И. Г. Ахметов

Программирование для студентов и школьников на примере Small Basic

БАЗОВЫЕ ПОНЯТИЯ ПРОГРАММИРОВАНИЯ

ПРОСТОЙ ОБЪЕКТНО-ОРИЕНТИРОВАННЫЙ ЯЗЫК

ВОЗМОЖНОСТЬ СОЗДАНИЯ ИНТЕРАКТИВНЫХ ПРОГРАММ

СВОБОДНО РАСПРОСТРАНЯЕМАЯ РУСИФИЦИРОВАННАЯ СРЕДА ПРОГРАММИРОВАНИЯ

ИНФОРМАТИКА И ИНФОРМАЦИОННО-КОММУНИКАЦИОННЫЕ ТЕХНОЛОГИИ

Ильдар Ахметов

Программирование для студентов и школьников на примере Small Basic

Санкт-Петербург
«БХВ-Петербург»
2012

УДК 681.3.068(07)
ББК 32.973.26-018.1я7
А95

Ахметов И. Г.

А95 Программирование для студентов и школьников на примере Small Basic. — СПб.: БХВ-Петербург, 2012. — 160 с.: ил. — (ИиИКТ)

ISBN 978-5-9775-0785-1

Книга предназначена для начинающих программировать школьников и студентов. Материал излагается доступным языком на примерах из повседневной жизни. Раскрыты основные определения: алгоритм, программа, программирование. Рассмотрены базовые понятия языков программирования: объекты, переменные, присваивание, типы данных, ввод/вывод. Разобрана работа условных операторов, циклов, обработка одномерных и двухмерных массивов, математические функции и функции работы со строками. Описывается работа с графикой, анимация, обработка событий. Материал излагается на примере объектно-ориентированного языка свободно распространяемой русифицированной среды Small Basic. В каждом разделе имеются задания для самостоятельного решения.

Для образовательных учреждений

УДК 681.3.068(07)
ББК 32.973.26-018.1я7

Группа подготовки издания:

Главный редактор	*Екатерина Кондукова*
Зам. главного редактора	*Людмила Еремеевская*
Зав. редакцией	*Григорий Добин*
Редактор	*Анна Кузьмина*
Компьютерная верстка	*Натальи Караваевой*
Корректор	*Наталия Першакова*
Дизайн серии	*Инны Тачиной*
Оформление обложки	*Марины Дамбиевой*
Зав. производством	*Николай Тверских*

Подписано в печать 30.12.11.
Формат 60×90 1/16. Печать офсетная. Усл. печ. л. 10.
Тираж 1500 экз. Заказ №
"БХВ-Петербург", 190005, Санкт-Петербург, Измайловский пр., 29.

Отпечатано с готовых диапозитивов
в ГУП "Типография "Наука"
199034, Санкт-Петербург, 9 линия, 12

ISBN 978-5-9775-0785-1

Оглавление

Введение **1**

Глава 1. Знакомимся с языком **3**

Что такое программирование 4
Привет, мир! 6
Основы 7
  Объекты 7
  Переменные 8
  Типы данных 11
  Ввод и вывод 13
  Математические функции 16

Глава 2. Усложняем код **19**

Условный оператор 20
Операторы сравнения 24
Логические операторы 25
Циклы 28
  Цикл *For* ("для каждого") 29
  Цикл *While* ("до тех пор, пока истинно") 32
Массивы 40
Двумерные массивы 54
Работа со строками 64
Стек 74
Файлы 80
  Работа с файлом 80
  Работа с файловой системой 83

Глава 3. Совершенствуем интерфейс 87

Графика 88
Подпрограммы 100
Обработка событий 105
Движение фигур 113
Элементы интерфейса 120
Игра 124
Другие объекты 135
Объект *Clock* 135
Объект *Desktop* 136
Объект *Dictionary* 137
Объект *Flickr* 137
Объект *ImageList* 138
Объект *Network* 138
Объект *Program* 139
Объект *Sound* 140
Объект *Turtle* 141

Напутствие 143

Приложение. Цвета Small Basic 145

Предметный указатель 151

Введение

Итак, вы решили заняться программированием. Но с чего начать? Языков программирования много, и растеряться в море их разнообразия не мудрено. Да и просто изучение синтаксиса какого-либо языка еще не сделает вас программистами. С одной стороны, надо научиться строить алгоритмы, которые, как известно, представляют собой наборы инструкций, описывающих порядок действий для достижения результата за конечное время. А с другой стороны, хочется увидеть результат, причем не на бумаге, а на экране монитора. Поэтому в качестве очень простого в освоении языка с несложной и дружественной средой программирования, на базе которого вы постигнете принципы алгоритмизации и, разумеется, научитесь программировать, и был выбран Small Basic.

