Федеральное агенство по образованию

ГОУ ВПО Тульский государственный педагогический

университет им. Л.Н. Толстого

Курсовая работа

"Массивы в языке Паскаль"

Выполнила

студентка 3 курса

группы Б, ф-та МФиИ

Дикшева О.А.

Проверила

Торина

Тула 2009

Оглавление

Введение

1. Виды массивов

1.1. Одномерные массивы

1.2. Примеры задач

1.3. Двумерные массивы

1.4. Примеры задач

2. Сортировка массивов

2.1Метод простых обменов (Пузырьковая сортировка)

2.2. Сортировка простым выбором

2.3 Сортировка простым включением (Метод вставки и сдвига)

3. Параметры-массивы и параметры-строки

Список литературы

Введение

Существуют различные типы данных в языке Паскаль. Рассмотрим производные типы. Каждое значение любого из этих типов в общем случае представляет собой уже нетривиальную структуру, т.е. обычно это значение имеет более чем одну компоненту. При этом каждая компонента структуры может быть как отдельным данным, так и в свою очередь нетривиальной структурой, т.е, значением любого из производных типов. Таким образом, значения производных типов в общем случае имеют иерархическую структуру, на самом нижнем уровне которой фигурируют только отдельные данные. Этим компонентам нижнего уровня могут присваиваться значения и они могут присутствовать в выражениях, как и значения переменных скалярного типа. Данные, являющиеся значениями скалярных типов, занимают сравнительно мало места в памяти ЭВМ. Отдельная литера, например, обычно представляется одним байтом (8 двоичных разрядов). Для чисел различны типов в зависимости от реализации отводят несколько байтов. Данные же, составляющие значение производного типа, обычно занимают значительный объем памяти ЭВМ. В связи с этим при написании программ для ЭВМ, имеющих сравнительно небольшой объем памяти, встает проблема экономного ее использования. В паскале предусмотрена возможность указания транслятору на необходимость экономного представления значений производных типов. Для этого задание производного типа необходимо начать со служебного слова packed , что означает упакованный. Но введя требование на упакованность данных, необходимо четко представлять себе, что, с одной стороны, это требование не всегда может быть выполнено транслятором (если, например, более экономного представления, чем обычное неупакованное представление для данных этого типа, в ЭВМ просто не существует). А с другой стороны, если оно выполнимо, то приводит к увеличению времени исполнения программы. Поясним на примере, за счет чего это происходит. Как уже указывалось ранее, одна литера занимает один байт. Машинная ячейка памяти, с которой работают команды ЭВМ, в общем случае состоит из нескольких байтов. Поэтому, если в ячейку поместить одну литеру, го большая ее часть не будет использована. На самом деле в одну ячейку можно поместить несколько литер (упакованное представление). Но тогда каждый раз, когда необходимо выполнить действие над отдельной литерой, придется производить выделение этой литеры из ячейки (распаковку литеры из ячейки). Аналогично, при записи отдельной литеры в память машины придется определять то место в ячейке, куда ее необходимо поместить, и заносить литеру именно туда, не изменяя содержимое остальных разрядов (запаковка литеры в ячейку). Такие дополнительные действия могут занимать значительную часть общего времени работы программы. Поэтому принимать решение об использовании упакованного представления данных должен всегда программист, в зависимости от конкретных условий и целей, которые он преследует. Итак, значения производных типов могут быть представлены в памяти ЭВМ в упакованном и неупакованном виде. Упакованное представление требует, вообще говоря, меньшего объема памяти, но замедляет процесс выполнения программы. Мы рассмотрим наиболее употребительный производный тип, а именно регулярный тип. Значение регулярного типа обычно называют массивом. Итак, массив — это упорядоченный набор фиксированного количества некоторых значений (компонент массива). Все компоненты должны быть одного и того же типа, который называют типом компонент или базовым (для массива) типом.

Тип данных Массив позволяет одному идентификатору задать несколько значений, которые отличаются порядковым номером. Номер элемента массива указывается после идентификатора в квадратных скобках {M[5] – пятый элемент массива М}. При описании массива указывается диапазон номеров элементов массива и тип, к которому относится каждый его элемент. Массивы могут быть одно-, двух- и многомерными.

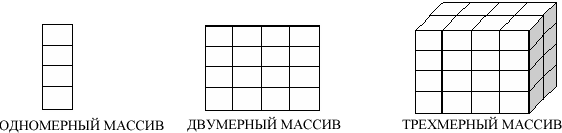


Рис. Изображение одно-, двух- и трехмерных массивов.

Пример описания и заполнения элементов массива.

