Переменные в Visual Basic

- **Переменные**
  - Каждая переменная имеет своё имя.
  - Оно может достигать 255 символов в длину, начинается всегда с буквы латинского алфавита, за которой могут следовать другие буквы, цифры и знак подчёркивания.
  - Регистр символов значения не имеет.

Переменные.

При выборе имен переменных рекомендуется придерживаться венгерского соглашения:

- Постарайтесь чтобы именование переменных согласно такой схеме у вас вошло в привычку.
- Я советую, но не настаиваю.

Тип	Схема именования	Пример
Константа	Имя константы должно состоять только из заглавных букв.	**HWND_BROADCAST**
Переменная	Имя переменной должно начинаться с маленькой буквы, далее следующие слова с большой.	**numOfFonts**
Функция	Имя функции должно начинаться с заглавной буквы, далее следующие слова тоже с заглавной.	**SetForegroundWindow**

Всего в VB 14 типов переменных.

- Кроме того, программист может определить и свой тип.

Тип	Описание	Размер (байт)	Суффикс
Byte	целые числа от 0 до 255	**1**	
Integer	целые числа от -32768 до 32767	**2**	**%**
Long	целые числа от -2147483648 до +2147483647	**4**	**&**

Тип	Описание	Размер (байт)	Суф фикс
String	строковая (символьная) информация	**до 2 Гб**	**$**
Single	дробные числа, с точностью до 7 цифр от 1.401298E-45 до 3.402823E38. длина числа может достигать 38 знаков	**4**	**!**
Double	дробные числа, с точностью до 16 цифр от 1.7976931348623 2E308 до -4.94065645841247E-324. длина числа может достигать 300 знаков	**8**	**#**
Currency	создан для того, чтобы избежать ошибок при преобразовании чисел из десятичной формы в двоичную и наоборот; может иметь до 4 цифр после запятой, и до 14 перед ней от -922337203685477.5808 до 922337203685477.5807	**8**	**@**

Типы переменных в Visual Basic.

Тип	Описание	Размер (байт)
Date	значения времени и даты в промежутке от полуночи 1 января 100 года до полуночи 31 декабря 9999 года если переменной присваивается только значение даты, то время равняется 00:00.	**8**
Boolean	булевы значения **True** и **False** в VB False - это нуль, а любое, не нулевое значение – True.	**2**
Variant	переменная типа Variant может содержать данные любого типа. Visual Basic автоматически производит необходимые преобразования данных	4 для числовых типов 22 для строковых

Типы переменных в Visual Basic.

В Visual Basic переменные объявляются с помощью оператора **Dim**, после которого следует **As** и **Имя_Типа** переменной.

Например:

```
Dim a As Long
Dim b As Byte
Dim c As Long
Dim numOfLetters As Long
Dim myString As String
Dim isLoaded As Boolean
```

Если не указывать **As Имя_Типа**, то переменная будет объявлена как **Variant**.

После объявления переменной ей присваивается значение по умолчанию.

- Для строки это - "" (пустая строка).
- Для чисел - 0.
- Для Boolean - False.

Переменная может быть использована сразу после объявления.

Переменные можно объявлять и в одной строчке, разделяя объявления запятыми:

```
Dim a As Long, Dim b As Long, Dim c As Long
```

важно заметить следующую особенность

Логично было бы объявить 3 переменные типа **Long** следующим образом:

```
Dim a, b, c As Long
```

*В результате такого объявления VB объявит 3 переменные, первая и вторая будут иметь тип **Variant**, и только третья - **Long**!*

Запомните это!

Для объявления переменной можно, а иногда и нужно использовать суффикс, но без использования зарезервированных слов **As Тип**.

```
Dim myLongParam&

Dim myString$
```

Также можно использовать суффиксы для явного указания типа константам

```
Call MyProc (myParam1, myParam2, 5&)
```

При вызове процедуры MyProc, последний параметр имеет тип **Long**.
Если бы мы не указали значок **&**, то он (параметр) имел бы тип **Byte**.

