Университет машиностроения

Кафедра «Автоматика и процессы управления»

Дисциплина

Информационные технологии

2 семестр

Тема 04

Основные алгоритмические конструкции

(Язык VBA for Ms Excel)

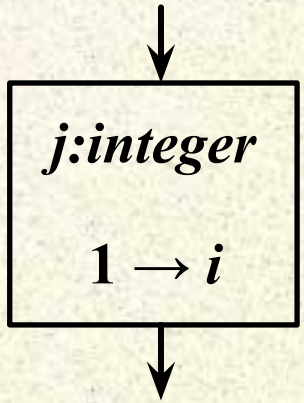
Инициализация

```
Dim  A

Dim  B  As Integer

Dim  C  As Integer = 1

D  =  1   'Неявное  объявление  через  присвоение
          значения

Function MyFunc(E As Integer) As Byte

'Объявление как аргумент функции
```

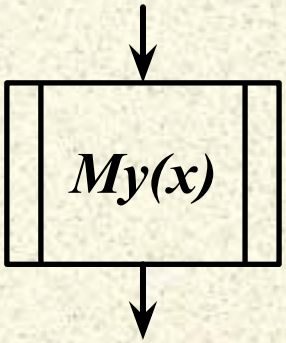
```
Y = Sin (X)

Function My (byVal F As Integer) As Byte

'Возврат значения при передаче параметра по-ссылке
```

Отображение:

Блок процесса



Сложные вычисления либо вызов функции с возможной неявной установкой значения



Вычисления и обработка данных

A = B + C / (D – E)

****

Y = Math.Sin(X) ‘ Вызов «стандартной функции»

****

IF (A>B) THEN Max = A ELSE Max = B

‘ Простая алгоритмическая конструкция, которая
‘ может интерпретироваться как очевидная
‘ «алгоритмическая» функция

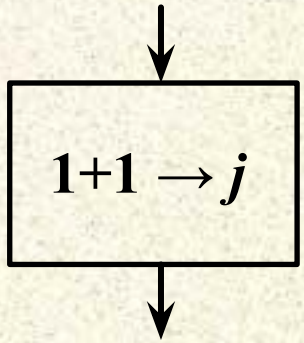
Y = GetSmthValue (SmthParams)

Call DoSmth (SmthParams)

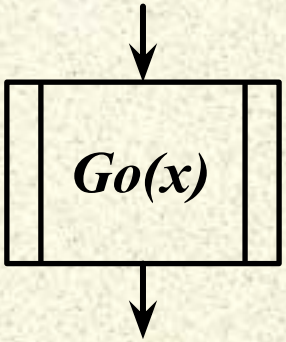
‘Вызов функции с возвратом значения и процедуры

Отображение:

Блок процесса



Предопределенный процесс (функция)



Блоки процесса можно объединять!!!

Потоки исполнения – нити (*threads*) и процессы

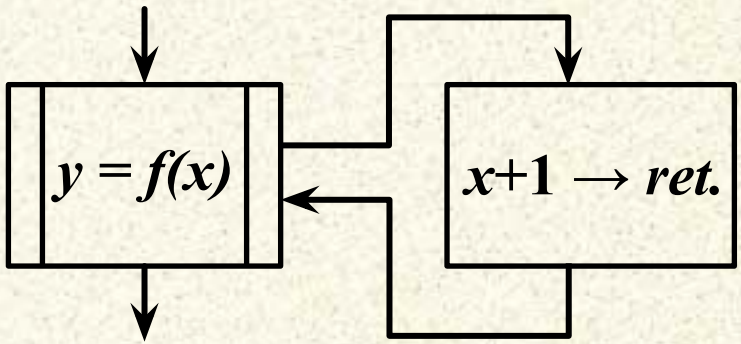
Поток исполнения – фрагмент исходного кода, в виде блока инструкций, которые должны выполняться последовательно.

На логическом уровне (при рассмотрении «логики» приложения) поток исполнения может включать ветвления и вызов функций.

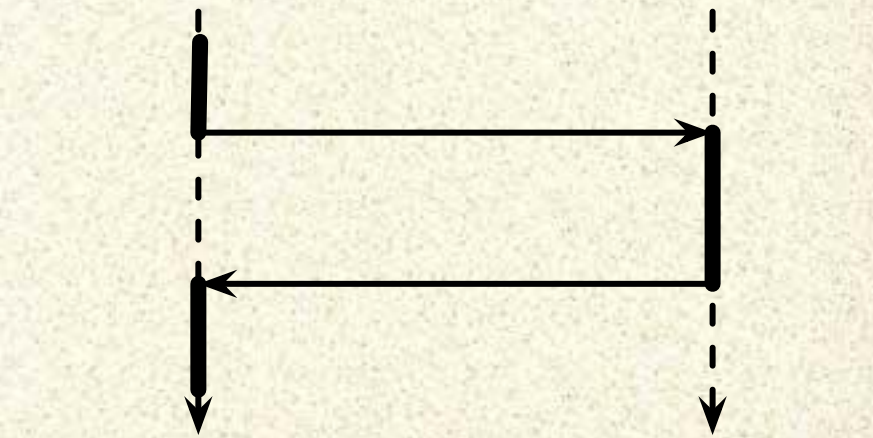
На уровне реализации поток получает исполнителя (виртуальный процессор) для независимого исполнения, либо объединяется с другим потоком (встраивается)