Var {описание массивов}

M: array [1..5] of integer; {одномерный массив М с номерами элементов от 1 до 5, состоящий из целых чисел}

M1: array [2..3,11..15] of char; {двумерный массив М1 с номерами строк от 2 до 3, с номерами столбцов от 11 до 15, состоящий из символов}

Begin {заполнение массива}

М[2]:=100; {второму элементу численного массива М присвоено значение 100}

М1[2,3]:=’d’; {элементу второй строки и третьего столбца символьного двухмерного массива М1 присвоено значение ’d’}

End.

# 1. Виды массивов

# 1.1 Одномерные массивы

Каждому используемому в программе конкретному массиву должно быть дано свое имя. Это имя будем называть полной переменной, поскольку ее значение есть весь массив. Каждая компонента массива может быть явно обозначена путем указания имени массива, за которым следует селектор компоненты — взятый в квадратные скобки индекс, задающий правило вычисления номера нужной компоненты. Это отличие от привычной записи индекса в математике, когда он указывается справа в нижней позиции, объясняется необходимостью использования линейной записи программы, так что многоуровневая запись должна быть исключена. При ссылке на компоненты массива индекс записывается на одном уровне с именем и заключается в квадратные скобки. Таким образом, для ссылки на отдельные компоненты используется запись вида (имя массива) [<индекс>] которую будем называть частичной переменной (поскольку ее значением является не весь массив, а отдельная его компонента, номер которой задается индексом) — применительно к массивам она называется переменной с индексом. В нашем примере массив получит имя v, а ссылки на отдельные его компоненты производятся с помощью частичных переменных v[ 1], v[2], ..., v[1ОО]. В общем случае в качестве индекса может, быть использовано выражение, значение которого и определяет номер компоненты массива. При этом важно, что в индексное выражение могут входить переменные, так что при изменении их значений меняется и значение индекса, которое определяет номер компоненты массива. Таким образом, одна и та же переменная с индексом в процессе выполнения программы может обозначать различные компоненты массива. Тип значения индексного выражения называют типом индекса. Множество значений типа индекса должно быть перенумерованным множеством, тем самым определяя количество компонент и их упорядоченность. При задании регулярного типа кроме типа индекса необходимо задать тип компонент. Задание такого регулярного типа, как одномерный массив, т.е. вектор, имеет вид:

аrrау [(тип индекса)] оf <тип компонент> ,где <тип компонент> — имя или задание типа.

# 1.2 Примеры задач

Задача 1. Дан линейный массив целых чисел. Подсчитать, сколько в нем различных чисел.

{Подсчет количества различных чисел в линейном массиве}.

ИДЕЯ РЕШЕНИЯ: заводим вспомогательный массив, элементами

которого являются логические величины (False - если элемент

уже встречался ранее, True - иначе)}

Program Razlichnye\_Elementy;

Var I, N, K, Kol : Integer;

A : Array [1..50] Of Integer;

Lo : Array [1..50] Of Boolean;

Begin

Write('Введите количество элементов массива: '); ReadLn(N);

FOR I := 1 TO N DO

Begin

Write('A[', I, ']='); ReadLn (A[I]);

Lo[I] := True; {Заполняем вспомогательный массив значениями True}

End;

Kol := 0; {переменная, в которой будет храниться количество различных чисел}

FOR I := 1 TO N DO

IF Lo[I] THEN

Begin

Kol := Kol + 1;

FOR K := I TO N DO

{Во вспомогательный массив заносим значение False,

если число уже встречалось ранее или совпадает с текущим элементом A[I]}

Lo[K] := (A[K] <> A[I]) And Lo[K];

End;

WriteLn('Количество различных чисел: ', Kol)

END.

Тест: N = 10; элементы массива - 1, 2, 2, 2, -1, 1, 0, 34, 3, 3. Ответ: 6.

Задача 2. Дан линейный массив. Упорядочить его элементы в порядке возрастания.

{Сортировка массива выбором (в порядке возрастания)}.

Идея решения: пусть часть массива (по K-й элемент включительно)

отсортирована. Нужно найти в неотсортированной части массива

минимальный элемент и поменять местами с (K+1)-м}

Program Sortirovka;

Var N, I, J, K, Pr : Integer; A : Array [1..30] Of Integer;

Begin

Write('Введите количество элементов: '); ReadLn(N);

For I := 1 To N Do

Begin

Write('Введите A[', I, '] '); Readln(A[I]);

End;

WriteLn;

For I := 1 To N - 1 Do

Begin

K := I;

For J := I + 1 To N Do If A[J] <= A[K] Then K := J;

Pr := A[I]; A[I] := A[K]; A[K] := Pr;

End;

For I := 1 To N Do Write(A[I], ' ');

End.

