



innopolis
UNIVERSITY

Введение в SmallBasic

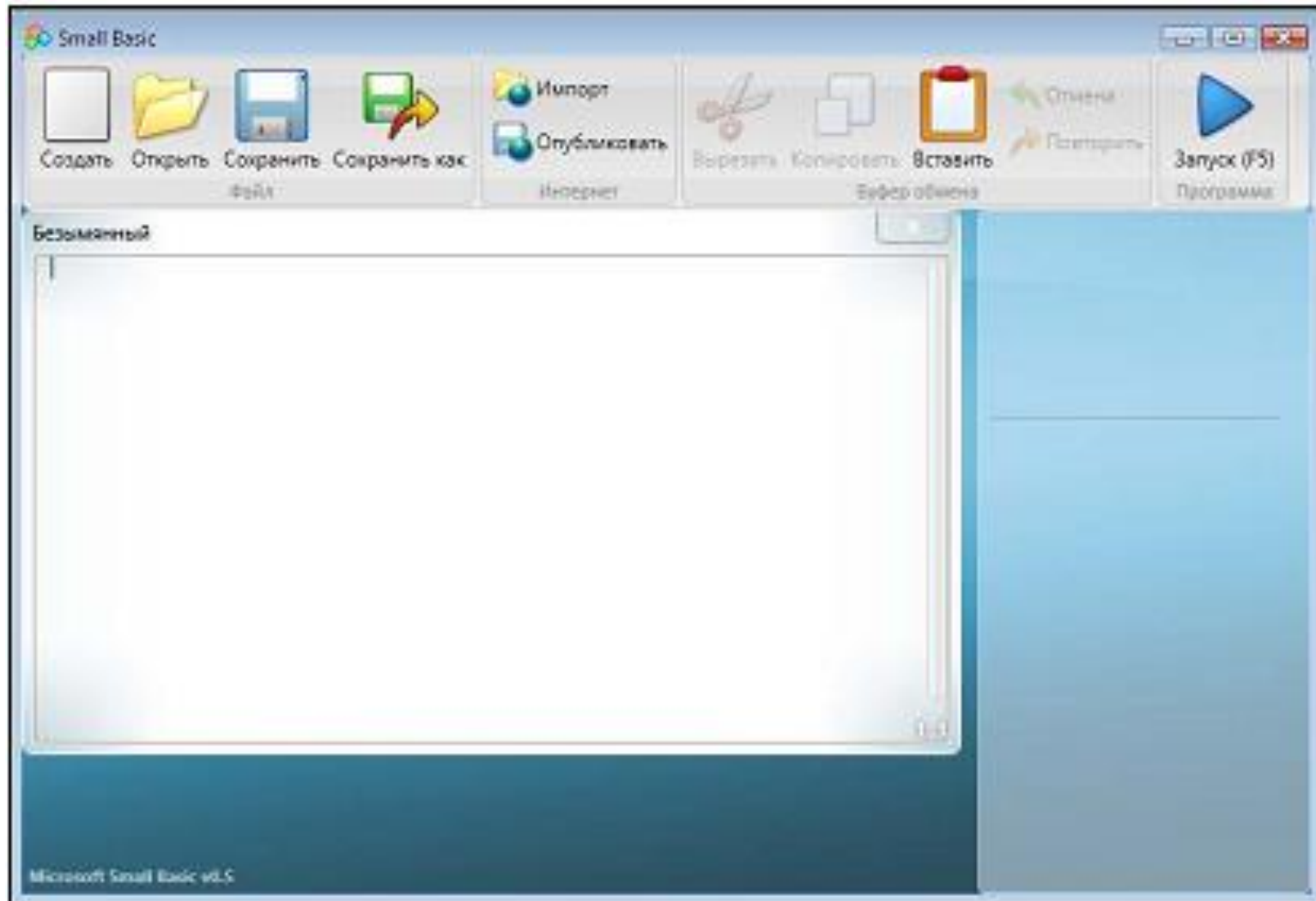
Microsoft SmallBasic

Microsoft Small Basic — язык программирования и среда разработки.

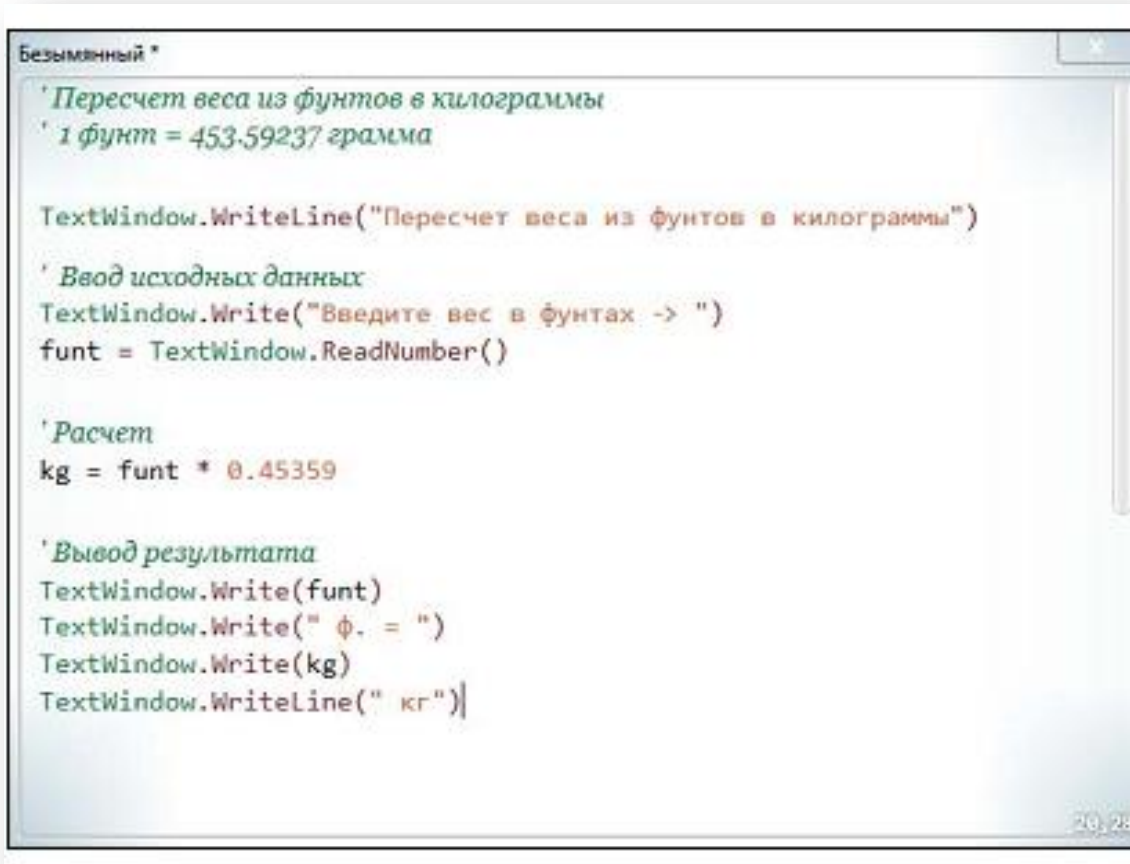
“+”