А дальше что? Начнем, пожалуй, с установки.

В Интернете существует "Русскоязычное сообщество для начинающих программистов". Это сайт, посвященный Small Basic. Проект поддерживается компанией Microsoft. Найти сообщество очень просто: в любой поисковой системе наберите запрос "Small Basic", и вы увидите результат с веб-адресом **www.smallbasic.ru**. Вам туда!

Отыскать ссылки, позволяющие скачать Small Basic с сайта этого проекта, совершенно легко: слева в окне обозревателя есть ссылка **Скачать**, а также справа среди прочих вкладок присутствует вкладка **СКАЧАТЬ SMALL BASIC**.

На открывшейся странице перечислены системные требования для установки и находится ссылка **Скачать Small Basic 1.0**. Щелчок по этой ссылке открывает окно загрузки дистрибутива

Small Basic. Скачайте его на свой компьютер, а затем запустите установку двойным щелчком по файлу SmallBasic.msi. Скорее всего, вы не первый раз устанавливаете программы, поэтому справитесь с этой задачей и без наших комментариев.

После установки в главном меню Windows появится группа **Small Basic**, из которой и запускается среда разработки.

Хочется добавить, что на сайте сообщества есть много полезной информации, в том числе документация, существует форум и публикуются интересные коды программ. Когда вы более-менее освоитесь с языком Small Basic, посмотрите материалы на сайте.

Ну, а мы начинаем...

Глава 1

Знакомимся с языком

Что такое программирование

Давайте для начала определимся, чем будем заниматься. А будем мы писать программы. Так что с определением слова *"программирование"*, думаю, все должно быть ясно. Как, например, есть еда — и мы ее едим, т. е. уничтожаем. А тут наоборот — мы программируем, т. е. создаем (пишем, придумываем) программы.

Что же такое *программа*? Слово вроде вполне знакомое. Есть программа передач на следующую неделю, программа развития сельского хозяйства России, школьная программа пятого класса. Что общего у этих программ? То, что они задают какой-то план действий, определяют, как и что будет происходить в будущем. Здесь есть, заметим, важное отличие программы от плана. В плане мы предусматриваем, что нечто будет происходить определенным образом, потом делаем и смотрим, что в итоге получилось — насколько близко к плану. В случае с программой такого нет. Программа — понятие очень четкое. Пусть только попробуют изменить внезапно программу передач или не показать передачу "Спокойной ночи, малыши" в нужное время!

Так же и компьютерная программа — штука очень четкая. Программист говорит железной машине, что нужно делать, и железная машина делает. Именно так, как сказал программист. Машина, хоть и умная, но совершенно лишена инициативы. Она как слишком дрессированная собака. Выполняет все команды — "Сидеть", "Лежать", "Дай лапу" — но если на хозяина кто-то нападает, будет сидеть и смотреть, пока не услышит команду "Фас".

> ***Кстати***
>
> К счастью, можно предусмотреть такой случай и заранее проинструктировать собаку: "Если на хозяина напали, надо кусать". Это можно сделать с помощью условного оператора или обработки событий.

Для того чтобы написать программу, нужно сначала придумать *алгоритм*. Алгоритм — это последовательность действий, которые нужно сделать. На самом деле, мы постоянно пользуемся алгоритмами — просто не задумываемся об этом. Вот, напри-

мер, решили мы сварить картошку. Для этого есть четкий порядок действий. Что-то вроде такого:

1. Почистить картошку.
2. Налить воду в кастрюлю.
3. Поставить кастрюлю на плиту.
4. Добавить соль.
5. Включить плиту.
6. Положить почищенную картошку в воду.
7. Варить картошку, пока она не станет мягкой.
8. Выключить плиту.
9. Снять кастрюлю.
10. Слить воду.