Вызов функции

Функциональное представление

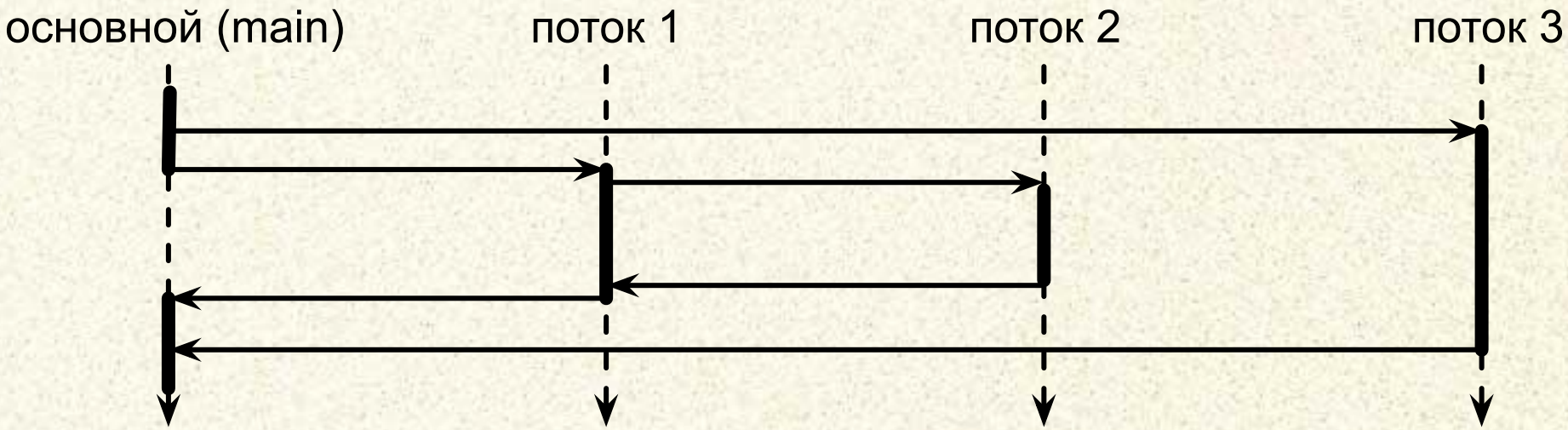


Представление потоков



Многопоточность, конкуренция за ресурсы

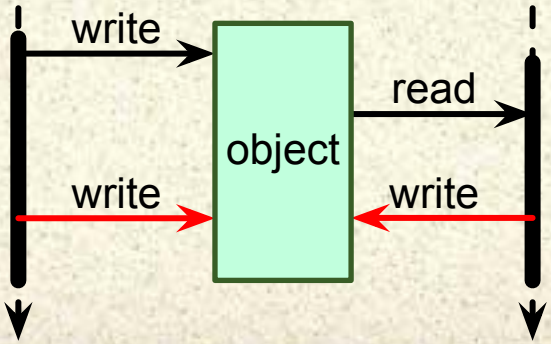
При наличии нескольких исполнителей, в системе может поддерживаться и быть реализована многопоточность.



Существует две реализации потоков: нить *(thread)* и процесс *(process)*.

Реализации многопоточности: невытесняющая (фоновая), кооперативная и приоритетная (вытесняющая). В одноядерных системах используется временно́е мультиплексирование (переключение контекста процессора).

Важная проблема многопоточности – конкуренция за ресурсы (доступ к общим ячейкам памяти и объектам)



Особенности исполнения кода VBA

Поскольку концептуально исполнение кода на VBA эквивалентно действиям пользователя в интерфейсе хост-приложения, то на уровне реализации код (инструкции) VBA всегда выполняется в одном потоке исполнения.

Во время исполнения кода интерфейс хост-приложения «заморожен».

Исполнение кода может быть приостановлено постановкой на паузу (Ctrl + Break или из меню IDE VBA), прервано ошибкой исполнения или точкой отладки *(Breakpoint)*.

Во время приостановки интерфейс хост-приложения будет доступен, в том числе и для редактирования данных в документах.

Однако при этом:

- **Исполнение другого потока кода VBA будет невозможно.**
- **Формулы в ячейках, содержащие пользовательские функции, будут возвращать неопределённые значения.**
- **Обработчики событий на пользовательских формах и в документах не будут вызываться. При этом сами элементы управления будут активны, например, для ввода значений.**

Операции ввода-вывода

Операция ввода-вывода подразумевает обмен данными между объектами, непосредственно используемыми в вычислениях, и внешними сущностями и (или) их инкапсуляциями в виде объектов.

