Тема 9

Основы алгоритмизации и программирования

9.1. Понятие алгоритма

Алгоритмом называется строго определенное и понятное предписание исполнителю совершить последовательность действий, направленных на решение поставленной задачи.

Термин «алгоритм» происходит от латинской формы имени среднеазиатского математика Аль-Хорезми – *Algorithmi*. Алгоритм является одним из основных понятий информатики и математики.

Исполнителем алгоритма предстает некоторая абстрактная или реальная (техническая, биологическая или биотехническая) система, которая способна выполнить действия, предписываемые алгоритмом.

Для характеристики исполнителя используют несколько понятий:

• среда;

• система команд;

• элементарные действия;

• отказы.

Среда (или обстановка) представляет собой «место обитания» исполнителя.

Любой из исполнителей может выполнять команды только из некоторого строго заданного списка, который является *системой команд* исполнителя. Для каждой команды задаются условия применимости (в каких состояниях среды может быть выполнена команда) и приводятся результаты выполнения команды.

После вызова команды исполнитель производит соответствующее *элементарное действие.*

Может возникнуть и *отказ* исполнителя в случае, если команда вызывается при недопустимом для нее состоянии среды. Чаще всего

исполнитель ничего не знает о цели алгоритма. Он выполняет все предложенные ему действия, не задавая вопросов «почему» и «зачем».

В информатике универсальным исполнителем алгоритмов является компьютер.

К основным свойствам алгоритмов относятся:

1) понятность для исполнителя – исполнитель алгоритма должен знать, как его выполнять;

2) дискретность (прерывность, раздельность) – алгоритм должен представлять процесс решения задачи как последовательное исполнение простых (или ранее определенных) шагов (этапов);

3) определенность – каждое правило алгоритма должно быть четким, однозначным и не оставлять места для произвола. Это свойство обеспечивает выполнение алгоритма механически, не требуя никаких дополнительных указаний или сведений о решаемой задаче;

4) результативность (или конечность) – алгоритм должен приводить к решению задачи за конечное число шагов;

5) массовость – алгоритм решения задачи производится в общем виде, т. е. его можно будет применять для некоторого класса задач, различающихся лишь исходными данными. При этом исходные данные могут выбираться из определенной области, которая называется областью применимости алгоритма.

На практике чаще всего встречаются следующие формы представления алгоритмов:

• словесная – записывается на естественном языке;

• графическая – с помощью изображения из графических символов;

• псевдокоды – полуформализованные описания алгоритмов на некотором условном алгоритмическом языке, которые включают в себя как элементы языка программирования, так и фразы естественного языка, общепринятые математические обозначения и др.;

• программная – тексты на языках программирования.

Словесный способ записи алгоритмов является описанием последовательных этапов обработки данных. Алгоритм может быть задан в произвольном изложении на естественном языке. Например, алгоритм нахождения наибольшего общего делителя двух натуральных чисел можно представить как следующую последовательность действий:

1) задание двух чисел;

2) если числа равны, то выбор любого из них в качестве ответа и остановка, в противном случае – продолжение выполнения алгоритма;

3) определение большего из чисел;

4) замена большего из чисел разностью большего и меньшего из чисел;

5) повтор алгоритма с шага 2.

Приведенный алгоритм используется для любых натуральных чисел и должен приводить к решению поставленной задачи.

Словесный способ не имеет широкого распространения, так как обладает некоторыми недостатками:

• данные описания строго не формализуемы;

• отличаются многословностью записей;

• допускают неоднозначность толкования отдельных предписаний.

Графический способ представления алгоритмов оказывается более компактным и наглядным по сравнению со словесным. При данном виде представления алгоритм изображается в виде последовательности связанных между собой функциональных блоков, каждый из которых соответствует выполнению некоторого числа действий.

Для графического представления алгоритм использует изображение в виде последовательности связанных между собой функциональных блоков, каждый из которых соответствует выполнению одного или нескольких действий. Это графическое представление называется *схемой алгоритма,* или *блок-схемой.*

В блок-схеме каждый из типов действий (ввод исходных данных, вычисление значений выражений, проверка условий, управление повторением

действий, окончание обработки и т. п.) соответствует геометрической фигуре, представленной в виде блочного символа. Блочные символы соединены линиями переходов, которые определяют очередность выполнения действий.