1. Очень простая среда – текстовый редактор.
2. 20 команд.
3. Встроенная контекстная подсказка.
4. Возможность расширения компонентов.

Главное окно Small Basic.



Набор текста программы



```
Безымянный *
'Пересчет веса из фунтов в килограммы
' 1 фунт = 453.59237 грамма

TextWindow.WriteLine("Пересчет веса из фунтов в килограммы")

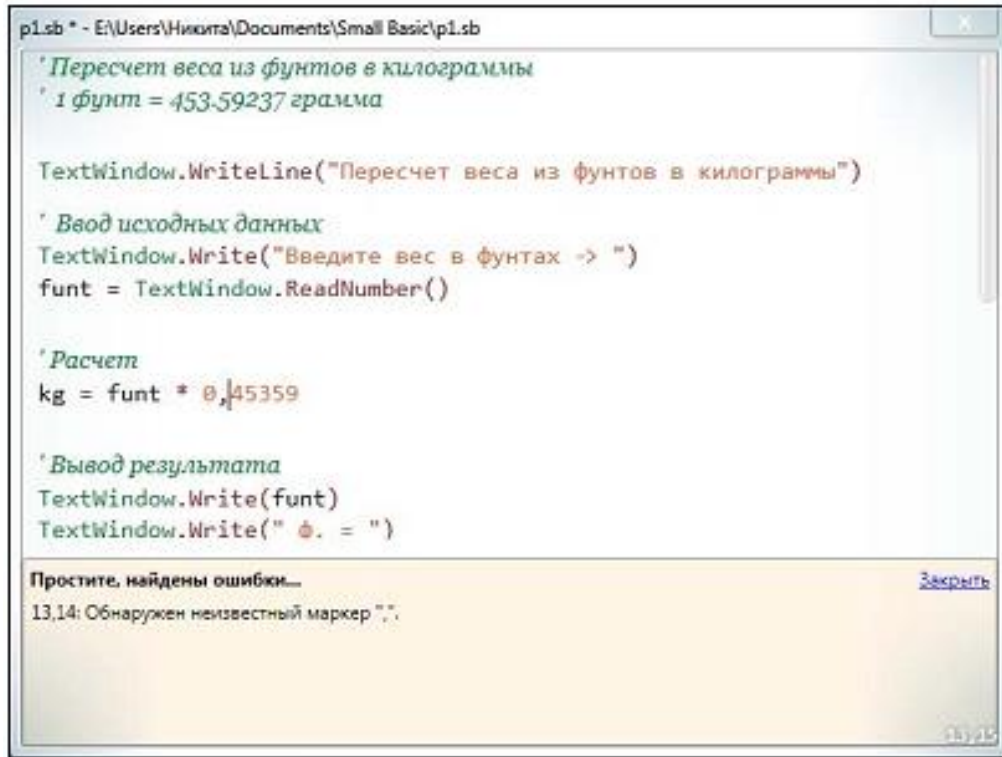
' Ввод исходных данных
TextWindow.Write("Введите вес в фунтах -> ")
funt = TextWindow.ReadNumber()

' Расчет
kg = funt * 0.45359

' Вывод результата
TextWindow.Write(funt)
TextWindow.Write(" ф. = ")
TextWindow.Write(kg)
TextWindow.WriteLine(" кг")
```

Редактор кода автоматически выделяет цветом константы (числовые и символьные) и ключевые слова языка, а также выделяет курсивом комментарии.

