Основы языка программирования Visual Basic 6

Типы данных. Переменные, массивы, записи и константы в Visual Basic

Тип данных	Описание	Диапазон значений	Занимаемая память
Byte	Двоичные данные	от 0 до 255	1 байт
Integer	Целые числа	от -32768 до 32767	2 байта
Long	Целые числа (длинные)	от -2147483648 до +2147483647	4 байта
Single	Вещественные числа	от $-3.402823*10^{38}$ до $3.402823*10^{38}$	4 байта
Double	Вещественные числа	от $-1.79769313486232*10^{308}$ до $1.79769313486232*10^{308}$	8 байтов
String	Символьный	от 0 до 65535 символов	1 или 2 байта на 1 символ
Currency	Число с фиксированной десятичной точкой	от -922337203685477.5808 до 922337203685477.5807	8 байтов
Boolean	Логический	True или False	1 байт
Date	Дата	от January (Январь) 1, 100, до December (Декабрь) 31, 9999	8 байтов
Variant	Произвольный тип	Любой, из перечисленных выше	Зависит от значения
Object	Объект		4 байта

Для объявления переменных используются операторы: **Static** (уровень процедуры), **Dim**, **Private** (уровень модуля или процедуры), **Public**, **Global** (уровень всех модулей приложения и формы). Примеры объявления переменных:

Dim i **As** Byte, n **As** Integer, max **As** Integer

Однотипный набор элементов (данных) можно объявить в виде ***массива***, где каждый ***элемент*** имеет свой ***индекс*** (порядковый номер). По умолчанию ***Visual Basic*** индексирует элементы массива, начиная с нуля, т.е., индекс **0** обозначает первый элемент массива, индекс **1** – второй и т.д. Массивы могут быть ***статическими*** (с указанием границ индексов), и ***динамическими*** (без предварительного указания границ индексов), ***одномерными*** (1 индекс) и ***многомерными*** (несколько индексов).

Пример объявления статического массива:

Dim ***Array1*****(**[***n1*** **To**] ***n2,*** **(**[***m1*** **To**] ***m2, …*****) As** ***Integer***,

где ***n1***, ***m1*** – нижние, ***n2***, ***m2*** верхние индексы (границы) массива.

Пример объявления динамического массива:

Dim ***Array2*****() As** ***Single***

В том случае, когда несколько переменных различных типов должны быть связаны между собой, их объединяют в ***записи*** (***пользовательские типы данных***). Оператор объявления ***записи*** помещается в ***раздел*** **Declarations** ***модуля*** и имеет вид:

[Public|Private] Type ***НазваниеТипа***
Элемент1 **As** ***Тип***
Элемент2 **As** ***Тип***
….
End Type

Константы – величины, значения которых не могут меняться. Как и переменные, ***константы*** объявляются в начале текста программного кода оператором:

Const ***ИмяКонстанты*** [**As** ***Тип***] = ***Значение***

В качестве значения константы допускается использование только постоянных значений и их комбинаций, включая арифметические и/или логические операторы.

Встроенные константы (соответствующие управляющим символам):

vbCrLf	**Chr(13) & Chr(10)**	возврат каретки и перевод строки
vbCr	**Chr(13)**	возврат каретки
vbLf	**Chr(10)**	перевод строки
vbTab	**Chr(9)**	табуляция

Операторы и функции

Для работы с данными используются операторы и функции. Ниже приводятся основные операторы, а также встроенные функции языка ***Visual Basic***.

Арифметические операторы:

+	оператор сложения	**-**	оператор смены знака
-	оператор вычитания	**^**	оператор возведения в степень
*****	оператор умножения	****	оператор целочисленного деления
/	оператор деления	**Mod**	оператор вычисления остатка от деления

Оператор конкатенации: **&** (или **+**)

Математические функции:

Abs(x)	возвращает абсолютное значение числа **x**	**Sin(x)**	возвращает синус угла **x** (угол задается в радианах)
Sqr(x)	возвращает квадратный корень числа **x**	**Cos(x)**	возвращает косинус угла **x** (угол задается в радианах)
Fix(x), Int(x)	возвращают число **x**, округленное до целого значения	**Tan(x)**	возвращает тангенс угла **x** (угол задается в радианах)
Round(x,n)	возвращает число **x**, округленное до **n** десятичных знаков	**Atn(x)**	возвращает арктангенс угла **x** (угол задается в радианах)
Rnd()	возвращает случайное число **>=0** и **<1**	**Log(x)**	возвращает натуральный логарифм числа **x**
Sgn(x)	возвращает знак числа **x**	**Exp(x)**	возвращает **e^x** (**e^x**)

