№ 372

С.Э. Мурадханов
А.И. Широков

Алгоритмические языки высокого уровня

Курс лекций

№ 372

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ «МИСиС»

Кафедра автоматизированных систем управления

С.Э. Мурадханов
А.И. Широков

Алгоритмические языки высокого уровня

Курс лекций

Рекомендовано редакционно-издательским
советом университета

МИСиС
ИЗДАТЕЛЬСКИЙ ДОМ

Москва 2011

УДК 681.3
М91

Рецензент
канд. техн. наук, доц. *С.В. Никифоров*

Мурадханов, С.Э.

М91 Алгоритмические языки высокого уровня : курс лекций / С.Э. Мурадханов, А.И. Широков. – М. : Изд. Дом МИСиС, 2011. – 170 с.

ISBN 978-5-87623-421-6

В курсе лекций рассмотрены основные элементы алгоритмических языков программирования. Приводятся многочисленные примеры, в которых изложено все, что нужно современному специалисту для создания приложений: конструкции языка, динамические структуры данных и основы объектно-ориентированного подхода при разработке программ.

Соответствует учебному плану курса «Алгоритмические языки высокого уровня».

Предназначен для студентов специальностей «Автоматизированные системы обработки информации и управления» и «Прикладная информатика».

УДК 681.3

ISBN 978-5-87623-421-6

ОГЛАВЛЕНИЕ

Предисловие

Умение разрабатывать и отлаживать программы является неотъемлемым для современного специалиста в области информационных технологий. В предлагаемом курсе лекций рассмотрены основные элементы алгоритмического языка Pascal: типы данных (как простые, так и структурированные) и операторы. Но знание только конструкций для современного специалиста недостаточно, поэтому с их помощью изложены такие важные в программировании понятия, как разнообразные динамические структуры данных. Отдельные разделы пособия посвящены процедурным типам, файлам и устройствам. Последняя часть пособия посвящена объектно-ориентированному подходу при разработке программ. В приложении приводятся необходимые сведения по системам счисления, без чего, как считают авторы, специалисты по компьютерным технологиям разрабатывать программы не должны.

Материал пособия соответствует учебному плану курса «Алгоритмические языки высокого уровня», который преподается студентам специальностей «Автоматизированные системы обработки информации и управления» и «Прикладная информатика».

1. ОСНОВЫ ЯЗЫКА ПРОГРАММИРОВАНИЯ PASCAL

Известно, что компьютерные устройства функционируют на основе машинных языков, называемых языками низкого уровня, которые позволяют выполнять только простые операции в системе кодов. Создание сложных программ с использованием машинных кодов встречается с большими трудностями, поэтому программисты разработали специальные программные средства, облегчающие работу на компьютере. Первыми машинно-ориентированными языками были Ассемблеры, которые упростили подготовку машинных кодов путём введения определённых мнемонических обозначений (assembler – сборщик частей в одно целое). Каждый тип компьютера характеризуется своим Ассемблером, что делает невозможным запуск программы, составленной для одного процессора, на компьютере с иным типом процессора. Другой недостаток Ассемблера связан с трудностью разработки больших программных проектов. Указанные обстоятельства послужили стимулом для развития теории программного обеспечения в направлении создания алгоритмических языков высокого уровня, независимых от типа компьютера и по своей форме и содержанию напоминающих обычные языки общения между людьми.

К настоящему времени разработаны десятки алгоритмических языков: Алгол-60, Кобол, Модула 2, Фортран, Паскаль (Pascal), Бейсик, Си, Пролог и др. Перевод программы с алгоритмического языка высокого уровня на машинный язык производится с помощью специальных программ, называемых трансляторами. Трансляторы бывают двух типов: компиляторы и интерпретаторы.

Первые ЭВМ предназначались для решения задач вычислительного характера по заказу военных ведомств. Поэтому не случайно, что один из первых языков высокого уровня Фортран, созданный в 1957 г. группой специалистов фирмы IBM (США) под руководством Джона Бэкуса, предназначался для использования в математических приложениях. Само слово FORTRAN происходит от FORmula TRANslator, что означает «переводчик формул». Этот язык прекрасно адаптирован к решению научно-технических вычислительных задач и находит применение до настоящего времени.