Отладка программы



The screenshot shows a Basic IDE window titled 'p1.sb * - E:\Users\Никита\Documents\Small Basic\p1.sb'. The code is as follows:

```
' Пересчет веса из фунтов в килограммы
' 1 фунт = 453.59237 грамма

TextWindow.WriteLine("Пересчет веса из фунтов в килограммы")

' Ввод исходных данных
TextWindow.Write("Введите вес в фунтах -> ")
funt = TextWindow.ReadNumber()

' Расчет
kg = funt * 0,45359

' Вывод результата
TextWindow.Write(funt)
TextWindow.Write(" фунт. = ")

Простите, найдены ошибки...
13,14: Обнаружен неизвестный маркер " , ".
```

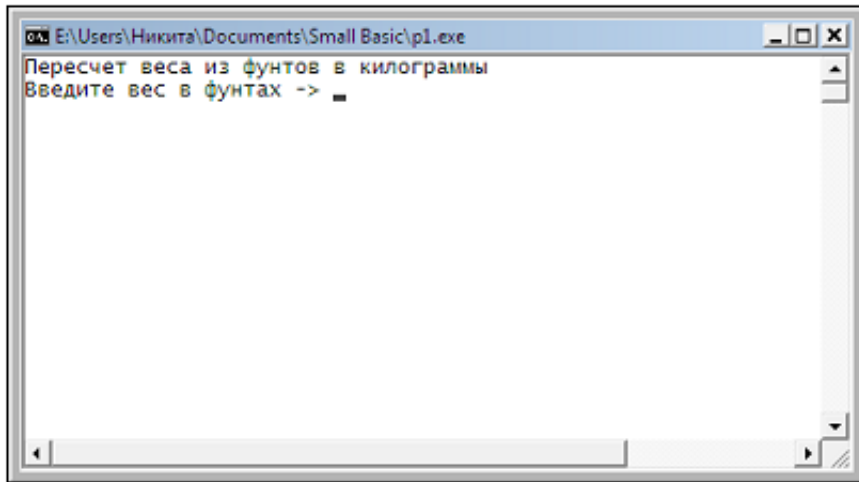
The error message at the bottom states: 'Простите, найдены ошибки... 13,14: Обнаружен неизвестный маркер " , ".' (Sorry, errors were found... 13,14: An unknown character ' , ' was detected.). A 'Закрыть' (Close) button is visible next to the error message.

Если в программе есть ошибки, то в нижней части окна редактора кода выводится сообщение об ошибке, с указанием номера строки и номера ошибочного символа.

Запуск программы



Кнопка «Запуск»



Окно программы

Для запуска программы необходимо щелкнуть по кнопке «Запуск» или нажать клавишу «F5».

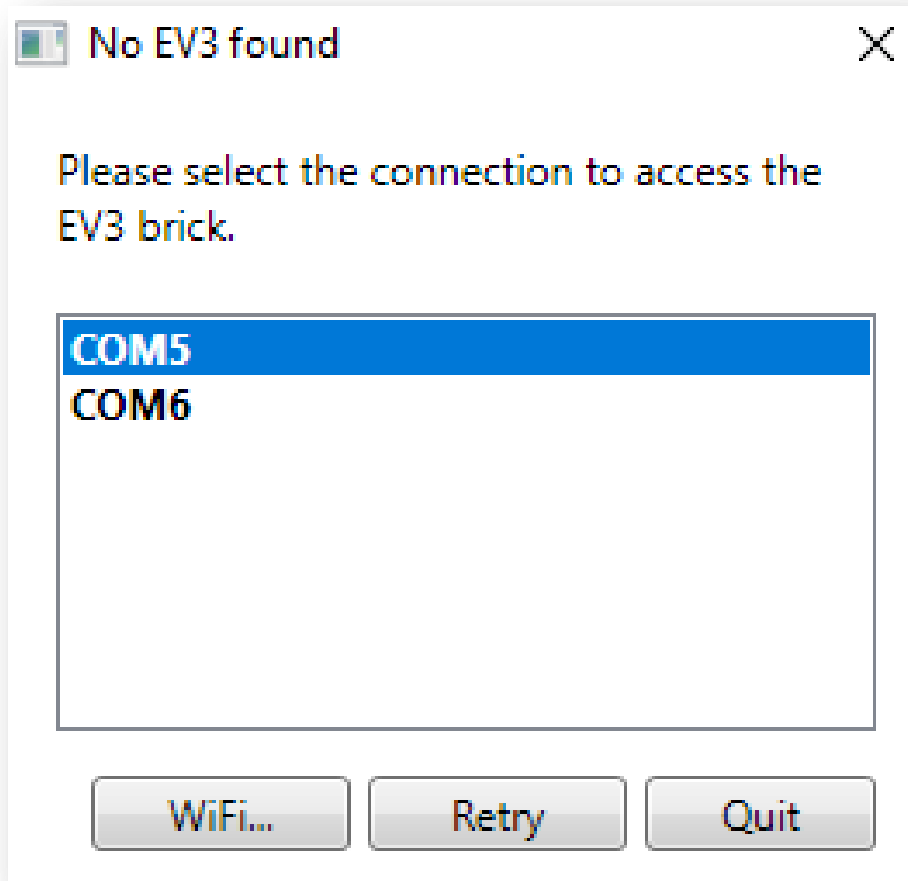
Кнопка «Запуск» находится на панели инструментов.

Если в программе нет ошибок, то появится окно программы.

Загрузка программы на блок EV3

Для загрузки программы на блок осуществляется с помощью программы EV3Explorer.

