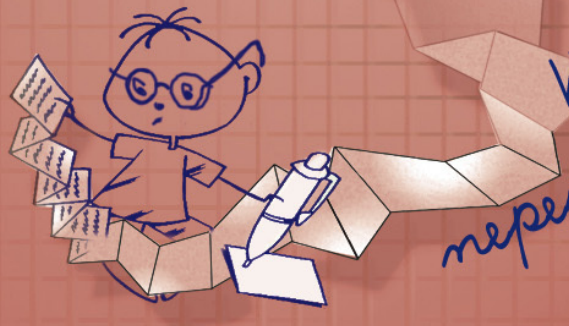
ИНФОРМАТИКА И ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ

шпаргалки



Используй сам,
передай 5 однокурсникам,
и будет вам счастье
во время сессии

Шпаргалки

Коллектив авторов

Информатика и информационные технологии

«Научная книга»

Коллектив авторов

Информатика и информационные технологии / Коллектив авторов — «Научная книга», — (Шпаргалки)

ISBN 5-699-24023-3

Информативные ответы на все вопросы курса «Информатика и информационные технологии» в соответствии с Государственным образовательным стандартом.

ISBN 5-699-24023-3

Содержание

А. В. Цветкова
Информатика и информационные технологии

1. Информатика. Информация

Представление и обработка / информации. Системы счисления

Информатика занимается формализованным представлением объектов и структур их взаимосвязей в различных областях науки, техники, производства. Для моделирования объектов и явлений используются различные формальные средства, например логические формулы, структуры данных, языки программирования и др.

В информатике такое фундаментальное понятие, как информация имеет различные значения:

1) формальное представление внешних форм информации;

2) абстрактное значение информации, ее внутреннее содержание, семантика;

3) отношение информации к реальному миру.

Но, как правило, под информацией понимают ее абстрактное значение – семантику. Если мы хотим обмениваться информацией, нам необходимы согласованные представления, чтобы не нарушалась правильность интерпретации. Для этого интерпретацию представления информации отождествляют с некоторыми математическими структурами. В этом случае обработка информации может быть выполнена строгими математическими методами.

Одно из математических описаний информации – это представление ее в виде функции

$y = f(x,t)$

где t – время,

x – точка некоторого поля, в которой измеряется значение y. В зависимости от параметров функции x и t информацию можно классифицировать.

Если параметры – скалярные величины, принимающие непрерывный ряд значений, то полученная таким образом информация называется непрерывной (или аналоговой). Если же параметрам придать некоторый шаг изменений, то информация называется дискретной. Дискретная информация считается универсальной.

Дискретную информацию обычно отождествляют с цифровой информацией, которая является частным случаем символьной информации алфавитного представления. Алфавит – конечный набор символов любой природы. Очень часто в информатике возникает ситуация, когда символы одного алфавита надо представить символами другого, т. е. провести операцию кодирования.

Как показала практика, наиболее простым алфавитом, позволяющим кодировать другие алфавиты, является двоичный, состоящий из двух символов, которые обозначаются, как правило, через 0 и 1. С помощью n символов двоичного алфавита можно закодировать 2n символов, а этого достаточно, чтобы закодировать любой алфавит.

Величина, которая может быть представлена символом двоичного алфавита, называется минимальной единицей информации или битом. Последовательность из 8 бит – байт. Алфавит, содержащий 256 различных 8-битных последовательностей, называется байтовым.

Под системой счисления подразумевается набор правил наименования и записи чисел. Различают позиционные и непозиционные системы счисления.

Система счисления называется позиционной, если значение цифры числа зависит от местоположения цифры в числе. В противном случае она называется непозиционной. Значение числа определяется по положению этих цифр в числе.

2. Представление чисел в ЭВМ. Формализованное понятие алгоритма

32-разрядные процессоры могут работать с оперативной памятью емкостью до 232-1, а адреса могут записываться в диапазоне 00000000 – FFFFFFFF. Однако в реальном режиме процессор работает с памятью до 220-1, а адреса попадают в диапазон 00000 – FFFFF. Байты памяти могут объединяться в поля как фиксированной, так и переменной длины. Словом называется поле фиксированной длины, состоящее из 2 байтов, двойным словом – поле из 4 байтов. Адреса полей бывают четные и нечетные, при этом для четных адресов операции выполняются быстрее.