Тест: N = 10; элементы массива - 1, 2, 2, 2, -1, 1, 0, 34, 3, 3.

Ответ: -1, -1, 0, 1, 2, 2, 2, 3, 3, 34.

# 1.3 Двумерные массивы

Двумерный массив (прямоугольная таблица (матрица, набор векторов)) - это пример массива, в котором элементы нумеруются двумя индексами.

В качестве номера (индекса) элемента массива используется выражение порядкового типа (чаще integer).

Двумерным массивом называется таблица, состоящая из строк и столбцов. Для описания массива используются два индекса.

А11 А12 А13 … А1m

A21 A22 A23 ... А2m

... ... ... ... ...

... ... ... ... ...

... ... ... ... ...

An1 An2 An3 ... Anm

Описание массива

Способ 1. В разделе описания переменных

var

ИмяМассива: array [Верх.Гр.1..Ниж.Гр.1,Верх.Гр.2..Ниж.Гр.2] of ТипЭлементов;

Способ 2. В разделе описания типов

ИмяМассива: array [Верх.Гр.1..Ниж.Гр.1,Верх.Гр.2..Ниж.Гр.2] of ТипЭлементов;

Способ 3. В разделе описания констант

const

ИмяМассива: array[1..3,1..3] of real=((1.2,2.4,0.4),(0.045,-0.47,0.003),(1.24,1,-7.4));

Заполнение массива данными (ввод элементов)

Массив, описанный как типизированная константа, уже содержит данные. Массивы, объявленные в разделе описания переменных, необходимо заполнить данными, прежде чем выполнять с ними какие-либо действия.

Значения элементов массива также можно задать следующими способами: при вводе данных с клавиатуры:

write('Введите количество строк и столбцов');

readln(n,m);

for i:=1 to n do

for j:=1 to m do

begin

write('a[',i,',',j,']='); {Можно эту строчку в программе не использовать}

readln(a[i,j]);

end;

с помощью датчика случайных чисел:

randomize;

writeln('Введите количество элементов массива');

readln(n);

for i:=1 to n do

begin

a[i]:=random(50);

writeln('a[',i,',',j,']=',a[i,j]);

end;

присваением заданных значений (например по формуле i\*i/i+2):

writeln('Введите количество элементов массива');

readln(n);

for i:=1 to n do

begin

a[i]:=i\*i/i+2;

writeln('a[',i,',',j,']=',a[i,j]);

end;

Вывод массива

вывод в столбец:

for i:=1 to n do

writeln(a[i,j]);

по строкам и столбцам:

for i:=1 to n do

begin

for j:=1 to m do

begin

write(a[i,j]:3);

end;

readln;

end;

Обработка массивов

Часто требуется вычислить сумму элементов массива, их среднее арифметическое значение или найти значения и номера максимального и минимального элементов, а также изменить значения элементов массива и т.д. Особенность работы с двумерными массивами заключается в том, что расширяется возможность обработки массива (появились новые элементы: строки, столбцы - являющиеся одномерными массивами). Подробно все действия можно рассмотреть в задачах разобранных в этом разделе.

Квадратная матрица

Квадратной называется такая матрица, в которой количество строк равно количеству столбцов. Выделяют следующие элементы квадратной матрицы:

главная диагональ;

побочная диагональ;

элементы, расположенные выше главной диагонали;

элементы, расположенные ниже главной диагонали;

элементы, расположенные выше побочной диагонали;

элементы, расположенные ниже побочной диагонали;

Главная диагональ. Если значения индексов (i, j) элемента равны, то элементы расположены на главной диагонали.

А11 А12 А13 А14

A21 A22 A23 А24

A31 A32 A33 А34

A41 A42 A43 А44

if i=j then <инструкции>

Побочная диагональ. Если для значений индексов (i, j) элементов выполняется равенство: i+j=n+1, то элементы расположены на побочной диагонали.