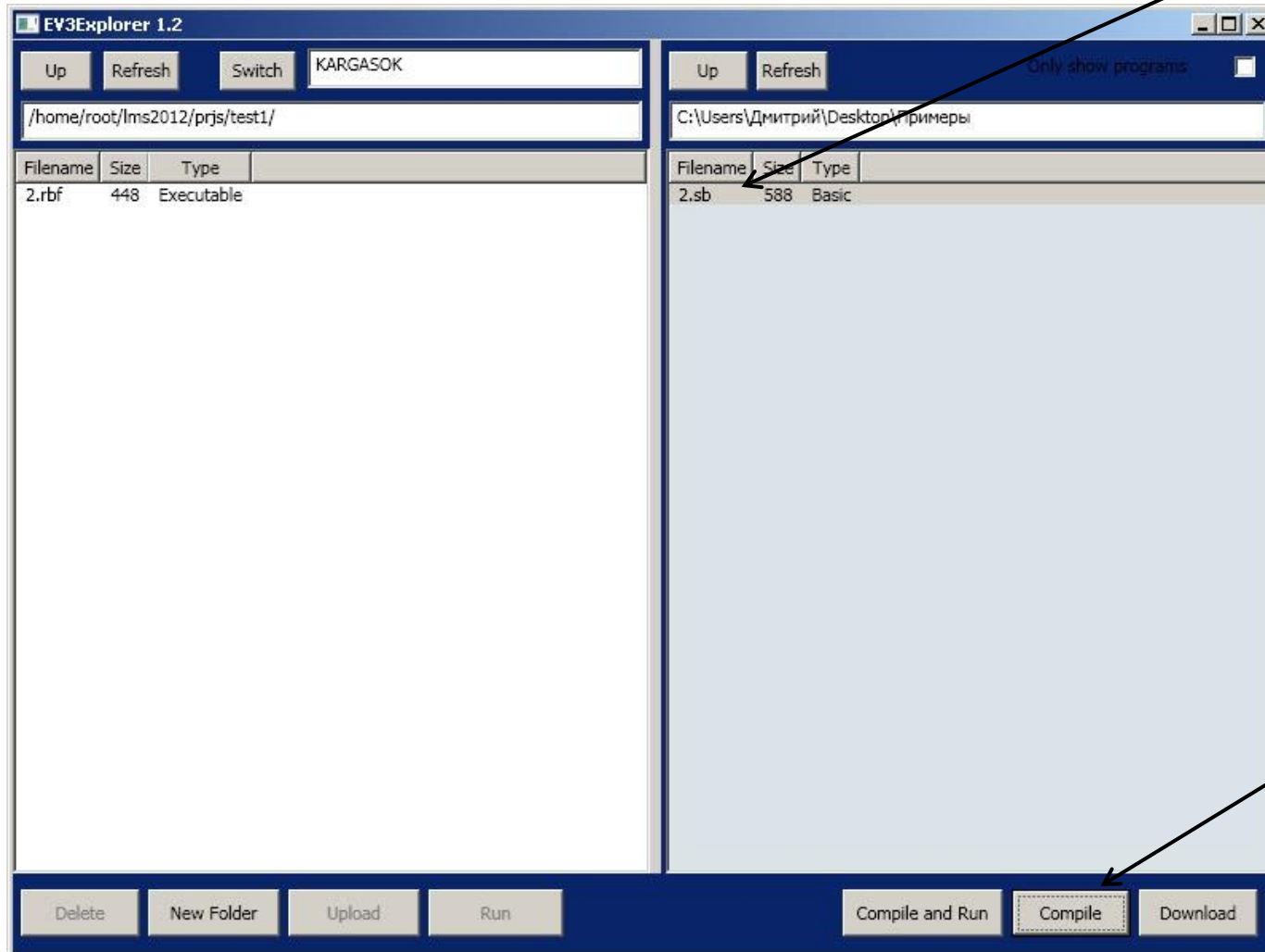
1. Необходимо установить соединение с блоком



- Соединение можно установить через USB-порт либо через Bluetooths используя Com-порт.

2.

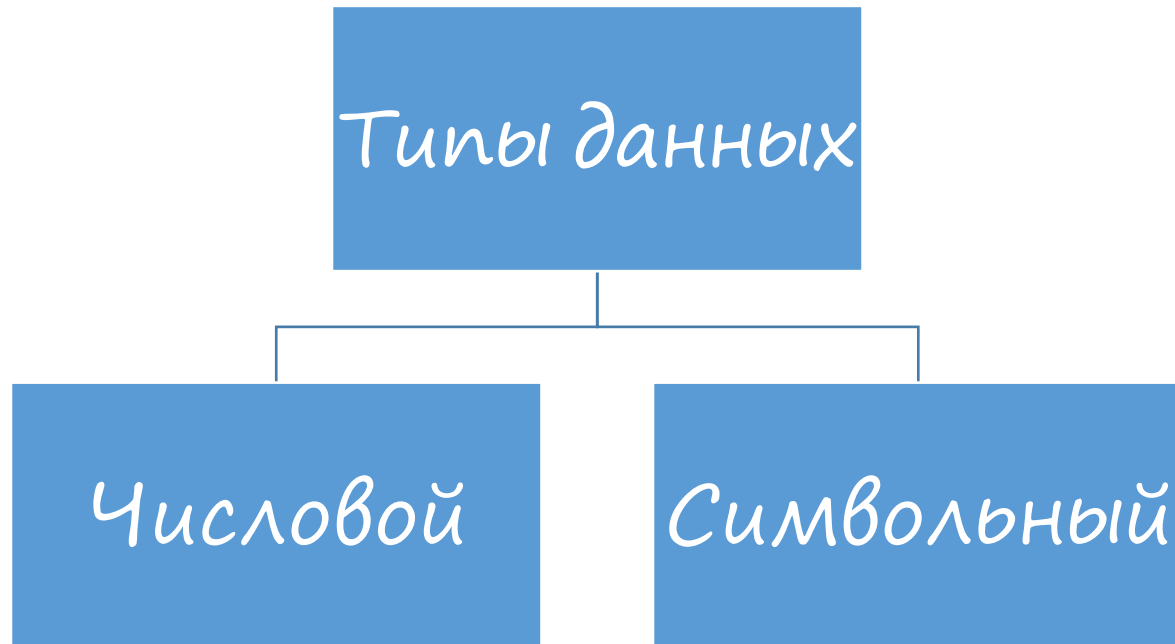
Выбрать
программу



Нажать
кнопку

Переменные. Математические и логические операции.

Типы данных



К числовому типу относятся целые и дробные числа,

к символьному типу — символы и строки символов.

Объявление переменной

В Small Basic **заранее объявлять** все переменные **не надо**.