Появление ЭВМ второго и последующего поколений подготовили условия для решения задач невычислительного характера. Новые

задачи потребовали специального программного обеспечения. Оно обсуждалось на международном форуме в Париже в 1960 г., где в качестве универсального алгоритмического языка был выбран Алгол-60. При решении экономических задач, требующих проведения операций над большими массивами данных, используется язык Кобол (COBOL – Computer Bisnes Oriented Language).

Алгоритмический язык Pascal занимает особое место в программировании. История его создания начинается в 1965 г., когда Международная федерация по обработке информации (IFIP) предложила нескольким специалистам в области информатики принять участие в разработке нового языка программирования – преемника Алгол-60. Среди них был швейцарский учёный, работавший в то время доцентом на факультете информатики Стэндфордского университета, Николаус Вирт. Он создал версию языка Алгол под названием Алгол-W. Федерация IFIP отклонила этот проект в пользу работы Аад ван Вейнгаартена под названием Алгол-68. Николаус Вирт не прекратил работы по созданию новой среды для программирования и в 1968 г. вместе со своими сотрудниками из Швейцарского федерального института технологии (ETH) в Цюрихе предложил первую версию языка Pascal, которая предназначалась для обучения студентов программированию. Язык назвали в честь французского математика, физика и философа Блеза Паскаля (1623–1662). Через два года, в 1971 г., появился компилятор этого языка. Новый язык получил высокую оценку у специалистов, что стимулировало его развитие. Он воплотил в себе ряд новшеств, в частности, идею структурного программирования, и стал прародителем более поздних языков программирования (язык Модула-2 и Ада). Благодаря работе А. Хейлсберга язык Pascal превратился в современную систему программирования, удовлетворяющую самым высоким профессиональным требованиям к процессу разработки программ в операционной системе MS DOS. В 80-х годах прошлого века для этого языка появились диалоговые оболочки, так называемые турбосреды. В 1985 г. американская фирма Borland выпустила на рынок улучшенную версию языка под названием Turbo Pascal 3.0 (TP), которая получила большую популярность среди программистов В 1984 г. бывший студент Николауса Вирта французский математик Филип Кан, основал фирму Borland International и начал продавать по почте разработанную им программу – среду программирования Turbo Pascal для ПК. Предлагаемый им продукт был настолько удачен, что Turbo Pascal вскоре вывел

фирму Borland в ряды основных производителей программного обеспечения.

Постоянное усовершенствование языка Turbo Pascal (версии 4, 5, 5.5, 6) принесли ему заслуженный успех. Появилась возможность создавать программы с использованием принципов объектно-ориентированного программирования, применять многооконный режим, использовать программы, составленные на языке Ассемблер, применять мышь и другие новшества. Седьмая версия языка Turbo Pascal унаследовала всё положительное от ранних разработок. Она выгодно отличается от других алгоритмических языков тем, что построена на небольшом количестве базовых понятий, имеет простой синтаксис и осуществляет транслирование программы на машинные коды относительно простым компилятором. Более эффективный компилятор и редактор предоставляют программисту возможность создавать сложные программные проекты. Работа в среде языка Turbo Pascal обеспечивается набором модулей, содержащих библиотеки стандартных процедур и функций. Компилятор имеет две версии, поддерживаемые различными файлами. Один из них (файл turbo.exe) предназначен для программирования в интегрированной среде разработки языка (ИСР), другой (файл tpc.exe) поддерживает пакетный режим работы.

Лингвистические концепции Turbo Pascal строятся по принципу «Разделяй и властвуй», в основе которого предполагается разбиение большой программы на более мелкие фрагменты (модули, объекты). Отдельные фрагменты создаются независимо друг от друга. Данные могут объединяться с операциями. Подобные структуры и называются объектами.

