Глава 4

ЯЗЫКИ ПРОГРАММИРОВАНИЯ. БЕЙСИК

4.1. Общие сведения о языках программирования

Для решения задачи на компьютере необходимо исходные данные задачи и алгоритм ее решения записать на формальном языке программирования в виде программы, ввести программу и данные в память компьютера и подать компьютеру команду на ее исполнение. По степени зависимости языков программирования от структуры конкретного компьютера их условно можно разделить на языки низкого, среднего и высокого уровней.

К языкам **низкого уровня** относят языки машинных команд конкретных компьютеров. Программа на машинном языке представляет собой последовательность команд, содержащих коды выполняемых компьютером операций и адреса, участвующих в этих операциях операндов. Это **машинно-зависимые** языки. Программа, записанная на машинном языке одного компьютера, не всегда может быть выполнена на другом компьютере ввиду различий в системах команд и структурах этих компьютеров. На машинных языках до недавнего времени записывались программы операционных систем, программы для специализированных компьютеров. Такие программы отличает высокое быстродействие и малые затраты памяти. Программирование на машинных языках связано с большими затратами времени.

Недостатки машинных языков частично отсутствуют в **машинно-ориентированных языках среднего уровня**. Многие действия, затрудняющие программирование на машинном языке, например,

распределение памяти, переданы компьютеру. Вместо конкретных адресов и кодов операций применяется символика, используются макрокоманды, объединяющие несколько действий в одной команде. К языкам среднего уровня относят так называемые автокоды и языки Ассемблера — переводчика автокодов на машинный язык. Такие переводчики называют трансляторами. Языки Ассемблера позволяют наиболее полно использовать все возможности структуры компьютера.

Языки программирования **высокого уровня машинно-независимы**. Достаточно только, чтобы компьютер имел транслятор с данного языка на свой машинный язык. Языки программирования высокого уровня делят на проблемно-ориентированные и универсальные.

Проблемно-ориентированные языки разрабатываются с целью сделать более эффективным программирование задач одного класса, например, задач экономических (Кобол), искусственного интеллекта (Лисп, Пролог), обучения (ЛОГО), управления в реальном времени и работой многопроцессорных комплексов (АДА, Linda Modnla) и т.п.

Универсальные языки программирования позволяют успешно решать широкий круг задач различного характера. Среди них следует назвать один из первых процедурно-ориентированных языков Фортран (1950 г.), который и в настоящее время продолжает развиваться и совершенствоваться, Алгол–60 (1960 г.). К ним относятся также последние версии Бейсика, Паскаль, Си, APL и др.

Как правило, каждый из языков программирования в течение своей «жизни» добавляется, модернизируется, усиливается. Так появляются новые версии языка. Например, язык Бейсик, предложенный профессором математики Дортмутского колледжа (США) Томасом Куртцем как простой язык для начинающих, постепенно превратился в мощные универсальные языки высокого уровня: GW Basic, QBasic, Qnick Basic, Basic PDS 7.1 (Basic Professional Development System — профессиональное расширение языка для DOS), Visual Basic.

Современная среда Windows включила Бейсик как базовый язык программирования. Приложения под Windows используют версии Бейсика в качестве языков управления. Все это указывает на хорошие перспективы самого популярного в мире языка.

Одним из перспективных направлений развития технологий программирования считается создание объектно-ориентированных языков.

Это перспективные системы нового поколения. В качестве основного понятия систем объектно-ориентированного программирования выступает объект, объединяющий свойства и поведение модели, то есть включающий в себя как описывающие модель данные, так и средства обработки этих данных. В системах объектно-ориентированного программирования широко используется графический интерфейс. Процесс программирования сводится к конструированию программы с помощью мыши из готовых или вновь создаваемых объектов.

Принцип визуального программирования, при котором процесс создания программы сводится к ее сборке на экране дисплея из готовых программных конструкций — картинок, в большей степени реализован в системах объектно-ориентированного программирования Дельфи (Delphi), Visual Age, Visual Java («ява» или «джава»).

Близок к объектно-ориентированным языкам и Microsoft Visual Basic, однако его по-прежнему считают процедурным языком. Он широко распространен и используется в программах Microsoft Office Word, Excel и Access.

В Интернет используются: язык гипертекстовой разметки HTML, язык моделирования виртуальной реальности VRML.

При обсуждении выбора начального языка программирования многие отмечают несомненные преимущества QBasic'а и Quick Basic'а перед другими языками. Во-первых, они широко распространены, не требуют больших ресурсов компьютера, отлично работают на персональных компьютерах. Во-вторых, они при своей очевидной простоте освоения в большей степени «заставляют думать», помогают развивать воображение и чувство стиля, позволяют прививать хорошие начальные программистские навыки.