Print #1, *smthvalue* ‘Вывод значения в 1-й поток

Line Input #1, *smthvar* ‘Загрузка текстовой строки

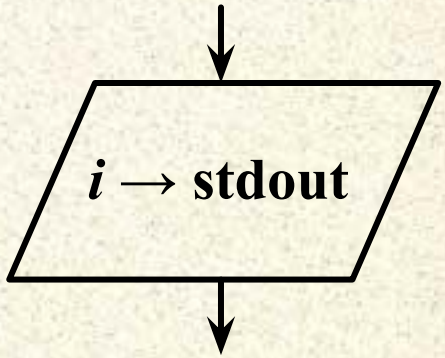
Ввод-вывод начальных и исходных значений разумнее обозначать специальными блоками для повышения читаемости алгоритма

Debug.Print *smthvalue* ‘Вывод в отладочное окно

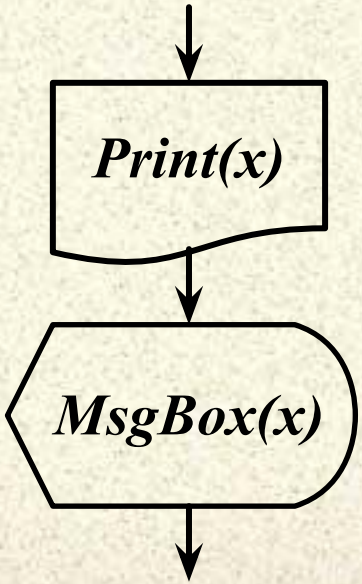
MsgBox (*smthvar*) ‘Использование диалогового окна

Отображение:

Блок ввода-вывода



Вывод результатов вычислений

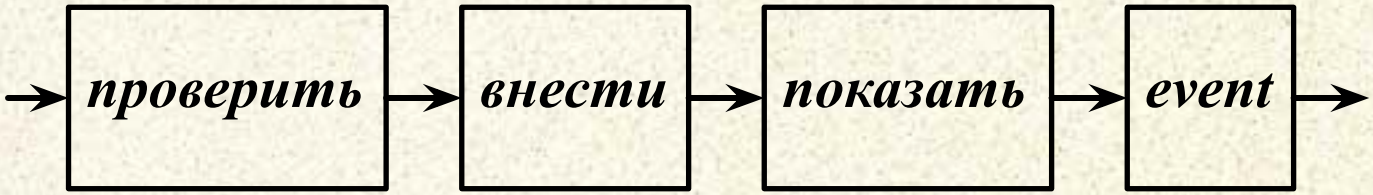


Объектное представление визуальных компонент и «синтаксический сахар»

Программная реализация визуальных компонент (элементов управления) и элементов документов инкапсулирована (скрыта) в соответствующих им объектах и управление ими выполняется через соответствующие методы и свойства.

Согласно принципам ООП доступ к свойствам реализован через акцессоры *(accessor)* и мутаторы *(mutator)*.

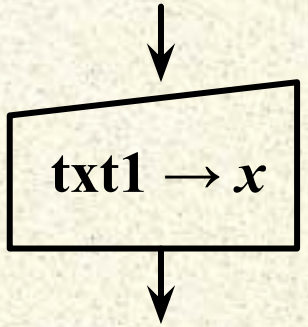
Используются специальные методы вместо «прямого» доступа к памяти объекта потому, что есть дополнительная функциональность!



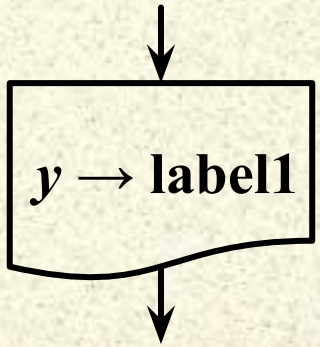

```
TextBox1.SetText("Hello")  'Использование мутатора
TextBox1.Text = "Hello"  'Присвоение (через «сахар»)
```

Отображение:

Ввод исходных данных



Вывод результатов вычислений



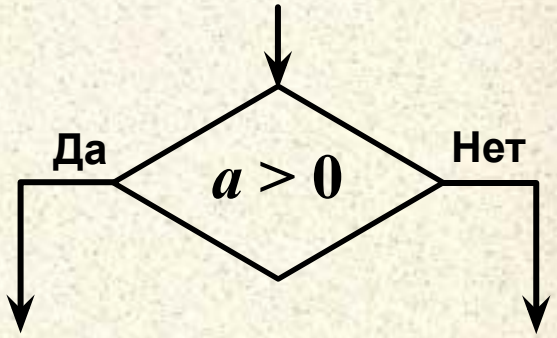
Управляющие конструкции: Простое ветвление

