И. Г. Ахметов

bhv®

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

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

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.

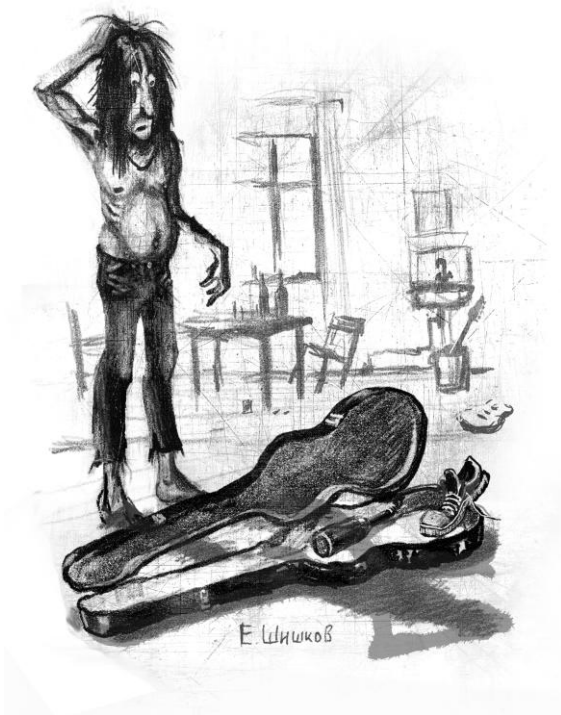
Типы данных

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`