Память для данных выделяется в момент обращения к переменной, например, при выполнении инструкции присваивания значения переменной.

В качестве имени переменной можно использовать любую последовательность из букв латинского алфавита и цифр, однако **первым символом должна быть буква**.

SmallBasic регистрозависим в названии переменных, т.е. идентификаторы (имена) **Sum** и **sum** обозначают разные переменные.

Константы

В тексте программы числовые константы записываются обычным образом. При записи дробных констант в качестве десятичного разделителя используется **точка**.

Строковые константы, для того чтобы их можно было отличить от имен переменных, в тексте программы заключаются **в кавычки**.

Пример строковых констант:
"Microsoft Small Basic"

Инструкция присваивания

В общем виде инструкция присваивания записывается так:

Переменная = Выражение

Пример:

$Sum = Cena * Kol$

$Kg = 0.495 * Funt$

$Total = 0$

Выражения

Выражение состоит из операндов и операторов. Операнды, в качестве которых могут выступать константы, переменные, а также функции — это объекты, над которыми выполняются действия.

Операторы обозначают действия, выполняемые над операндами.

Оператор	Действие
+	Сложение
-	Вычитание
*	Умножение
/	Деление

Понятие функции

Функция — это подпрограмма, обеспечивающая решение некоторой задачи и возвращающая в вызвавшую ее программу результат, который называется значением функции.

Чтобы получить значение функции, ее имя надо вызвать — указать функцию в качестве операнда выражения инструкции присваивания.

Пример:

```
r = Math.Sqrt(x)
```

```
y = L * Math.Sin(a)
```

Основные математические функции

Функция	Значение
Math.Abs(x)	Абсолютное значение x
Math.Power(a,b)	Возведение в степень b числа a
Math. SquareRoot(x)	Квадратный корень из числа x
Math. Remainder(a,b)	Остаток от деления a на b как целое значение
Math.Sin(a)	Синус угла a , заданного в радианах
Math.Cos(a)	Косинус угла a , заданного в радианах
Math.Tan(a)	Тангенс угла a , заданного в радианах
Math. Round(x)	Целое — x , округленное в соответствии с известными правилами
Math. Ceiling(x)	Целое — x , округленное "с избытком"
Math. Floor(x)	Целое, x округленное "с недостатком"

Вывод на монитор

Вывод результата на экран обеспечивают инструкции:

`TextWindow.Write`

`TextWindow.WriteLine`

В общем виде инструкция вывода сообщения записывается так:

`TextWindow.Write(Сообщение)`

где `Сообщение` — текст (строковая константа), который надо вывести на экран.

Пример:

`TextWindow.Write("Microsoft Small Basic")`

`TextWindow.WriteLine("Micrisift Small Basic")`

Отличие инструкций `TextWindow.Write` и `TextWindow.WriteLine`

Инструкция `TextWindow.Write` выводит сообщение сразу за текстом, выведенным предыдущей инструкцией.

Инструкция `TextWindow.WriteLine` выводит сообщение в начале следующей строки.

Вывод значений переменных

Инструкция вывода значения переменной в общем виде выглядит так:

`TextWindow.Write(Переменная)`

где *Переменная* — имя переменной, значение которой надо вывести на экран монитора.

Пример:

`TextWindow.Write(profit)`

`TextWindow.WriteLine(profit)`

Вывод текста и переменной

При необходимости вывести на экран текст и значение переменной, можно использовать следующий пример записи инструкции:

```
TextWindow.Write("Доход: " + profit + " руб.")
```

Ввод данных

Инструкции

`TextWindow.ReadNumber()`

`TextWindow.Read()`

обеспечивают ввод данных с клавиатуры.

Ввод чисел

Функция `TextWindow.ReadNumber()` используется для ввода с клавиатуры чисел.

В общем виде инструкция ввода числа с клавиатуры выглядит так:

`переменная = TextWindow.ReadNumber()`

где `переменная` — имя переменной, значение которой надо ввести с клавиатуры во время работы программы.

Пример:

`sum = TextWindow.ReadNumber()`

Ввод строк

Функция **TextWindow.Read()** используется для ввода с клавиатуры строк символов.

В общем виде инструкция ввода строки с клавиатуры выглядит так:

переменная = TextWindow.Read()

где **переменная** — имя переменной, значение которой надо ввести с клавиатуры во время работы программы.

Пример:

name = TextWindow.Read()

Пример программы

```
TextWindow.WriteLine("Введите первое число")  
a=TextWindow.ReadNumber()  
TextWindow.WriteLine("Введите второе число")  
b=TextWindow.ReadNumber()  
c=a+b  
TextWindow.WriteLine(c)  
TextWindow.WriteLine("Произведение a и b="+a*b)
```

Управляющие структуры

Условие

Операторы сравнения

Простое условие состоит из двух операндов и оператора сравнения, находящегося между ними. Операндами выражения-условия могут быть переменные, константы (числовые, строковые) и функции.

Оператор	Условие
> (больше)	$a > b$
< (меньше)	$a < b$
= (равно)	$a = b$
<> (не равно)	$a <> b$
>= (больше или равно)	$a \geq b$
<= (меньше или равно)	$a \leq b$

Логические операторы

Операторы сравнения позволяют записывать простые условия, из которых путем применения логических операторов **And** (логическое И), **Or** (логическое ИЛИ) и **Not** (отрицание) можно строить сложные условия.

Примеры сложных условий:

$(x \geq x1) \text{ And } (x \leq x2)$

$(x < x1) \text{ Or } (x > x2)$

$(Sum \geq 1000) \text{ And } (sum < 10000)$

Условие (Инструкция if)

Условие — это выражение логического типа, которое может принимать одно из двух значений: "истина" или "ложь".