Изначально все программы писались в машинных кодах. С появлением алгоритмических языков высокого уровня (к которым, в частности, относятся Pascal, C++, Delphi и др.) методика создания программ изменилась. Сначала программист в программе–редакторе пишет исходный текст на языке высокого уровня, затем программа компилятор (транслятор) транслирует этот текст в машинные коды.

Алгоритмические языки и ассемблеры относятся к языкам символьного кодирования, т.е. к языкам, которые оперируют не машинными кодами, а условными символьными обозначениями, поэтому программы, составленные на этих языках, не могут быть непосредственно выполнены на компьютере. Чтобы такая программа заработала, ее текст нужно преобразовать в машинный код. Для этого существуют специальные программы-переводчики (трансляторы). Разли-

чают два вида трансляторов: компилятор и интерпретатор. **Компилятор** транслирует программу сразу целиком, и лишь после этого возможно ее выполнение. **Интерпретатор** – это более простой транслятор, он последовательно транслирует операторы программы и также по частям ее выполняет. После этого начинается процесс «линковки» или «компоновки», в результате которого происходит объединение всех модулей программы (с подключением необходимых для выполнения программы библиотек, драйверов и т.п.) в единое целое – исполняемый загрузочный модуль. Затем этот исполняемый модуль может быть загружен в оперативную память ПК для выполнения.

Обычный разговорный язык состоит из четырех основных элементов: символов, слов, словосочетаний и предложений. Приблизительно такие же конструкции можно выделить и в языках программирования. Перейдем к их описанию.

2. ЭЛЕМЕНТЫ АЛГОРИТМИЧЕСКОГО ЯЗЫКА PASCAL

Алгоритмический язык содержит подобные элементы, только слова называют элементарными конструкциями, словосочетания – выражениями, предложения – операторами.

Символы, элементарные конструкции, выражения и операторы составляют иерархическую структуру. Элементарные конструкции – это последовательность символов. Выражения – это последовательность элементарных конструкций и символов. Оператор – это последовательность выражений, элементарных конструкций и символов.

Описание алгоритмического языка – описание четырех названных элементов. Описание символов заключается в перечислении допустимых символов языка. Под описанием элементарных конструкций понимают правила их образования. Описание выражений – правила образования любых выражений, имеющих смысл в данном языке. Описание операторов состоит из рассмотрения всех типов операторов, допустимых в языке.

СИМВОЛЫ языка – это основные неделимые знаки, в терминах которых пишутся все тексты на языке.

ЭЛЕМЕНТАРНЫЕ КОНСТРУКЦИИ – это минимальные единицы языка, имеющие самостоятельный смысл. Они образуются из основных символов языка.

ВЫРАЖЕНИЕ в алгоритмическом языке состоит из элементарных конструкций и символов, оно задает правило вычисления некоторого значения.

ОПЕРАТОР задает полное описание некоторого действия, которое необходимо выполнить. Для описания сложного действия может потребоваться группа операторов. В этом случае операторы объединяются в СОСТАВНОЙ ОПЕРАТОР или БЛОК.

Действия, заданные операторами, выполняются над ДАННЫМИ. Предложения алгоритмического языка, в которых даются сведения о типах данных, называются ОПИСАНИЯМИ или неисполняемыми операторами. Объединенная единым алгоритмом совокупность описаний и операторов образует ПРОГРАММУ на алгоритмическом языке.

2.1. Алфавит языка Pascal

Текст программы на языке Turbo Pascal (TP) представляет собой последовательность строк, состоящих из символов, образующих ал-

фавит языка. Строки программы завершаются специальными управляющими символами, не входящими в алфавит (<CR> – возврат каретки, клавиша <Enter> и <LF> – новая линия). Максимальная длина строки составляет 126 символов.

В алфавит языка входят:

1. **Буквы латинского алфавита** (строчные и прописные) от `a` до `z` и от `A` до `Z`, а также знак подчеркивания «_», который приравнивается к буквам. (В TP нет различия между прописными и строчными буквами алфавита, кроме случаев, когда они входят в символьные и строковые выражения.)

2. **Арабские цифры** от `0` до `9`.