Вот это и есть алгоритм. Видите, как все просто — вроде бы всего-навсего сварили картошку, а на самом деле использовали при этом алгоритм!

Программа пишется на основе алгоритма с использованием какого-нибудь *языка программирования*, который понимает ком-

пьютер. Языков программирования много. Люди говорят на русском, английском языке, иврите или африкаансе, а для машин есть языки C, Pascal, Basic, Java, PHP, Perl и т. д. Некоторые похожи друг на друга, некоторые — совсем ни на что не похожи. И точно так же, как человеку совсем не обязательно (да и невозможно!) говорить на языке австралийских аборигенов и при этом в совершенстве владеть эсперанто, так и программист вовсе не обязан знать десятки языков программирования. Достаточно овладеть хотя бы одним (а профессионалу — тремя-пятью).

Мы с вами будем разбираться с языком программирования Small Basic. Это вариант известного языка Basic. Small Basic — очень простой, но при этом современный язык.

Привет, мир!

Есть такая традиция. Когда кто-то начинает изучать программирование, то самая первая программа — это всегда программа "Hello, world", или "Привет, мир". Проще ничего не бывает — программа просто выводит на экран этот текст. Можно вывести и что-нибудь другое, конечно, но не будем нарушать традиций.

> ***Кстати***
>
> В одной книге автор предлагал в первой программе выводить фразу "Наше вам с кисточкой".

Итак, вот она — ваша первая программа:

```
TextWindow.WriteLine("Привет, мир!")
```

Попробуйте ввести эту программу в открытом окне Small Basic и запустить ее (запускается программа большим синим треугольником с подписью "Запуск"). Получилось? Видите черное окно, в котором написано "Привет, мир!"? Тогда поздравляю — первую программу вы написали.

> ***Кстати***
>
> Кроме приветствия, в черном окне вы увидите еще одну строку — "Press any key to continue...". Разумеется, вы знаете, как она переводится. Ну а если забыли, то напомним: "Нажмите любую клавишу для продолжения...".

Основы

Объекты

Что же значит эта строка с непонятными словами? Здесь все довольно просто. `TextWindow` — это объект "окно с текстом", то самое черное окошко, в котором можно отображать текст.

Объект — это нечто, чем вы можете пользоваться. У каждого объекта есть свойства и методы. *Свойство* объекта — это какая-то его характеристика, а *метод* объекта — это то, что объект может делать.

Например, у вас дома есть микроволновка. Это — объект. Свойства микроволновки — цвет (белый, черный, красный, синий в крапинку), объем в литрах (20, 30, 130), название (Samsung, Electrolux, "Лысьва"). Методы микроволновки — разогреть, разморозить, поджарить на гриле.

Так же и здесь. `TextWindow` — объект, а `WriteLine` — его метод, который означает "вывести строку". Точка используется как разделитель. Метод `WriteLine` принимает *параметр* — он же должен знать, что именно надо вывести в черное окно! Параметры всегда указываются в скобках.

Давайте теперь усложним программу. Например, вот так:

```
TextWindow.ForegroundColor = "Red"
TextWindow.WriteLine("Привет, мир!")
```

Теперь "Привет, мир!" написано в черном окне красным цветом — и это все благодаря первой строке. `ForegroundColor` — свойство объекта `TextWindow`, которое обозначает "цвет текста". Мы хотим, чтобы цвет был красным, поэтому и присваиваем этому свойству значение `"Red"` — "красный". Можете попробовать теперь раскрасить строку в другие цвета — синий (`"Blue"`), желтый (`"Yellow"`), зеленый (`"Green"`) и т. д.

Теперь, когда первая программа (из целых двух строк кода!) готова, давайте немного разберемся с теорией.

Переменные

Переменная — самое важное понятие. Представьте себе деревянный ящик. Помните, в каком ящике нашли Чебурашку? Да, вот такой деревянный ящик. Это и есть *переменная*. На ящик приклеена бумажка с названием — это *имя переменной*. В ящик можно что-то класть, а потом вытаскивать — обычно кладут числа и буквы (ну и еще всякие штуки, о которых вы узнаете позже).