Операторы сравнения:

<	меньше	**=**	равно
>	больше	**<>**	не равно
<=	меньше или равно	**Is**	оператор сравнения объектов
>=	больше или равно	**Like**	оператор сравнения строк с шаблоном

Логические операторы:

And	оператор логического умножения	**=**	оператор логической эквивалентности
Or	оператор логического сложения	**<>**	оператор логической импликации
Not	оператор логического отрицания	**Is**	оператор логического исключающего сложения

Строковые функции:

Len(s)	возвращает количество символов строки **s**	**Left(s, k)**	возвращает **k** символов с начала строки
Asc(s)	возвращает ASCII код первого символа строки **s**	**Right(s,k)**	возвращает **k** символов в конце строки
Chr(n)	возвращает символ по ASCII коду **n**	**Mid(s,n,k)**	возвращает **k** символов строки, начиная с **n** символа
Str(x)	возвращает преобразованную число **x** в строку	**Lcase(s)**	преобразовывает строку **s** в нижний регистр
Val(s)	возвращает преобразованную строку **s** в число	**Ucase(s)**	преобразовывает строку **s** в верхний регистр
InStr([n,]s,w)	возвращает позицию первой найденной подстроки **w** в строке **s**, начиная с **n** символа	**LTrim(s), Rtrim(s), Trim(s)**	возвращает строку **s** без (начальных, конечных, начальных и конечных) пробелов
StrComp(s,w)	возвращает результат сравнения строк **s** и **w**	**Space(k)**	возвращает строку из **k** пробелов
Hex(n)	возвращает шестнадцатеричное представление числа **n**	**Oct(n)**	возвращает восьмеричное представление числа **n**
String(n,c)	возвращает строку, состоящую из повторяющихся **n** символов **c** (**c** – символ, первый символ строки или код символа)		
Format(v,"f")	возвращает выражение **v**, в виде значения (строки), в соответствии с заданным форматом **f** ("`00.0`", "`#.00`", "`0%`", "`h:m:s`", **...**)		
Replace(s,f,r[,n][,k][,cm])	возвращает строку **s**, подстрока **f** в которой, **k** раз будет заменена подстрокой **r**, начиная с номера символа **n**		

Функции даты и времени:

Date	возвращает текущую дату	**Month(d)**	возвращает номер месяца последовательной даты **d**
Time	возвращает текущее время	**Day(d)**	возвращает число месяца последовательной даты **d**

Now	возвращает текущие дату и время	**Weekday(d[,w])**	возвращает номер дня недели даты **d** (**w** – указывает, какой день недели считать первым)
Hour(t)	возвращает час дня времени **t**	**DateSerial(y,m,d)**	преобразовывает в последовательную дату числа **y, m, d** (день, месяц, год)
Minute(t)	возвращает количество минут времени **t**		
Second(t)	возвращает количество секунд времени **t**	**DateDiff(i,d1,d2)**	возвращает число интервалов **i** времени между датами **d2**, **d1**
Year(d)	возвращает год последовательной даты **d**	**DateAdd(i,n,d)**	возвращает дату – результат добавления **n** интервалов **i** к дате **d**

Функции преобразования типов:

CByte(x)	преобразует выражение к типу **Byte** (байтовый)	**CStr(x)**	преобразует выражение к типу **String** (строка)
CInt(x)	преобразует выражение к типу **Integer** (целое)	**CBool(x)**	приводит выражение к типу **Boolean** (логический)
CLng(x)	преобразует выражение к типу **Long** (длинное целое)	**CDate(x)**	преобразует выражение к типу **Data** (дата)
CSng(x)	преобразует выражение к типу **Single** (вещественное число)	**CVar(x)**	преобразует выражение к типу **Variant**
CDbl(x)	преобразует выражение к типу **Double** (вещественное число двойной точности)	**CCur(x)**	преобразует выражение к типу **Currency** (денежный формат)

Функции проверки типов данных:

IsArray()	возвращает True, если переменная представляет массив	**IsDate()**	возвращает True, если выражение представляет допустимую дату
IsEmpty	возвращает True, если переменная не инициализирована или равна пустому значению	**IsError**	возвращает True, если числовое выражение представляет ошибку
IsMissing()	возвращает True, если значение в функцию не передано	**IsNull()**	возвращает True, если выражение не содержит допустимых данных или равно Null
IsNumeric()	возвращает True, если выражение представляет число	**IsObject()**	возвращает True, если выражение представляет объект