Изменение порядка следования инструкций

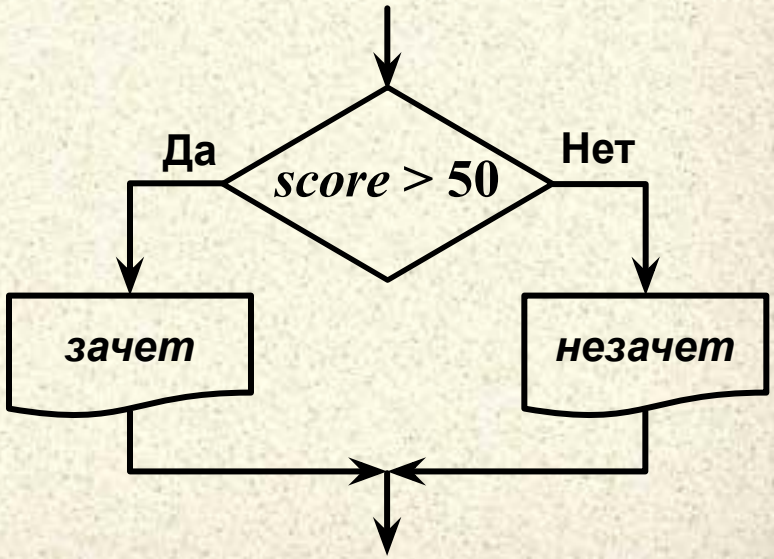
```
IF (condition) THEN statements
‘ Однострочная запись с положительной ветвью

IF (condition) THEN statements ELSE statements
‘ Однострочная запись с двумя ветвями

‘Многострочная запись
IF (condition) THEN    ‘ если …. то
    Statements
ELSE                   ‘ иначе …
    Statements
END IF
```

Отображение:

Блок «решение» с проверкой условия



Пример вывода оценки за тест



Управляющие конструкции: Каскад по отрицательной ветви

Использование уточняющих условий

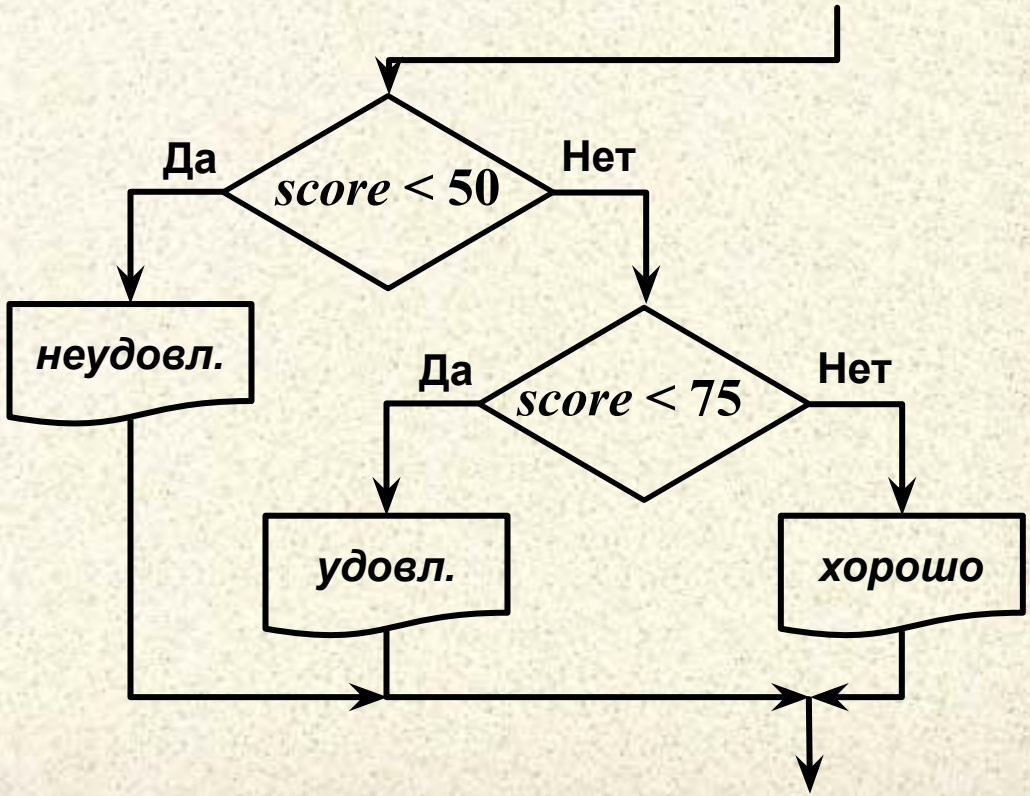
```
If (Condition) Then    ' Первое условие отбора
    Statements
ElseIf (Condition) Then ' Дополнительное условие
    Statements
Else ' Остальные случаи
    Statements
End If
```

Из стандарта оформления кода:

- **Использование отступов для выделения внутренних блоках**

Отображение: Каскад обычно «растет» вправо-вниз

Пример вывода дифференцированной оценки за тест



Управляющие конструкции: Проверка состояния

Выбор – это альтернативная реализация множественного ветвления через анализ значения одной переменной «простого» типа

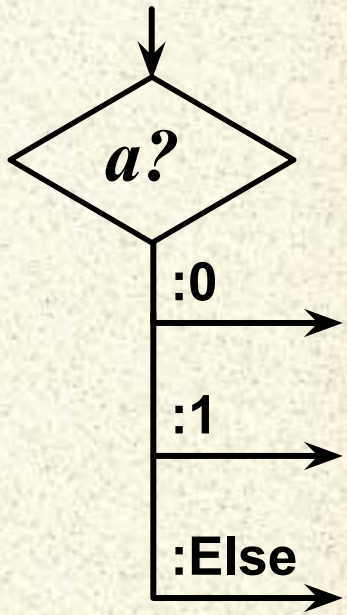
```
SELECT CASE property   'Многострочная запись
    CASE Expressions: Statements
    …
    CASE ELSE: Statements
END SELECT
```