Числа с фиксированной точкой в ЭВМ представляются как целые двоичные числа, и занимаемый ими объем может составлять 1, 2 или 4 байта.

Целые двоичные числа представляются в дополнительном коде. Дополнительный код положительного числа равен самому числу, а дополнительный код отрицательного числа может быть получен по такой формуле:

x = 10n – \x\, где n – разрядность числа.

В двоичной системе счисления дополнительный код получается путем инверсии разрядов, т. е., заменой единиц нулями и наоборот, и прибавлением единицы к младшему разряду.

Количество битов мантиссы определяет точность представления чисел, количество битов машинного порядка определяет диапазон представления чисел с плавающей точкой.

Формализованное понятие алгоритма

Алгоритм может существовать только тогда, когда в то же самое время существует некоторый математический объект. Формализованное понятие алгоритма связано с понятием рекурсивных функций, нормальных алгоритмов Маркова, машин Тьюринга.

В математике функция называется однозначной, если для любого набора аргументов существует закон, по которому определяется единственное значение функции. В качестве такого закона может выступать алгоритм; в этом случае функция называется вычислимой.

Рекурсивные функции – это подкласс вычислимых функций, а алгоритмы, определяющие вычисления, называются сопутствующими алгоритмами рекурсивных функций. Сначала фиксируются базовые рекурсивные функции, для которых сопутствующий алгоритм тривиален, однозначен; затем вводятся три правила – операторы подстановки, рекурсии и минимизации, при помощи которых на основе базовых функций получаются более сложные рекурсивные функции.

Базовыми функциями и их сопутствующими алгоритмами могут выступать:

1) функция n независимых переменных, тождественно равная нулю. Тогда, если знаком функции является φn, то независимо от количества аргументов значение функции следует положить равным нулю;

2) тождественная функция n независимых переменных вида Ψ ni. Тогда, если знаком функции является Ψ ni, то значением функции следует взять значение i-го аргумента, считая слева направо;

3) λ—функция одного независимого аргумента. Тогда, если знаком функции является λ, то значением функции следует взять значение, следующее за значением аргумента.

3. Введение в язык Pascal

Основные символы языка – буквы, цифры и специальные символы – составляют его алфавит. Язык Pascal включает следующий набор основных символов:

1) 26 латинских строчных и 26 латинских прописных букв:

2) _ (знак подчеркивания);

3) 10 цифр: 0 1 2 3 4 5 6 7 8 9;

4) знаки операций:

+ – O / = <> < > <= >= := @;

5) ограничители:. , ( ) [ ] (. .) { } (* *) .. : ;

6) спецификаторы: ^ # $;

7) служебные (зарезервированные) слова: ABSOLUTE, ASSEMBLER, AND, ARRAY, ASM, BEGIN, CASE, CONST, CONSTRUCTOR, DESTRUCTOR, DIV, DO, DOWNTO, ELSE, END, EXPORT, EXTERNAL, FAR, FILE, FOR, FORWARD, FUNCTION, GOTO, IF, IMPLEMENTATION, IN, INDEX, INHERITED, INLINE, INTERFACE, INTERRUPT, LABEL, LIBRARY, MOD, NAME, NIL, NEAR, NOT, OBJECT, OF, OR, PACKED, PRIVATE, PROCEDURE,

PROGRAM, PUBLIC, RECORD, REPEAT, RESIDENT, SET,

SHL, SHR, STRING, THEN, TO, TYPE, UNIT, UNTIL, USES,

VAR, VIRTUAL, WHILE, WITH, XOR.

Кроме перечисленных, в набор основных символов входит пробел.

В языке Pascal существует правило: тип явно задается в описании переменной или функции, которое предшествует их использованию. Концепция типа языка Pascal имеет следующие основные свойства:

1) любой тип данных определяет множество значений, к которому принадлежит константа, которые может принимать переменная или выражение либо вырабатывать операция или функция;

2) тип значения, задаваемого константой, переменной или выражением, можно определить по их виду или, описанию;

3) каждая операция или функция требуют аргументов фиксированного типа и выдают результат фиксированного типа.

В языке Pascal существуют скалярные и структурированные типы данных. К скалярным типам относятся стандартные типы и типы, определяемые пользователем. Стандартные типы включают целые, действительные, символьный, логические и адресный типы.

Целые типы определяют константы, переменные и функции, значения которых реализуются множеством целых чисел, допустимых в данной ЭВМ.