3. **Специальные символы**:

`+ - * / = . , : ; ' < > ( ) { } [ ] $ @ # ^`

и пробел, не имеющий графического изображения (используется для отделения лексем друг от друга).

4. **Знаки операций**, которые формируются из одного или нескольких специальных символов или служебных слов:

а) *арифметические операции*: `+` (сложение), `-` (вычитание), `*` (умножение), `/` (деление вещественных чисел), `mod` (деление целых чисел), `div` (остаток от деления двух целых чисел);

б) *операции отношения*: `<` (меньше), `>` (больше), `<=` (не больше), `>=` (не меньше), `=` (равно), `<>` (не равно);

в) *логические операции*: `and` (логическое И), `or` (логическое ИЛИ), `not` (логическое НЕ), `xor` (исключительное ИЛИ);

г) *операции над множествами*: `*` (пересечение множеств), `+` (объединение множеств), `IN` (принадлежность множеству).

5. **Символы национального алфавита, например русского**.

6. **Комментарии** – произвольная последовательность символов, в том числе и русских букв, заключенных в фигурные скобки {…} или `(* … *)`, предназначенная для пояснений в программе. Комментарии могут находиться в любой части программы.

2.2. Константы

В качестве констант в Pascal могут использоваться целые, вещественные и шестнадцатеричные числа, логические константы, символы, строки символов, конструкторы множеств и признак неопределенного указателя `NIL`.

Целые числа в Pascal записываются в обычном виде, например: `0, +100,-56498.` Следует учесть, что если целочисленная константа выходит за указанные границы, компилятор дает сообщение об ошибке. Такие константы должны записываться с десятичной точкой, т.е. определяться как вещественные числа.

Вещественные числа в Pascal представляются в одной из двух форм, которые называются: запись числа с фиксированной точкой и запись с плавающей точкой.

Первая форма – запись числа в виде целой и дробной частей: `–3.15, 0.1, +23.0125.`

Вторая форма (с плавающей точкой) – это запись числа с мантиссой и десятичным порядком, разделенными латинской буквой `E`. Такая запись означает, что мантисса (которая может быть целым числом или вещественным числом в форме с фиксированной точкой) умножается на 10 в степени, задаваемой порядком (который всегда должен быть целым числом): `-18.7E+3, 2.123E+4, 2.34E-2, 6E-1.`

В языке программирования Pascal запрещается запись вещественных чисел в виде `.5` или `5,` их необходимо записывать как `0.5` и `5.0` соответственно. Если в записи числа содержится точка, то, по крайней мере, одна цифра ей должна предшествовать и следовать за ней.

Выражение-константа представляет собой выражение, которое может вычисляться компилятором без необходимости выполнения программы. Поскольку компилятор должен иметь возможность полностью вычислить выражение-константу во время компиляции, в качестве выражений-констант не допускается использовать следующие конструкции:

- ссылки на переменные и типизированные константы (кроме констант в адресных выражениях);
- вызовы функций (кроме тех, которые отмечены далее);
- оператор получения адреса @ (кроме констант в адресных выражениях).

В выражениях-константах допускается использовать следующие стандартные функции: `Abs, Chr, Hi, High, Length, Lo, Low, Odd, Ord, Pred, Ptr, Round, SizeOf, Succ, Swap, Trunc.`

Примеры использования выражений-констант

```
Const
   Min = 0;
   Max = 100;
   Center = (Max - Min) div 2;
   Beta = Chr(255);
   NumChars = Ord('Z') - Ord('A') + 1;
   Message = 'Out of memory';
   ErrStr = 'Error:' + Message + '.';
   ErrPos = 80 - Length(Error) div 2;
   ErrAttr = Blink + Red * 16 + White;
   Ln10 = 2.302585092994095684;
   Ln10R = 1 / Ln10;
   Numeric = ['0'..'9'];
   Alpha = ['A'..'Z','a'..'z'];
   AlphaNum = Alpha + Numeric;
   DD = $7F;
```

Шестнадцатеричное число состоит из шестнадцатеричных цифр, которым предшествует знак доллара $ (код 36 в ASCII). Диапазон шестнадцатеричных чисел – от $00000000 до $FFFFFFFF.

