Язык программирования скриптов Digitals

Функциональность инструментальных панелей (Окно|Создать панель инструментов) была существенно расширена. В новой версии языка скриптов можно использовать переменные и формулы. Организация условных переходов и циклов позволяет создавать программы скриптов, выполняющих групповую обработку, как на уровне объектов, так и всех открытых карт. Добавлены функции вызова диалогов для создания интерактивных скриптов.

Переменные

Переменные представляют собой именованные пользователем ячейки, которым можно присваивать значения возвращаемые функциями, а также подавать их в функции в качестве параметров. Все переменные должны начинаться с символа **$** и могут состоять из латинских букв, цифр и знака подчеркивания. Предварительное обьявление переменных не требуется и переменная "начинает работать" сразу по мере ее появления в тексте скрипта.

Например:

$C=@Map.Count	в переменную $C будет занесено количество объектов активной карты
@Map.SelectObject $C	будет помечен последний по порядку объект карты

Функции

Функции (команды) позволяют выполнять различные действия над объектами карты и получать информацию об их текущем состоянии. Функции всегда начинается с символа **@** и могут иметь входные параметры, которые следуют после ее имени через пробел. Так, например, для функции **@Map.SelectObject** в качестве параметра указывается номер объекта, который необходимо пометить.

Вызов функции **@Map.SelectObject 1** пометит первый по порядку объект.

Некоторые функции не выполняют никаких действий, но возвращают значение, например **@Map.Count** возвращает количество объектов карты, а **@Map.NextSelected** вернет номер помеченного объекта (или нулевое значение, если ни один объект не помечен). Для использования возвращаемого функцией значения его необходимо присвоить какой-либо переменной.

Например:

$Name=@Map.ClearFilename	занести имя файла карты (без расширения) в переменную $Name
@Map.SaveToFile $Name.in4	сохранит карту в файл формата In4 (заменив расширение на In4)

Выражения

Выражения служат для присвоения переменным значений, которые вычисляются на основании значений других переменных и числовых констант.

Например:

$A=1	заносит в переменную $A значение 1
$B=$A+5	заносит в переменную $B сумму значения переменной $A и числового значения 5
$B=$B+1	увеличивает значение переменной $B на единицу

(в результате выполнения скрипта, переменная **$A** будет содержать значение **1**, а **$B** – значение **7**)

Метки

Метки служат для присвоения имени любой строке программы и дают возможность условного либо безусловного перехода в определенное место программы при помощи функции **@Goto**. Все метки должны начинаться с символа **%** и располагаться в отдельной строке. Обычно используются для организации цикла.

Например:

$Count=@MapCount	заносит количество всех открытых карт в переменную $Count
@If $Count=0 @Break	прерывает выполнение если число открытых карт нулевое
$I=1	присваивает переменной цикла $I начальное значение - 1
%Start	метка - начало цикла
$Name=@Map[$I].ClearFilename	занести имя $I-й карты (без расширения) в переменную $Name
@Map[$I].SaveToFile $Name.dxf	сохранить карту в файл DXF (с тем же именем заменив расширение)
$I=$I+1	увеличивает на единицу номер обрабатываемой карты
@If $I<=$Count @Goto %Start	если номер карты меньше или равен $Count, то перейти на Start

В результате выполнения скрипта все открытые в Digitals карты будут сохранены в формат DXF.

Проверка условия

Для проверки условия служит функция **@If**, которая может быть использована в сокращенном и расширенном формате. Сокращенный формат имеет следующий вид: **@If** Выражение Действие. В качестве действия может использоваться функция (выражение) которая будет выполнена если выражение истинно.

Например:

@If $C=0 @Break вызывает @Break (прерывание скрипта), если значение переменной $C нулевое

Расширенный формат функции @If позволяет использовать более сложные выражения, а также задавать действие, которое выполняется в том случае, когда выражение ложно. Расширенный формат имеет следующий вид **@If** Выражение **then** Действие1 **else** Действие2.

