ИНФОРМАТИКА И ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ

КОНСПЕКТ ЛЕКЦИЙ

EKSMO EDUCATION

Хит сезона

ЭКЗАМЕН

В КАРМАНЕ

А. В. Цветкова

Информатика и информационные технологии: конспект лекций

«Научная книга»

Цветкова А. В.

Информатика и информационные технологии: конспект лекций / А. В. Цветкова — «Научная книга»,

Конспект лекций соответствует требованиям Государственного образовательного стандарта высшего профессионального образования РФ и предназначен для освоения студентами вузов специальной дисциплины «Информатика и информационные технологии». Лаконичное и четкое изложение материала, продуманный отбор необходимых тем позволяют быстро и качественно подготовиться к семинарам, зачетам и экзаменам по данному предмету.

Содержание

А. В. Цветкова
Информатика и информационные технологии: конспект лекций

ЛЕКЦИЯ № 1. Введение в информатику

1. Информатика. Информация. Представление и обработка информации

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*. Интерпретируя представления информации, мы получим ее смысл, семантику. Поэтому, 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 хи(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, так как для каждых конкретных параметров можно получить значение функции с заданной степенью точности.

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.* Если количество символов кодируемого алфавита меньше количества символов кодирующего алфавита, то сама операция кодирования не является сложной, в противном случае необходимо использовать фиксированный набор символов кодирующего алфавита для однозначного правильного кодирования.

Как показала практика, наиболее простым алфавитом, позволяющим кодировать другие алфавиты, является двоичный, состоящий из двух символов, которые обозначаются, как правило, через О 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.*

В качестве стандартного сегодня в информатике принят код, в котором каждый символ кодируется 1 байтом. Существуют и другие алфавиты.

2. Системы счисления

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.

3. Представление чисел в ЭВМ

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, соответственно числа с фиксированной точкой представляются в дополнительном коде. При этом если число занимает 2 байта, то структура числа записывается по следующему правилу: старший разряд отводится под знак числа, а остальные – под двоичные цифры числа. Дополнительный код положительного числа равен самому числу, а дополнительный код отрицательного числа может быть получен по такой формуле: $x = 10и - \backslash x \backslash$, 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.

4. 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 ψni. Тогда, если знаком функции является ψni, 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. Разные ученые предлагали свои подходы к формализованному

представлению алгоритма. Например, американский ученый Черч предположил, что класс вычислимых функций исчерпывается рекурсивными функциями и, как следствие, каким бы ни был алгоритм, перерабатывающий один набор целых неотрицательных чисел в другой, найдется алгоритм, сопутствующий рекурсивной функции, эквивалентный данному. Следовательно, если для решения некоторой поставленной задачи нельзя построить рекурсивную функцию, то и не существует алгоритма для ее решения. Другой ученый, Тьюринг, разработал виртуальную ЭВМ, которая перерабатывала входную последовательность символов в выходную. В связи с этим им был выдвинут тезис, что любая вычислимая функция вычислима по Тьюрингу.

ЛЕКЦИЯ № 2. Язык Pascal

1. 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:

ABCDEFGHIJKLMNOPQRSTUVWXYZ

abcdefghijklmnopqrstuvwxyz;

2) _ (знак подчеркивания);

3) 10 цифр: 0123456789;

4) знаки операций:

+ – x / = <> < > <= >= := @;

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. Пробелы нельзя использовать внутри сдвоенных символов и зарезервированных слов.

Концепция типа для данных

В математике принято классифицировать переменные в соответствии с некоторыми важными характеристиками. Производится строгое разграничение между вещественными, комплексными и логическими переменными, между переменными, представляющими отдельные значения и множество значений, и т. д. При обработке данных на ЭВМ такая классификация еще более важна. В любом алгоритмическом языке каждая константа, переменная, выражение или функция бывают определенного типа.

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.

Отсюда следует, что транслятор может использовать информацию о типах для проверки вычислимости и правильности различных конструкций.

Тип определяет:

1) возможные значения переменных, констант, функций, выражений, принадлежащих к данному типу;

2) внутреннюю форму представления данных в ЭВМ;

3) операции и функции, которые могут выполняться над величинами, принадлежащими к данному типу.

Следует заметить, что обязательное описание типа приводит к избыточности в тексте программ, но такая избыточность является важным вспомогательным средством разработки программ и рассматривается как необходимое свойство современных алгоритмических языков высокого уровня.