В языке Pascal принят следующий приоритет операций:

Тип	Диапазон значений	Требуемая память (байт)
Byte	0...255	1
Word	0...65535	2
Shortint	–128...127	1
Integer	–32768...32767	2
Longint	–2147483648...2147483647	4

Тип	Диапазон значений	Кол-во цифр мантиссы	Требуемая память (байт)
Real	2.9e – 39...1.7e + 38	11—12	6
Single	1.5e – 45...3.4e + 38	7—8	4
Double	5.0e – 324...1.7e + 308	15—16	8
Extended	3.4e – 4932...1.1e + 4932	19—20	10
Comp	–9.2e + 18...9.2e + 18	19—20	2

1) вычисления в круглых скобках;
2) вычисления значений функций;
3) унарные операции;
4) операции * / div mod and;
5) операции + – or xor;
6) операции отношения = <> < > <= >=.

4. Стандартные процедуры и функции

Арифметические функции

1. Function Abs(X); возвращает абсолютное значение параметра.
2. Function ArcTan(X: Extended): Extended; возвращает арктангенс аргумента.
3. Function Exp(X: Real): Real; возвращает экспоненту.
4. Function Frac(X: Real): Real; возвращает дробную часть аргумента.
5. Function Int(X: Real): Real; возвращает целочисленную часть аргумента.
6. Function Ln(X: Real): Real; возвращает натуральный логарифм (Ln e = 1) выражения X вещественного типа.
7. Function Pi: Extended; возвращает значение Pi, которое определено как 3.1415926535.
8. Function Sin(X: Extended): Extended; возвращает синус аргумента.
9. Function Sqr(X: Extended): Extended; возвращает квадрат аргумента.
10. Function Sqrt(X: Extended): Extended; возвращает квадратный корень аргумента.

Процедуры и функции преобразования величин

1. Procedure Str(X [: Width [: Decimals]]; var S); преобразовывает число X в строковое представление.
2. Function Chr(X: Byte): Char; возвращает символ с порядковым номером X в ASCII-таблице.
3. Function High(X); возвращает наибольшее значение в диапазоне параметра.
4. Function Low(X); возвращает наименьшее значение в диапазоне параметра.
5. Function Ord(X): Longlnt; возвращает порядковое значение выражения перечислимого типа.
6. Function Round(X: Extended): LongInt; округляет значение вещественного типа до целого.
7. Function Trunc(X: Extended): LongInt; усекает значение вещественного типа до целого.
8. Procedure Val(S; var V; var Code: Integer); преобразовывает число из строкового значения S в числовое представление V.

Процедуры и функции работы с порядковыми величинами

1. Procedure Dec(var X [; N: LongInt]); вычитает единицу или N из переменной X.
2. Procedure Inc(var X [; N: LongInt]); прибавляет единицу или N к переменной X.
3. Function Odd(X: LongInt): Boolean; возвращает True, если X – нечетное число, и False – в противном случае.
4. Function Pred(X); возвращает предыдущее значение параметра.
5. Function Succ(X); возвращает следующее значение параметра.

5. Операторы языка Pascal

Условный оператор

Формат полного условного оператора определяется следующим образом:

If B then S1 else S2

где B – условие разветвления (принятия решения), логическое выражение или отношение; S1, S2 – один выполняемый оператор, простой или составной.

При выполнении условного оператора сначала вычисляется выражение B, затем анализируется его результат: если B – истинно, то выполняется оператор S1 – ветвь then, а оператор S2 пропускается; если B – ложно, то выполняется оператор S2 – ветвь else, а оператор S1 – пропускается.

Оператор выбора

Структура оператора имеет следующий вид:

case S of

c1: insruction1;

c2: insruction2;

…

cn: insructionN;

else instruction

end;

где S – выражение порядкового типа, значение которого вычисляется;

c1, c2,…, on – константы порядкового типа, с которыми сравниваются выражения S; instructionl,…, instructionN – операторы, из которых выполняется тот, с константой которого совпадает значение выражения S;

instruction – оператор, который выполняется, если значение выражения S не совпадает ни с одной из констант c1, o2, on.

Оператор цикла с параметром

Когда начинает выполняться оператор for, начальное и конечное значения определяются один раз, и эти значения сохраняются на протяжении всего выполнения оператора for. Оператор, который содержится в теле оператора for, выполняется один раз для каждого значения в диапазоне между начальным и конечным значением. Счетчик цикла всегда инициализируется начальным значением.