Инструкция If используется в том случае, если нужно выбрать одно из двух возможных действий. В общем виде она записывается так:

If Условие Then

Действие 1

Else

Действие 2

EndIf

Пример инструкции if

If t = 1 Then

r = r1+r2

Else

r = r1*r2/(r1+r2)

EndIf

Неполная инструкция if

If *Условие* Then
 Действие
EndIf

При помощи нескольких инструкций If можно осуществить множественный выбор. Например, если необходимо выбрать один из трех вариантов, то реализовать это можно так:

```
If Условие 1 Then
    Действие 1
Elseif Условие 2 Then
    Действие 2
Else
    Действие 3
EndIf
EndIf
```

Пример инструкции if

```
If format = 1 Then
    cena = 2.5
    fname = "9x12"
Elseif format = 2 Then
    cena = 4
    fname = "10x15"
Else
    cena = 8
    fname = "18x24"
EndIf
EndIf
```

Циклы

Цикл с параметром

Инструкция For

Инструкция **For** используется, если некоторую последовательность действий надо выполнить несколько раз, причем число повторений можно задать (вычислить) заранее.

В общем виде инструкция **For** записывается так:

```
For Счетчик = Нач_знач To Кон_знач Step Приращение  
    Инструкции  
EndFor
```

Обратите внимание: если начальное значение счетчика больше указанного конечного значения счетчика, инструкции цикла ни разу не будут выполнены.

Пример инструкции For

$m = 1$

For $i = 1$ To n

$m = m * 2$

EndFor

Инструкция *While*

Инструкция **While** используется для реализации циклов с предусловием. В общем виде она записывается так:

```
While Условие  
    Инструкции  
EndWhile
```

Пример инструкции While

$dx = 0.25$

$x = x1$

While ($x \leq x2$)

$y = 0.5 * x * x + x - 5$

TextWindow.Write($x + " "$)

TextWindow.WriteLine(y)

$x = x + dx$

EndWhile

Процедуры

Процедура

Объявление подпрограммы в общем виде выглядит так:

```
Sub Имя  
    'здесь инструкции подпрограммы  
EndSub
```

Слово **Sub** (от англ. *subprogram* — подпрограмма) показывает, что далее следуют инструкции подпрограммы, слово **EndSub** означает конец подпрограммы. Идентификатор **Имя** определяет имя подпрограммы

Пример подпрограммы

```
Sub WriteArray
  For k=1 To 4
    TextWindow.Write(a[k] + ",")
  EndFor
  TextWindow.WriteLine(a[5])
EndSub
```

Вызов процедуры
WriteArray()

Особенности процедур

Следует **обратить внимание** на то, что подпрограмма Small Basic, в отличие от подпрограмм в других языках программирования (Visual Basic, Pascal, C++ и др.), для передачи в подпрограмму информации **не используют** механизм параметров. Информация в подпрограмму передается через **внешние (глобальные)** переменные.

Работа с датчиками

Инициализация датчиков

EV3 Бейсик совместим со всеми стандартными EV3 и NXT датчиками. Для работы с датчиками в SmallBasic необходимо инициализировать датчики для работы в необходимом режиме, так как датчики могут работать в нескольких режимах.

Инициализация датчиков производится командой со следующими параметрами:

Sensor.SetMode(порт, режим)

Пример:

Sensor.SetMode(1,0)

Датчик касания

Датчик касания (кнопка) имеет только один режим работы.

Команда инициализации датчика касания:

Sensor.SetMode(norm,0)

Датчик касания (кнопка) используется с функцией **Sensor.ReadPercent**(norm), которая возвращает 0, если кнопка не нажата и 100 в нажатом положении.

Ультразвуковой датчик

Ультразвуковой датчик имеет два режима работы:

0 – возвращает значение расстояния в **миллиметрах** (датчик NXT в данном режиме возвращает значение в **сантиметрах**);

1 – возвращает значение расстояния в **десятих дюйма**.

Получение показаний ультразвукового датчика

Для определения расстояния, которое
возвращает ультразвуковой датчик
используйте функцию

Sensor.ReadRawValue(порт, 0).

Пример программы:

‘датчик подключен к 1 порту

Sensor.SetMode(1,0)

While “true”

k=Sensor.ReadRawValue(1,0)

EndWhile

Гироскопический датчик

Гироскопический датчик имеет два режима работа:

0 – измеряет угол **в градусах** относительно позиции датчика на момент старта программы или сброса его показаний;

1 – измеряет скорость изменения отклонения **в градусах в секунду**.

Получение показаний гироскопического датчика

Для получения показаний датчика в обоих режимах используется функция

Sensor.ReadRawValue(порт, 0),

которая возвращает массив из **единственного 0-го элемента**.

Пример программы:

' Гироскоп подключен к 1 порту

Sensor.SetMode(1,0)

While "True"

k=Sensor.ReadRawValue(1,0)

EndWhile

Инфракрасный датчик

Инфракрасный датчик EV3 может работать в следующих режимах:

0 – измерение расстояния до объекта **в см;**

1 – измерение расстояния и направления на ИК-маяк;

2 – сигналы, принятые от ИК-маяка (или от маяков, **до 4 одновременно**).

Получение результатов с инфракрасного датчика

По умолчанию датчик работает в режиме 0 и функция **Sensor.ReadPercent(порт)** возвращает целое число **от 0 до 100** – **расстояния до объекта в см.**

Это расстояние не особо точное, зависит от освещенности объекта. Более точно измеряется расстояние до ярко освещенных объектов.

Пример программы:

' Подключите ИК-датчик к 4 порту

Sensor.SetMode(4,0)

While "True"

k=Sensor.ReadPercent(4)

EndWhile

Получение результатов с инфракрасного датчика

В **режиме 1** ИК датчик начинает возвращать расстояние и направление на ИК-маяк. При этом он возвращает оба значения одновременно по функции **Sensor.ReadRaw(порт, 2)**.

