Порядковые и ограниченные типы

**Порядковые типы данных**

Это типы, в диапазоне значений которых величины идут по порядку, а интервалом является целое число. Например, в типе Byte значения следуют так:

0, 1, 2, 3, … , 255

То есть все значения порядкового типа можно перечислить подряд с целочисленным интервалом. Иными словами, каждое значение порядкового типа можно пронумеровать и тогда, за исключением первого и последнего значений диапазона, у всех значений будут существовать такие понятия как «следующее» и «предыдущее».

Вещественные числа не являются порядковыми, так как значения вещественного диапазона невозможно пронумеровать. Почему? Предположим, у нас есть диапазон вещественных чисел:

0,00…3,00

Тогда в этом диапазоне может быть бесконечно большое количество значений, потому что количество знаков после запятой может быть равно 0, а может быть равно 10, 50, или 100. То есть мы не знаем заранее, сколько значений будет в этом диапазоне, следовательно, мы не сможем пронумеровать их.

Порядковые типы данных очень широко используются при разработке программ. Это самые простые типы данных, с ними легко работать.

К порядковым типам относятся целые числа, логические значения, символьные типы (но не строки), перечисляемые типы и интервальные типы. Порядковые типы можно использовать в циклах, для индексации массивов, с функциями Inc, Dec, Ord и т.п.

**Ограниченные типы**

Ограниченный тип данных представляет собой интервал значений порядкового типа, называемого базовым типом. Описание типа задаёт наименьшее и наибольшее значения, входящие в этот интервал.

Например,

Var a: 1..25; ch: 'a' ..'z';

Здесь переменные а и ch могут принимать значения только из указанного интервала; базовым типом для переменой а является целый тип, а для переменной ch - символьный.

Переменная ограниченного типа сохраняет все свойства переменных базового типа.

Для чего вводится ограниченный тип данных? Использование ограниченного типа делает программу наиболее понятной и наглядной. Например, если в программе переменная b может принимать только значения 3, 4, 5, 6, 7, 8, то лучше описать её следующим образом: Var b: 3..8;, чем Var b: Integer; так как в случае выхода значения b за диапазон 3..8 в первом случае будет выдано диагностическое сообщение, которое поможет найти ошибку. Во втором случае будет получен неправильный результат, что затруднит поиск ошибки. Таким образом, второй вариант описания переменной следует использовать в тех случаях, когда диапазон значений заранее неизвестен либо занимает весь допустимый интервал значений для рассматриваемого типа.

Пример 1:

Напишите программу, которая переменной t присваивает значения true, если первая дата предшествует (в рамках года) второй дате, и значение false в противном случае.

Решение

Так как в условии задачи оговаривается, что обе даты должны находится в рамках года, то дата должна задаваться днём и месяцем. Количество любого месяца года не может быть более 31, количество месяцев в году равно 12. Значение переменной t равно true, если номер первого месяца меньше второго, либо значение первого дня меньше второго при условии, что номера месяцев совпали.

Program Example\_24;  
Var d1, d2 : 1..31;  
m1,m2: 1..12;  
t:Boolean;  
Begin  
Write( 'Введите первую дату (день, месяц)');  
Readln(d1, m1);  
Write( 'Введите вторую дату (день, месяц)');  
Readln(d2, m2);  
t:=(m1<m2) Or ((m1=m2) And(d1<d2));  
Writeln(t);  
End.

Оператор варианта

В программах очень часто используются операторы Case ... of. Эта конструкция предназначена для замены конструкции из вложенных операторов If...then и применяется для обработки ситуаций с несколькими исходами, т.е. когда условие может принимать более двух значений. Оператор выбора имеет следующий вид:

|  |
| --- |
| Оператор варианта |
| Case <порядковая переменная> Of |
| <константа1> : <оператор1>; |
| <константа2> : <оператор2>; |
| ... |
| <константа n> : <оператор n>; |
| [ Else <оператор>; ] |
| End; |

Выполнение оператора выбора начинается с вычисления выражения, которое должно принимать значение, принадлежащее порядковому типу. В случае, если результат вычисления равен одной из перечисленных констант, то выполняется соответствующий оператор. Затем управление передаётся за пределы оператора выбора. Если значение выражения не совпадает ни с одной константой, то выполняется оператор, стоящий после else, если он есть, либо управление передаётся на оператор end и выполнение программы продолжается.

Примечание.

*Тип констант должен совпадать с типом выражения.*

*Здесь ветвь else заключена в квадратные скобки, что говорит о том, что эта часть оператора выбора не обязательна.*

*В этой конструкции перед else стоит точка с запятой.*

*В качестве оператора может быть и составной оператор.*

Пример 2:

Составьте программу, которая определяет, является ли введенный символ гласной или согласной буквой английского алфавита.

Решение

По условию задачи все символы делятся на следующие группы:

\* гласные буквы английского алфавита;

\* согласные буквы английского алфавита;

\* символы, не являющиеся буквами английского алфавита.

Рассмотрим следующую программу:

Program Exampl\_25  
Var ch:Char;  
Begin  
Write('Введите символ');  
Readln(ch);

Case Upcase(ch); Of  
'A','E','I','O','U': Writeln('Это гласная буква английского алфавита');  
'A'..'Z': Writeln('Это буква английского алфавита');  
Else  
Writeln('Этот символ не является буквой английского алфавита');  
End.

Обратите внимание на то, что константы здесь в первом случае перечисляются через запятую, а во втором используется интервал значений.

Литература

<https://info-master.su/programming/kurs/ordinal-data-types.php>

<http://www.borlpasc.narod.ru/docym/prac/ograni.htm>