Оператор цикла с предусловием

While B do S;

где B – логическое условие, истинность которого проверяется (оно является условием завершения цикла)$;

S – тело цикла – один оператор. Выражение, с помощью которого осуществляется управление повторением оператора, должно иметь логический тип. Вычисление его производится до того, как внутренний оператор будет выполнен. Внутренний оператор выполняется повторно до тех пор, пока выражение принимает значение Тгие. Если выражение с самого начала принимает значение False, то оператор, содержащийся внутри оператора цикла с предусловием, не выполняется.

Оператор цикла с постусловием

repeat S until B;

где B – логическое условие, истинность которого проверяется (оно является условием завершения цикла);

S – один или более операторов тела цикла. Результат выражения должен быть логического типа. Операторы, заключенные между ключевыми словами repeat и until, выполняются

последовательно до тех пор, пока результат выражения не примет значение True. Последовательность операторов выполнится, по крайней мере, один раз, поскольку вычисление выражения производится после каждого выполнения последовательности операторов.

6. Понятие вспомогательного алгоритма

Алгоритм решения задачи проектируется путем декомпозиции всей задачи в отдельные подзадачи. Обычно подзадачи реализуются в виде подпрограмм.

Подпрограмма – это некоторый вспомогательный алгоритм, многократно использующийся в основном алгоритме с различными значениями некоторых входящих величин, называемых параметрами.

Подпрограмма в языках программирования – это последовательность операторов, которые определены и записаны только в одном месте программы, однако их можно вызвать для выполнения из одной или нескольких точек программы. Каждая подпрограмма определяется уникальным именем.

В языке Pascal существуют два типа подпрограмм – процедуры и функции. Процедура и функция – это именованная последовательность описаний и операторов. При использовании процедур или функций программа должна содержать текст процедуры или функции и обращение к процедуре или функции. Параметры, указанные в описании, называются формальными, указанные в обращении подпрограммы – фактическими. Все формальные параметры можно разбить на следующие категории:

1) параметры-переменные;
2) параметры-константы;
3) параметры-значения;
4) параметры-процедуры и параметры-функции, т. е. параметры процедурного типа;
5) нетипизированные параметры-переменные.

Тексты процедур и функций помещаются в разделописаний процедур и функций.

Передача имен процедур и функций в качестве параметров

Во многих задачах, особенно в задачах вычислительной математики, необходимо передавать имена процедур и функций в качестве параметров. Для этого в TURBO PASCAL введен новый тип данных – процедурный, или функциональный, в зависимости от того, что описывается. (Описание процедурных и функциональных типов приводится в разделе описания типов.)

Функциональный и процедурный тип определяется как заголовок процедуры и функции со списком формальных параметров, но без имени. Можно определить функциональный, или процедурный тип без параметров, например:

```
type
Proc = Procedure;
```

После объявления процедурного, или функционального, типа его можно использовать для описания формальных параметров – имен процедур и функций. Кроме того, необходимо написать те реальные процедуры или функции, имена которых будут передаваться как фактические параметры.

7. Процедуры и функции в Pascal

Процедуры в Pascal

Описание процедуры состоит из заголовка и блока, который, за исключением раздела подключения модулей, не отличаются от блока программы. Заголовок состоит из ключевого слова Procedure, имени процедуры и необязательного списка формальных параметров в круглых скобках:

Procedure <имя> [(<список формальных параметров>)];

Для каждого формального параметра должен быть определен его тип. Группы параметров в описании процедуры разделяются точкой с запятой.

По структуре процедура почти полностью аналогична программе. Однако в блоке процедуре отсутствует раздел подключения модулей. Блок состоит из двух частей: описательной и исполнительной. В описательной части содержится описание элементов процедуры. А в исполнительной части указываются действия с доступными процедуре элементами программы (например, глобальные переменные и константы), позволяющие получить требуемый результат. Раздел инструкций процедуры отличается от раздела инструкций программы только тем, что после ключевого слова End, завершающего этот раздел, ставится точка с запятой, а не точка.

Для обращения к процедуре используется инструкция вызова процедуры. Она состоит из имени процедуры и списка аргументов, заключенного в круглые скобки. Операторы, которые должны выполняться при запуске процедуры, содержатся в операторной части модуля процедуры.

