Государственное образовательное учреждение
высшего профессионального образования
Санкт-Петербургский
государственный технологический институт
(технический университет)

ДОБРО ПОЖАЛОВАТЬ

Кафедра системного анализа

Оператор безусловного перехода

Синтаксис:

GOTO m

m - номер строки или метка.

Пример:

GOTO 10
GOTO мет

Оператор END

Форма записи	Назначение
End If	Конец блока If…Then…Else
End Function	Конец определения функции Function
End Sub	Конец определения процедуры Sub.

Логические выражения

Логическое выражение – это выражение, результатом выполнения которого является ИСТИНА или ЛОЖЬ.

Операции отношения

Операции отношения используются для сравнения арифметических и строковых выражений. Результат операции – логическое значение.

Операция	Название операции	Пример
=	Равно	X=Y
>	Больше	X>Y
>=	Больше или равно	X>=Y
<	Меньше	X<Y
<=	Меньше или равно	X<=Y
<>	Неравно	X<>y

В общем случае операция отношения в VB имеют следующий вид:

выражение1 **операция_отношения** *выражение2*

Пример. A+SIN(X) > B + 3.4

A > 2

Логические операции

Логические операции применяются к величинам логического типа:

Not	Отрицание	**Not A** истинно тогда и только тогда, когда A ложно.
And	Логическое умножение	**A And B** истинно тогда и только тогда, когда истинно A и истинно B.
Or	Логическое сложение	**A Or B** истинно тогда и только тогда, когда A или B истинно.
Xor	Исключающее ИЛИ	**A Xor B** истинно тогда и только тогда, когда значения A и B не совпадают.
Eqv	Эквивалентность	**A Eqv B** истинно тогда и только тогда, когда значения A и B совпадают.
Imp	Импликация	**A Imp B** принимает значение ЛОЖЬ, если A истинно, а B ложно, и значение ИСТИНА, в остальных случаях.

Логические операции приведены в порядке старшинства операций.

Структурированные операторы

Структурированными операторами являются операторы, которые состоят из других операторов. К ним относятся:

- условный оператор **If**;
- оператор выбора **Select Case**;
- операторы цикла.

Условный оператор If

Оператор **If**…**Then**…имеет линейный и блочный синтаксис.

Линейный синтаксис:

If Условие **Then** Операторы_1 [**Else** Операторы_2]

Данный оператор является однострочным, т.е. записывается в одну строчку.

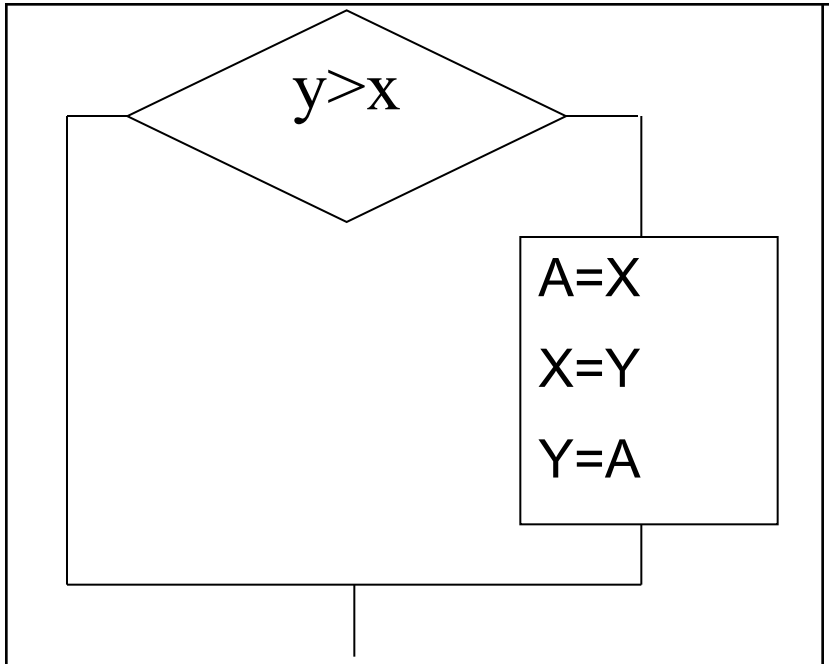
Порядок выполнения операторов:

- Вычисляется значение Условия;
- Если значение Условия «истина», то выполняются Операторы_1, а затем оператор, следующий за условным;
- Если значение Условия «ложь», то выполняются Операторы_2, а затем оператор, следующий после условного.

Пример.