Выбор языка QBasic обусловлен его большей простотой и распространенностью по сравнению с языком Quick Basic. К тому же все программы, написанные на QBasic выполняются в среде Quick Basic. Эти языки имеют так много общего, что переход от одного языка к другому не вызывает затруднений. Важным преимуществом Quick Basic'а является возможность создавать исполняемые вне среды языка файлы. Отметим также, что QBasic является

интерпретатором, а Quick Basic — компилятором. Отличие первого от второго заключается в том, что **трансляторы интерпретирующего типа** рассматривают, переводят программу на машинный язык и исполняют ее построчно, одну строку программы за другой. **Трансляторы-компиляторы** анализируют и переводят на машинный язык всю программу в целом.

В учебнике язык QBasic рассматривается как средство, позволяющее продолжить цепочку «задача — алгоритм — программа — компьютер — результат». Операторы и функции языка, их синтаксис даются в объеме необходимом для демонстрации тех или иных особенностей реализации алгоритмов.

4.2. Язык программирования Бейсик

Алфавит языка Бейсик

Алфавит языка включает следующий набор символов:

A — Z, a — z	прописные и строчные буквы латинского алфавита;
0,1,2,...,9	арабские цифры с добавлением букв A — F (a — f) для обозначения чисел в шестнадцатеричной системе;

знаки арифметических операций:

- \+ сложение,
- \- вычитание,
- * умножение,
- / деление,
- \ целочисленное деление (деление нацело),
- ^ возведение в степень;

знаки операций отношения:

- = равно,
- > больше,
- < меньше;

знаки препинания и разделители:

. точка,
, запятая,
: двоеточие,
; точка с запятой,
— тире,
! восклицательный знак,
? вопросительный знак,
"" кавычки,
' апостроф,
( ) круглые скобки,
[ ] квадратные скобки,
" " пробел,
_ подчеркивание;

символы объявления типа:

% целое число одинарной точности (проценты),
& целое число двойной точности (амперсант),
! вещественное число одинарной точности,
вещественное число двойной точности,
$ знак строковой (символьной, текстовой) величины.

Клавиатура компьютеров позволяет также выдавать символы:

@ символ Эт,
~ знак эквивалентности,
| прямая черта,
{ } фигурные скобки,
№ знак номера,
а также элементы псевдографики, ранее использовавшиеся для построения таблиц, и другие символы.

Бейсик позволяет использовать при записи программ также национальные алфавиты, в частности: А — Я, а — я — прописные и строчные буквы русского алфавита.

Классификация данных

В языке используют числовые, строковые (текстовые) данные и данные, тип которых определяется пользователем.

Числовые данные

Среди числовых данных различают целые и вещественные числа, которые могут быть заданы в формате одинарной и двойной точности. Для увеличения диапазона представления числа могут задаваться в формате с плавающей точкой. Познакомимся с этой формой представления чисел, чтобы не удивиться, неожиданно получив результат в необычном виде.

Число в форме с плавающей точкой представляется в виде $A*10^P$, где A — мантисса, P — порядок. В формате одинарной точности пример записи числа следующий: a! = 2.407153E+7. Обычная запись этого числа такая: a = $2.407153*10^7$ = 24071530.

Пример записи числа в формате двойной точности следующий: b# = 5.472953369410318D+21, что в обычном виде выглядит так: b = $5.472953369410318*10^{21}$ = 5472953369410318000000.

Заметим, что при решении рассматриваемого ниже примера на компьютерах последних моделей результаты могут быть более впечатляющими.

Потребуем от компьютера последовательно возводить число 2 в степень n = 10, 14, 15, 30, 40, 125, 130, 333, 1023, 1024. При этом будем постепенно переходить от целых чисел одинарной точности (a%), к целым двойной точности (a&), затем к вещественным числам одинарной точности (a!), от которых к вещественным двойной точности (a#). В итоге получим следующие результаты, в которых левый столбец — это данные, вводимые в компьютер в окне для быстрого выполнения (нажатие клавиши F6), а правый столбец — ответ компьютера на задание (распечатка на экране дисплея):

```
a% = 2^10: ? a%          1024
a% = 2^14: ? a%          16384
a% = 2^15: ? a%          Overflow — переполнение
a& = 2^15: ? a&          32768
a& = 2^30: ? a&          1073741824
a& = 2^40: ? a&          Overflow — переполнение
a! = 2^40: ? a!          1.099512E+12
```

```
a! = 2^125: ? a!          4.25353E+37
a! = 2^130: ? a!          Overflow -- нереполнение
a# = 2^130: ? a#          1.361129467683754D+39
a# = 2^333: ? a#          1.74980057982641D+100 (10100
a# = 2^1023: ? a#         8.988465674431158D+307
a# = 2^1024: ? a#         Overflow — нереполнение
```

Выполним краткий анализ полученных результатов. Начинаем работу с формата целых чиеел одинарной точности (a%). Увеличивая показатель степени, видим, что при n = 15 результат выходит за диапазон представления чиеел. Переходим к двойной точности (a&). Убеждаемся, что в формате целых чисел двойной точности можно работать с очень большими числами, но при n = 40 дианазон целых чисел онять исчерпан полностью — переполнение диапазона.