Иногда требуется, чтобы процедура вызывала сама себя. Такой способ вызова называется рекурсией. Рекурсия полезна в случаях, когда основную задачу можно разбить на подзадачи, каждая из которых реализуется по алгоритму, совпадающему с основным.

Функции в Pascal

Описание функции определяет часть программы, в которой вычисляется и возвращается значение. Описание функции состоит из заголовка и блока. Заголовок содержит ключевое слово Function, имя функции, необязательный список формальных параметров, заключенный в круглые скобки, и тип возвращаемого функцией значения. Общий вид заголовка функции следующий:

Function <имя> [(<список формальных параметров>)]: <тип возвращаемого результата>;

В реализации Turbo Pascal 7.0 фирмы Borland возвращаемое функцией значение не может иметь составной тип. А язык Object Pascal, используемый в интегрированных средах разработки Borland Delphi, допускает любой тип возвращаемого результата, кроме файлового типа.

Блок функции представляет собой локальный блок, по структуре аналогичный блоку процедуры. В теле функции должна быть хотя бы одна инструкция присваивания, в левой части которой стоит имя функции. Именно она и определяет значение, возвращаемое функцией. Если таких инструкций несколько, то результатом функции будет значение последней выполненной инструкции присваивания.

Функция активизируется при вызове функции. При вызове функции указывается идентификатор функции и какие-либо параметры, необходимые для вычисления функции. Вызов функции может включаться в выражения в качестве операнда. Когда выражение вычисляется, функция выполняется и значением операнда становится значение, возвращаемое функцией.

В операторной части блока функции задаются операторы, которые должны выполняться при активизации функции. В модуле должен содержаться, по крайней мере, один оператор присваивания, в котором идентификатору функции присваивается значение. Результатом функ-

ции является последнее присвоенное значение. Если такой оператор присваивания отсутствует или он не был выполнен, то значение, возвращаемое функцией, не определено.

Если идентификатор функции используется при вызове функции внутри модуля – функции, то функция выполняется рекурсивно.

8. Опережающие описания и подключение подпрограмм. Директива

В программе может содержаться несколько подпрограмм, т. е. структура программы может быть усложнена. Однако эти подпрограммы могут располагаться на одном уровне вложенности, поэтому сначала должно идти описание подпрограммы, а затем обращение к ней, если только не используется специальное опережающее описание.

Описание процедуры, содержащее вместо блока операторов директиву forward, называется опережающим описанием. В каком-либо месте после этого описания с помощью определяющего описания процедура должна определяться. Определяющее описание – это описание, в котором используется тот же идентификатор процедуры, но опущен список формальных параметров, и в которое включен блок операторов. Описание forward и определяющее описание должны присутствовать в одной и той же части описания процедуры и функции. Между ними могут описываться другие процедуры и функции, которые могут обращаться к процедуре с опережающим описанием. Таким образом, возможна взаимная рекурсия.

Опережающее описание и определяющее описание представляют собой полное описание процедуры. Процедура считается описанной с помощью опережающего описания.

Если в программе будет содержаться довольно много подпрограмм, то программа перестанет быть наглядной, в ней будет тяжело ориентироваться. Во избежание этого некоторые подпрограммы хранят в виде исходных файлов на диске, а при необходимости они подключаются к основной программе на этапе компиляции при помощи директивы компиляции.

Директива – это специальный комментарий, который может быть размещен в любом месте программы, там, где может находиться и обычный комментарий. Однако они различаются тем, что у директивы имеется специальная форма записи: сразу после закрывающей скобки без пробела записывается знак $, а затем, опять же без пробела, указывается директива.

Пример:

1) {$E+} – эмулировать математический сопроцессор;
2) {$F+} – формировать дальний тип вызова процедур и функций;
3) {$N+} – использовать математический сопроцессор;
4) {$R+} – проверять выход за границы диапазонов.

Некоторые ключи компиляции могут содержать параметр, например:

{$I имя файла} – включить в текст компилируемой программы названный файл

9. Параметры подпрограмм

В описании процедуры или функции задается список формальных параметров. Каждый параметр, описанный в списке формальных параметров, является локальным по отношению к описываемой процедуре или функции, и в модуле, связанным с данной процедурой или функцией, на него можно ссылаться по его идентификатору.

Существует три типа параметров: значение, переменная и нетипизированная переменная. Они характеризуются следующим:

1. Группа параметров без предшествующего ключевого слова является списком параметров-значений.