Например:

@If ($I>0) and ($I<=$Count) then @Goto %Next else @Goto %Start

Если значение $I больше нуля и меньше или равно $Count, то перейти на метку %Next (продолжить исполнение), а в противном случае перейти на метку %Start (например вернуться к вводу данных).

Диалоги

Диалоги это специальные функции, которые позволяют скрипту взаимодействовать с пользователем, запрашивая у него необходимые данные, которые заносятся в переменные для дальнейшего использования.

В текущей версии реализованы следующие диалоги:

Dialog.Ask Text

Выводит окно с полем ввода и возвращает введенное пользователем значение.

Например: **$Dopusk=@Dialog.Ask Введите значение допуска**

Если пользователь нажмет в диалоге кнопку **Отмена** или клавишу **Esc**, то выполнение скрипта будет прервано, а при нажатии кнопки **OK** (**Enter**) введенное значение будет занесено в переменную **$Dopusk**.

Dialog.Confirm Text

Выводит окно с запросом подтверждения выполняемого действия и кнопками **OK** и **Отмена**.

Например: **$C=@Dialog.Confirm Удалить объект?**

Если пользователь нажмет в диалоге кнопку **OK,** то переменная **$C** вернет значение **1**, а если кнопку **Отмена,** то значение **0**.

Dialog.Message Text

Выводит текстовое окно с сообщением.

Например: **@Dialog.Message Обработка завершена успешно**

Dialog.Select Title|Text1|Text2|...

Выводит окно с возможностью выбора значения из заданного списка.

Например: **$A=@Dialog.Select Задайте угол разворота|90º|180º|270º**

Если пользователь нажмет в диалоге кнопку **Отмена** или клавишу **Esc**, то выполнение скрипта будет прервано, а при нажатии кнопки **OK** (**Enter**) выбранное значение будет занесено в переменную **$A**.

Dialog.OpenFile Filter Filename

Выводит диалог выбора файла для открытия. Необязательный параметр **Filter** задает маску, а необязательный параметр **Filename** – имя файла по умолчанию.

Например:

$Name=@Dialog.OpenFile

$Name=@Dialog.OpenFile *.in4

$Name=@Dialog.OpenFile *.txt C:\Digitals\1.txt

Если пользователь нажмет в диалоге кнопку **Отмена** или клавишу **Esc**, то функция вернет пустое значение, а при нажатии кнопки **OK** (**Enter**) полное имя выбранного файла будет занесено в переменную **$Name**. Для проверки пустого значения можно использовать функцию **@If**, например **@If $Name= @Break**.

Dialog.SaveFile Filter Filename

Выводит диалог выбора файла для сохранения. Необязательный параметр **Filter** задает маску, а необязательный параметр **Filename** – имя файла по умолчанию. Например:

```
$Name=@Dialog.SaveFile *.dxf
$Name=@Dialog.SaveFile *.txt C:\Digitals\1.txt
```

Если пользователь нажмет в диалоге кнопку **Отмена** или клавишу **Esc**, то функция вернет пустое значение, а при нажатии кнопки **OK** (**Enter**) полное имя выбранного файла будет занесено в переменную **$Name**. Для проверки пустого значения можно использовать функцию **@If**, например **@If $Name= @Break**.

Комментарии

Комментарием считается любая строка скрипта, начинающаяся с символа **;** (точка с запятой). Комментарии не участвуют в выполнении скрипта и могут использоваться для вставки пояснений, а также для временного отключения определенных строк программы.

Массивы

Массивы представляют собой разновидность функций позволяющих напрямую обращаться к различным перечисляемым (списочным) объектам Digitals. Доступ к конкретному элементу массива осуществляется по его порядковому номеру - индексу. Индекс элемента массива указывается в квадратных скобках. Массивы могут быть вложенными, так что элемент массива более высокого уровня может содержать массивы более низкого уровня. Например, каждый элемент массива **Карта** содержит массив **Объекты карты**, а каждый объект, в свою очередь, содержит массив **Параметры объекта**.

В текущей версии программы реализованы следующие массивы:

Map[N]

Массив **Map** содержит все открытые в программе карты. Чтобы получить количество открытых в настоящий момент карт (окон) нужно использовать функцию **MapCount** (не путать с **Map.Count**, которая возвращает количество объектов в активной карте). При работе с одной (активной картой) ее индекс можно опускать.