Присвоить значение переменной можно при помощи знака равно "=".

```
Dim a As Long , Dim b As Byte , Dim c As Long
Dim myString As String
Dim isLoaded As Boolean
    a = 1234567
    b = 234
    c = 133
    myString = "Visual Basic рулит”
    isLoaded = True
```

Совместимость и преобразование типов

```
a = b       ' можно, т.к. переменная b "умещается" в тип Long
a = b + c   ' теперь в a хранится сумма b + c.
b = c           ' тоже возможно (133 < 255)
b = a       ' нельзя, т.к. переменная a не укладывается в диапазон [0-255].
myString = a
    ' VB сам преобразует число 1234567 в строку "1234567 ",
    ' а потом присвоит это значение переменной myString.
    ' Возможно и обратное действие.
isLoaded = True  ' всё ОК
myString = myString & " однозначно!"
    ' здесь происходит слияние двух строк,
    ' в результате переменная myString содержит строку:
    ' "Visual Basic рулит однозначно!".
isLoaded = a
    ' возможно, т.к. VB сам преобразует тип Long в
    ' тип Boolean. isLoaded будет содержать True.
```

Рассмотрим распространённую ошибку

```
Dim a As Byte
Dim b As Byte
Dim c As Long
   a = 200
    b = 200
     c = a + b
```

Казалось бы, что если запустить такой код на выполнение, то в переменной **c** будет находиться значение **400** (200 + 200).

Visual Basic на строке c = a + b сгенерирует ошибку Overflow

Дело в том, что в выражении справа от знака равно складываются 2 переменные типа **Byte**, и Visual Basic решает, что после вычисления этого выражения, должен остаться тот же тип - **Byte**.

Visual Basic предоставляет в ваше распоряжение несколько функций преобразования типов: **CLng, CBool, CDate, CStr** и т.д.

```
Dim a As Byte
Dim b As Byte
Dim c As Long
   a = 200
    b = 200
     c = CLng(a) + CLng(b) 'Всё в порядке
```

- Объявлять переменные можно в самых разных местах:
  - Внутри процедуры (функции)
  - В разделе General Declarations формы
  - В разделе General Declarations модуля

Область "видимости" переменных.

Переменная объявлена внутри процедуры (функции)

- В этом случае переменная будет "видна" только в коде этой процедуры (функции).
- Если вы попытаетесь обратиться к такой переменной внутри кода другой процедуры, то Visual Basic сгенерирует ошибку.

Переменная объявлена в разделе
General Declarations
формы

- Такие переменные будет "видны" в любом месте кода формы, т. е. в любой процедуре (или функции) формы.
- Переменные в данном месте могут быть объявлены с помощью зарезервированных слов **Private** и **Public**.

(В самом верху кода формы должен находиться оператор Option Explicit)

Область "видимости" переменных.

Переменная объявлена в разделе General Declarations

- Здесь действуют те же правила, что и в разделе General Declarations формы.
- **Private** (или **Dim**) будут "видны" только в коде модуля, а **Public** - везде.
- Отличие наблюдается только в способе доступа к переменной. Не обязательно указывать имя модуля перед такой переменной.

Рассмотрим 3 определения:

```
Dim myLocalVar1 As Byte

Private myLocalVar2 As Integer

Public myGlobalVar1 As Long
```

Первые 2 определения абсолютно эквивалентны.

- Переменные объявленные таким образом будут видны в любом месте кода формы (но только той формы, где они объявлены).

Третья переменная будет видна всему приложению в любом месте.

- Чтобы добраться к такой переменной из кода другой формы, необходимо перед именем переменной указат формы, где эта переменная

```
Form1.myGlobalVar1 = 234
```

Область "видимости" переменных.

Переменные, объявленные в процедуре (функции) будут "живы" только пока выполняется эта процедура (функция).

- При выходе из процедуры - переменная удаляется.
- При очередном вызове этой процедуры - переменная заново инициализируется.

Дополнительно к словам Private, Public и Dim, в процедурах и функциях можно использовать зарезервированное слово **Static**.

- Такая переменная при повторном вызове этой процедуры не будет заново инициализироваться.
- Она будет сохранять то значение, которое было в ней после предыдущего вызова.

```
Static myStat As String ' Private Static переменная
```

Период существования переменных.

Переменные уровня формы будут "живы" только пока "жива" форма.

- Как только объектная переменная формы будет установлена в **Nothing** (или после выполнения оператора **Unload**), все переменные уровня этой формы удаляются.

Переменные уровня модуля "живы", пока "живёт" ваше приложение.

- Т.е. "живы" всегда.

Период существования переменных.

Чтобы объявить константу необходимо использовать зарезервированное слово **Const**, за которым следует имя и значение (и возможно тип) константы

```
Const   PI = 3.1415
```

Для констант с плавающей точкой тип по умолчанию - **Double**

Для констант - целых чисел тип по умолчанию – **Integer**

Для того чтобы явно задать тип константы, необходимо после имени задать тип

```
Const PI As Long = 3 ' PI = 3, PI имеет тип Long
```

В Visual Basic массивы определяются следующим образом:

```
Dim ИмяМассива(НомПерв1 To НомПосл1 [, НомПерв2 To НомПосл2, ...]) [As [New] ИмяТипа]
```

Например:

```
Dim myArray (10) As Long
```

*массив **myArray** будет содержать 11 элементов, потому что нижняя граница массива начинается с нуля*

Чтобы задать определённую размерность можно использовать зарезервированное слово **To**:

```
Dim myArray (5 To 10) As Long
```

Массивы можно делать многомерными:

```
Dim chessTable (1 To 8, 1 To 8) As String
```

Динамические массивы - это такие массивы, размерность которых может меняться в ходе работы программы.