2. Группа параметров, перед которыми следует ключевое слово const и за которыми следует тип, является списком параметров-констант.

3. Группа параметров, перед которыми стоит ключевое слово var и за которыми следует тип, является списком параметров-переменных.

Параметры-значения

Формальный параметр-значение обрабатывается, как локальная по отношению к процедуре или функции переменная, за исключением того, что он получает свое начальное значение из соответствующего фактического параметра при активизации процедуры или функции. Изменения, которые претерпевает формальный параметр-значение, не влияют на значение фактического параметра. Соответствующее фактическое значение параметра-значения должно быть выражением, и его значение не должно иметь файловый тип или какой-либо структурный тип, содержащий в себе файловый тип.

Фактический параметр должен иметь тип, совместимый по присваиванию с типом формального параметра-значения. Если параметр имеет строковый тип, то формальный параметр будет иметь атрибут размера, равный 255.

Параметры-константы

В теле подпрограммы значение параметра-константы изменить нельзя. Параметрами-константами можно оформить те параметры, изменения которых в подпрограмме нежелательно и должно быть запрещено.

Параметры-переменные

Параметр-переменная используется в случаях, когда значение должно быть передано из подпрограммы в вызывающий блок. В этом случае при вызове подпрограммы формальный параметр замещается аргументом-переменной, и любые изменения формального параметра отражаются на аргументе.

Процедурные переменные

После определения процедурного типа появляется возможность описывать переменные этого типа. Такие переменные называют процедурными переменными. Как и целая переменная, которой можно присвоить значение целого типа, процедурной переменной можно присвоить значение процедурного типа. Таким значением может быть, конечно, другая процедурная переменная, но оно может также представлять собой идентификатор процедуры или функции. В таком контексте описания процедуры или функции можно рассматривать как описание особого рода константы, значением которой является процедура или функция.

Как и при любом другом присваивании, значения переменной в левой и в правой части должны быть совместимы по присваиванию. Процедурные типы, чтобы они были совместимы по присваиванию, должны иметь одно и то же число параметров, а параметры на соответствующих позициях должны быть одинакового типа. Имена параметров в описании процедурного типа никакого действия не вызывают.

Кроме того, для обеспечения совместимости по присваиванию, процедура или функция, если ее нужно присвоить процедурной переменной, не должна быть стандартной или вложенной.

10. Типы параметров подпрограмм

Параметры-значения

Формальный параметр-значение обрабатывается как локальная переменная, он получает свое начальное значение из соответствующего фактического параметра при активизации процедуры или функции. Изменения, которые претерпевает формальный параметр-значение, не влияют на значение фактического параметра. Соответствующее фактическое значение параметра-значения должно быть выражением, и его значение не должно иметь файловый тип.

Параметры-константы

Формальные параметры-константы получают свое значение при активизации процедуры или функции. Присваивания формальному параметру-константе не допускаются. Формальный параметр-константа не может передаваться в качестве фактического параметра другой процедуре или функции.

Параметры-переменные

Параметр-переменная используется, когда значение должно передаваться из процедуры или функции вызывающей программе. При активизации формальный параметр-переменная замещается фактической переменной, изменения формального параметра-переменной отражаются на фактическом параметре.

Нетипизированные параметры

Когда формальный параметр является нетипизированным параметром-переменной, то соответствующий фактический параметр может представлять собой ссылку на переменную или константу. Нетипизированный параметр, описанный с ключевым словом var, может модифицироваться, а нетипизированный параметр, описанный с ключевым словом const, доступен только по чтению.

Процедурные переменные

После определения процедурного типа появляется возможность описывать переменные этого типа. Такие переменные называют процедурными переменными. Процедурной переменной можно присвоить значение процедурного типа.

Процедура или функция при присваивании должна быть:

1) не стандартной;
2) не вложенной;
3) не процедурой типа inline;
4) не процедурой прерывания (interrupt).

Параметры процедурного типа

Поскольку процедурные типы допускается использовать в любом контексте, то можно описывать процедуры или функции, которые воспринимают процедуры и функции в качестве параметров. Параметры процедурного типа особенно полезны в том случае, когда над множеством процедур или функций нужно выполнить какие-то общие действия.