Например, давайте создадим переменную с именем `a` и положим туда число 17:

```
a = 17
```

Это значит "положить число 17 в переменную `a`", т. е. "положить число 17 в ящик, на котором написано `a`". Знак равенства в этом случае называется *оператором присваивания*, потому что с его помощью *присваивают* значения переменным. Теперь в ящике с надписью `a` лежит число 17. Давайте теперь напишем вот такую конструкцию:

```
a = a + 5
```

Это не уравнение! Это — присваивание. Вот что значит эта строка:

1. Взять значение переменной `a`.
2. Прибавить к нему 5.
3. Положить новое значение в переменную `a`, стерев из нее предыдущее.

У оператора присваивания есть две части — левая и правая. Левая часть находится слева, а правая — да, вы правильно догадались. В левой части обычно пишется переменная, которая будет меняться. А в правой — то, как она вычисляется. То есть оператор присваивания работает *справа налево*, и только так!

Давайте посмотрим еще раз, что происходит. Итак:

```
a = a + 5
```

1. Начинаем обрабатывать правую часть. Открываем ящик `a`. Там находится число 17, которое мы положили туда раньше. Берем это число из ящика. При этом из ящика вытаскивается только копия значения — другая копия остается лежать в ящике!
2. Теперь в правой части вместо имени `a` подставится число 17, которое мы взяли из ящика `a`. Считаем: 17 плюс 5 — будет 22. Правая часть вычислена, но число 22 еще никуда не записано — в ящике `a` все еще лежит 17.
3. А вот теперь работает присваивание. То есть в ящик `a` (который написан слева) кладется число 22. Старое число 17 из ящика исчезает.

> ***Кстати***
>
> Поэтому и название такое — переменная, потому что значения меняются.

Давайте приведем пример посложнее:

```
b = a * 2 + 10
```

Это значит "взять число 17 из ящика `a`, умножить его на 2, прибавить к нему 10, а затем положить в ящик `b`". Обратите внимание — ящика `b` вообще не существовало, а теперь он появился, и в нем лежит число 44 (т. е. 17 × 2 + 10).

А вот еще один пример:

```
a = 5
b = 8
a = a + b
b = b - 1
a = b
```

В этой программе пять строк. Посмотрим, что происходит (табл. 1.1).

Таблица 1.1

Действие	**a**	**b**
В самом начале	Ничто	Ничто
После 1-й строки (`a = 5`)	5	Ничто
После 2-й строки (`b = 8`)	5	8
После 3-й строки (`a = a + b`)	13	8
После 4-й строки (`b = b - 1`)	13	7
После 5-й строки (`a = b`)	7	7

Посмотрите еще раз: переменные, которые находятся в левой части оператора присваивания, не меняются. Меняются только те, что стоят справа.

И еще один важный момент. Обратите внимание на последнюю строку (`a = b`). Она не делает переменные `a` и `b` синонимами, т. е. не перевешивает таблички с названиями с одного ящика на другой. Просто два разных ящика содержат одинаковое значение.

Типы данных

Переменные могут быть разных типов. Тут снова хорошо работает сравнение с ящиками. Представьте себе ящик для апельсинов, футляр для гитары, коробочку для обручального кольца. Все они нужны для того, чтобы что-то в них класть. Но гитара не влезет в коробочку для кольца, а кольцо затеряется в ящике из-под апельсинов.

Кстати

Представьте себе глаза той девушки, которой преподнесут кольцо в деревянном ящике.

Так же и с переменными — не все они одинаковые, отличает их *тип данных*.

Часто типы данных вызывают сложности при изучении программирования. Зачастую языки имеют очень много разных типов, и разобраться в них довольно трудно.

В языке Small Basic типов данных всего два:

- число;
- строка.

Что такое число, всем понятно. Примеры чисел: 2, –17, 0, 3,14. Числа можно складывать, умножать, вычитать, делить. Над ними можно совершать все математические действия — вычислять логарифмы, синусы, косинусы и т. д.