Возвращаемое ей значение – массив, в 0 элементе – **направление на ИК маяк**, в 1 элементе – **расстояние до ИК-маяка в см.**

Получение результатов с инфракрасного датчика

В **режиме 2** ИК – датчик, подключенный, начинает определять, какие кнопки нажаты на удаленном маяке (маяках).

А – левая верхняя, В – левая нижняя, С – правая верхняя, D – правая нижняя. Е – средняя

Коды, принимаемые

`Sensor.ReadRawValue(4, канал_передачи)`

А = 1

А и С = 5

А и В = 10

В = 2

В и С = 7

В и D = 8

С = 3

А и D = 6

С и D = 11

D = 4

Е = 9

Все другие комбинации выдадут 0

Получение результатов с инфракрасного датчика

В **режиме 2** ИК датчик начинает возвращать расстояние и направление на ИК-маяк.

В случае, если только один маяк передает коды нажатых кнопок на ИК-датчик, необходимо использовать **Sensor.ReadRawValue(4, канал_передачи)** для получения кодов кнопок.

Если несколько маяков одновременно передают коды нажатых кнопок на ИК-датчик на разных каналах, необходимо использовать функцию **Sensor.ReadRaw(4, 4)**, которая возвращает массив из 4 значений, одного на каждый канал.

0-й элемент массива соответствует маяку на 1 канале и т.д

Энкодер как датчик угла

В Small Basic EV3 LEGO моторы могут использоваться в качестве датчиков угла.

Функция **Motor.GetCount** ("порт")

показывает **угол поворота оси мотора в градусах**, подключенного к порту, указанному в ее параметрах.

Функция **Motor.ResetCount**("порт") сбрасывает показания угла поворота оси мотора **до 0**.

Пример:

```
k=Motor.GetCount("B")
```

```
Motor.ResetCount("B")
```

Цветосветовой датчик

В EV3 Бейсике цветосветовой датчик может работать в 4 режимах:

- 0 – режим отраженного света;
- 1 – режим измерения уровня внешней освещенности;
- 2 – режим измерения цвета;
- 4 – режим измерения RGB-составляющих цвета.

Режим отраженного света

В **режиме 0** датчик возвращает по функции **Sensor.ReadPercent(порт)** уровень отраженного света **в процентах** (от 0 до 100).

0 – черный цвет

100 – белый цвет

Пример:

‘датчик подключен ко 2-ому порту

```
Senor.SetMode(2,0)
```

```
k=Sensor.ReadRepcent(2)
```

Режим измерения уровня внешней освещенности

В **режиме 1** датчик возвращает **0** при минимуме внешнего освещения и **199** на ярком свету. Для получения результата измерений используется функция

Sensor.ReadPercent(порт).

Пример:

‘датчик подключен ко 2-ому порту

```
Senor.SetMode(2,1)
```

```
k=Sensor.ReadPercent(2)
```

Режим измерения цвета

В **режиме 2** датчик возвращает **код цвета**.
Для работы с датчиком в этом режиме используйте функцию

Sensor.ReadRawValue(порт, 0),

которая возвращает массив из **единственного 0-го элемента**.
Коды цветов:

0 – цвет не определен,

1 – черный,

2 – **синий**,

3 – **зеленый**,

4 – **желтый**,

5 – **красный**,

6 – белый,

7 – **коричневый**.

Обратите внимание, датчик цвета LEGO откалиброван на цвета кубиков LEGO, по остальным оттенкам цветов, даже если они кажутся очевидными, датчик может выдать неожиданный результат.

Пример программы

Sensor.SetMode(3,2) *Устанавливаем режим 2 датчика цвета на порту 3*

While "True"

code=Sensor.ReadRawValue(3, 0)

If code =0 Then

col="UNKNOWN"

Elseif code =1 Then

col="BLACK"

Elseif code =2 Then

col="BLUE"

Elseif code =3 Then

col="GREEN"

Elseif code =4 Then

col="YELLOW"

Elseif code =5 Then

col="RED"

Elseif code =6 Then

col="WHITE"

Elseif code =7 then

col="BROWN"

EndIf

LCD.Text(1,33,75, 2, col)

EndWhile

Режим измерения RGB-составляющих цвета

В режиме 4 датчик цвета возвращает массив из RGB-составляющих цвета. Это позволит Вам определить любой оттенок цвета, ориентируясь на его составляющие.

Для работы с датчиком в этом режиме используйте функцию

Sensor.ReadRaw(порт, количество_показаний)

Пример:

Sensor.ReadRaw(2,2)

Режимы работы датчиков

Датчик	Режим	Допустимые функции	Возвращаемое значение
Касания	0	Sensor.ReadPercent	0=не нажат, 100=нажат
Ультразвуковой	0	Sensor.ReadRawValue	Расстояние в миллиметрах
	1	Sensor.ReadRawValue	Расстояние в десятых дюйма
Гироскопический	0	Sensor.ReadRawValue	Угол отклонения в градусах
	1	Sensor.ReadRawValue	Скорость изменения угла отклонения в градусах в секунду
Инфракрасный	0	Sensor.ReadPercent	Расстояние в см
	1	Sensor.ReadRaw	value0=направление на ИК-маяк value1=Расстояние до ИК-маяка
	2	Sensor.ReadRawValue	value0=сигнал на канал 1 value1= сигнал на канал 2...