Способы записи условий выбора:

```
Case 5, 6: smth  ' Явное указание списка значений
Case 5 To 7: smth  ' Указание диапазона
Case Is>7: smth  ' Указание условия на значение
```

Отображение:

Блок «решение» с проверкой состояния



Управляющие конструкции:

Организация циклов

Принято различать циклы:

- **С предусловием**
- **С постусловием**
- **Со счетчиком**

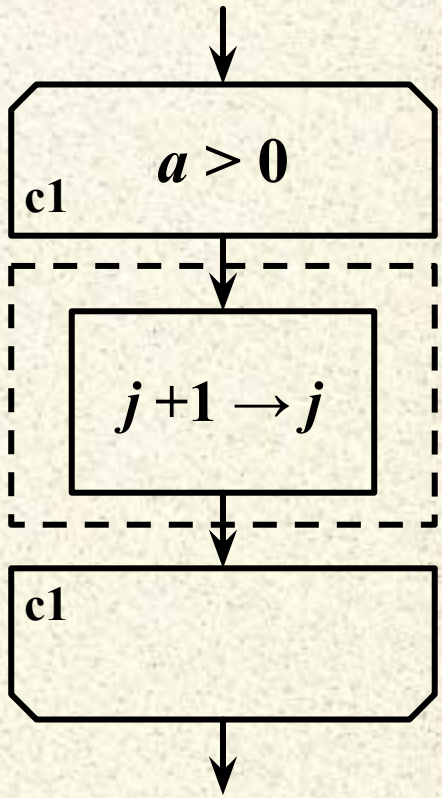
```
Do While … Loop   ‘ Цикл «пока» с предусловием
Do … Loop While   ‘ Цикл «пока» с постусловием
For … Next        ‘ Цикл со счетчиком
```

Альтернативные формы:

```
While … Wend      ‘ Цикл «пока» с предусловием (устар.)
Do … Loop Until   ‘ Цикл «пока не» с постусловием
Do Until … Loop   ‘ Цикл «пока не» с постусловием
```

Отображение:

Блок «циклическое повторение»*



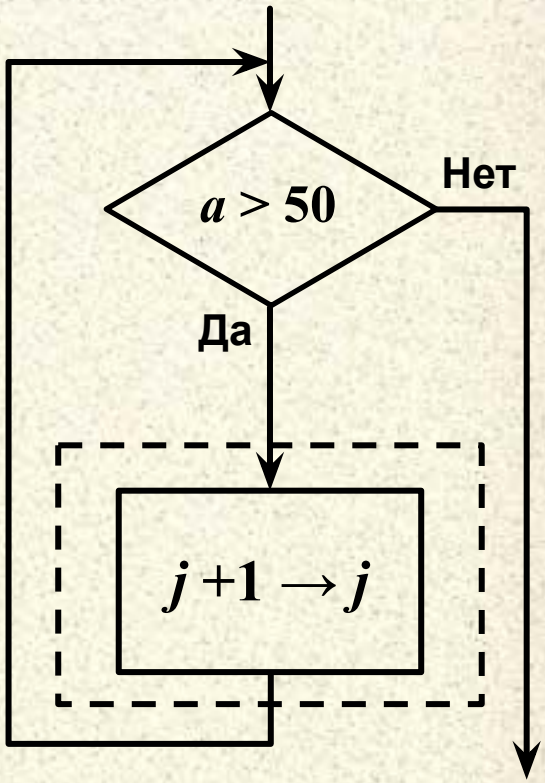
* *Примечание: Это основная форма согласно ГОСТ 19.701-90*

Управляющие конструкции:

Реализация цикла «пока» с предусловием

Отображение:

Повторять пока условие истинно



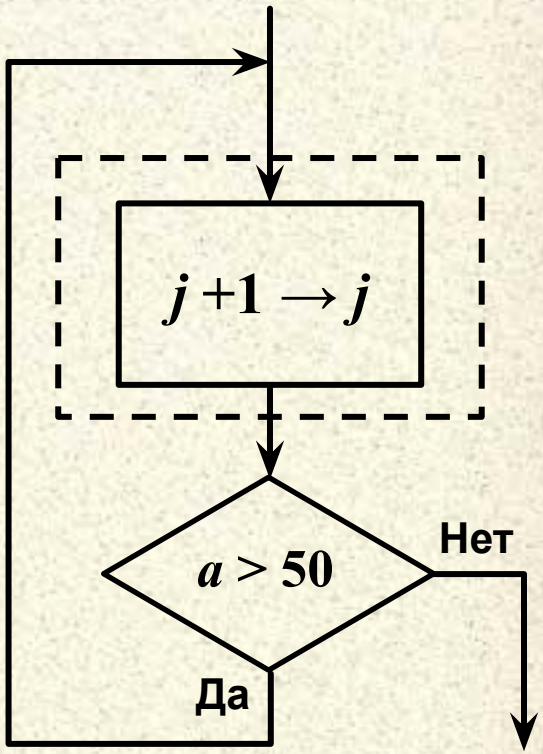

```
Do While (Condition) ‘ Сначала проверяем условие
    Statements
Loop

‘ либо

While (Condition) ‘ Сначала проверяем условие
    Statements
Wend
```