Перехоцим к вещественным числам одинарной точности (a!), использующим форму представления чисел с плавающей точкой. Это нозволяет нам получить результат нри n = 40. Он равен:

$$1.099512E+12 = 1.099512 \cdot 10^{12} = 1099512 \cdot 10^{6} = 1099512000000.$$

Видим, что результат нолучен с округлением. Можно сделать вывод об уменьшении точности чисел с увеличением их величины. Увеличивается шаг, т.е. расстояние между соседними числами, которые могут быть представлены в компьютере. При n = 130 онять нереполнение диапазона. Переходим к формату вещественных чисел двойной точности (a#). Убеждаемся в значительном увеличении диапазона. Пошли числа, которым нет названий. Компьютер сдался при n = 1024. Это 308-значное десятичное число.

Известно, что 10^6 — миллион, 10^9 — миллиард, 10^{12} — триллион, 10^{15} — квадриллион, 10^{18} — квинтиллион, 10^{21} — секстиллион, 10^{24} — октиллион, 10^{27} — ниниллион, 10^{30} — дециллион. Физики считают, что во всей Вселенной количество элементарных частиц, из которых состоят атомы находящегося в ней вещества, не больше, чем 10^{88}. Поэтому, практической необходимости пользоваться числами, большими, чем 10^{100}, нет. Для этого числа придумали название Гугол. Компьютер даже без использования специальных подпрограмм позволяет работать с числами, имеющими в три раза больше разрядов, чем Гугол.

По умолчанию действия в программе выполняются с одинарной точностью, которая, как правило, достаточна при решении учебных задач. Поэтому, с целью упрощения записи программ суффиксы %, &, ! и # будем опускать.

Строковые данные

Представляют собой текст (строку) от одного до 32767 символов. Строковые данные, как правило, заключаются в кавычки, напримерː в$ = «XX век».

Константы

Константами называют используемые в процессе выполнения программы заранее определенные величины. Различают числовые и строковые константы. *Числовые константы* — это положительные или отрицательные числа, содержащие только символы, задающие величину числа. Они могут быть целые и вещественные, одинарной и двойной точности. *Строковые константы* — это последовательность любых символов.

Переменные

Переменными называют величины, которые в процессе выполнения программы могут менять свое значение. Переменные имеют свое имя, называемое идентификатором. Различают простые переменные и переменные-массивы. Под переменной можно понимать область памяти компьютера, в которой хранится значение этой переменной.

Простые переменные могут представлять в программе одно число, строку или запись. Переменной присваивают результаты выполнения операций. Пока переменной в программе не присвоено значения, числовая переменная равна нулю, строковая переменная равна пустой строке.

Переменные-массивы представляют в программе организованную группу элементов одного типа. В качестве элементов массива выступают переменные с индексами — номерами элементов в массиве. Индексы — целые числа. Они начинаются с нуля и заканчиваются величиной размера массива. Различают размерность массива (число измерений) и размеры размерностей массива (число элементов в каждом измерении).

Например, массив A(16) — одномерный массив. Он имеет одну размерность (одно измерение). Номера элементов массива: 0,1,2,...,16. Количество элементов в массиве 17. A(i) — обозначение i-го элемента массива. Индекс i может в данном примере принимать значения от 0 до 16. Такой массив можно записать еще так: A(0 TO 16).

Массив Z(10,10) — двухмерный. Он имеет две размерности по 11 элементов в каждой. Двухмерный массив C(N,M) можно представить в виде таблицы, имеющей N+1 строк и M+1 столбцов. Величина C(i,j) соответствует элементу такой таблицы, стоящему на пересечении i-й строки и j-го столбца. Второй вариант записи Z(0 TO 10, 0 TO 10) или Z(0 TO 10,10).

Максимальное число размерностей в массиве 60. Максимальное количество элементов каждой размерности 32767.

Операции и выражения

Выражением называют числовую или строковую константу, переменную или по определенным правилам записанную комбинацию констант, переменных и функций, соединенных знаками операций и скобками, определяющими последовательность выполнения этих операций.

Различают арифметические операции, операции отношения, логические операции и функциональные операции. В соответствии с названиями операций называются и выражения. Выражение может содержать различные операции. Например, арифметическое выражение может содержать функции. Величины, сравниваемые с помощью операций отношения, могут представлять собой арифметические выражения, логические выражения и т.д.