Логическая константа – это слово FALSE (ложь), либо слово TRUE (истина).

Символьная константа – это любой символ ПК, заключенный в апострофы: 'z' – символ z; 'Ф' – символ Ф. Если необходимо записать как символьную константу сам апостроф ('), он удваивается: ''''. Допускается использование записи символа путем указания его внутреннего кода, которому предшествует символ # (код 35): #97 – символ a; #90 – символ Z; #39 – символ '; #13 – символ CR.

Строковая константа – любая последовательность символов (кроме символа CR – возврат каретки), заключенная в апострофы. Если в строке нужно указать сам символ апострофа, он удваивается.

A='Это - строка символов' – строка на экране:[Это - строка символов]

B= 'That''s string' – строка на экране:[That's string]

Строка символов может быть пустой, т.е. не иметь никаких символов в обрамляющих ее апострофах.

Строку можно составлять из кодов нужных символов с предшествующими каждому коду символами #: К примеру такая запись #83#121#109#98#11#108 эквивалентна строке 'Symbol'. Здесь 83 – код символа S, 121 – y и т.д.

В строке можно чередовать части, записанные в обрамляющих апострофах, с частями, записанными кодами. Таким способом можно

вставлять в строки любые управляющие символы, в том числе и символ CR (код 13): CC=#7'Ошибка !'#13'Нажмите любую клавишу ...'#7 .

2.3. Идентификаторы и переменные

Важным элементом исходного текста программы являются идентификаторы.

Идентификаторы – это имена констант, типов переменных, переменных, меток, процедур (функций), программ, модулей, библиотек, объектов, полей в записях и т.п. Использование идентификаторов осуществляется с помощью описаний. В качестве идентификаторов можно использовать любые последовательности символов, которые удовлетворяют следующим условиям:

- идентификатор может состоять из букв латинского алфавита, цифр, знака подчеркивания (другие символы в описании недопустимы);
- идентификатор не может начинаться с цифры;
- идентификатор не может совпадать ни с одним из зарезервированных слов;
- длина идентификатора может быть произвольной, но значащими считаются первые 63 символа.

Пример программы (использования описания идентификаторов)

```
Program Stepen;
Var
   A,B: Real;
   N: Integer;
Begin
   Writeln('Программа возведения числа в степень');
   Write('Введите число = '); Readln(A);
   Write('Укажите степень = '); Readln(N);
   Writeln;
   B:=Exp(N*Ln(A));
   Writeln('Результат:');
   Writeln('B= ',B:8:2);
   Readln
End.
```

В примере `Stepen, A,B,N` – правильные идентификаторы.

Переменная – это область оперативной памяти, занимающая несколько ячеек и имеющая свое имя. Переменная обладает следующими свойствами:

- переменная хранит не более 1 значения;
- переменная способна хранить значения только одного типа;

• переменная хранит значение до тех пор, пока в нее не поместят новое значение, при этом предыдущее значение переменной стирается;

• значение переменной может быть вызвано для использования сколько угодно раз без изменения оригинала (значения);

• в начале выполнения программы содержимое переменной считается неопределенным;

• ячейки памяти, отведенные под переменную путем ее описания, заполняются значениями в ходе выполнения программы с помощью оператора присваивания (этим переменная отличается от константы, которой значение присваивается до выполнения основной программы, в разделе определения констант).

Различные типы данных занимают в оперативной памяти ПК разное количество ячеек (емкость одной ячейки – 1 байт). Перед началом вычислений следует сообщить ПК, сколько ячеек памяти надо зарезервировать под ту или иную переменную. Для этого в блоке «VAR» раздела описаний программы должны быть описаны все используемые переменные, т.е. должно быть указано имя каждой переменной и типы данных, которые будут храниться в этих переменных.

В настоящее время в профессиональном программировании принято записывать имена переменных с использованием так называемой венгерской нотации.