Управляющие конструкции:

Реализация цикла «пока» с постусловием

Отображение:

Повторять пока условие истинно




```
Do                              ‘ Сначала выполняем
    Statements
Loop While (Condition) ‘ Потом проверяем
```

Альтернативная форма – «пока не» (*Until*)

While (NOT(*Condition*)) ⇨ **Until (*Condition*)**

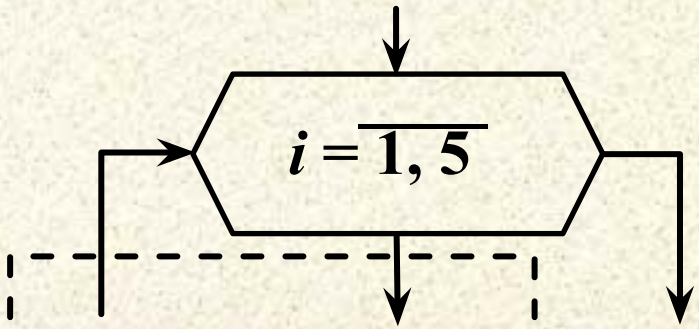
Управляющие конструкции:

Реализация цикла со счетчиком

```
I = 1                  ‘ Инициализация счётчика
While ( I <= 50)       ‘ Проверка достижения границы
    Statements         ‘ Что-то делаем
    I = I + 1          ‘ Наращиваем счётчик
Wend                   ‘ Закрывающая скобка
```

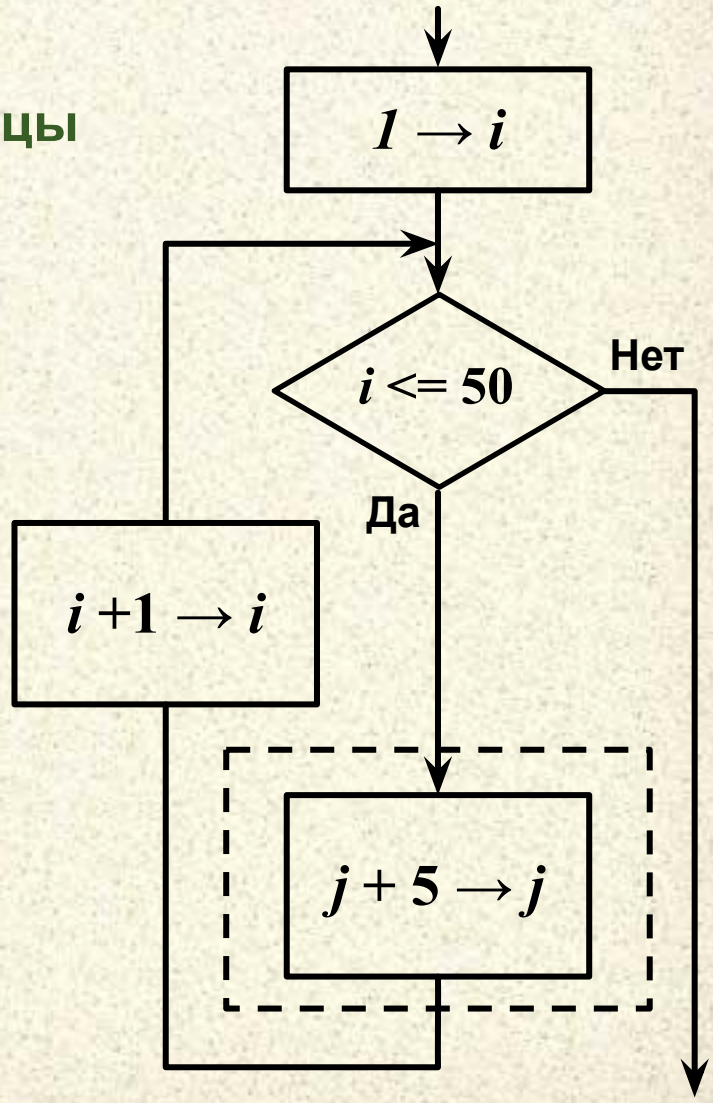
Альтернативная форма – «для» (*For*)

```
FOR I = 1 TO 5 STEP 1
    Statements
NEXT I
```



Отображение:

Повторять пока счётчик не достигнет границы



Однако…

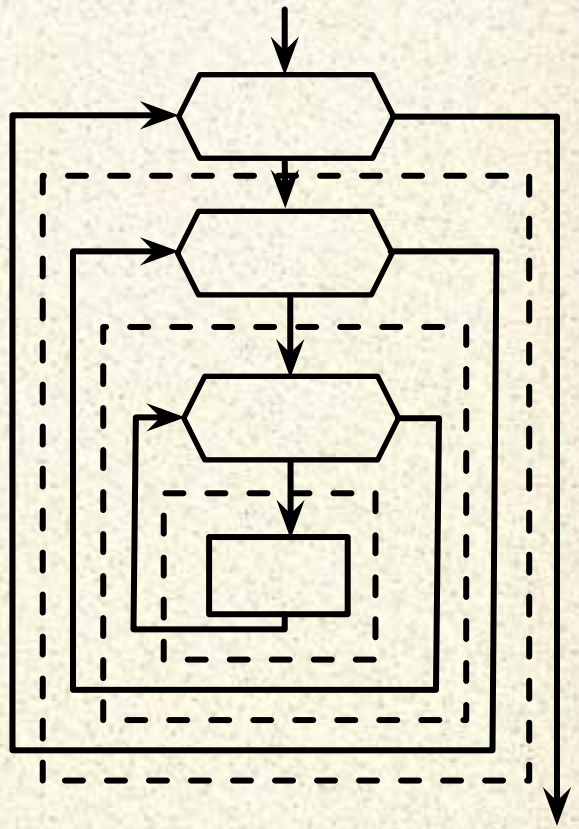
Структурный код «громоздок»