Даны вещественные числа x и y. Присвоить переменной x значение max{x,y}, а y min{x,y}.

$$x = \begin{cases} x, \text{если } x \geq y \\ y, \text{если } x < y \end{cases} \qquad y = \begin{cases} y, \text{если } x \geq y, \\ x, \text{если } x < y \end{cases}$$



If y>x **Then** a=x: x=y: y=a

Полный текст программы:

```
Private Sub Комманда1_Click()
Dim x As Single, y As Single
x = CSng(InputBox("Введите x"))
y = CSng(InputBox("Введите y"))
If y > x Then a = x: x = y: y = a
MsgBox(“x=” + Cstr(x))
MsgBox(“y=“ + Cstr(y))
End Sub
```

Блочный синтаксис

If Условие_1 **Then**
[Блок операторов_1]
[**ElseIf** Условие_2 **Then**
    Блок операторов_2]

……

[**ElseIf** Условие_N **Then**
    Блок операторов_N]
[**Else**
    Блок операторов_N+1]
End If

Блок операторов – один или несколько операторов.

Порядок выполнения:

- Вычисляется значение Условия_1;
- Если значение Условия_1 «истина», то выполняется Блок операторов_1, а затем оператор, следующий после условного;
- Если значение Условия_1 «ложь», то проверяется по порядку каждое условие **Elself**. Как только найдется верное условие, выполняются операторы данного блока, а затем оператор, следующий после условного;
- Если ни одно из условий **Elself** не выполнено (все «ложь»), то выполняются операторы блока **Else** (Блок операторов_N+1).

Преимущества блочной структуры

1. Программа легче читается.
2. Допустимы более длинные операторы и структуры.
3. Допустимо больше сложных условий.
4. Больше структурированности и гибкости, благодаря использованию нескольких условий.

Пример. Рассчитать y от заданного x.

$$y = \begin{cases} x^2, \text{если } x < 0 \\ 2x + 5, \text{если } 0 \le x \le 1 \\ \sqrt{x}, \text{если } x > 1 \end{cases}$$

```
Private Sub Комманда1_Click()
Dim x As Single, y As Single
x = CSng(InputBox("Введите x"))
If x<0 Then
   y=x^2
ElseIf 0<=x And x<=1 Then
   y=2*x+5
Else
   y=Math.Sqrt(x)
End if
MsgBox(“y=“ +CStr(y))
End Sub
```

Операторы цикла

Многократно повторяемая последовательность операторов программы называется ***циклом***.

Для организации цикла в VB можно использовать операторы цикла. Эти операторы состоят из заголовка цикла, тела цикла и конца цикла.

VB поддерживает следующие операторы цикла:

- Оператор цикла **Do/Loop**;
- Оператор цикла **While/Wend**;
- Оператор цикла **For /Next**;
- Оператор **For Each/Next.**

Оператор цикла Do/Loop

Оператор цикла **Do/Loop** это управляющий оператор, который повторяет блок операторов, пока условие – «истина», или до тех пор, пока условие «ложно».

Синтаксис:

Do [{**While/Until**} Условие_1]
  Операторы
[**Exit Loop**]
……
Loop [{While/Until} Условие_2]

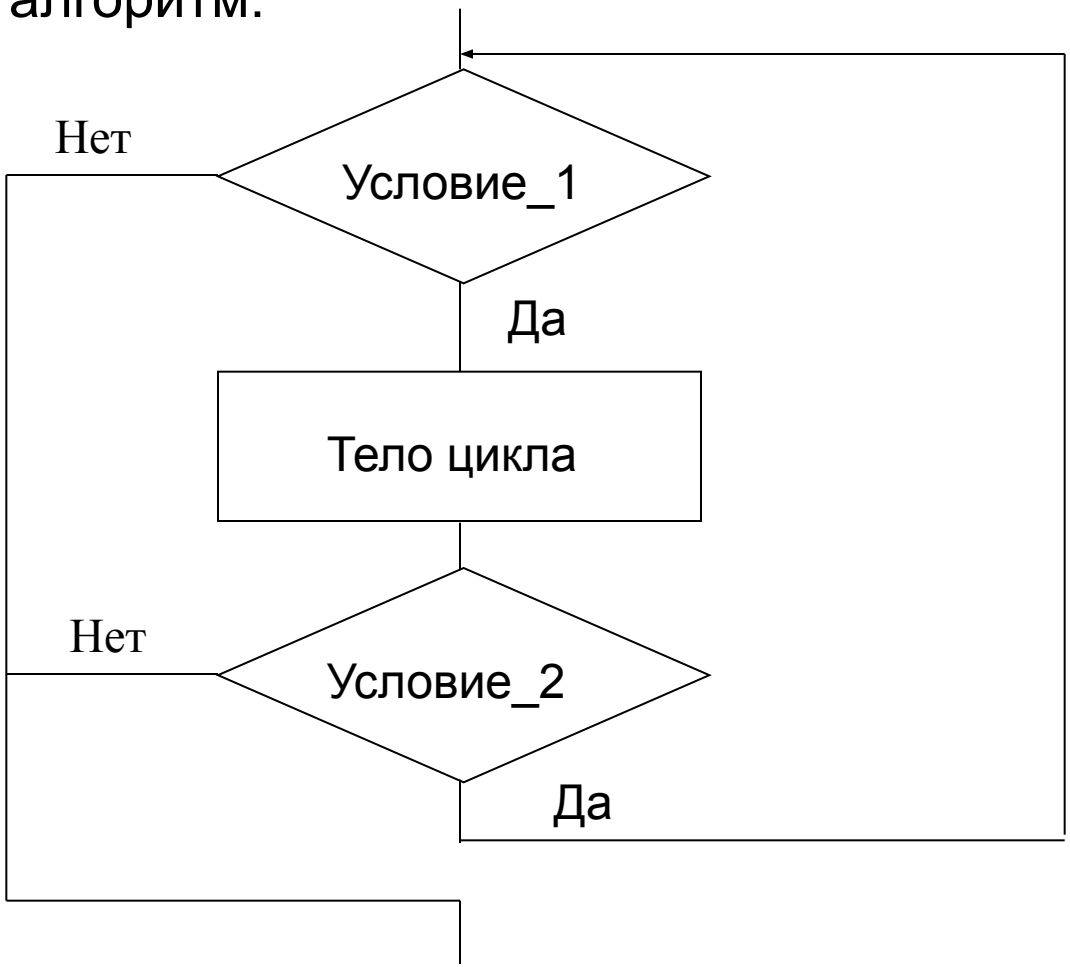
Оператор цикла **Do/Loop** позволяет создавать циклы, с проверкой условия завершения в *верхней* части цикла, в *нижней* части цикла и в *обеих частях одновременно.*