А11 А12 А13 А14

A21 A22 A23 А24

A31 A32 A33 А34

A41 A42 A43 А44

if i+j=n+1 then <инструкции>

Для элементов, расположенных выше главной диагонали необходимо использовать один из следующих фрагментов программы:

А11 А12 А13 А14

A21 A22 A23 А24

A31 A32 A33 А34

A41 A42 A43 А44

for i:=1 to n do

for j:=1 to n do

if i < j then <инструкции>

for i:=1 to n-1 do

for j:=i+1 to n do

<инструкции>

Если элементы расположены на главной диагонали и выше её необходимо использовать следующий фрагмент программы:

А11 А12 А13 А14

A21 A22 A23 А24

A31 A32 A33 А34

A41 A42 A43 А44

for i:=1 to n do

for j:=1 to n do

if i<=j then <инструкции>

Для элементов, расположенных ниже главной диагонали необходимо использовать следующий фрагмент программы:

А11 А12 А13 А14

A21 A22 A23 А24

A31 A32 A33 А34

A41 A42 A43 А44

for i:=1 to n do

for j:=1 to n do

if i>j then <инструкции>

Для элементов, расположенных ниже главной диагонали и не ней необходимо использовать следующий фрагмент программы:

А11 А12 А13 А14

A21 A22 A23 А24

A31 A32 A33 А34

A41 A42 A43 А44

for i:=1 to n do

for j:=1 to n do

if i>=j then <инструкции>

Если элементы, расположены выше побочной диагонали, то необходимо использовать следующий фрагмент программы:

А11 А12 А13 А14

A21 A22 A23 А24

A31 A32 A33 А34

A41 A42 A43 А44

for i:=1 to n-1 do

for j:=1 to n-1 do

if i+j<=n then <инструкции>

Если элементы, расположены ниже побочной диагонали, то необходимо использовать следующий фрагмент программы:

А11 А12 А13 А14

A21 A22 A23 А24

A31 A32 A33 А34

A41 A42 A43 А44

for i:=2 to n do

for j:=2 to n-1 do

if i+j>n+1 then <инструкции>

Транспонирование матрицы.

Транспонированной матрицей называется матрица, у которой столбцы соответствуют строкам исходной квадратной матрицы. При этом элементы главной диагонали исходной и транспонированной матриц, одни и те же.

Операция транспонирования сводится к обмену элементов матрицы, расположенных симметрично главной диагонали.

Исходная матрица

Транспонированная матрица

1

5

9

13

1

2

3

4

Фрагмент программы транспонирования матрицы:

for i:=1 to n do {Просмотр всех строк матрицы}

for j:=i+1 to n do {Просмотр всех элементов в строке, расположенных выше главной диагонали}

begin

k:=a[i,j];

a[i,j]:= a[j,i];

a[j,i]:= k;

end;

1.4 Примеры задач

1. Найти сумму всех элементов некоторого двумерного массива и сравнить их с произведением элементов некоторой строки.

program zadacha\_1;

uses crt;

var

a: array[1..50,1..50] of integer; {массив}

i,j: integer; {переменные счетчики}

n,m: integer; {количество строк и столбцов массива}

s: integer; {сумма элементов массива}

p: integer; {произведение элементов некоторой строки}

q: integer; {некоторая строка}

begin

clrscr;

write('Введите количество строк: ');

readln(n);

write('Введите количество столбцов: ');

readln(m);

for i:=1 to n do

for j:=1 to m do

begin

write('a[',i,',',j,']=');

readln(a[i,j]);

end;

writeln('Матрица:');

for i:=1 to n do

begin

for j:=1 to m do

begin

write(a[i,j]:3);

end;

readln;

end;

for i:=1 to n do

for j:=1 to m do

begin

s:=s+a[i,j];

end;

write('Введите номер строки для работы: ');

readln(q);

p:=1;

for j:=1 to m do

begin

p:=p\*a[q,j];

end;

writeln('Сумма элементов матрицы: ',s);

writeln('Произведение элементов строки ',q,' равна ',p);

if s>p then

begin

writeln('Сумма больше произведения');

end

else

begin

writeln('Произведение больше произведения');

end;

readln;

end.

2.Поменять второй столбец матрицы с предпоследним.

program zadacha\_2;

uses crt;

var

a: array [1..50,1..50] of integer;

b: array [1..50,1..50] of integer;

m,n,i,j: integer;

begin

clrscr;

writeln('Количество строк');

readln(n);

writeln('Количество столбцов');

readln(m);

for i:= 1 to n do

for j:= 1 to m do

begin

write ('a[',i,',',j,']=');

readln (a[i,j]);

end;

writeln('Исходная матрица:');

for i:=1 to n do begin

for j:=1 to m do

write (a[i,j]);

writeln;

end;

for i:=1 to n do begin

for j:=1 to m do

b[i,j]:=a[i,j];

end;

for i:=1 to n do begin

a[i,2]:=b[i,m-1];

end;

for i:=1 to n do begin

a[i,m-1]:=b[i,2];

end;

writeln('Полученная матрица:');

for i:=1 to n do begin

for j:=1 to m do

write (a[i,j]);

writeln;

end;

readln;

end.