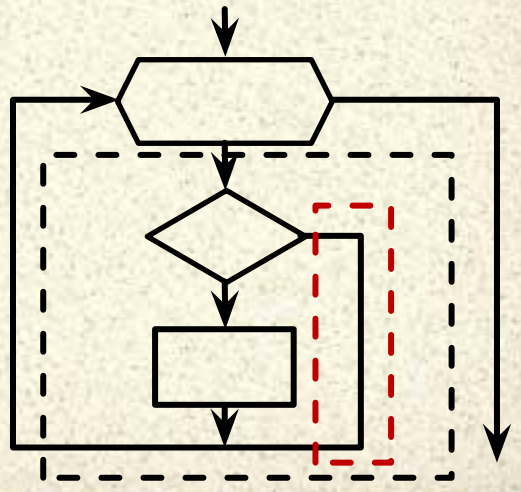
Цикл со счетчиком предполагает доступ к переменной счетчика «только по чтению» и только в коде тела цикла

```
FOR I = 1 TO 5 STEP 1
    X = I + 1  ' Можно
    I = 5  ' Нельзя
NEXT I
X = I + 1  ' Нельзя
```

Циклы не могут пересекаться – операторы цикла полностью изолируют тело цикла. Выйти из «лабиринта» вложенных циклов можно только пройдя все итерации. При невозможности досрочного выхода нужно создавать «ветви обхода».

«Лабиринт» тройного цикла





Другие :

Безусловный переход (Goto)

Можно реализовать «пропускание» тела цикла, подпрограммы…

```
FOR I = 1 TO 5 STEP 1
    If I > 3 Then GoTo FastNext1 ' Переход в конец
    Statements
FastNext1:
NEXT I
```

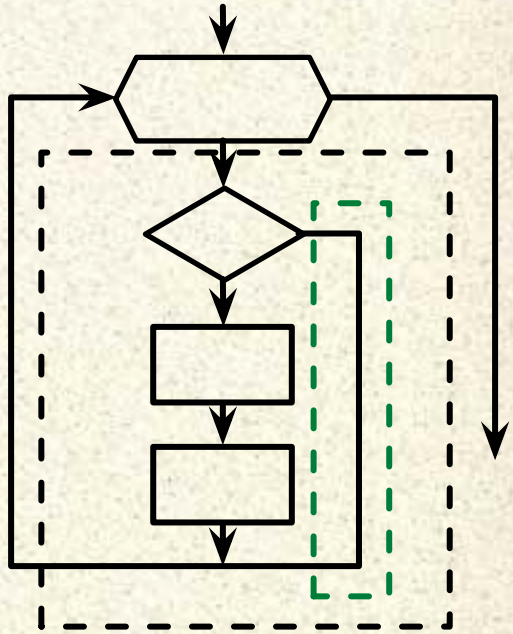
…обработчики или циклы

```
loop1:
    Statements
If (Condition) Then Goto loop1
```

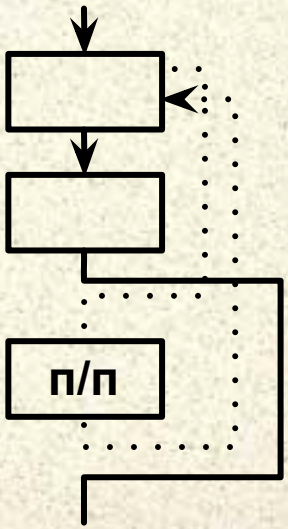
Ухудшается «читаемость» → «тарелка спагетти»
+ нет логичного повода сделать отступы

Отображение:

Обход тела цикла



«Подпрограмма»



Дополнительные конструкции:
Перехват исключений (on Error)