Функции ввода-вывода данных

Функция **InputBox** – показывает диалоговое окно с подсказкой и полем ввода текста и возвращает введенную пользователем строку. Для управления вводом предназначены кнопки "**OK**" и "**Cancel**". "Образец" ввода данных, используя **InputBox**:

ВводимаяСтрока = **InputBox**("***Сообщение***"[, "***ЗаголовокОкна***"][, ("***ЗначениеПо-Умолчанию***"][, "***КоординатыЛевого***", "***ВерхнегоУглаОкна***"][, "***ИмяФайла***", "***Контекстно-ЗависимойСправки***"])

Для вывода информации используется диалоговое окно **MsgBox**:

MsgBox("***Сообщение***")[, "***параметрButtons***"][, "***ЗаголовокОкна***"] [,"***ИмяФайла***", "***Контекстно-ЗависимойСправки***"]

Параметр ***Buttons*** складывается из параметров других категорий:

Buttons = ***Button*** + ***Icon*** + ***Default*** + ***Modal*** + ***Extras***

Button:

vbOKOnly	**0**	только кнопка **OK**
vbOKCancel	**1**	кнопки **OK** и **Cancel**
vbAbortRetryIgnore	**2**	кнопки **Abort**, **Retry** и **Ignore**
vbYesNoCancel	**3**	кнопки **Yes**, **No** и **Cancel**
vbYesNo	**4**	кнопки **Yes** и **No**
vbRetryCancel	**5**	кнопки **Retry** и **Cancel**

Icon:

vbCritical	**16**	значок "**Ошибка**" –
vbQuestion	**32**	значок "**Запрос**" –
vbExclamation	**48**	значок "**Предупреждение**" –
vbInformation	**64**	значок "**Информация**" –

Default:

vbDefaultButton1	**0**	по умолчанию активна 1-ая кнопка
vbDefaultButton2	**256**	по умолчанию активна 2-ая кнопка
vbDefaultButton3	**512**	по умолчанию активна 3-ая кнопка
vbDefaultButton4	**768**	по умолчанию активна 4-ая кнопка

Диалоговое окно MsgBox возвращает определенные значения в зависимости от нажатой кнопки. В этом случае вызов MsgBox имеет вид:

Значение = **MsgBox**("***Сообщение***"[, "***параметрButtons***"][*][*])

Возвращаемые значения и константы диалогового окна MsgBox:

Константа	Значение	Описание	Константа	Значение	Описание
vbOK	1	OK	**vbCancel**	2	Отмена
vbAbort	3	Прервать	**vbRetry**	4	Повторить
vbIgnore	5	Пропустить	**vbYes**	6	Да
vbNo	7	Нет			

Ветвления и выбор

При программировании ветвлений могут применяться конструкции: **If ... Then ...** – когда действия необходимо выполнять только при ***соблюдении условия***, и **If ... Then ... Else ...** – когда необходимо выполнять разные действия в зависимости от соблюдения или ***несоблюдения условия***. Основанием для принятия решений в управляющих конструкциях ветвлений являются **логические** (условные) **выражения**, в которых используются ***операторы сравнения*** [и ***логические операторы***].

В случае, если в конструкциях ветвлений требуется выполнение ***блока операторов***, используются многострочные операторы, имеющие следующие виды:

IF ***логическое_выражение*** **Then**
  оператор1_ЕслиИстина
  оператор2_ЕслиИстина
  оператор3_ЕслиИстина
...
End If

IF ***логическое_выражение*** **Then**
  оператор1_ЕслиИстина
  оператор2_ЕслиИстина
...
Else
  оператор1_ЕслиЛожь
  оператор2_ЕслиЛожь
...
End If

Конструкция **Select Case** (множественный выбор) позволяет обрабатывать в программе несколько условий. Эта конструкция состоит из анализируемого ***выражения*** и ***набора операторов*** **Case** на каждое возможное значение ***выражения***:

Select Case ***селектор***
  Case ***значение1***
    оператор1
  Case ***значение2***
    оператор2
  ...
  Case Else
    операторИначе
End Select,

где ***селектор*** может быть числовым, символьным типом,
а ***значение...*** – числовым, символьным или логическим выражением.

Циклы в Visual Basic

Циклы бывают трех типов – циклы **For** (***со счетчиком***), циклы с ключевыми словами **While** и **Until** (циклы **While** и **Until** бывают ***с предусловием*** и ***постусловием***).