Например, для сохранения активной карты нужно использовать функцию **Map.Save**, а для сохранения карты открытой во втором по счету окне – функцию **Map[2].Save**.

Все функции, которые имеют приставку **Map.** также можно использовать и с приставкой **Map[N].** выполняя определенные действия над любой из открытых карт, а не только над активной.

Map[N].Object[N]

Массив **Object** содержит все объекты карты. Чтобы получить количество объектов карты нужно использовать функцию **Map[N].Count**. Для получения количества точек в объекте используется функция **Map[N].Object[N].Count**.

Например **Map[1].Object[1].Count** вернет количество точек в первом объекте первой открытой карты, а **Map.Object[2].Count** – число точек во втором объекте активной карты.

Map[N].Object[N].Point[N]

Массив **Point** содержит все точки объекта карты.

Пример скрипта для сохранения всех точек помеченного объекта в текстовый файл:

```
$Sel=@Map.NextSelected
@if $Sel<=0 @Break Необходимо пометить объект
$C=@Map.Object[$Sel].Count
$I=1
%Start
$P=@Map.Object[$Sel].Point[$I]
@Text.Add $P
$I=$I+1
@If $I<=$C @Goto %Start
@Text.Save C:\test.txt
```

Массив **Point** можно использовать не только для получения но и для присвоения координат любой точки. Например вызов **@Map.Object[$Sel].Point[1] $P** присвоит координату первой точки из переменной **$P**.

Помимо доступа ко всем трем координатам, также можно получать/присваивать значения каждой координаты (X,Y,Z) в отдельности, например:

$X=@Map[1].Object[1].Point[1].x или **@Map[1].Object[1].Point[1].z 0**

Map[N].Object[N].Parameter[N]

Массив **Parameter** хранит все значения параметров объекта карты. В отличие от остальных массивов, параметры можно адресовать не только по их порядковому номеру, но также по ID и идентификатору In4.

Например:

```
$Value=@Map.Object[1].Parameter[0]
$Value=@Map.Object[1].Parameter[ID10000]
$Value=@Map.Object[1].Parameter[CN]
```

Подобно точкам, значения параметров можно не только считывать, но и устанавливать, например

@Map.Object[1].Parameter[SC] 0010389 или **@Map.Object[1].Parameter[TX] $NONE**

(встроенная переменная **$NONE** используется для присвоения пустого значения)

Text[N]

Массив **Text** предназначен для работы с текстовыми списками (файлами). Каждый элементов массива представляет собой произвольный набор текстовых строк, которые можно добавлять, изменять, сохранять в файл и загружать из файла. В пределах одного скрипта можно, при необходимости, использовать до 63 разных списков адресуемых как **Text[1], Text[2] ... Text[63]**. Если в скрипте используется всего один текстовый список, то его можно адресовать как **Text** без указания индекса.

Например:

```
@Text.Load C:\test.txt
$Value=@Map.Object[1].Parameter[CN]
Text.Add $Value
@Text.Save C:\test.txt
```

Для работы со списками доступны следующие команды:

Text[N].Load ИмяФайла – загружает список из текстового файла

Text[N].Save ИмяФайла – записывает список в текстовый файл

Text[N].Add СтрокаТекста – добавляет текстовую строку в конец списка

Text[N].Count – возвращает количество строк в списке

Text[N].Clear – очищает список

Text[N].Line[N]

Массив **Line** предназначен для доступа к каждой строке списка по ее порядковому номеру (начиная с единицы). Значение строки можно прочитать (присвоить переменной) и записать – присвоить новое значение из переменной или напрямую.

Например:

```
@Text.Load C:\test.txt
$L=@Text.Line[1]
$Value=@Map.Object[1].Parameter[CN]
@Text.Line[1] $L $Value
```

При необходимости присвоить строке списка пустое значение нужно использовать встроенную переменную $NONE, например **@Text.Line[3] $NONE**

Прочие новые функции

Break Text

Прерывает выполнение скрипта, а при наличии параметра Text выводит окно с текстовым сообщением.