While - используется при необходимости *повторения* цикла, если *Условие - истинно* и его прекращения, если *Условие ложно*.

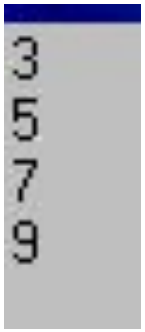
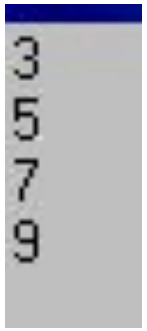
Until –выполнение оператора цикла противоположно **While,** т. е. цикл будет *прекращен*, если *Условие – истинно* и *повторяться*, если *Условие ложно*.

В любом месте цикла можно использовать оператор *альтернативного выхода* **Exit Loop**. В этом случае управление передается оператору, который стоит после завершения оператора цикла.

Оператору цикла **Do/Loop** в общем случае соответствует следующий алгоритм:



Пример.
Что будет напечатано?

Private Sub Комманда1_Click() x = 3 Do While x < 10 Print x: x = x + 2 Loop End Sub	Ответ: 
Private Sub Комманда1_Click() x = 3 Do Until x > 10 Print x: x = x + 2 Loop End Sub	Ответ: 

Массивы

Переменные бывают простыми переменными и переменными с индексами, образующими массив.

Массив – это последовательность (совокупность) величин одного типа, обозначенным одним именем и отличающихся индексом.

Отдельные величины, образующие массив называются элементами массива. Элементы массива определяются именем массива и индексом, заключенным в скобки.

Элементы массива образуют переменные с индексом. Индекс указывает положение элемента в массиве. Элемент массива имеет столько индексов, какова размерность массива.

При использовании массива нужно указать не только тип элементов массива, но и сколько ячеек памяти необходимо зарезервировать для данного массива. Поэтому в программном коде, прежде чем начнутся операции с элементами массива, массив нужно описать.

При описании массивов указывается ***количество индексов*** и ***наибольшее значение*** каждого индекса, т.е. задаются *размерность* и *размер* массива.

Размерность – количество индексов (одномерный, двумерный). *Размер* массива – количество элементов массива.

Различают ***статические*** и ***динамические*** массивы.

Границы ***статического*** массива устанавливаются на этапе разработки и могут меняться только в новой версии программы.

Динамические массивы изменяют свои границы в ходе выполнения программы. С их помощью можно динамически задавать размер массива в соответствии с конкретными условиями.

Объявление массивов

Объявление ***статического*** массива используется оператор ***Dim*** с указанием в круглых скобках после имени массива *границ индексов*:

Пример:

Dim A(6) As Single – объявление одномерного массива с элементами A(0),A(1),…,A(6) вещественного типа.

Dim *X(4,5)****As Integer*** **-** объявлен массив(матрица), имеющий 5 строк и 6 столбцов.

Массивы (как и переменные) могут быть локальными, контейнером и глобальными. Как и при объявлении переменных при объявлении массива вместо ключевого слова ***Dim*** можно использовать: ***Public***, ***Static, Private.*** Ключевое слово ***Private*** как и ***Dim*** означает, что массив локальный.

Объявление динамического массива

Динамический массив создается в два этапа:

Определить массив

```
Dim R( ) As Single
```

С помощью оператора ReDim установить фактический размер массива.

Пример

```
Dim A() As
Dim n as integer
N= Csng(inputBox(“n=“))
Redim A(0 to n)
```