Режимы работы датчиков

Датчик	Режим	Допустимые функции	Возвращаемое значение
Цветосветовой датчик	0	Sensor.ReadPercent	0=нет отраженного света, 100=макс
	1	Sensor.ReadPercent	0=нет внеш. освещ., 100=макс
	2	Sensor.ReadRawValue	0=неизвестно, 1=черный, 2=синий, 3=зеленый, 4=желтый, 5=красный, 6=белый, 7=коричневый
	4	Sensor.ReadRaw	value0=красная составляющая, value1=зеленая составляющая, value2=синяя составляющая

Работа с моторами

Режим измерения RVG- составляющих цвета

EV3 Бейсик совместим со средними и большими моторами EV3, а также с моторами NXT и, по большому счету, не делает различий при работе с ними.

EV3 Бейсик имеет **9 команд**, которые могут использоваться для управления моторами.

4 из них являются основными: **Motor.Move**, **Motor.MoveSync**, **Motor.Start** и **Motor.StartSync**.

И, конечно же, **Motor.Stop** для того, чтобы останавливать мотор.

Параметры команд для работы с моторами

Команды для работы с моторами используют следующие параметры:

порт – порт, к которому подключен мотор. Если моторов в параметре несколько, они всегда записываются в алфавитном порядке. **Пример: "BC". "A";**

угол – угол поворота мотора. **Всегда положительное значение**, в случае отрицательного – **знак игнорируется**;

тормоз = "True", когда после остановки мотор должен затормозить, иначе "False" – мотор останавливается по инерции;

Motor.Move

Поворачивает один или несколько моторов с заданной скоростью **на указанный угол** (в **градусах**). Программа не будет переходить к выполнению следующих команд до тех пор, пока моторы не повернутся на требуемый угол.

Motor.Move (“порты”, скорость, угол, “торможение”)

Порты: Порты моторов

Скорость: Скорость от -100 (полный назад) до 100 (полный вперед)

Угол: Угол поворота

Возвращает: "True", если необходимо удерживать положение после остановки моторов

Пример:

Motor.Move (“AB”, 50, 180, “True”)

Motor.MoveSync

Поворачивает **два мотора синхронно** на определенное количество градусов. Синхронная работа двигателей означает, что когда один двигатель нагружен и что-то препятствует его вращению, второй двигатель пропорционально замедлится или даже вообще остановится. Угол, на который будут повернуты моторы, относится к мотору с наибольшей скоростью вращения, угол поворота второго мотора будет рассчитан пропорционально его скорости.

Motor.Move (“порты”, скорость1, скорость2, угол, “торможение”)

Порты: Имена двух портов для моторов (например "AB" или "CD")

Скорость1: Скорость от -100 (полный назад) до 100 (полный вперед) мотора с младшим по алфавиту номером порта

Скорость2: Скорость от -100 (полный назад) до 100 (полный вперед) мотора со старшим по алфавиту номером порта

Угол: Угол поворота (мотора с наибольшей скоростью)

Возвращает: "True", если необходимо удерживать положение после остановки моторов.

Пример:

Motor.Move (“AB”, 50, 30, 180, “True”)

Motor.Start

Запустить один или несколько моторов с указанной скоростью или изменить скорость уже запущенных моторов на указанную.

Motor.Start (“порт”,
скорость)

Порты: Имя порта мотора

Скорость: Скорость от -100 (полный назад) до 100 (полный вперед) мотора.

Пример:

Motor.Start (“A”,50)

Motor.StartSync

Синхронно запустить два мотора с указанными скоростями в режиме контроля за их вращением. Если один мотор будет испытывать нагрузку, которая замедлит его скорость, второй мотор пропорционально замедлится, чтобы сохранить траекторию движения.

Motor.StartSync (“порты”,
скорость1, скорость2)

Порты: Имена портов моторов

Скорость1: Скорость от -100 (полный назад) до 100 (полный вперед) мотора с младшим по алфавиту номером порта

Скорость2: Скорость от -100 (полный назад) до 100 (полный вперед) мотора со старшим по алфавиту номером порта.

Пример:

72 | university.innopolis.ru robolymp.ru

Особенности команд Motor.Start, Motor.StartSync

Команды Motor.Start, Motor.StartSync могут быть использованы только внутри циклов.

Пример:

While "True"

Motor.Start("BC",50)

EndWhile

Motor.Stop

Остановить один или несколько моторов. Команда завершает так же все запланированные или незавершенные команды управления этими моторами.

Motor.StartSync (“порты”, “торможение”)

Порты: Порт(ы) моторов

Торможение: "True", если необходимо удерживать положение после остановки моторов

Пример:

Motor.Stop(“BC”, “True”)

Рекомендации по выбору команд для управления моторами

	Двигаться x градусов, программа ждет завершения	Включиться на постоянное вращение
Регулировка скорости	Motor.Move	Motor.Start
Синхронизация моторов	Motor.MoveSync	Motor.StartSync

Работа с дисплеем

Дисплеей

EV3 имеет черно-белый экран с разрешением 178×128 пикселей. Левый верхний угол экран имеет координаты $(0,0)$, правый нижний $(178,128)$.

Основные операции для работы с дисплеем

Для работы с экраном EV3 Бейсик имеет следующие основные команды:

LCD.Clear() – очищает экран

LCD.Text (цвет, x, y, размер, "текст") – пишет текст заданного размера и цвета в указанной позиции

LCD.Write (x, y, "текст") – пишет текст среднего размера в позиции x, y

LCD.Update () – обновляет экран

Задержка выполнения команды

Для задержки выполнения команды используется функция:

Program.Delay(время)

Время – время задержки в миллисекундах

Пример программы

```
LCD.Clear() ' очищаем экран  
LCD.Write(45,60,"Hello World") ' печатаем  
начиная с точки (45,60) на экране  
Program.Delay(10000) ' ждем 10 секунд прежде  
чем завершить программу
```


Спасибо за
внимание!