Опеределяется такой массив следующим образом:

```
Dim ИмяМассива() [As [New] ИмяТипа]
```

Например:

```
Dim myArray () As Byte
```

К элементам динамического массива сразу обращаться нельзя

- т.к. они ещё не инициализированы.

Для начала нужно указать размерность массива

- для этого в VB есть оператор **ReDim**.

Например:

```
ReDim myArray (4)
```

*Если в дальнейшем возникнет необходимость снова изменить размерность массива, можно ещё раз использовать **ReDim**.*

Рассмотрим пример:

```
Dim myLong As Long
Dim myArray() As Long   ' объявляем массив
ReDim myArray (2)    ' одна размерность [0,1,2]
myArray (1) = 234    ' присваиваем второму элементу чило 234
myLong = myArray (1)     ' сохраняем его в переменной myLong
ReDim myArray (3)    ' снова меняем размерность - теперь [0,1,2,3]
myLong = myArray (1)     ' снова пытаемся сохранить второй элемент
```

В результате, переменной **myLong** присвоится 0 вместо 234!

Оператор **ReDim** заново инициализирует все элементы массива к значению по умолчанию.

Если мы хотим изменить размеры массива, сохранив все старые элементы нужно после оператора **ReDim** поставить слово **Preserve**.

```
ReDim Preserve myArray (3)    ' сохраняем старые элементы
myLong = myArray (1)   ' всё в порядке
```

Массивы могут храниться в переменных типа **Variant**.

- В некоторых случаях без этого просто не обойтись!
- Например, когда вы хотите, чтобы ваша функция возвращала массив.

Чтобы сохранить массив в переменной типа **Variant** необходимо просто присвоить его этой переменной

```
Dim myVariantArray      ' переменная Variant по умолчанию
myVarianrArray = chessTable
```

Обратите внимание, никакие индексы указывать не нужно!

Чтобы узнать текущие размеры массива, можно использовать встроенные функции Visual Basic

LBound	•возвращает нижнюю границу массива
UBound	•возвращает верхнюю границу массива

Полезная информация.

Чтобы определить **запись**, в программе нужно использовать зарезервированное слово **Type**, описание заканчивается словом **End Type**.

```
Private Type Student    ' вместо Private могло быть и Public
    FIO As String
    Age As Byte
    HasGramot As Boolean
End Type
```

*Слово **Student** синим выделяться не будет, т.к. синюю подсветку имеют только зарезервированные слова*

***Dim** перед именем переменной указывать не нужно!*

Записи.

Теперь можно объявлять переменные, имеющий тип – **Student**:

```
Dim newStud As Student
```

К полям записи можно обращаться при помощи точки:

```
newStud.FIO = "Василий Петрович Пупкин»
newStud.Age = 19
newStud.HasGramot = False
```

Всё как в Pascal.

Visual Basic предоставляет возможность не указывать каждый раз имя переменной типа запись, при обращении к её элементам.

- Для этого есть зарезервированное слово **With**.

```
With newStud
     .FIO = "Бабай Бабаевич Бабаев”
     .Age = 20
     .HasGramot = True
End With
```

Записи.

В Visual Basic определен новый тип - **перечисление**.

- Перечисление - это список констант.

Перед использованием перечисления в программе его необходимо определить с помощью зарезервированного слова **Enum**.

```
Enum Ocenka
    Neud = 3
    Horosho = 4
    Otlichno = 5
End Enum
```

Присваивать значения константам внутри **Enum** не обязательно.

- Если этого не сделать, то константы будут принимать значения 0,1,2... и т.д..

Введем переменную типа Ocenka:

```
Dim oc1 As Ocenka
```

Если теперь вы попытаетесь присвоить такой переменной значение - Visual Basic выдаст список (**Neud, Horosho** и **Otlichno**) из которого вы сможете выбрать нужное значение.

При программировании на паскале мы могли использовать множества.

- **К сожалению множеств в Visual Basic нет**.

Но в принципе, никто не мешает вам реализовать их самостоятельно, написав соответствующие функции.

Замечание.