3.Дана матрица размерности m\*n. Расположить элементы последнего столбца по убыванию.

program zadacha\_3;

uses crt;

var

a:array [1..50] of integer;

b:array [1..50] of integer;

k,i,m,j,n,r,l:integer;

begin

clrscr;

write('Введите количество строк');

readln(n);

write('Введите количество столбцов');

readln(m);

for i:=1 to n do

for j:=1 to m do

begin

write('a[',i,']={b[',j,']=}');

readln(a[i]);

end;

for i:=1 to n-1 do

for k:=i+1 to n do

{for j:=1 to m do}

if a[k]>a[i] then

begin

r:=a[i];

a[i]:=a[k];

a[k]:=r;

end;

writeln('Отсортированый массив:');

for i:=1 to n do

writeln(a[i]:4);

readln;

end.

4. Дана квадратная матрица. Транспонировать её. Посчитать сумму всех нечётных элементов транспонированной матрицы.

program zadacha\_4;

uses crt;

var

a:array [1..5,1..5] of integer;

S,i,j,n,c:integer;

begin

clrscr;

write ('введите кол-во строк и столбцов ');

readln (n);

for i:=1 to n do

for j:=1 to n do

begin

write ('a[',i,',',j,']=');

readln (a[i,j]);

end;

for i:=1 to n do

for j:=i+1 to n do

begin

c:=a[i,j];

a[i,j]:=a[j,i];

a[j,i]:=c;

end;

S:=0;

for j:=1 to n do

for i:=1 to n do

begin

if a[j,i] mod 2 <>0 then

begin

S:=S+a[j,i];

end;

end;

writeln ('S=',S);

readln;

end.

5. Дан двумерный массив. Посчитать сумму его двух столбцов, вывести большую сумму.

program zadacha\_5

uses crt;

var

a:array[1..3,1..3] of integer;

i,j,m,n,s,s1,p,p1,max,p3:integer;

begin

clrscr;

write('введите количество строк');

readln(n);

write('введите количество столбцов');

readln(m);

for i:=1 to n do

for j:=1 to m do

begin

write('a[',i,',',j,']=');

readln(a[i,j]);

end;

begin

write('введите номер столбца');

readln(p);

for i:=1 to n do

for j:=1 to m do

if j=p then s:=s+a[i,p];

write('s=',s);

readln;

end;

s1:=0;

begin

write('введите номер другого столбца');

readln(p1);

for i:=1 to n do

for j:=1 to m do

if j=p1 then s1:=s1+a[i,p1];

write ('s1=', s1);

readln;

end;

begin

max:=s;

if s < s1 then

begin

max:=s1;

write('максимальная сумма',max);

readln;

write('вывести на экран номер столбца большей суммы', p3);

readln(p3);

end;

end;

end.

end.

6. Заполнить матрицу по образцу:

0

0

0

0

program zadacha\_6;

uses crt;

var

a:array [1..50,1..50] of integer;

n:integer;

i,j,k:integer;

begin

clrscr;

write ('Введите количество строк и столбцов в матрице');

readln (n);

k:=0;

for i:=1 to n do

for j:=1 to n do

if i=j then

begin

a[i,j]:=k;

k:=k+1;

end

else

begin

a[i,j]:=0;

end;

for i:=1 to n do

for j:=1 to n do

begin

writeln('a[',i,',',j,']=',a[i,j]);

readln;

end;

readln; end.

7. Дана квадратная матрица порядка N. В матрице вычислить среднее арифметическое положительных элементов, стоящих на главной диагонали

program zadacha\_7;

uses crt;

Var

a:array[1..50,1..50] of integer;{массив}

i,j:integer;

s,n,k:integer;

sr:real;

begin

clrscr;

write('введите кол-во строк');

readln(n);

write('введите кол-во столбцов');

readln(n);

write ('введите кол-во чисел');

readln(k);

for i:=1 to n do

for j:=1 to n do

begin

write('a[',i,',',j,']=');

readln(a[i,j]);

end;

s:=0;

begin

for i:=1 to n do

for j:=1 to n do

if a[i,j] > 0 then

s:=s+a[i,j];

sr:=s/n;

end;

write('sr=',sr);

readln;

end.

8. Найти сумму всех элементов квадратной матрицы, расположенных по главной диагонали и выше ее.

program zadacha\_8;

uses crt;

var

a: array [1..30,1..30] of integer;

i,j,s,n: integer;

begin

clrscr;

writeln ('введите количество строк и столбцов: ');

readln (n);

for i:=1 to n do

for j:=1 to n do

begin

write('a[',i,',',j,']=');

readln (a[i,j]);

end;

for i:=1 to n do

for j:=i+1 to n do

begin

if j>=i then

begin

s:=s+a[i,j];

end;

end;

writeln('s=',s);

readln;

end.