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.

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

Действительные типы определяет те данные, которые реализуются подмножеством действительных чисел, допустимых в данной ЭВМ.

Тип	Диапазон значений	Количество 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

Типы, определяемые пользователем, – перечисляемый и интервальный. Структурированные типы имеют четыре разновидности: массивы, множества, записи и файлы.

Кроме перечисленных, Pascal включает еще два типа – процедурный и объектный.

Выражение языка состоит из констант, переменных, указателей функций, знаков операций и скобок. Выражение задает правило вычисления некоторого значения. Порядок вычисления определяется старшинством (приоритетом) содержащихся в нем операций. В языке Pascal принят следующий приоритет операций:

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

Выражения входят в состав многих операторов языка Pascal, – также могут быть аргументами встроенных функций.

2. Стандартные процедуры и функции

Арифметические функции

1. Function Abs(X);
Возвращает абсолютное значение параметра.
X – выражение вещественного или целочисленного типа.

2. Function ArcTan(X: Extended): Extended;
Возвращает арктангенс аргумента.
X – выражение вещественного или целочисленного типа.

3. Function Exp(X: Real): Real;
Возвращает экспоненту.
X – выражение вещественного или целочисленного типа.

4. Function Frac(X: Real): Real;
Возвращает дробную часть аргумента.
X – выражение вещественного типа. Результат – дробная часть X, т. е.
Frac (X) = X–Int (X).

5. Function Int(X: Real): Real;
Возвращает целочисленную часть аргумента.
X – выражение вещественного типа. Результат – целочисленная часть X, т. е. X, округленный к нулю.

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.
X – выражение вещественного типа. Sin возвращает синус угла X в радианах.

9. Function Sqr(X: Extended): Extended;
Возвращает квадрат аргумента.
X – выражение с плавающей запятой. Результат того же самого типа, что и X.

10. Function Sqrt(X: Extended): Extended;
Возвращает квадратный корень аргумента.
X – выражение с плавающей запятой. Результат – квадратный корень X.

Процедуры и функции преобразования величин

1. Procedure Str(X [: Width [: Decimals]]; var S);

Преобразовывает число X в строковое представление согласно

Width и параметрам форматирования Decimals. X – выражение вещественного или целого типа. Width и Decimals – выражения целого типа. S – переменная типа String или символьный массив с нулевым окончанием, если допускается расширенный синтаксис.

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): Longint;

Возвращает порядковое значение выражения перечислимого типа. X – выражение перечислимого типа.

6. Function Round(X: Extended): Longint;

Округляет значение вещественного типа до целого. X – выражение вещественного типа. Round возвращает значение Longint, которое является значением X, округленным до ближайшего целого числа. Если X находится точно посередине между двумя целыми числами, возвращается число с наибольшей абсолютной величиной. Если округленное значение X выходит за диапазон Longint, генерируется ошибка времени выполнения программы, которую вы можете обработать с использованием исключительной ситуации EInvalidOp.

7. Function Trunc(X: Extended): Longint;

Усекает значение вещественного типа до целого. Если округленное значение X выходит за диапазон Longint, генерируется ошибка времени выполнения программы, которую вы можете обработать с использованием исключительной ситуации EInvalidOp.

8. Procedure Val(S; var V; var Code: Integer);

Преобразовывает число из строкового значения S в числовое

представление V. S – выражение строкового типа – последовательность символов, которая формирует целое или вещественное число. Если выражение S недопустимо, индекс неверного символа сохраняется в переменной Code. В противном случае Code устанавливается в нуль.

Процедуры и функции работы с порядковыми величинами

1. Procedure Dec(varX [; N: Longlnt]);

Вычитает единицу или N из переменной X. Dec(X) соответствует X:= X – 1, и Dec(X, N) соответствует X:= X – N. X – переменная перечислимого типа или типа PChar, если допускается расширенный синтаксис, и N – выражение целочисленного типа. Процедура Dec генерирует оптимальный код и особенно полезна в длительных циклах.

2. Procedure Inc(varX [; N: Longlnt]);

Прибавляет единицу или N к переменной X. X – переменная перечислимого типа или типа PChar, если допускается расширенный синтаксис, и N – выражение целочисленного типа. Inc (X) соответствует инструкции X:= X + 1, и Inc (X, N) соответствует инструкции X:= X + N. Процедура Inc генерирует оптимальный код и особенно полезна в длительных циклах.

