .. 65535.

Таким образом массив окружностей будет описываться:

**CircleAr : array[1..Nmax] of CircleType;**

Описание файла:

**CircleFile : file of CircleType;**

Для ввода параметров задающих прямоугольник будем использовать типизированный файл rectangle.dat с двумя элементами типа запись:

**RectangleType=record**

**x : integer;**

**y : integer;**

**end;**

Таким образом каждый компонент файла состоит из координат x и y угловых точек прямоугольника, массив точек прямоугольника и файл будут описываться:

**RectangleAr : array[1..2] of RectangleType;**

**RectangleFile : file of RectangleType;**

3. В программе реализована возможность ввода исходных данных из файла или клавиатуры. После запуска программы выдаётся запрос о направлении ввода “Откуда произвести ввод исходных данных? f – файл, k-клавиатура”. В соответствии с нажатой клавишей осуществляется переход к вводу данных.

4. Результаты выполнения программы выводятся на экран и в текстовый файл result.txt, создаваемый в каталоге из которого запущена программа.

5. В программе осуществляется контроль вводимых данных:

При вводе числового ответа с клавиатуры существует определенная вероятность ошибочного набора (например, буква вместо цифры). При вводе ответа с помощью оператора Read это вызвало бы прерывание программы и, как следствие, необходимость ее повторного запуска. Чтобы блокировать такое прерывание, ввод переменной производится с помощью процедуры ReadInt, ReadWord в которых при помощи стандартной процедуры IOResult осуществляющей контроль формата вводимого числа. При неправильном формате ввода процедура ReadInt, ReadWord выдает на экран сообщение об ошибке и предлагает пользователю повторить ввод.

6. При запуске программы на экране в первую очередь представлены условия задачи и реквизиты исполнителя, реализованные следующим участком кода:

ClrScr;

Writeln('▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒');

Writeln('▒ Лабораторная работа №2 ▒');

Writeln('▒ студента гр. ▒');

Writeln('▒ ▒');

Writeln('▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒');

Writeln('▒ Условие задачи по аналитической геометрии: ▒');

Writeln('▒ Найти количество окружностей на плоскости ▒');

Writeln('▒ имеющих пересечение с прямоугольником стороны ▒');

Writeln('▒ которого параллельны осям координат. ▒');

Writeln('▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒');

1. В программе используются следующие процедуры и функции, которые структурируют программу: FrazaReadError, ReadWord, ReadInt, PrintInData, Cross1, Cross2, PrintHead.

**2. Листинг программы**

Program Lab2;

uses Crt;

label ExitFromCalc;

Const Nmax=100;

Type

CircleType = record

x : integer;

y : integer;

R : word

end;

RectangleType=record

x : integer;

y : integer;

end;

Var

CircleAr : array[1..Nmax] of CircleType;

CircleFile : file of CircleType;

RectangleAr : array[1..2] of RectangleType;

RectangleFile : file of RectangleType;

ResultFile : text;

n : word; {Кол-во окружностей для проверки}

i : byte; {параметр цикла}

S : byte; {счётчик пересечений окружностей с прямоугольником}

ch : char;

size : longint;

{ ------------------------------------------------------------ }

Procedure FrazaReadError(k:integer);

{ Сообщение о неправильном формате вводимого числа с подачей }

{ звукового сигнала }

Begin

If k<>0 then

Begin

Writeln(#7'Неправильный формат числа');

Writeln('Повторите ввод');

End;

End { FrazaReadError };

{ ------------------------------------------------------------ }

Procedure ReadWord(Var Number:word);

{ Ввод с клавиатуры и проверка формата переменной типа word}

Var k : word;

Begin

Repeat

{$I-} Read(Number); {$I+}

k:=IOResult;

FrazaReadError(k);

Until k=0;

End { ReadWord };

{ ------------------------------------------------------------ }

Procedure ReadInt(Var Number:integer);

{ Ввод с клавиатуры и проверка формата переменной типа integer }

Var k : integer;

Begin

Repeat

{$I-} Read(Number); {$I+}

k:=IOResult;

FrazaReadError(k);

Until k=0;

End { ReadInt };

{ ------------------------------------------------------------ }

Procedure PrintInData;

Var i : byte;

Begin

Writeln(' Исходные данные');

Writeln('Окружности:');

For i:=1 to n do

Begin

Write(i,':','R=',CircleAr[i].R,' (',CircleAr[i].x,':',CircleAr[i].x,')','; ');

End;

Writeln;

Writeln('Прямоугольник:');

Writeln('Верхняя правая точка: (',RectangleAr[1].x,':',RectangleAr[1].y,')');

Writeln('Верхняя правая точка: (',RectangleAr[2].x,':',RectangleAr[2].y,')');

End {PrintInData};

{ ------------------------------------------------------------ }

Function Cross1(Var R:word; P,C:integer):boolean;

Begin

Cross1:=false;

if (R\*R-(P-C\*C))>=0 then Cross1:=true;

End {Cross1};

{ ------------------------------------------------------------ }

Function Cross2(Var Inter1,Inter2:integer; R:word; P,C1,C2:integer):boolean;