Начнем с функциональных операций, чтобы использовать их в примерах других выражений.

Функциональные операции

Бейсик имеет много встроенных, так называемых, стандартных функций. Различают числовые функции, текстовые (символьные) функции, функции преобразования типов, другие функции. Кроме стандартных имеются так называемые функции пользователя, тип и вид которых имеет возможность задавать программист. Стандартные функции языка Бейсик сведены в табл. 4.1.

Стандартные функции языка Бейсик — Таблица 4.1

Название функции	Обозначение в математике	Запись на Бейсике	Примечание
Синус	$\sin x$	SIN(x)	x задан в радианах
Косинус	$\cos x$	COS(x)	x задан в радианах
Тангенс	$\mathrm{tg}\ x$	TAN(x)	x задан в радианах
Арктангенс	$\mathrm{arctg}\ x$	ATN(x)	возможно Overflow
Корень квадратный	$\sqrt{x}$	SQR(x)	$X \geq 0$
Абсолютная величина	$\lvert x \rvert$	ABS(x)	x — числовое выражение
Экспонента	e^x	EXP(x)	e — основание натурального логарифма, e = 2,7
Целая часть числа		INT	округляет до ближайшего целого
Ближайшее целое		FIX	отбрасывает дробную часть
Логарифм	$\ln x$	LOG(x)	натуральный логарифм, $x > 0$
Случайное число		RND	выдает случайное число в диапазоне от 0 до 1
Знак числа		SGN(x)	выдает знак x, x — числовое выражение

Добавим к примечаниям в таблице следующее. В качестве аргументов выступают числовые выражения. Часто говорят, что программа задает аргумент, а компьютер возвращает значение функции. Возвращаемые значения по точности совпадают с точностью задания аргумента. Перевод в тригонометрических функциях угла из размерности в градусах в радианы выполняется умножением на pi/180 (pi = 3.14...).

Примеры выполнения операции **INT** (округление до ближайшего меньшего целого):

a) INT(10.51) = 10; b) INT(−10.22) = −11.

Примеры выполнения операции **FIX** (отбрасывает дробную часть числа):

a) FIX(10.51) = 10; b) FIX(−10.22) = −10.

Для получения случайных чисел в различных диапазонах можно пользоваться следующими соотношениями:

a = INT(RND*N)
a — целое положительное в диапазоне [0, N–1];

a = INT(RND*(N+1))
a — целое положительное в диапазоне [0, N];

a = INT(RND*N+1)
a — целое положительное в диапазоне [1, N];

a = INT(RND*1000)/10
a — вещественное положительное [0,99.9];

a = INT(RND*100-RND*100)
a — целое в диапазоне [–99, 99];

a = INT(RND*1000-RND*100)/10
a — вещественное в диапазоне [–9.9, 99.9] и т. д.

С каждым запуском программы генератор случайных чисел будет возвращать одну и ту же последовательность. Чтобы каждый раз получать разные последовательности, нужно базу функции RND привязать, например, к таймеру компьютера. Это делается в начале программы записью оператора RANDOMIZE TIMER.

Функция SGN(x) возвращает знак указанного в качестве аргумента числового выражения. При положительном аргументе возвращается 1, при нулевом — 0, при отрицательном — минус 1.

Арифметические операции и выражения

К арифметическим операциям сложение, вычитание, умножение, деление и возведение в степень добавим две дополнительные.

Целочисленное деление. Знак операции — обратная наклонная черта «\». Запись операции: a\b. Перед выполнением операции операнды округляются до целого значения, а в частном отбрасывается дробная часть.

Примеры выполнения операции деления нацело:
a) 19\4 = 4; b) 19.57\ 3.32 = 20\3 = 6.

Остаток от деления. Знак операции — слово **MOD**. Запись операции: a MOD b. Читается эта запись так: «a по модулю b». Результат операции — остаток от деления операндов. Перед выполнением операции операнды округляются.

Примеры выполнения операции деления по модулю:
a) 19 MOD 4 = 3; b) 19.57 MOD 3.32 = 20 MOD 3 = 2.

Последовательность арифметических операций, расположенных в порядке убывания приоритета выполнения, следующая: возведение в степень, умножение и деление, деление нацело, сложение и вычитание. Действия внутри круглых скобок выполняются первыми.

В арифметических выражениях могут быть записаны числовые константы, переменные, неременные с индексами, функции. Все эти элементы выражения объединены знаками арифметических операций и круглыми скобками.

Запись всех элементов арифметических выражений выполняется в одну строку. Поэтому суммы и разности в числителях и знаменателях дробей, а также произведения в знаменателях необходимо заключать в скобки. Нельзя ставить два знака арифметических действий подряд. Две последовательные операции должны разделяться круглыми скобками. Например, запись a/–b ошибочна. Необходимо записывать это выражение так: a/(–b). Ошибочен пропуск знака операции умножения. Часты ошибки при возведении в степень тригонометрических функций.