9. Дана вещественная матрица размерности n\*m. Удалить k столбец матрицы.

program zadacha\_9;

uses crt;

var

a: array [1..100,1..100] of real;

b: array [1..100,1..100] of real;

i,j: integer; {переменные счётчики}

n,m: integer; {количество строк и столбцов в массиве}

k: integer; {№ строки которую необходимо удалить}

begin

clrscr;

write ('Введите количество строк в массиве');

readln (n);

write ('введите количество столбцов в массиве');

readln (m);

write ('Введите № строки которую надо удалить');

readln (k);

randomize; {ввод массива случайных чисел}

for i:=1 to n do

for j:=1 to m do

begin

a[i,j]:=random(100);

end;

for i:=1 to n do

for j:=1 to m do

begin

writeln ('a[',i,',',j,']=',a[i,j]);

end;

writeln ('Новый массив');

for i:=1 to n do

for j:=1 to m do

begin

if j<>k then

{Проверка условия № столбца

неравен № столбца,

который необходимоудалить}

begin

b[i,j]:=a[i,j];

{если да, то новому массиву

присваиваем проверяемый элемент }

end;

end;

for i:=1 to n do {вывод нового массива}

for j:=1 to m do

writeln ('b[',i,',',j,']=',b[i,j]);

readln;

end.

10. Дана вещественная матрица размерности n\*m. Вывести номера столбцов, содержащих только отрицательные элементы.

program zadacha\_10;

uses crt;

var

a: array[1..50,1..50] of real;

i,j: integer;

n,m: integer;

begin

clrscr;

write('введите кол-во строк ');

readln(n);

write('введите кол-во столбцов ');

readln(m);

for i:=1 to n do

for j:=1 to m do

begin

write('a[',i,',',j,']= ');

readln(a[i,j]);

end;

for j:=1 to m do

begin

if a[i,j]<0 then

begin

writeln ('номер столбца, в котором все элементы отрицательные= ', J);

readln;

end

else

begin

writeln ('в столбце ',J,' нет отрицательных или не

все отрицательные элементы ');

end;

end;

readln;

end.

11. В двумерном массиве найти минимальное число и определить в какой строке и каком столбце он находится.

program zadacha\_11;

uses crt;

var

a:array [1..50,1..50] of integer; {описание масива}

i,j:integer; {переменные-счетчики}

min: real; { минимальное число}

n,m: integer; { кол-во строк, кол-во столбцов}

begin {начало программы}

clrscr;

write('введите кол-во строк '); {ввод кол-ва строк}

readln(n);

write('введите кол-во столбцов '); {ввод кол-ва столбцов}

readln(m);

for i:=1 to n do

for j:=1 to m do

begin

write('a[',i,',',j,']= '); { ввод элементов массива }

readln(a[i,j]);

end;

min:=a[1,1];

for i:=1 to n do

for j:=1 to m do

begin

if a[i,j] < min then {поиск минимального числа}

min:=a[i,j];

end;

for i:=1 to n do

for j:=1 to m do

begin

if a[i,j]=min then

begin

writeln('минимальное число = ',min:5);

{вывод минимального числа}

writeln('номер элемента i=',i,',','j=',j);

{вывод номера мин. чис-ла}

end;

end;

readln;

end.

12. Развернуть квадратную матрицу на 90 градусов по часовой стрелке.

program zadacha\_12;

uses crt;

var

a: array [1..30,1..30] of integer; {исходная матрица}

b: array [1..30,1..30] of integer; {промежуточная матрица}

c: array [1..30,1..30] of integer; {Матрица развёрнута на 90?}

i,j: integer; {переменные счётчики}

m,n: integer; {количество строк и столбцов}

begin

clrscr;

write ('введите количество строк и столбцов ');

readln (n);

for i:=1 to n do {ввод элементов массива}

for j:=1 to n do

begin

write ('a[',i,',',j,']=');

readln (a[i,j]);

end;

for i:=1 to n do

for j:=1 to n do

begin

b[i,j]:=a[n+1-i,j]; {промежуточной матрицы присваиваем

элементы первоначальной матрицы по закону: первому элементу

присваиваем строки последний, последнему первый, второму

элементу предпоследний, предпоследнему второй и тд.}

end;

writeln ('Матрица развёрнута на 90?.');

for i:=1 to n do

for j:=1 to n do

begin

c[i,j]:=b[j,i]; {третьей матрице присваиваем

элементы промежуточной по закону: первая строка

становится первым столбцом и тд. }

writeln ('c[',i,',',j,']=',c[i,j]); {печать массива развёрнутого на 90?}

readln;

end;

end.