Венгерская нотация – это соглашение о наименованиях переменных и функций. Соглашение широко используется при программировании на языках Pascal, C и в среде WINDOWS. Венгерская нотация основывается на следующих принципах:

• имена переменных и функций должны содержать префикс, описывающий их тип;

• имена переменных и функций записываются полными словами или словосочетаниями или их сокращениями, но так, чтобы по имени можно было понять назначение переменной или действие, выполняемое функцией.

Префиксы записываются малыми буквами, первая буква каждого слова – заглавная, префиксы и слова записываются либо слитно, либо через символ _ (подчеркивание).

Для языка Pascal рекомендованы префиксы для скалярных переменных и функций, приведенные в табл. 2.1.

Таблица 2.1

Префиксы, используемые в именах переменных

№	Префикс	Тип	№	Префикс	Тип
1	by	Byte	11	ch	Char
2	sh	Shortint	12	b	Boolean
3	i	Integer	13	p	Pointer
4	w	Word	14	ar	Array
5	l	Longint	15	s	String
6	r	Real	16	sz	Stringz
7	si	Single	17	se	Set
8	d	Double	18	re	Record
9	e	Extended	19	f	File
10	c	Comp	20	t	Text

Пример: `Var rV, arVector[1..20], sName, iCount.`

В откомпилированной программе для всех переменных отведено место в памяти, и всем переменным присвоены нулевые значения.

2.4. Служебные (ключевые и зарезервированные) слова

Любая программа на Pascal начинается словом `PROGRAM` и содержит объявление имени программы. Слово `PROGRAM` зарезервировано в Pascal, т.е. не может использоваться ни в каких иных целях, кроме как для объявления имени программы. В Pascal имеется множество зарезервированных слов. Любое из них нельзя использовать в качестве идентификатора (имени) какого-либо объекта программы – переменной, константы и т.д. Смысл зарезервированных слов строго зафиксирован (каждое слово предназначено для описания определенных действий). Служебные слова нельзя использовать не по назначению. При этом набор зарезервированных слов может меняться от версии к версии. Происходит это потому, что в качестве новой версии языка появляются дополнительные возможности, для реализации которых нужны новые зарезервированные слова. В то же время некоторые из старых зарезервированных слов перестают быть таковыми. Делается это для лучшей переносимости программ.

Основные зарезервированные слова:

`absolute` – директива, с помощью которой определяется абсолютный адрес переменных, функций или процедур;

`and, or, xor, not` – директивы логических операций;

`array` – используется для описания переменных типа «массив»;

`begin, end` – «операторные скобки». Любая программа начинается со слова `begin` и заканчивается словом `end` с точкой (`end.`). Внутри «операторных скобок» может находиться любой набор операторов, действий или выражений. Операторы всегда отделяются друг от друга точкой с запятой «`;`»;

`case` – оператор выбора вариантов (для реализации разветвления процесса);

`if <операторы> then <операторы> else <операторы>` – оператор «разветвления» вычислительного процесса (возможно два разветвления: да – в этом случае выполняются операторы, стоящие после `then`, или нет – выполняются операторы, стоящие после `else`);

`const` – описание констант;

`object, constractor, destructor, virtual` – директивы для работы с объектами;

`do(downto), for, to, while, repeat, until` – операторы цикла;

`file` – описание переменной типа «файл»;

`forward` – директива, которая говорит о том, что вызывается процедура или функция, описание которой в программе будет после;

`function, procedure` – директива описания процедур и функций;

`implementation, interface, unit, uses` – директивы для описания модулей и библиотек;

`mod` – операция взятия остатка от деления по модулю;

`div` – операция целочисленного деления;

`program` – идентификатор названия программы;

`set` – описание переменной типа «множество»;

`string` – описание переменной типа «строка» (массив символов);

`var` – описание переменных;

`type` – описание типов переменных;

`goto(label)` – переход на определенное место в программе;

`in` – операция проверки принадлежности элемента множеству;

`nil` – нулевая ссылка (нулевой адрес динамической переменной);

`record` – описание переменной типа «запись».