Псевдокод является системой обозначений и правил, которая предназначена для единообразной записи алгоритмов. Он занимает промежуточное место между естественным и формальным языками. С одной стороны, псевдокод похож на обычный естественный язык, поэтому алгоритмы могут на нем записываться и читаться как обычный текст. С другой стороны, в псевдокоде используются некоторые формальные конструкции и математическая символика, благодаря чему запись алгоритма приближается к общепринятой математической записи.

В псевдокоде не применяются строгие синтаксические правила для записи команд, которые присущи формальным языкам, что облегчает запись алгоритма на стадии его проектирования и дает возможность использовать более широкий набор команд, рассчитанный на абстрактного исполнителя. Однако в псевдокоде чаще всего имеются некоторые конструкции, присущие формальным языкам, что облегчает переход от записи на псевдокоде к записи алгоритма на формальном языке. Например, в псевдокоде, также каки в формальных языках, существуют служебные слова, смысл которых определен раз и навсегда. Их выделяют в печатном тексте жирным шрифтом, а в рукописном тексте подчеркивают. Единый или формальный подход к определению псевдокода не существует, поэтому используются различные псевдокоды, отличающиеся набором служебных слов и основных (базовых) конструкций.

Программная форма представления алгоритмов иногда характеризуется некоторыми структурами, состоящими из отдельных базовых (основных) элементов. При данном подходе к алгоритмам изучение основных принципов их конструирования следует начинать с этих базовых элементов. Их описание осуществляется с использованием языка схем алгоритмов и алгоритмического языка.

9.2. Системы программирования

Машинно-ориентированные языки относятся к *машинно-зависимым* языкам программирования. Основные конструктивные средства таких языков позволяют учитывать особенности архитектуры и принципов работы определенной ЭВМ, т. е. они имеют те же возможности и требования к программистам, что и машинные языки. Однако в отличие от последних они требуют предварительной трансляции на машинный язык составленных с их помощью программ.

Данными видами языков программирования могут быть: автокоды, языки символического кодирования и ассемблеры.

Для *машинно-независимых языков* не требуется полного знания специфики компьютеров. С их помощью можно записывать программу в виде, допускающем ее реализацию на ЭВМ с различными типами машинных операций, привязка к которым возлагается на соответствующий транслятор.

Причина бурного развития и применения высокоуровневых языков программирования заключается в быстром росте производительности ЭВМ и хронической нехватке программистских кадров.

Промежуточное место между машинно-независимыми и машинно-зависимыми языками отводится *языку Си.* Он создавался при попытке объединения достоинств, присущих языкам обоих классов. Данный язык обладает рядом особенностей:

• максимально использует возможности конкретной вычислительной архитектуры; из-за этого программы на языке Си компактны и работают эффективно;

• позволяет налучшим образом использовать огромные выразительные средства современных языков высокого уровня.

Языки разделяют на процедурно-ориентированные и проблемно-ориентированные.

Процедурно-ориентированные языки, например Фортран, Кобол, Бейсик, Паскаль, наиболее часто используются для описания алгоритмов решения широкого класса задач.

Проблемно-ориентированные языки, в частности РПГ, Лисп, АПЛ, GPSS, применяются для описания процессов обработки информации в более узкой, специфической области.

Объектно-ориентированные языки программирования позволяют разрабатывать программные приложения для большого круга разнообразных задач, имеющих общность в реализуемых компонентах.

Рассмотрим методы использования языков программирования.

Интерпретация представляет собой пооператорную трансляцию и последующее выполнение оттранслированного оператора исходной программы. Существует два основных недостатка метода интерпретации:

1) интерпретирующая программа должна располагаться в памяти ЭВМ на протяжении всего процесса выполнения исходной программы. Другими словами, она должна занимать некоторый установленный объем памяти;

2) процесс трансляции одного и того же оператора повторяется такое число раз, которое должна исполнять эта команда в программе. Это приводит к резкому снижению производительности работы программы.

Трансляторы-интерпретаторы являются достаточно распространенными, так как они поддерживают диалоговый режим.

Процессы трансляции и выполнения при *компиляции* разделяются во времени: сначала исходная программа в полном объеме переводится на машинный язык, после чего оттранслированная программа может многократно исполняться. Для трансляции методом компиляции необходим неоднократный «просмотр» транслируемой программы, т. е. *трансляторы-компиляторы* являются многопроходными. Трансляция методом компиляции носит название *объектного модуля,* который представляет собой эквивалентную программу в машинных кодах. Необходимо, чтобы перед

исполнением объектный модуль обрабатывался специальной программой ОС и преобразовывался в *загрузочный модуль.*

Применяют также трансляторы *интерпретаторы-компиляторы,* объединяющие в себе достоинства обоих принципов трансляции.