Математическое выражение	Запись на Бейсике
$\frac{a+b}{c-d}+\frac{e}{fg}$	(a + b)/(c – d) + e/(f*g)
$0{,}05a\sin^2 bx^3+2{,}5e^{2x}$	.05 * a * SIN(b * x^3)^2 + 2.5 * EXP(2 * x)
$\left\lvert 12{,}6-\sqrt{1+3\mathrm{tg}^2 x}\right\rvert$	ABS(12.6 – SQR(1 + TAN(x)^2))

Операции отношения

Операции отношения производят сравнение двух величин. Результат сравнения может быть истиной (TRUE) или ложью (FALSE). Этим логическим значениям присваивается, соответственно, значения 1 и 0. Перечень операций отношения приведен в табл. 4.2.

Операции отношения *Таблица 4.2*

Знак операции	Проверяемое отношение	Пример выражения
=	равно	a=b
<>	не равно	a<>b
<	меньше	a<b
>	больше	a>b
<=	меньше или равно	a<=b
>=	больше или равно	a>=b

При объединении в одном выражении арифметических операций и операций отношения сначала выполняются арифметические операции.

Логические операции

К логическим относят следующие операции.

Логическое умножение (многоместная операция И). Запись операции: X AND Y. Результат операции принимает значение TRUE (истина) только в случае, если все операнды имеют значение TRUE.

Логическое сложение (многоместная операция ИЛИ). Запись операции: X OR Y. Результат операции принимает значение FALSE (ложь) только в случае, если все операнды имеют значение FALSE.

Логическое отрицание (одноместная операция НЕ). Запись операции: NOT X. Логическое значение результата операции противоположно логическому значению аргумента.

Кроме этих трех несколько реже используются операции XOR (сумма операндов по модулю 2), EQV (равнозначность, эквивалентность операндов), IMP (импликация). Подробнее о логических операциях будет рассказано в главах 12 и 13.

Логические операции находят применение, например, для записи сложных логических условий. Математическое выражение a ≤ x ≤ b записывается на Бейсике так: x > = a AND x < = b. Если функция f(x) определена на двух отрезках числовой оси: a ≤ x ≤ b и c ≤ x ≤ d, то на Бейсике это условие запишется так:

x < = a AND x < = b OR x < = c AND x < = d.

Строковые операции

Строковые операции включают показанные в табл. 4.2 операции отношения и операцию конкатенации (сцепления) строк. Операции отношения используются для сравнения строк. Так как кодирование букв в алфавитах идет в возрастающем порядке величины кодов, то справедливы неравенства «б» > «а», «баба» < «дед», «abcd» < «abcf», «од» + «ин» > «два» и т.д. Поэтому строковые выражения могут быть элементами условий при ветвлении алгоритмов.

Операция конкатенации (знак операции «+») позволяет объединять строки. Например, при желании объединить строки «21-й» и «век» нужно записать следующее выражение: «21-й» + + « » + «век». В результате получаем строку: «21-й век».

4.3. Краткие сведения о среде Qbasic

Загрузка Qbasic

Загрузка среды Qbasic зависит от той среды, в которой она выполняется.

При работе с оболочкой Norton Commander нужно в каталоге найти файл qbasic.exe, выделить его и нажать на клавишу Enter. Обычно этот файл отсутствует в корневом каталоге. В этом случае следует раскрыть каталог с языками программирования и раскрыть подкаталог QBASIC, где должен находиться файл qbasic.exe. Заметим, что все перечисленные действия удобно выполнять с

помощью манипулятора мышь. Например, для раскрытия каталога, запуска на исполнение файла qbasic.exe следует подвести к ним курсор мыши и дважды быстро нажать на ее левую кнопку.

Если требуется войти в Qbasic из операционной системы Windows 95, 98, 2000, то запуск его можно осуществить через Проводника: Пуск / Программы / Проводник / Обзор /.../ QBASIC / qbasic.exe. Значок-пиктограмма Qbasic может находиться на рабочем столе. В этом случае следует дважды нажать левую кнопку мыши на значке Qbasic.

Характеристика окна редактирования

После загрузки Qbasic на экране монитора появляется приглашение для работы: Welcome to MS-DOS Qbasic. Удалить приглашение можно, нажав клавишу Esc (escape — убежать) или с помощью мыши. После этого среда готова к вводу с клавиатуры новых программ или редактированию и исполнению ранее написанных программ.

Главное окно редактирования схематично показано на рис. 4.1. Рассмотрим элементы окна. Они пронумерованы.

1. **Рабочее окно.** В нем выполняется ввод с клавиатуры и редактирование текста программ.