Строка — это последовательность символов. Примеры строк: "собака", "Мама мыла раму", "Съешь еще этих мягких французских булок, да выпей же чаю", "A8bfhGGT71bHtd71vfa". Строки можно склеивать и делить на части, в них можно искать символы и заменять их другими.

Типы в Small Basic задаются косвенно. То есть вам не нужно описывать типы, как во многих других языках программирования. Вы просто пишете:

```
k = 5
s = "It is raining cats and dogs"
```

И Small Basic понимает, что тип переменной `k` — число, а `s` — строка.

Интересный момент есть с оператором `+`. Для чисел он означает сложение, а для строк — склеивание. Но если "сложить" число со строкой — они тоже будут склеены (табл. 1.2).

Таблица 1.2

Выражение	Результат
`2 + 2`	`4`
`"око" + "рок"`	`"окорок"`
`"абв" + 10`	`"абв10"`

Ввод и вывод

Каждая программа должна делать что-то полезное. Обычно программы берут какие-нибудь данные *(входные данные)*, обрабатывают их и предоставляют результат *(выходные данные)*.

Вот, например, приходит человек на вокзал, подходит к справочному окну и спрашивает: "Сколько стоит билет на поезд до Таганрога в плацкартный вагон?" Ему отвечают: "2312 рублей". Здесь входные данные — название города (Таганрог) и тип вагона (плацкартный). Выходные данные — цена. Заметьте, что значение на выходе напрямую зависит от значений входных параметров.

Кстати

Мы-то с вами, конечно, знаем, что вместо справочной службы есть **www.rzd.ru**.

Соответственно, программа должна как-то получать входные данные и выдавать выходные. Для этого есть специальные *операторы ввода* и *вывода*. Эти операторы — методы объекта `TextWindow` (ведь работают они внутри "черного окна").

Есть два оператора ввода, в зависимости от типа данных (табл. 1.3).

Таблица 1.3

Метод	Описание
`TextWindow.Read()`	Ввод строки
`TextWindow.ReadNumber()`	Ввод числа

В конце оператора ввода всегда ставятся пустые скобки.

Операторы вывода не зависят от типа данных, но их тоже два (табл. 1.4).

Таблица 1.4

Метод	Описание
`TextWindow.Write(data)`	Обычный вывод
`TextWindow.WriteLine(data)`	Вывод с переходом на следующую строку экрана

Разница между двумя вариантами вывода представлена в табл. 1.5.

Таблица 1.5

Пример использования метода	Результат
`TextWindow.Write("око")` `TextWindow.Write("рок")`	`окорок`
`TextWindow.WriteLine("око")` `TextWindow.WriteLine("рок")`	`око` `рок`

Вот пример простой программы с вводом и выводом:

```
a = TextWindow.ReadNumber()
b = TextWindow.ReadNumber()
x = a + b
TextWindow.WriteLine(x)
```

Первая строка — ввод переменной `a`. Посмотрите: это такой же оператор присваивания, который мы уже знаем. В левой части — переменная `a`, в которую будем записывать значение. А в правой части — оператор ввода `ReadNumber()`, который будет ждать, какое число мы введем с клавиатуры. То есть в черном окне начинает мигать курсор, мы вводим какое-нибудь число, нажимаем клавишу <Enter> — и это число кладется в переменную `a`. Точно так же работает вторая строка — следующее введенное число окажется в переменной `b`. Третья строка складывает значения переменных `a` и `b`, а результат записывает в переменную `x`. Ну и, последняя строка выводит значение `x` на экран.