9.3. Классификация языков программирования высокого уровня

Высокоуровневые языки используются в машинно-независимых системах программирования. Такие системы программирования в сравнении с машинно-ориентированными системами предстают более простыми в использовании.

Языки программирования высокого уровня подразделяют на процедурно-ориентированные, проблемно-ориентированные и объектно-ориентированные.

Процедурно-ориентированные языки применяются для записи процедур или алгоритмов обработки информации на каждом определенном круге задач. К ним относятся:

а) язык Фортран (Fortran), название которого происходит от слов Formulae Translation – «преобразование формул». Фортран представляет собой один из старейших языков программирования высокого уровня. Длительность его существования и применения можно объяснить простотой структуры данного языка;

б) язык Бейсик (Basic), который расшифровывается как Beginner's All-purpose Symbolic Instruction Code, что в переводе означает – «многоцелевой символический обучающий код для начинающих», разработан в 1964 г. как язык для обучения программированию;

в) язык Си (C), применяемый с 1970-х гг. как язык системного программирования специально для написания ОС UNIX. В 1980-е гг. на основе языка С был разработан язык С++, практически включающий в себя язык С и дополненный средствами объектно-ориентированного программирования;

г) язык Паскаль (Pascal), который назван в честь французского ученого Б. Паскаля, начал применяться с 1968–1971 гг. Н. Виртом. При создании Паскаль использовался для обучения программированию, но со временем стал широко применяться для разработки программных средств в профессиональном программировании.

Проблемно-ориентированные языки используются для решения целых классов новых задач, возникших в связи с постоянным расширением области применения вычислительной техники:

а) язык Лисп (Lisp – List Information Symbol Processing), который был изобретен в 1962 г. Дж. Маккарти. Первоначально он применялся как средство для работы со строками символов. Лисп употребляется в экспертных системах, системах аналитических вычислений и т. п.;

б) язык Пролог (Prolog – Programming in Logic), используемый для логического программирования в системах искусственного интеллекта.

Объектно-ориентированные языки развиваются и в настоящий момент. Большинство из этих языков являются версиями процедурных и проблемных языков, но программирование с помощью языков этой группы является более наглядным и простым. К наиболее часто употребляемым языкам относятся:

а) Visual Basic (~ Basic);

б) Delphi (~ Pascal);

в) Visual Fortran (~ Fortran);

г) C++ (~ C);

д) Prolog++ (~ Prolog).

9.4. Система VBA

Система VBA представляет собой подмножество VB и вклю – чает себя средства образования приложений VB, его структуры данных и управляющие структуры, дающие возможность создавать пользовательские типы данных. Так же как и VB, VBA – является системой визуального программирования, управляемого событиями. В ней имеется возможность создания форм со стандартным набором элементов управления и написания процедур,

обрабатывающих события, которые возникают при тех или иных действиях системы и конечного пользователя. Также она позволяет использовать элементы ActiveX и автоматизации. Система VBA представляет собой полноценную систему программирования, но не имеет полного набора возможностей, которыми обладает последняя версия VB.

Программирование в среде VBA обладает рядом особенностей. В частности, в ней нельзя создавать проект независимо от этих приложений.

Из-за того что VBA является визуальной системой, программист способен создавать видимую часть приложения, которая является основой интерфейса «программа – пользователь». Благодаря этому интерфейсу производится взаимодействие пользователя с программой. На принципах объектно-ориентированного подхода, который реализуется в VBA применительно к приложениям, выполняемым под управлением Windows, разрабатывается программный интерфейс.

Характерным для данных приложений является то, что на экране в любой момент присутствует множество *объектов* (окон, кнопок, меню, текстовых и диалоговых окон, линеек прокрутки). С учетом алгоритма программы пользователь обладает определенной свободой выбора относительно использования этих объектов, т. е. он может сделать щелчок по кнопке, перенести объект, ввести данные в окно и т. п. При создании программы программист не должен ограничивать действия пользователя, он должен разрабатывать программу, правильно реагирующую на любое действие пользователя, даже некорректное.

Для любого объекта определяется ряд возможных *событий.* Одни события обусловлены действиями пользователя, например одинарным или двойным щелчком мыши, переносом объекта, нажатием клавиши клавиатуры и т. п. Некоторые события происходят в результате свершения других событий: окно открывается или закрывается, элемент управления становится активным или теряет активность.