2. **Главное меню.** Это так называемое падающее (ниспадающее) меню. Подробнее о функциях этого меню и работе с ним рассказано в Приложении 1. Некоторые вопросы работы с главным меню рассмотрены ниже.

3. **Курсор.** Указывает на место, где будет напечатан вводимый с клавиатуры символ. Для перемещения курсора можно использовать клавиши перемещения курсора (со стрелками). Для быстрого перемещения курсора удобно применять мышь. Достаточно переместить указатель мыши в нужное место и нажать на левую кнопку.

4. **Заголовок.** В эту рамку вписывается имя файла, с которым идет работа в рабочем окне. Вписывание имени происходит автоматически при вызове бейсик-программы из памяти в рабочее окно. При сохранении новой программы среда запрашивает пользователя о том, под каким именем эта программа должна быть записана в память компьютера. До присваивания программе имени в рамке записано Untitled (без имени).

5. **Указатель прокрутки.** Служит индикатором положения курсора в тексте программы.

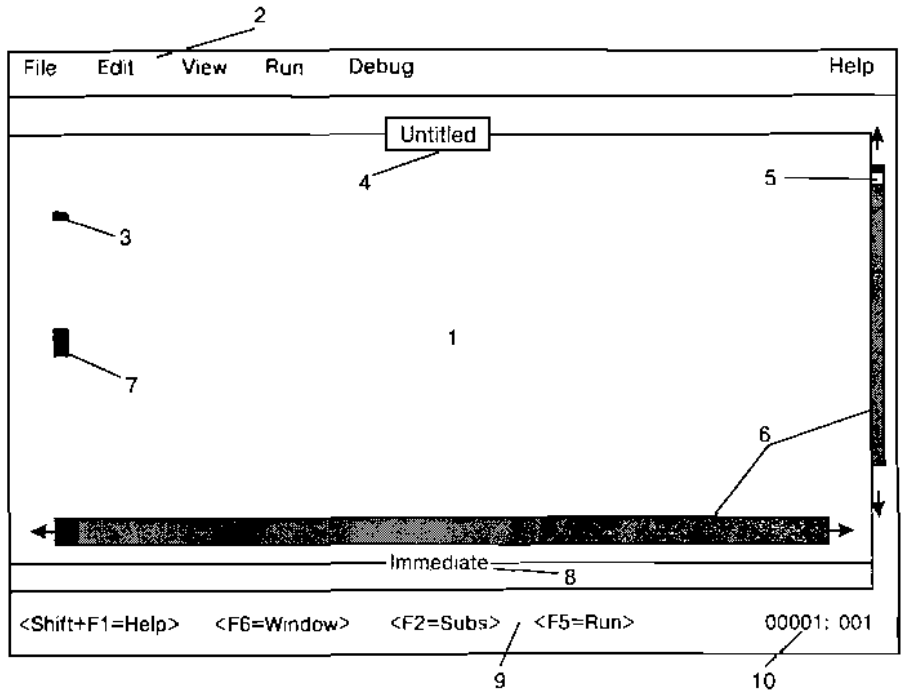

Рис. 4.1. *Окно среды языка Qbasic*

6. **Линейки прокрутки.** Предназначены для перемещения указателя прокрутки. Для перемещения курсора по тексту программы достаточно установить курсор мыши на указатель прокрутки и, удерживая нажатой кнопку мыши, перемещать указатель в нужное место. При этом вся линейка прокрутки примерно соответствует размеру программы.

7. **Курсор манипулятора мышь.** Показывает положение мыши на экране монитора.

8. **Окно для быстрого выполнения (Immediate).** Позволяет быстро получать результаты выполнения записываемых в окне операторов и функций. Запуск на выполнение — нажатие клавиши Enter. Окно удобно использовать при отладке программ, при проведении небольших экспериментов с вычислениями, цветом и музыкой. Для входа в окно и выхода из него следует нажать на клавиатуре клавишу F6.

9. **Подсказка по работе с функциональными клавишами.** Показывает назначение функциональных клавиш, часто используемых при работе с программами.

10. **Координаты текущего положения курсора.** Сообщает номера строки и столбца места нахождения курсора. Такая информация полезна при работе с большими программами.

4.4. Общие сведения о Бейсик-программах

Непосредственный режим

Различают непосредственный и программный режимы. Непосредственный режим еще называют режимом калькулятора. Переход в этот режим и обратно выполняется при нажатии клавиши F6. При этом активизируется нижнее окно для быстрого выполнения. Перед записью вычисляемого выражения ставится знак вопроса. Это требование к компьютеру вычислить и вывести на экран результат. Команда на выполнение подается нажатием клавиши Enter.

Примеры (слева — запись задания, справа — результат):

? 2^10	1024;
? "коло"+"бок"	колобок;
? 2+5>5	–1 (истина);
? "2+5" > "5"	0 (ложь);
? "пять" < "десять"	0 (ложь).