3. Function Odd(X: Longlnt): Boolean;

Возвращает True, если X – нечетное число, и False – в противном случае.

4. Function Pred(X);

Возвращает предыдущее значение параметра. X – выражение перечислимого типа. Результат того же самого типа.

5. Function Succ(X);

Возвращает следующее значение параметра. X – выражение перечислимого типа. Результат того же самого типа.

3. Операторы языка Pascal

Условный оператор

Формат полного условного оператора определяется следующим образом: If B then SI else S2; где B – условие разветвления (принятия решения), логическое выражение или отношение; SI, 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.

Также существует сокращенная форма условного оператора. Она записывается в виде: *If B then S.*

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…., *cn* – константы порядкового типа, с которыми сравниваются выражения

S; instruction1,…, instructionN – операторы, из которых выполняется тот, с константой которого совпадает значение выражения *S;*

instruction – оператор, который выполняется, если значение выражения *Sylq* совпадает ни с одной из констант c1, c2…. cn.

Данный оператор является обобщением условного оператора If для произвольного числа альтернатив. Существует сокращенная форма оператора, при которой ветвь *else* отсутствует.

Оператор цикла с параметром

Операторы цикла с параметром, которые начинаются со слова *for,* вызывают повторяющееся выполнение оператора, который может быть составным оператором, пока управляющей переменной присваивается возрастающая последовательность значений.

Общий вид оператора *for*:

for <счетчик цикла> := <начальное значение> to <конечное значение> do <оператор>;

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 инициализиру-

ется начальным значением. Когда работает оператор *for*, значение счетчика цикла увеличивается при каждом повторении на единицу. Если начальное значение превышает конечное значение, то содержащийся в теле оператора for оператор не выполняется. Когда в операторе цикла используется ключевое слово *downto*, значение управляющей переменной уменьшается при каждом повторении на единицу. Если начальное значение в таком операторе меньше, чем конечное значение, то содержащийся в теле оператора цикла оператор не выполняется.

Если оператор, содержащийся в теле оператора *for*, изменяет значение счетчика цикла, то это является ошибкой. После выполнения оператора *for* значение управляющей переменной становится неопределенным, если только выполнение оператора *for* не было прервано с помощью оператора перехода.

Оператор цикла с предусловием

Оператор цикла с предусловием (начинающийся с ключевого слова while) содержит в себе выражение, которое управляет повторным выполнением оператора (который может быть составным оператором). Форма цикла:

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 True. Если выражение с самого начала принимает значение False, то оператор, содержащийся внутри оператора цикла с предусловием, не выполняется.

Оператор цикла с постусловием

В операторе цикла с постусловием (начинающимся со слова *repeat*) выражение, которое управляет повторным выполнением последовательности операторов, содержится внутри оператора *repeat*. Форма цикла:

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.

ЛЕКЦИЯ № 3. Процедуры и функции

1. 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 раздел описаний 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.

2. Процедуры в Pascal

Каждое описание процедуры содержит заголовок, за которым следует программный блок. Общий вид заголовка процедуры следующий:

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

Процедура активизируется с помощью оператора процедуры, в котором содержатся имя процедуры и необходимые параметры. Операторы, которые должны выполняться при запуске процедуры, содержатся в операторной части модуля процедуры. Если в содержащемся в процедуре операторе внутри модуля процедуры используется идентификатор процедуры, то процедура будет выполняться рекурсивно, т. е. будет при выполнении обращаться сама к себе.

3. Функции в Pascal

Описание функции определяет часть программы, в которой вычисляется и возвращается значение. Общий вид заголовка функции следующий:

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

Функция активизируется при ее вызове. При вызове функции указываются идентификатор функции и какие-либо параметры, необходимые для ее вычисления. 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 функции 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, то функция выполняется рекурсивно.

4. 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 S, а затем, опять же без пробела, указывается директива.

Пример

1) {SE+} – эмулировать математический сопроцессор;
2) {SF+} —формировать дальний тип вызова процедур и функций;
3) {SN+} – использовать математический сопроцессор;
4) {SR+} – проверять выход за границы диапазонов.

Некоторые ключи компиляции могут содержать параметр, например:

{$1 имя файла} – включить в текст компилируемой программы названный файл.

ЛЕКЦИЯ № 4. Подпрограммы

1. 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, связанном с данной процедурой или функцией, на него можно ссылаться по его идентификатору.

Конец ознакомительного фрагмента.