CheckErrors 1/0

Включает/отключает встроенный обработчик ошибок. По умолчанию обработчик включен и в случае возникновения ошибки выполнение скрипта будет прервано с выдачей соответствующего сообщения. Если после этого войти в режим редактирования скрипта, то строка с ошибкой будет подсвечена красным.

Например, такой скрипт выдаст сообщение об ошибке и команда **Dialog.Message** не будет выполнена:

```
$I=0
@Map.SelectObject $I
@Dialog.Message Скрипт завершен
```

А в таком варианте ошибка будет проигнорирована и выполнение скрипта будет продолжено:

```
@CheckErrors 0
$I=0
@Map.SelectObject $I
@Dialog.Message Скрипт завершен
```

В случае отключения контроля ошибок разработчику желательно самому выполнять все необходимые проверки значений переменных, входных данных и т.д. Функцию **@CheckErrors** можно использовать многократно, включая/отключая встроенный контроль ошибок по ходу исполнения скрипта.

Map.ClearFilename

Возвращает полное имя файла карты без расширения, например для карты хранящейся в файле **C:\Digitals\MyMap.dmf** функция вернет строку **C:\Digitals\MyMap**

Функция @Calc

Функция @Calc служит для вычисления значений формул, содержащих математические выражения или операции со строками. Вызов функции имеет следующий вид **$S=@Calc Выражение**. Внутри выражений можно использовать следующие операции и функции:

```
Variable types
  x,y         : numeric - (integer, float)
  a,b         : boolean (1 or 0)
  s,t,v       : string
  d           : DateTimeString  (StampString)

Basic operations
  numeric:          x + y , x - y , x * y, x / y, x ^ y
  compare:          x > y, x < y, x >= y, x <= y, x = y, x <> y
  ansi compare:     s > t, s < t, s >= t, s <= t, s = t, s <> t
  boolean (1/0):    a AND b,  a OR b,  NOT(a)
  set variable :    x:=formula (or value) ;
  destroy variable: FreeVar(s);    // s=variable name
  logical:          ExistVar(s)  // s=variable name
  formula separation with semicolon :  formula1 ; formula2

Type conversion
  boolean (1/0):   Logic(x)
  numeric:         Numeric(s)
  string:          String(x)
  char:            Char(x)
  integer:         Ascii(s)

  all types:       Eval(f)  // where f is formula in [...]
  string :         NumBase(x,base) // base from <2..16>
  integer:         BaseNum(s,base) // base from <2..16>

Math operations
numeric (integer): x Div y,  x Mod y

Math functions
   Abs(x), Frac(x), Trunc(x), Heaviside(x) or H(x), Sign(x),
   Sqrt(x), Ln(x), Exp(x),
   Cos(x), CTg(x), Ch(x), CTh(x), Sin(x),  Sh(x), Tg(x), Th(x),
   ArcSin(x), ArcCos(x), ArcTg(x), ArcCtg(x),
   MaxVal(x [,y, ...]),  MinVal(x [,y, ...]),
   SumVal(x [,y,...]),   AvgVal(x [,y, ...])

String operations
   s || t ,
   s Like t,      // (%,_)
   s Wildcard t   // (*,?)
```

String functions

```
  integer: Length(s), Pos(t,s)
  string:  Trim(s), TrimLeft(s), TrimRight(s), Upper(s), Lower(s),
           Copy(s,x,[y]), CopyTo(s,x,[y]), Delete(s,x,[y]),
           Insert(s,t,x);
           Replace(s,t,v,[1/0=ReplaceAll,[1/0=IgnoreCase]] );
           IFF(a,s,t);    //IF a>=1 then Result:=s else Result:=t
```

Date & Time functions

```
  integer: Year(s), Month(s), Day(s), WeekDay(s),
           Hour(s), Minute(s), Sec(s)
  numeric: StrToStamp(d)  LastDay(x) // last day in Month (28-31)
  string:  StampToStr(x), StampToDateStr(x), StampToTimeStr(x)
```