Программа

Программа представляет собой последовательность строк. Последовательность операторов в одной строке разделяется двоеточиями. Длина строки ограничена 256 символами. Строки могут быть пронумерованы, как это делалось в ранних версиях Бейсика. Номера строк в Quick Basic и Qbasic не определяют последовательность их выполнения. Строки могут иметь метки, позволяющие производить условные и безусловные переходы при необходимости изменить последовательность выполнения операторов. Отличие метки от номера строки в том, что метка может быть как числовым, так и алфавитно-цифровым идентификатором строки. Метка может иметь от 1 до 40 букв или цифр, должна начинаться с буквы и отделяться от операторов строки двоеточием. В общем виде синтаксис строки следующий:

[идентификатор строки:] [оператор] [:оператор] ... ['комментарий]

Набор и редактирование программы

Как правило, начинающий осваивать азы программирования набирает текст программы, работая одним пальцем и подолгу выискивая нужный символ. Но этого не избежать. Все так начинали. Квалифицированные пользователи, работая, например, с текстовыми редакторами, используют слепой метод набора, когда взгляд следит не за клавиатурой, а за вводимым текстом. Приблизиться к этому можно путем работы с клавиатурными тренажерами.

Клавиатурных тренажеров множество. Имеются детские тренажеры с увлекательными картинками. Имеются и профессиональные. Все они требуют правильной постановки рук, некоторой привязки пальцев к определенным клавишам. Напомним, что переучиваться труднее, чем научиться.

При наборе программы мало знать и уметь быстро находить нужные символы. Имеется множество полезных команд, подаваемых компьютеру нажатием на определенные клавиши, сочетания клавиш. Эти команды или необходимы при наборе, или существенно облегчают его.

К необходимым командам можно отнести следующие.

- Для перемещения курсора можно использовать клавиши со стрелками или мышь.
- Команды стирания неправильно введенного символа (слева от курсора — клавиша **Back Space**, справа от курсора — клавиша **Delete**).
- Команда замены символа другим символом (клавиша Insert и ввод нового символа).

- Команды перехода с латинского алфавита на русский алфавит и обратно (одновременно две клавиши **Shift** + **Shift** или **Shift** + **Ctrl**, что зависит от используемой программы-русификатора).
- Команды перехода от строчных букв к прописным и обратно (клавиша **Caps Lock**) и некоторые другие.

К полезным командам можно отнести, например, команды перемещения курсора в начало строки (клавиша **Home**), конец строки (**End**), выделения текста, копирования и удаления выделенного текста и многие другие. Осваивать все эти команды помогает таблица работы функциональных и управляющих клавиш, показанная в Приложении 2.

Главное меню содержит пункт **Edit** (редактирование). Переход к меню можно выполнить нажатием клавиши **Alt** или с помощью мыши. Система ниспадающих меню описана в Приложении 1. Эффективность работы при редактировании программы зависит от знания всех средств, предоставляемых пользователю средой языка Qbasic. Имеется двойное и тройное дублирование возможностей подачи одних и тех же команд. Например, команду можно подать: 1) выбором ее клавишами перемещения курсора и нажатием на клавишу **Enter**; 2) использованием «горячих» клавиш; 3) с помощью мыши. Это же можно сказать о командах, подаваемых при запуске программы на исполнение, при отладке программ, сохранении программ и других действиях.

Следует отметить высокое качество редакторов языков QBasic и Qnick Basic. Набор строк программы выполняется строчными буквами. Если строка набрана правильно, то при переходе к набору следующей строки (ввод строки программы осуществляется нажатием на клавишу **Enter**) все строчные буквы ключевых слов операторов и функций становятся прописными. Если этого не происходит, то в строке имеются синтаксические ошибки.

Ошибок при наборе большой программы избежать трудно. Но редактор среды языка Бейсик вовремя подсказывает о сделанных ошибках. Номера ошибок приведены в Приложении 3.

Запуск программы на исполнение

Следующим этапом работы с программой после набора — это ее запуск на выполнение. Запуск программы с ее начала выполняется нажатием клавиш **Shift** + **F5**. Нажатие клавиши **F5** запускает программу после ее остановки, например, оператором **STOP** или командой **Ctrl** + **Break**. Последняя команда может остановить выполнение программы, зациклившейся после запуска.

Команды запуска, как и многие другие, можно подавать, пользуясь падающим меню. Для активизации строки главного меню достаточно нажать на клавишу **Alt**. Передвигаясь по меню с помощью клавиш перемещения курсора, выбираем нужный пункт и раскрываем его, нажав на клавишу **Enter**. Затем в подменю выбираем нужную команду и подаем ее нажатием клавиши **Enter**. При выборе пунктов меню и подменю можно использовать «горячие клавиши», нажимая на клавиши, соответствующие ярко подсвеченным буквам в названиях пунктов меню и подменю. Все эти действия также можно выполнять с помощью манипулятора мышь.