Циклы **For** (***со счетчиком***) выполняют последовательность команд определенное ***счетчиком*** число раз. Конструкция цикла **For**:

For ***Счетчик*** = ***НачальноеЗначение*** **To** ***КонечноеЗначение*** [**Step** ***ЗначениеШага***]
  операторы
Next [***счетчи***к]

Конструкция **Do...Loop** с ключевым словом **While** выполняется до тех пор, пока задаваемое в цикле ***условие истинно***.

Do While *Условие*
  операторы
Loop

Do
  операторы
Loop While *Условие*

Конструкция **Do ... Loop** с ключевым словом **Until** выполняется до тех пор, пока задаваемое в цикле ***условие ложно***.

Do Until *Условие*
  операторы
Loop

Do
  операторы
Loop Until *Условие*

В некоторых случаях требуется **прервать *выполнение цикла*** до его завершения. В этих случаях, для безусловного завершения цикла, может быть использован оператор **Exit** (**Exit For**, **Exit Do**).

Работа с массивами в Visual Basic

Статические массивы

Статические массивы не меняют размерности и размера в процессе выполнения программы. Обычно работа с элементами массива выполняется в циклах, где в качестве индекса выступает счетчик цикла. Пример заполнения одномерного массива:

```
For i = 1 to 10
   q(i) = i ^ 2
Next i
```

* Объем памяти, который требуется для массива, равен произведению байтов, выделяемых для одной переменной, на количество элементов.

Динамические массивы

Динамические массивы используются в том случае, когда количество элементов массива заранее неизвестно и будет определяться в процессе выполнения программы.

* Немаловажно, что занимаемая динамическим массивом память может быть освобождена по окончании его обработки (для этого используется оператор Erase).

Описание динамических массивов осуществляется в два этапа:

1. Объявляют массив без указания размерности, например:
Dim arrA () **As** Byte

2. Когда необходимо, переопределяют размерность массива (задают или изменяют размерность). Пример:
ReDim arrA(5) **As** Byte

Необходимо иметь в виду, что при каждом выполнении оператора ReDim, все значения элементов массива будут потеряны. Для того чтобы значения элементов массива не пропали, следует использовать служебное слово Preserve, например:
ReDim Preserve arrA(10) **As** Byte

Однако в том случае, когда граница не увеличивается, а уменьшается, значения "лишних" элементов все равно будут потеряны.

* Следует обратить внимание, что при помощи ключевого слова Preserve может быть изменена только верхняя граница размерности массива.

В Visual Basic существует возможность определения значений нижней и верхней границы массива, для чего используются функции LBound и UBound... соответственно.

Массивы элементов управления

Кроме массивов числовых и текстовых данных в системе программирования ***Visual Basic 6*** имеется возможность использования массивов объектов (элементов управления). Такие массивы удобно использовать, когда на форме размещаются функционально однотипные элементы управления с однотипными задачами.

Для создания ***массива элементов***, необходимо создать первый объект на форме и скопировать его в буфер обмена. Затем выполнить операцию вставки этого элемента из буфера обмена, после которой на экране появится диалоговое окно с запросом: "Вы хотите создавать массив элементов управления?" При утвердительном ответе на запрос, на форме появится такой же элемент управления, но уже с (уникальным) номером элемента массива (Index), который можно использовать для обращения к элементу. Элементы массива Label обозначаются так: первый – Label(0), второй – Label(1) и т.д.

Работа со строками

Строкой является символьная (текстовая) информация (обозначается в тексте программы в двойных кавычках) и переменные (типа String), предназначенные для работы с такой информацией.

Существует два вида строк:

Строки переменной длины, которые не занимают фиксированного объема памяти. Пример объявления такой строки: **Dim** strA **As** String

Строки постоянной длины. Используются в том случае, когда длина строки указывается при ее объявлении и не может изменяться при выполнении программы, т.е., строка постоянной длины занимает фиксированный объем памяти. Пример объявления: **Dim** strA **As** String * 15, где 15 – длина (количество символов) строки.

Для работы со строками используются ***строковые функции***. Сцепление двух и более строк (***конкатенация***) реализуется с помощью оператора "&" или "+". Функ-

ции Len() определяет: для символьной информации – длину строки, а для иной информации – количество байтов, отводимой для переменной данного типа.

Пользовательские процедуры и функции

Под ***процедурой*** понимается последовательность записанных операций, к которым (для их выполнения) можно обращаться несколько раз и в любой части программы. Существует ***событийные*** и ***пользовательские процедуры***.

Событийными являются процедуры, которые вызываются при каком-либо событии элемента управления. ***Пользовательские*** процедуры – это группы операторов, создаваемые разработчиком для выполнения определенных задач.