Давайте теперь снабдим эту программу подсказками, чтобы было понятно, что за волшебную махинацию она производит:

```
TextWindow.Write("Введите первое число: ")
a = TextWindow.ReadNumber()
TextWindow.Write("Введите второе число: ")
b = TextWindow.ReadNumber()
x = a + b
TextWindow.WriteLine("Сумма чисел равна: " + x)
```

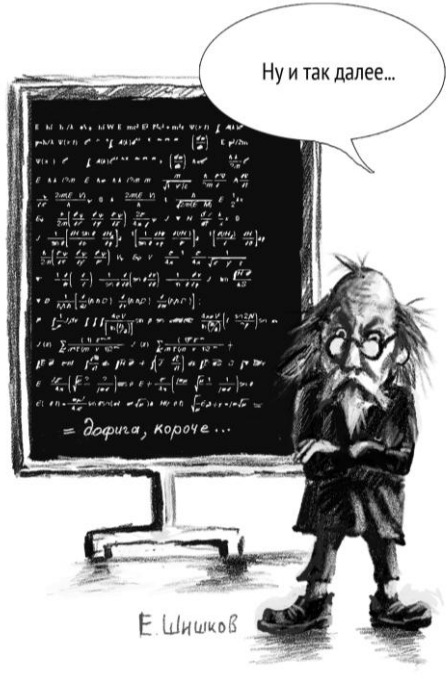
Теперь программа ведет себя гораздо вежливее. Она просит ввести первое число и призывно мигает курсором. Получив число, так же просит ввести второе. Затем считает сумму и пишет, чему она равна. Обратите внимание на знак "+" в последней строке — он склеивает фразу "Сумма чисел равна: " со значением переменной `x`, поэтому получается что-то вроде "Сумма чисел равна: 4".

Задания

1. Вычислите сумму не двух, а трех чисел.
2. А теперь произведение двух (или трех) чисел.
3. Напишите программу для перевода дней в часы (один день — 24 часа).
4. Аналогично напишите программу для конвертации цены из долларов в рубли (курс посмотрите в Интернете).
5. Даны стороны прямоугольника. Нужно найти его площадь.
6. Возведите число в восьмую степень за три операции умножения.

Математические функции

Куда ж без математики! Часто нужно что-то вычислять — синусы, косинусы, логарифмы... Все эти вычисления реализуются с помощью математических функций. В языке Small Basic эти функции находятся в объекте `Math`. Строго говоря, это методы объекта `Math`.



Вот такие математические функции есть в Small Basic (табл. 1.6).

Таблица 1.6

Метод	**Описание**
`Math.Abs(number)`	Модуль
`Math.ArcCos(number)`	Арккосинус
`Math.ArcSin(number)`	Арксинус
`Math.ArcTan(number)`	Арктангенс
`Math.Ceiling(number)`	Округление вверх
`Math.Cos(number)`	Косинус
`Math.Floor(number)`	Округление до ближайшего наименьшего целого
`Math.GetDegrees(angle)`	Перевод числа из радиан в градусы
`Math.GetRadians(angle)`	Перевод числа из градусов в радианы
`Math.GetRandomNumber(maxNumber)`	Случайное число в интервале от 1 до `maxNumber`
`Math.Log(number)`	Десятичный логарифм
`Math.Max(number1,number2)`	Максимум двух чисел
`Math.Min(number1,number2)`	Минимум двух чисел
`Math.NaturalLog(number)`	Натуральный логарифм
`Math.Pi()`	Число π
`Math.Power(baseNumber,exponent)`	Возведение числа `baseNumber` в степень `exponent`
`Math.Remainder(dividend,divisor)`	Остаток от деления
`Math.Round(number)`	Обычное округление
`Math.Sin(number)`	Синус
`Math.SquareRoot(number)`	Квадратный корень
`Math.Tan(number)`	Тангенс

Вот несколько примеров записи математических формул на Small Basic:

$a = x^3$
```
a = Math.Power(x,3)
```

$b = \sin x$
```
b = Math.Sin(x)
```

$c = \cos x^2$
```
c = Math.Cos(Math.Power(x,2))
```
или
```
c = Math.Cos(x*x)
```

$d = \mathrm{tg}^2 x$
```
d = Math.Power(Math.Tan(x),2)
```

$a = \ln\ x + y$
```
a = Math.NaturalLog(x+y)
```

$b = \sqrt{x}$
```
b = Math.SquareRoot(x)
```
или
```
b = Math.Power(x,1/2)
```

$c = \sqrt[3]{x}$
```
c = Math.Power(x,1/3)
```

$d = \dfrac{\sin x^2}{\ln x} + 2\pi$
```
d = Math.Sin(Math.Power(x,2)) /
Math.NaturalLog(x) + 2 * Math.Pi
```