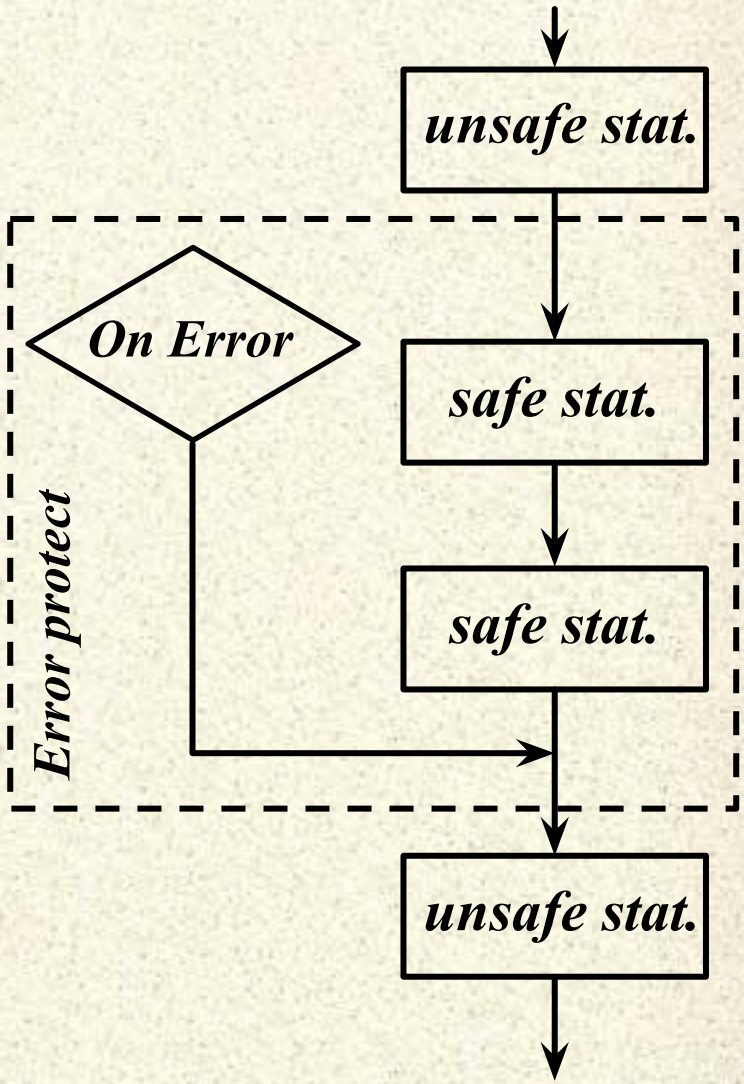
```
I = 1                    ‘ «Незащищенная» инструкция
On Error Goto endcalc1   ‘ Переход при ошибке
    Statements           ‘ Что-то делаем «под защитой»
endcacl1:                ‘ Метка для перехода
On Error Goto 0 ‘ Отключение «перехватчика»
Statements
```

Реализуемы «обработчики» и «сбросы»

```
On Error  Goto  ErrorHandler1
    Statements
    Exit Sub
ErrorHandler1:
    MsgBox(“Ошибка №” & Str( Err.Number ))
    Resume 0
```

```
On Error  Resume Next   ‘ Пропустить ошибку
On Error  Resume 0  ‘ Повторить ошибочный оператор
```

Отображение:
Условный блок
«на исключение»



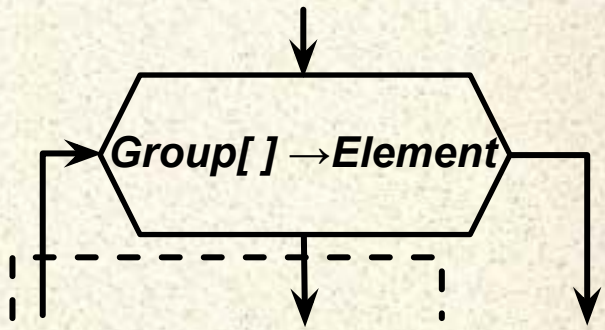
Дополнительные конструкции:

Перебор коллекции

```
For Each Element In Group
    Statements ' Обработка Element
NEXT Element
```

Отображение:

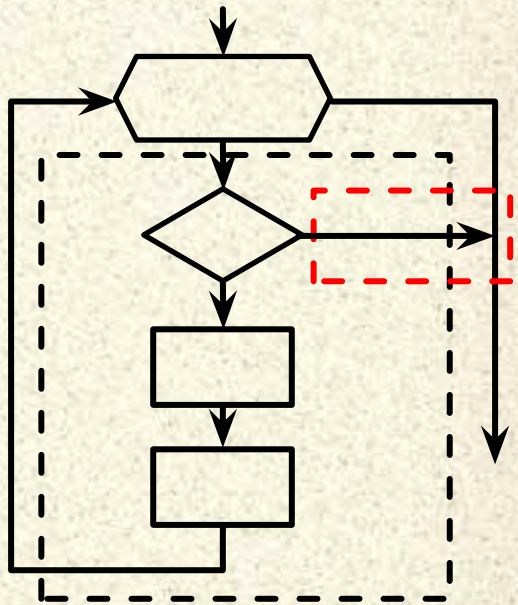
Модификация



Прерывание исполнения блока

```
Exit For  ' Прерывание цикла For (досрочный выход)
Exit Do   ' Прерывание цикла Do
Exit Sub  ' Досрочный выход из процедуры
End       ' Завершение программы
```

Прерывание цикла



```
For i = 1 To 5
    Statements
    If i > 3 Then Exit For
Next I
MsgBox("Цикл завершен")
```

Спасибо за внимание !!!

Далее:

- **Общие вопросы проектирования**
- **Обработка данных**
- **Основные алгоритмические конструкции**
- **Сложные типы данных. Работа со строками**

…

Контакты:
mami.testolog.ru
timid@mami.ru
inform437@gmail.com