Отладка программы

Отладка набранной программы начинается с ее запуска. Отладка заключается в выявлении и исправлении ошибок, допущенных при составлении программы и ее вводе в память компьютера.

Как правило, работа программ начинается с запроса исходных данных. Многократные вводы с клавиатуры исходных данных при отладке программы раздражают, приводят к затратам времени. Можно на время отладки исключить запросы, а ввод данных организовать добавочной строкой с операторами присваивания.

Важным моментом отладки является выбор значений исходных данных, позволяющих по возможности всесторонне проверить работу алгоритма и реализующей его программы. Полезно в качестве контрольного теста при первых запусках программы брать данные, при которых результат решения задачи известен заранее.

Главными помощниками при отладке программ служат эффективные средства отладки, предоставляемые пользователю средой языка Бейсик. Эти средства перечислены в Приложениях 1, 2, 3.

Сохранение программы

Для сохранения программы в памяти компьютера следует раскрыть пункт главного меню **File**. Описание функций подпунктов этого меню дано в Приложении 1. Сохраняемая программа в памяти компьютера должна иметь имя и выступать уже в качестве файла. Для этого в меню **File** следует выбрать подпункт **Save As...** (сохранить как...). На экране появится приглашение записать имя программы, которое с автоматическим добавлением расширения **.bas** будет в дальнейшем являться именем файла. Имя может содержать до восьми символов. После записи имени следует нажать **Enter** или щелкнуть мышью на команде **Ok**. Программа будет сохранена в памяти компьютера. Перед записью следует выбрать место, куда следует программу записать.

Часто выполняется работа с программой, вызванной из памяти, т.е. уже имеющей имя. В этом случае для сохранения ее под этим же именем следует использовать подпункт **Save** (сохранить). При желании изменить имя программы после ее, например, модернизации нужно снова использовать пункт подменю **Save As...**

Вызов программы из памяти

Вызов из памяти файла с программой для исполнения, доработки или модернизации выполняется по команде **Open** (открыть) пункта главного меню **File**. Команда **Open** раскрывает каталог с именами файлов. Вход в каталог можно выполнить клавишей **Tab** или с помощью мыши. Выбрав требуемый файл клавишами перемещения курсора или перемещением указателя мыши, нужно нажать на **Enter** или дважды на кнопку мыши.

Если при вызове в среду Бейсика из памяти новой программы в окне редактирования находилась еще не записанная в память старая программа, то среда спросит у пользователя: нужно ли ее сохранять (**Yes**) или нет (**No**). Поданная команда (**Y**) запишет старую программу в память под ее именем, после чего в окно редактирования будет из памяти записана новая программа.

4.5. Этапы решения задач на компьютере

На рис. 4.2 приведен вариант обобщенной блок-схемы последовательности действий при решении задачи на компьютере.

Первые три блока блок-схемы относятся к решению вопросов алгоритмизации, остальные — к программированию.

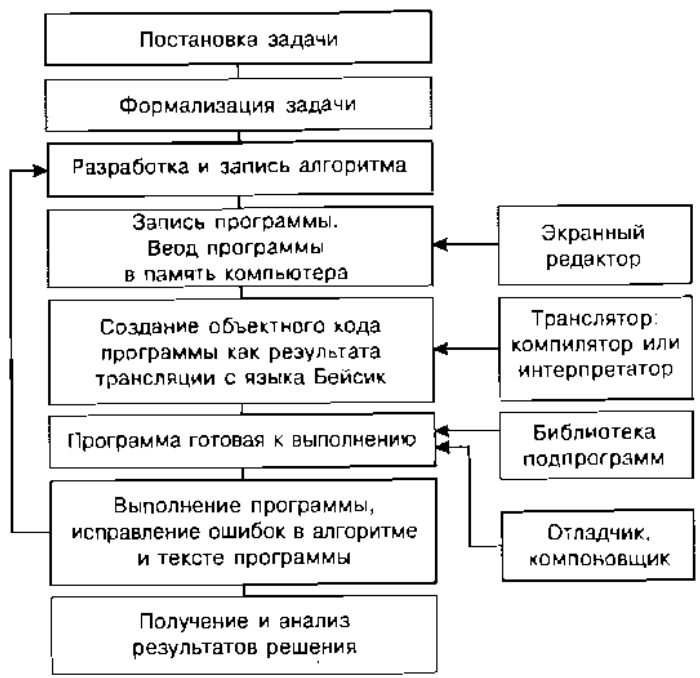

Рис. 4.2. *Блок-схема этапов решения задач на компьютере*