Задания

1. Даны два катета прямоугольного треугольника. Найдите его гипотенузу (это теорема Пифагора, разумеется).
2. Даны длины трех сторон треугольника (уже произвольного). Найдите его площадь (по формуле Герона).
3. Найдите корни квадратного уравнения, если известны его коэффициенты *a*, *b*, *c*. Формулы — стандартные, через дискриминант.
4. Дано трехзначное число. Нужно найти его первую цифру (вам помогут деление и округление).
5. Дано любое целое число. Найдите его последнюю цифру (с помощью остатка от деления).
6. Вычислите вот такое сложное выражение: $x = \dfrac{b^5 - a + \sqrt[3]{bc^2}}{\sqrt{ab}}$.
7. И вот такое: $y = \sin\ 3\pi a + bc\ - \cos a^2$.

Глава 2

Усложняем код

Условный оператор

В жизни постоянно возникают ситуации, которые требуют выбора. Например, собираетесь вы утром выходить на улицу и думаете: "Что же надеть?" Прежде всего, это зависит от погоды. Если льет дождь, то желательно надеть резиновые сапоги и взять зонтик. Если валит снег, то нужны валенки и шапка-ушанка. Ну а если нет ни снега, ни дождя — тогда просто надеваем какую-нибудь обычную одежду.

Смысл условного оператора именно в этом слове — *"если"*. В языке Small Basic (да и в большинстве других языков) это слово пишется просто — *If*.

Если условие истинно (т. е. правда), то оно примет значение "Истина" или `"True"`. Если условие ложно (т. е. совсем даже не правда), то это будет "Ложь" — `"False"` (обратите внимание — кавычки важны!). То есть условие "Париж — столица Франции" истинно (`"True"`), а "Рим — столица Камбоджи" — ложно (`"False"`).

В самом простом варианте условный оператор записывается так:

```
If <условие> Then
  <действия>
EndIf
<продолжение программы>
```

Работает он следующим образом:

1. Вычисляется условие.
2. Если оно истинно, то выполняются действия, а затем — продолжение программы.
3. Если оно ложно, то действия игнорируются, а сразу выполняется продолжение.

Давайте рассмотрим пример. Тот же самый, про погоду, но пока сделаем его попроще, с единственным условием — есть дождь или нет. Запишем это условие в виде псевдопрограммы.

> ***Кстати***
>
> Мы будем пользоваться таким приемом еще не раз. Псевдокод, конечно, нельзя запускать — он нужен для того, чтобы понимать суть.

```
If идет_дождь Then
  взять_зонт
EndIf
выйти_из_дома
```

Видите: все понятно, эту программу очень легко перевести с компьютерного языка на человеческий (т. е. с Бейсика на русский). "Если идет дождь, то взять зонт".

А теперь давайте усложнять. Если условий несколько, то условный оператор записывается так:

```
If <условие 1> Then
  <действия 1>
ElseIf <условие 2> Then
  <действия 2>
...
Else
  <действия, если все условия ложны>
EndIf
```

Слово `ElseIf` (пишется слитно) означает "иначе если", а слово `Else` — просто "иначе". Только самое первое условие записывается в виде `If` (если). Все остальные условия — это уже `ElseIf`. А уж если ни одно из условий не выполнено (все оказалось ложью — "Все врут!"), то срабатывает вариант "иначе" — `Else`.

Вернемся к нашему исходному погодному примеру и запишем его в виде псевдопрограммы.

```
If идет_дождь Then
  взять_зонт
  надеть_сапоги
ElseIf идет_снег Then
  надеть_валенки
  надеть_шапку
Else
  надеть_обычную_одежду
EndIf
выйти_из_дома
```

Посмотрите внимательно на пример. На русский она переводится совершенно очевидно, достаточно перевести ключевые слова. `If` — "если", `Then` — "то", `ElseIf` — "иначе если", `Else` — "иначе".

А вот как компьютер будет разбирать эту программу:

1. Проверить, идет ли дождь.
2. Если дождь идет (условие истинно), то взять зонт и надеть сапоги. Выйти из дома (выход из условного оператора — другие условия уже не проверяются).