Var Buf : real;

Begin

Cross2:=false;

Buf:=sqrt(R\*R-(P-C1\*C1))+C2;

if Buf<=Inter2 then

if Buf>=Inter1 then Cross2:=true;

End {Cross2};

{ ------------------------------------------------------------ }

Procedure PrintHead;

Begin

ClrScr;

Writeln('▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒');

Writeln('▒ Лабораторная работа №2 ▒');

Writeln('▒ студента гр▒');

Writeln('▒ ▒');

Writeln('▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒');

Writeln('▒ Условие задачи по аналитической геометрии: ▒');

Writeln('▒ Найти количество окружностей на плоскости ▒');

Writeln('▒ имеющих пересечение с прямоугольником стороны ▒');

Writeln('▒ которого параллельны осям координат. ▒');

Writeln('▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒');

End { ReadInt };

{ ------------------------------------------------------------ }

Begin

PrintHead;

Writeln('Откуда произвести ввод исходных данных?');

Writeln('f - файл, k-клавиатура');

Repeat

ch:=ReadKey;

Until ((ch='k') or (ch='f'));

if (ch='k') then

Begin

Writeln('Набор параметров завершайте нажатием клавиши Enter');

Write('Введите количество окружностей:'); ReadWord(n);

For i:=1 to n do

Begin

Writeln('Введите кординаты центра ',i,' окружности.');

Write('x='); ReadInt(CircleAr[i].x);

Write('y='); ReadInt(CircleAr[i].y);

Writeln('Введите радиус ',i,' окружности.');

Write('R='); ReadWord(CircleAr[i].R);

End;

For i:=1 to 2 do

Begin

Writeln('Введите кординаты ',i,' точки прямоугольника');

Write('x='); ReadInt(RectangleAr[i].x);

Write('y='); ReadInt(RectangleAr[i].y);

End;

PrintHead;

PrintInData;

Writeln('Записать введённые данные в файлы входных данных?');

Writeln('(y - Да, n - Нет)');

Repeat

ch:=ReadKey;

Until ((ch='y') or (ch='n'));

if (ch='y') then

Begin

Assign(CircleFile,'circle.dat'); Rewrite(CircleFile);

Assign(RectangleFile,'rectangl.dat'); Rewrite(RectangleFile);

For i:=1 to n do

Write(CircleFile,CircleAr[i]);

size:=FileSize(CircleFile);

Writeln('Файл circle.dat перезаписан!!! Текущий размер ',size,' компонент.');

For i:=1 to 2 do

Write(RectangleFile,RectangleAr[i]);

Writeln('Файл rectangle.dat перезаписан!!!');

Close(RectangleFile); Close(CircleFile);

End;

End

else

Begin

Assign(CircleFile,'circle.dat'); Reset(CircleFile);

Assign(RectangleFile,'rectangl.dat'); Reset(RectangleFile);

size:=FileSize(CircleFile);

n:=size;

For i:=1 to n do

Read(CircleFile,CircleAr[i]);

For i:=1 to 2 do

Read(RectangleFile,RectangleAr[i]);

Close(RectangleFile); Close(CircleFile);

Writeln('Данные из входных файлов загружены!!!');

PrintInData;

End;

S:=0;

{Цикл проверки}

For i:=1 to n do

Begin

if Cross1(CircleAr[i].R,RectangleAr[1].y,CircleAr[i].y) then

Begin

if Cross2(RectangleAr[1].x,RectangleAr[2].x,CircleAr[i].R,RectangleAr[1].y,CircleAr[i].y,CircleAr[i].x) then

Begin

S:=S+1; GoTo ExitFromCalc;

End

End;

if Cross1(CircleAr[i].R,RectangleAr[2].y,CircleAr[i].y) then

Begin

if Cross2(RectangleAr[1].x,RectangleAr[2].x,CircleAr[i].R,RectangleAr[2].y,CircleAr[i].y,CircleAr[i].x) then

Begin

S:=S+1; GoTo ExitFromCalc;

End

End;

if Cross1(CircleAr[i].R,RectangleAr[1].x,CircleAr[i].x) then

Begin

if Cross2(RectangleAr[1].y,RectangleAr[2].y,CircleAr[i].R,RectangleAr[1].x,CircleAr[i].x,CircleAr[i].y) then

Begin

S:=S+1; GoTo ExitFromCalc;

End

End;

if Cross1(CircleAr[i].R,RectangleAr[2].x,CircleAr[i].x) then

Begin

if Cross2(RectangleAr[1].y,RectangleAr[2].y,CircleAr[i].R,RectangleAr[2].x,CircleAr[i].x,CircleAr[i].y) then

Begin

S:=S+1; GoTo ExitFromCalc;

End

End;

ExitFromCalc:

End;

{Конец цикла проверки}

Writeln('С прямоугольником пересекаются ',S,' окружности(ей) из ',n,'.');

Assign(ResultFile,'result.txt');

Rewrite(ResultFile);

Write(ResultFile,'С прямоугольником пересекаются ',S,' окружности(ей) из ',n,'.');

Close(ResultFile);

Readln;

End.

4. **Экранные формы:**