# 2. Сортировка массивов

Задача сортировки (упорядочения) элементов массива в соответствии с их значениями относится к классу классических задач, которые решались еще на первых е- mail –ах.

В настоящее время разработано достаточно много различных методов сортировки. Одни из них относятся к методам простых сортировок. Другие к улучшенным. Однако до сегодняшнего момента задача разработки метода, сочетал бы в себе все лучшие качества остается открытой. Договоримся, что линейный массив, который необходимо упорядочить уже задан, т.е. описан и сгенерирован.

Различают следующие типы сортировок:

1) по возрастанию

2) по убыванию

3) по не убыванию

4) по не возрастанию

При рассмотрении каждого метода будем сортировать элементы по неубыванию.

# 2.1 Метод простых обменов (Пузырьковая сортировка)

Идея метода: Весь массив рассматривается несколько раз, причем при каждом рассмотрении сравниваются значения 2-х соседних элементов. Если они стоят в неправильном порядке, то производится их перестановка. Так происходит до тех пор, пока не будет выполнено ни одной перестановки. Метод называют пузырьковой сортировкой потому что меньшие значения элементов постепенно "всплывают", как пузырики воздуха в воде, перемещаясь в начало массива, а "тяжелые" элементы "оседают на дно".

7 0 -4 3 1 -2 5

-4 7 0 -2 3 1 5

-4 -2 7 0 1 3 5

-4 -2 0 7 1 3 5

-4 -2 0 1 7 3 5

-4 -2 0 1 3 7 5

-4 -2 0 1 3 5 5

Фрагмент:

For i:=2 to n do

For j:=n downto i do

if v[j] <v[j-1] then

begin

x:=v[j];

v[j]:=v[j-1];

v[j-1]:=x;

end;

# 2.2 Сортировка простым выбором

Идея метода: весь массив просматривается несколько раз и на каждом шаге ищется минимальный элемент и запоминается его порядковый номер. Затем найденный минимальный элемент меняется значением с первым, вторым, третьим и т.д. предпоследним элементом массива и исключается из рассмотрения

7 0 -4 3 1 -2 5

-4 0 7 3 1 -2 5

-4 -2 7 3 1 0 5

-4 -2 0 3 1 7 5

-4 -2 0 1 3 7 5

-4 -2 0 1 3 5 7

For i:= to n do

Begin

min:=v[i];

ind :=i;

for j:= i to n-1 do

if v[j]<min then

bedin

min:=v[j];

ind:=j;

end;

x:=v[i];

v[i]:=v[ind];

v[ind]:=x;

end;

# 2.3 Сортировка простым включением (Метод вставки исдвига)

Идея метода: делается предположение, что первые р элементов массива уже упорядочены и рассматривается р+1 элемент. Если окажется, что он меньше чем какой либо из первых р, то он занимает место большего, а участок массива ограниченный его новым местом и старым смещается в право.

7 0 -4 3 1 -2 5

0 7 -4 3 1 -2 5

-4 0 7 3 1 -2 5

-4 0 3 7 1 -2 5

-4 0 1 3 7 -2 5

-4 -2 0 1 3 7 5

-4 -2 0 1 3 5 7

For i:=2 to n do

For j:=1 to i-1 do

if v[i]<v[j] then

begin

x:=v[i];

for h:=1 downto j+1 do

v[h]:=i[h-1];

v[j]:=x;

end.

# 3. Параметры - массивы и параметры – строки

# Может сложиться впечатление, что объявление переменных в списке формальных параметров подпрограммы ничем не отличается от объявления их в разделе описания переменных. Действительно, в обоих случаях много общего, но есть одно существенное различие: типом любого параметра в списке формальных параметров может быть только стандартный или ранее объявленный тип. Поэтому нельзя, например, объявить следующую процедуру:

Procedure S (a: array [1..10] of Real);

так как в списке формальных параметров фактически объявляется тип-диапазон, указывающий границы индексов массива.