3. Если дождя нет (предыдущее условие ложно), то проверить, идет ли снег.
4. Если снег идет (условие истинно), то надеть валенки и шапку. Выйти из дома (выход из условного оператора).
5. Если ни дождя, ни снега нет (все условия ложны), то надеть обычную одежду. Выйти из дома.

Приведем теперь несколько работающих примеров — уже не псевдокод, а настоящие работающие программы. Попробуйте их написать и запустить в Small Basic.

```
TextWindow.Write("Введите число: ")
a = TextWindow.ReadNumber()
If a > 0 Then
  TextWindow.WriteLine("Число положительное")
EndIf
```

Эта программа просит нас ввести число и проверяет, положительное ли оно. Конструкция простейшая. Если условие `a > 0` истинно (т. е. если значение переменной `a` больше нуля), то на экран выводится строка "Число положительное".

А как же узнать, что число отрицательное или равно нулю? Давайте немного усложним программу.

```
TextWindow.Write("Введите число: ")
a = TextWindow.ReadNumber()
If a > 0 Then
  TextWindow.WriteLine("Число положительное")
ElseIf a < 0 Then
  TextWindow.WriteLine("Число отрицательное")
Else
  TextWindow.WriteLine("Ноль")
EndIf
```

Теперь сначала проверяется условие положительности — `a > 0`. Если оно истинно, то выводится фраза "Число положительное", и на этом проверки заканчиваются. Если же оно ложно, то проверяется следующее условие — `a < 0`. Если оно истинно, то выводится фраза "Число отрицательное". Если же оба условия оказа-

лись ложными, т. е. число и не больше нуля, и не меньше нуля, то остается только один вариант: это самый настоящий ноль! Соответственно, и выводится "Ноль".

А вот еще один любопытный пример. Помните нашу первую программу — "Привет, мир!"? Давайте сделаем ее более доброжелательной. Пусть она здоровается с миром по-разному в зависимости от времени суток — "Доброе утро", "Добрый день" или "Добрый вечер". Для этого воспользуемся объектом `Clock` (Часы), у которого есть свойство `Hour` (Час). `Clock.Hour` скажет нам, сколько сейчас часов.

> ***Кстати***
>
> Можно, конечно, здороваться, как в фильме "Шоу Трумана" — "Доброе утро, и, если не увидимся, добрый день и добрый вечер!"

```
If Clock.Hour < 12 Then
  TextWindow.WriteLine("Доброе утро, мир!")
ElseIf Clock.Hour < 18 Then
  TextWindow.WriteLine("Добрый день, мир!")
Else
  TextWindow.WriteLine("Добрый вечер, мир!")
EndIf
```

Посмотрите, если на часах меньше 12, то мы скажем миру "Доброе утро", если меньше шести вечера (т. е. 18), то "Добрый день", а иначе — "Добрый вечер".

Операторы сравнения

В условиях используются *операторы сравнения*, с помощью которых можно сравнивать значения. Мы уже пользовались оператором < (меньше), когда сравнивали `a` и 0.

А вот в табл. 2.1 представлен полный список операторов сравнения.

Таблица 2.1

Оператор	Описание	Пример
<	Меньше	`2 < 5`
>	Больше	`8 > 3`
<=	Меньше или равно	`3 <= 7, 7 <= 7`
>=	Больше или равно	`5 >= 1, 5 >= 5`
=	Равно	`6 = 6`
<>	Не равно	`2 <> 3`

Обратите внимание на важный момент. Символ = может означать как присваивание, так и сравнение. Все зависит от того, в каком месте программы он находится.

Например:

- `a = 2` — присваивание (кладем число 2 в переменную `a`);
- `If a = 2` — сравнение (проверяем, равно ли значение переменной `a` числу 2).

Логические операторы

С помощью *логических операторов* можно строить более сложные условия. Логических операторов в Small Basic два (табл. 2.2).

Таблица 2.2

Название	Оператор	Описание
И	`And`	Истина, если оба условия истинны
ИЛИ	`Or`	Истина, если хотя бы одно из условий истинно

Приведем пару примеров с логическими операторами.