Если процедура или функция должны передаваться в качестве параметра, они должны удовлетворять тем же правилам совместимости типа, что и при присваивании. То есть, такие процедуры или функции должны компилироваться с директивой far, они не могут быть встроенными функциями, не могут быть вложенными и не могут описываться с атрибутами inline или interrupt.

11. Строковый тип в Pascal. Процедуры и функции для переменных строкового типа

Последовательность символов определенной длины называется строкой. Переменные строкового типа определяются путем указания имени переменной, зарезервированного слова string, и возможно, но не обязательно указания максимального размера, т. е. длины строки, в квадратных скобках. Если не задавать максимальный размер строки, то по умолчанию он будет равен 255, т. е. строка будет состоять из 255 символов.

К каждому элементу строки можно обратиться по его номеру. Однако ввод и вывод строк осуществляются целиком, а не поэлементно, как это происходит в массивах. Число введенных символов не должно превышать указанного в максимальном размере строки, так если такое превышение будет иметь место, то «лишние» символы будут проигнорированы.

Процедуры и функции для переменных строкового типа

1. Function Copy(S: String; Index, Count: Integer): String;

Возвращает подстроку строки. S – выражение типа String. Index и Count – выражения целого типа. Функция возвращает строку, содержащую Count символов, начинающихся с позиции Index. Если Index больше, чем длина S, функция возвращает пустую строку.

2. Procedure Delete(var S: String; Index, Count: Integer);

Удаляет подстроку символов длиной Count из строки S, начиная с позиции Index. S – переменная типа String. Index и Count – выражения целого типа. Если Index больше, чем длина S, символы не удаляются.

3. Procedure Insert(Source: String; var S: String; Index: Integer); Объединяет подстроку в строку, начиная с определенной позиции. Source – выражение типа String. S – переменная типа String любой длины. Index – выражение целочисленного типа. Insert вставляет Source в S, начиная с позиции S.

4. Function Length(S: String): Integer;

Возвращает число символов, фактически используемое в строке S. Обратите внимание: при использовании строк с нуль-окончанием, число символов не обязательно равно числу байтов.

5. Function Pos(Substr: String; S: String): Integer; Ищет подстроку в строке. Pos ищет Substr внутри S

и возвращает целочисленное значение, которое является индексом первого символа Substr внутри S. Если Substr не найден, Pos возвращает нуль.

12. Записи

Запись представляет собой совокупность ограниченного числа логически связанных компонент, принадлежащих к разным типам. Компоненты записи называются полями, каждое из которых определяется именем. Поле записи содержит имя поля, вслед за которым через двоеточие указывается тип этого поля. Поля записи могут относиться к любому типу, допустимому в языке Pascal, за исключением файлового типа.

Описание записи в языке Pascal осуществляется с помощью служебного слова RECORD, вслед за которым описываются компоненты записи. Завершается описание записи служебным словом END.

Например, записная книжка содержит фамилии, инициалы и номера телефона, поэтому отдельную строку в записной книжке удобно представить в виде следующей записи:

```
type Row = Record
FIO: String[20];
TEL: String[7];
end;
var str: Row;
```

Описание записей возможно и без использования имени типа, например:

```
var str: Record
FIO: String[20];
TEL: String[7];
end;
```

Обращение к записи в целом допускается только в операторах присваивания, где слева и справа от знака присваивания используются имена записей одинакового типа. Во всех остальных случаях оперируют отдельными полями записей. Чтобы обратиться к отдельной компоненте записи, необходимо задать имя записи и через точку указать имя нужного поля. Такое имя называется составным. Компонентой записи может быть также запись, в таком случае составное имя будет содержать не два, а большее количество имен.

Обращение к компонентам записей можно упростить, если воспользоваться оператором присоединения with. Он позволяет заменить составные имена, характеризующие каждое поле, просто на имена полей, а имя записи определить в операторе присоединения.

Иногда содержимое отдельной записи зависит от значения одного из ее полей. В языке Pascal допускается описание записи, состоящей из общей и вариантной частей. Вариантная часть задается с помощью конструкции case P of, где P – имя поля из общей части записи. Возможные значения, принимаемые этим полем, перечисляются так же, как и в операторе варианта. Однако вместо указания выполняемого действия, как это делается в операторе варианта, указываются поля варианта, заключенные в круглые скобки. Описание вариантной части завершается служебным словом end. Тип поля P можно указать в заголовке вариантной части. Инициализация записей осуществляется с помощью типизированных констант.

Конец ознакомительного фрагмента.