Если мы хотим передать какой-то элемент массива, то проблем, как правило, не возникает, но если в подпрограмму передается весь массив, то следует первоначально описать его тип. Например:

type

atype = array [1..10]of Real;

Procedure S (a: atype);

Поскольку строка является фактически своеобразным массивом, ее передача в подпрограмму осуществляется аналогичным образом:

type

intype = String [15] ;

outype = String [30] ;

Function St (s : intype): outype;

Требование описать любой тип-массив или тип-строку перед объявлением подпрограммы на первый взгляд кажется несущественным. Действительно, в рамках простейших вычислительных задач обычно заранее известна структура всех используемых в программе данных, поэтому статическое описание массивов не вызывает проблем. Однако разработка программных средств универсального назначения связана со значительными трудностями. По существу, речь идет о том, что в Турбо Паскале невозможно использовать в подпрограммах массивы с "плавающими" границами изменения индексов. Например, если разработана программа, обрабатывающая матрицу 10х10 элементов, то для обработки матрицы 9x11 элементов необходимо переопределить тип, т.е. перекомпилировать всю программу (речь идет не о динамическом размещении массивов в куче, а о статическом описании массивов и передаче их как параметров в подпрограммы). Этот недостаток, как и отсутствие в языке средств обработки исключительных ситуаций (прерываний), унаследован из стандартного Паскаля и представляет собой объект постоянной и вполне заслуженной его критики. Разработчики Турбо Паскаля не рискнули кардинально изменить свойства базового языка, но, тем не менее, включили в него некоторые средства, позволяющие в известной степени смягчить отмеченные недостатки.Эти недостатки практически полностью устранены в языке Object Pascal, используемом в визуальной среде программирования Delphi.

Прежде всего, в среде Турбо Паскаля можно установить режим компиляции, при котором отключается контроль за совпадением длины фактического и формального параметра-строки (см. прил.1). Это позволяет легко решить вопрос о передаче подпрограмме строки произвольной длины. При передаче строки меньшего размера формальный параметр будет иметь ту же длину, что и параметр обращения; передача строки большего размера приведет к ее усечению до максимального размера формального параметра. Следует сказать, что контроль включается только при передаче строки, объявленной как формальный параметр-переменная. Если, соответствующий параметр объявлен параметром-значением, эта опция игнорируется и длина не контролируется.

Значительно сложнее обстоит дело с передачей массивов произвольной длины. Наиболее универсальным приемом в этом случае будет, судя по всему, работа с указателями и использование индексной арифметики. Несколько проще можно решить эту проблему при помощи нетипизированных параметров (см. п.8.5). В версии Турбо Паскаля 7.0 язык поддерживает так называемые открытые массивы, легко решающие проблему передачи подпрограмме одномерных массивов переменной длины.

Открытый массив представляет собой формальный параметр подпрограммы, описывающий базовый тип элементов массива, но не определяющий его размерности и границы:

Procedure MyProc(OpenArray: array of Integer);

Внутри подпрограммы такой параметр трактуется как одномерный массив с нулевой нижней границей. Верхняя граница открытого массива возвращается функцией HIGH, упоминавшейся в п.4.1.1. Используя 0 как минимальный индекс и значение, возвращаемое функцией HIGH, как максимальный индекс, подпрограмма может обрабатывать одномерные массивы произвольной длины:

{Иллюстрация использования открытых массивов: программа выводит на экран содержимое двух одномерных массивов разной длины с помощью одной процедуры ArrayPrint}

Procedure ArrayPrint(aArray: array of Integer);

var

k: Integer;

begin

for k := 0 to High(aArray) do

Write(aArray[k]:8);

WriteLn

end;

const

A:array [-1..2] of Integer = (0,1,2,3);

B: array [5..7] of Integer = (4,5,6);

begin

ArrayPrint(A);

ArrayPrint(B)

end.

Как видно из этого примера, фактические границы массивов А и В, передаваемых в качестве параметров вызова процедуре ArrayPrint, не имеют значения. Однако размерность открытых массивов (количество индексов) всегда равна 1 - за этим следит компилятор. Если бы, например, мы добавили в программу двумерный массив С

var

С: array [1..3,1..5] of Integer;

то обращение

ArrayPrint(С)

вызывало бы сообщение об ошибке

Error26: Type mismatch.

(Ошибка 26: Несоответствие типов.)

# Список литературы

1.Пильщиков В.Н. Сборник упражнений по языку Паскаль: Учеб. пособие для вузов.-М.:Наука, 1989.-160с.

2. Семашко Г.Л., Салтыков А.И. Программирование на языке Паскаль. М.: Наука 1988.-128с.

3. Введение в язык Паскалью./Абрамов В.Г. Трифонов Н.П. Трифонова Г.Н. Учеб. пособие .- М.: Наука 1988.-320с.

4.Могилев А.В. Практикум по инф-ке. Учебное пособие для студентов уч. Заведений\ Могилев А.В., Пак Н.И., Хеннер Е.К., М.:Академия, 2001.-608с.