

ifflowpseudo-1.ppt [Protected View] - PowerPoint

FILE HOME INSERT DESIGN TRANSITIONS ANIMATIONS SLIDE SHOW REVIEW VIEW ADD-INS STORYBOARDING

PROTECTED VIEW Be careful—files from the Internet can contain viruses. Unless you need to edit, it's safer to stay in Protected View. Enable Editing

Simple IF - processing on Y only

```
graph TD; A{IF AMT > 5000} -- N --> B(( )); B --> A; A -- Y --> C[AMT x 10 = ANS]; C --> B;
```

`if amt > 5000
ans = amt * 10
end if`

— cikon left

Assignment sign

In the flowchart the end if is the circle.

This is a simple IF. If the answer to the question is yes then processing is to be done. If the answer to the question is no then processing is not done. After the IF is over, we need to close up the IF as shown in the diagram and in the pseudocode.

SLIDE 2 OF 20 COMMENTS 107%

Protected View: Be careful—files from the Internet can contain viruses. Unless you need to edit, it's safer to stay in Protected View. Enable Editing

Simple IF - processing on Y or N

```
graph TD; A{IF AMT > 5000} -- Y --> B[ANS = AMT x 10]; A -- N --> C[ANS = AMT x 5]; B --> D(( )); C --> D;
```

```
if amt > 5000
  ans = amt * 10
else
  ans = amt * 5
end if
```

In this example, if the AMT is greater than 5000 then we want to multiply AMT by 10 and store the answer in ANS. If the AMT is not greater than 5000 then we want to multiply AMT by 5 and store the answer in ANS.

This is a simple IF. Different processing will happen depending on the answer to the question. Note that the yes and the no are joined together at the bottom to indicate that the decision has been terminated.

SLIDE 3 OF 20 | NOTES | COMMENTS | 107%

ifflowpseudo-1.ppt [Protected View] - PowerPoint

FILE HOME INSERT DESIGN TRANSITIONS ANIMATIONS SLIDE SHOW REVIEW VIEW ADD-INS STORYBOARDING

PROTECTED VIEW Be careful—files from the Internet can contain viruses. Unless you need to edit, it's safer to stay in Protected View. Enable Editing

AND relationship

```
graph TD; D1{IF INVCD = "A"} -- N --> J1(( )); D1 -- Y --> D2{IF AMT > 5000}; D2 -- N --> J1; D2 -- Y --> P[MOVE "OKAY" TO MSG]; P --> J1; J1 --> J2(( ));
```

In most programming languages the move "OKAY" to msg would be written as msg = "OKAY". The OKAY literal is enclosed in quotes.

```
if invcd = "A"  
  if amt > 5000  
    move "OKAY to msg"  
  end if  
end if
```

msg = "OKAY"

literal

Both statements must be true for the msg to receive the word OKAY. If either statement is false, no processing is done.

This is two IF questions in a simple AND relationship. This means that both questions have to be true for processing to be done. INVCD must be = A AND AMT must be > 5000 for processing to be done. In the simple AND, both answers must be YES.

SLIDE 4 OF 20

ifflowpseudo-1.ppt [Protected View] - PowerPoint

FILE HOME INSERT DESIGN TRANSITIONS ANIMATIONS SLIDE SHOW REVIEW VIEW ADD-INS STORYBOARDING

PROTECTED VIEW Be careful—files from the Internet can contain viruses. Unless you need to edit, it's safer to stay in Protected View. Enable Editing

AND relationship

This can be written using simple if statements or using a compound to say if both are true set msg = "OKAY"

```
graph TD; D1{IF INVC = "A"} -- N --> M1(( )); D1 -- Y --> D2{IF AMT > 5000}; D2 -- N --> M1; D2 -- Y --> P[MOVE "OKAY" TO MSG]; P --> M2(( )); M2 --> M1; M1 --> Exit(( ));
```

```
if invcd = "A"
  if amt > 5000
    move "OKAY to msg"
  end if
end if

if invcd = "A" and amt > 5000
  move "OKAY" to msg
end if
```

Because the no on both of these has no processing, this could be written as a compound. You can only write a compound if the processing for the no is the same for both questions.