Функции – это наборы команд, предназначенные для вычисления тех или иных значений на основании исходных данных. ***Функции*** могут быть ***встроенными*** и ***пользовательскими***. ***Встроенными*** функциями являются функции, имеющиеся в языке ***Visual Basic*** (встроенные функции приведены в разделе "Операторы и функции"). ***Пользовательские*** функции создаются разработчиком.

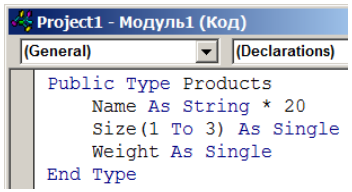
Для создания ***процедуры*** (или ***функции***) можно воспользоваться командой ***Добавить Процедуру*** (меню "**Инструменты**"). Сначала выбираем тип: Sub (процедура) или Функция, указываем ***область*** доступности и подтверждаем создание ***процедуры*** (***функции***). Затем указываем обязательные и необязательные параметры и записываем программный код ***процедуры*** (или ***функции***).

Работа с записями

Записи объединяют несколько переменных различных типов, каждая из которых называется полем. В этих полях могут быть описаны как переменные, так и массивы. Описание ***записи*** должна быть расположена в разделе ***Declarations*** соответствующего ***модуля*** в структуре Type... End Type.

Пример описания записи "**Товары**":

```
Public Type Products
   Name As String * 20
   Size(1 To 3) As Single
   Weight As Single
End Type
```



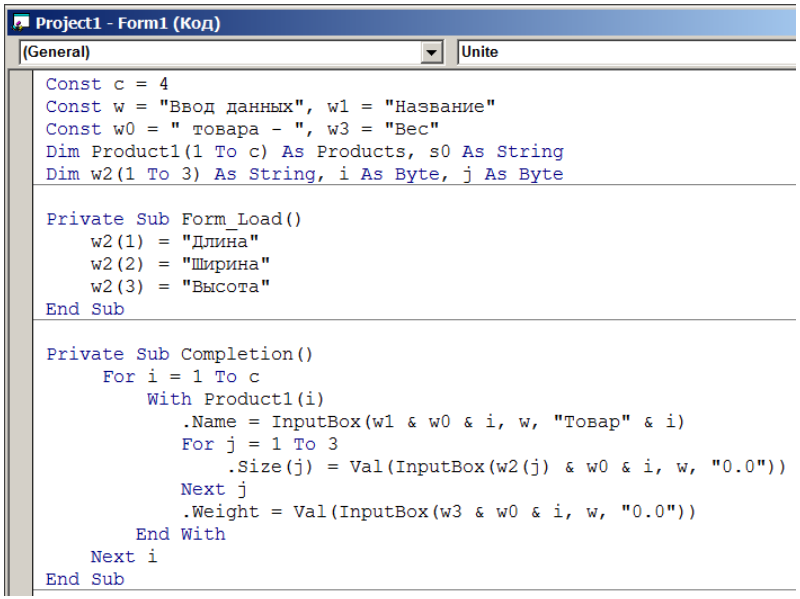
Следует сказать, что доступ к полям записей осуществляется таким же образом, как и к свойствам объектов (Запись.*Поле*). При заполнении полей записи, можно использовать конструкцию With...End With.

Пример кода заполнения полей массива записей приводится ниже:

```
Const c = 4
Const w = "Ввод данных", w1 = "Название"
Const w0 = " товара - ", w3 = "Вес"
Dim Product1(1 To c) As Products, s0 As String
Dim w2(1 To 3) As String, i As Byte, j As Byte

Private Sub Form_Load()
   w2(1) = "Длина"
   w2(2) = "Ширина"
   w2(3) = "Высота"
End Sub

Private Sub Completion()
   For i = 1 To c
      With Product1(i)
         .Name = InputBox(w1 & w0 & i, w, "Товар" & i)
         For j = 1 To 3
            .Size(j) = Val(InputBox(w2(j) & w0 & i, w, "0.0"))
         Next j
         .Weight = Val(InputBox(w3 & w0 & i, w, "0.0"))
      End With
   Next i
End Sub
```

Для объединения полей массива записей в строку используем функцию:

```
Private Function United_String(Product As Products) As String
   Dim s As String
   With Product
      s = .Name
      For j = 1 To 3
         s0 = Space(10 - Len(Str(.Size(j))))
         s = s & s0 & .Size(j)
      Next j
      s0 = Space(10 - Len(Str(.Weight)))
      United_String = s & s0 & .Weight
   End With
End Function
```