Любое из событий проявляется в определенных *действиях* программы, а виды возможных действий можно разделить на две группы. Действия первой группы являются следствием свойств объекта, устанавливающихся из некоторого стандартного перечня свойств, которые задаются системой программирования VBA и самой системой Windows, например свертывание окна после щелчка по кнопке Свернуть. Вторую группу действий на события может определить только программист. Для любого возможного события отклик обеспечивается созданием процедуры VBA. Теоретически возможно создать процедуру для каждого события, но практически программист заполняет кодом процедуры только для событий, представляющих в данной программе интерес.

Объекты VBA являются функциональными, т. е. они действуют определенным образом и способны откликаться на конкретные ситуации. Внешний вид объекта и его поведение влияют на его свойства, а методы объекта определяют функции, которые способен выполнять данный объект.

Свойствами-участниками являются свойства, которые задают вложенные объекты.

Объекты способны реагировать на события – инициируемые пользователем и генерируемые системой. События, *инициируемые пользователем,* появляются, например, при нажатии клавиши, щелчка кнопками мыши. Исходя из этого любое действие пользователя может привести к целому набору событий. События, *генерируемые системой,* проявляются автоматически в случае, предусмотренном программным обеспечением компьютера.

9.5. Язык программирования VBA

Язык программирования VBA предназначен для написания кода программы. Он обладает своим алфавитом, который включает в себя:

• строчные и прописные буквы латинского алфавита *(A, B....,Z,a,b....,z);*

• строчные и прописные буквы кириллицы (А—Я, а—я);

• неотображаемые символы, используемые для отделения лексем (лексических единиц) друг от друга;

• специальные символы, участвующие в построении конструкций языка: +-*?^=><[](){}' &©;

• цифры от 0 до 9;

• символ подчеркивания «_»;

• составные символы, воспринимаемые как один символ: <=, >=, <>.

Лексема является единицей текста программы, которая имеет определенный смысл для компилятора и не может быть разбита в дальнейшем.

Программный код VBA – это последовательность лексем, записанных в соответствии с принятыми синтаксическими правилами, которая реализует нужную семантическую конструкцию.

Идентификатор представляет собой последовательность букв, цифр и символов подчеркивания.

Система VBA определяет некоторые ограничения, которые накладываются на имена:

1) имя следует начинать с буквы;

2) имя не должно включать в себя точки, пробелы, разделительные символы, знаки операций, специальные символы;

3) имя должно быть уникальным и не совпадать с зарезервированными словами VBA или другими именами;

4) длина имени не должна превышать 255 символов;

5) при составлении имен необходимо соблюдать соглашения по стилю;

6) идентификатор должен ясно отражать назначение переменной для понимания программы;

7) в именах лучше применять строчные буквы; если имена включают в себя несколько названий, их нужно отделять друг от друга подчеркиванием или начинать новое слово с прописной буквы;

8) имена констант следует составлять из прописных букв;

9) название идентификатора необходимо начинать со специального знака, указывающего на тип данных, связанный с этим идентификатором.

Переменные являются объектами, которые предназначены для хранения данных. Перед применением переменных в программе необходимо их объявлять (декларировать). Правильный выбор типа переменной обеспечивает эффективное использование памяти компьютера.

Строковые переменные могут быть переменной и фиксированной длины.

Объекты, значения которых не изменяются и не могут быть изменены во время выполнения программы, носят название *констант.* Их подразделяют на именованные и неименованные.

Перечни используются для декларации группы констант, объединенных общим именем, к тому же они могут быть объявлены только в разделе глобальных объявлений модуля или формы.

Переменные подразделяют на два вида – простые и переменные структурного вида. Массивы бывают одномерными и многомерными.

После декларации значение переменной может оказаться произвольным. Для присвоения переменной необходимого значения применяется *операция присваивания.*

Математические операции используются для записи формулы, представляющей собой программный оператор, который содержит числа, переменные, операторы и ключевые слова.

Операции отношения могут привести к появлению значения, причем существуют только два результирующих значения: истина и ложно.

Логические операции используются в логических выражениях, это происходит при существовании нескольких условий выбора в операциях отношения.

Операции для работы со строками – это операции конкатенации, которые позволяют объединить значения двух или нескольких строковых

переменных или строковых констант. Результатом такой операции является более длинная строка, составленная из исходных строк.