SLIDE 5 OF 20

COMMENTS

12:40 PM 9/30/2014

iffloppseudo-1.ppt [Protected View] - PowerPoint

FILE HOME INSERT DESIGN TRANSITIONS ANIMATIONS SLIDE SHOW REVIEW VIEW ADD-INS STORYBOARDING

PROTECTED VIEW Be careful—files from the Internet can contain viruses. Unless you need to edit, it's safer to stay in Protected View. Enable Editing

AND relationship

```
graph TD; D1{IF INVCD = "A"} -- N --> A1[MOVE "PROBLEM" TO MSG]; D1 -- Y --> D2{IF AMT > 5000}; D2 -- N --> A2[MOVE "PROBLEM" TO MSG]; D2 -- Y --> A3[MOVE "OKAY" TO MSG]; A1 --> J(( )); A2 --> J; A3 --> J;
```

Since these are the same I can use a compound and under the else I can move the literal problem to msg.

```
if invcd = "A"
  if amt > 5000
    move "OKAY" to msg
  else
    move "PROBLEM" to msg
  end if
else
  move "PROBLEM" to msg
end if
```

```
if invcd = "A" and amt > 5000
  move "OKAY" to msg
else
  move "PROBLEM" to msg
end if
```

On this slide the no on both decisions is the same, so again you can use separate questions or a compound.

SLIDE 6 OF 20

Protected View: iflowpseudo-1.ppt [Protected View] - PowerPoint

IF Statement: Screenshots and pseudocode

```
graph TD; A{IF INVCD = "A"} -- N --> B[MOVE "PROBLEM" TO MSG]; A -- Y --> C{IF AMT > 5000}; C -- N --> D[MOVE "PROBLEM" TO MSG]; C -- Y --> E[MOVE "OKAY" TO MSG]; B --> F(( )); D --> F; E --> F; F --> G(( ));
```

Same
So I can do a compound

```
if invcd = "A"
  if amt > 5000
    move "OKAY" to msg
  else
    move "PROBLEM" to msg
  end if
else
  move "PROBLEM" to msg
end if
```

```
if invcd = "A" and amt > 5000
  move "OKAY" to msg
else
  move "PROBLEM" to msg
end if
```

On this slide the no on both decisions is the same, so again you can use separate questions or a compound.

SLIDE 6 OF 20

Microsoft PowerPoint interface showing a slide titled "AND relationship". The slide contains a flowchart, a code block, and handwritten annotations.

Flowchart:

```
graph TD; A{IF INVCD = "A"} -- N --> C(( )); A -- Y --> B{IF AMT > 5000}; B -- N --> D[MOVE "PROBLEM" TO MSG]; B -- Y --> E[MOVE "OKAY" TO MSG]; D --> C; E --> C; C --> F(( ));
```

Code Block:

```
if invcd = "A"  
  if amt > 5000  
    move "OKAY" to msg  
  else  
    move "PROBLEM" to msg  
  end if  
end if
```

Handwritten Annotations:

- Blue circle around the flowchart with text: "Not Scamp So no compound"
- Text: "One of the N does nothing and the other moves a literal."

Footer:

In this case, the conditions have different processing if the answer to the question is no. If INVCD is not equal to A, no processing is done. If AMT > 5000 is no then the PROBLEM message is moved. Therefore, the questions must be asked individually. Compound pseudocode or coding cannot be used.

ifflowpseudo-1.ppt [Protected View] - PowerPoint

FILE HOME INSERT DESIGN TRANSITIONS ANIMATIONS SLIDE SHOW REVIEW VIEW ADD-INS STORYBOARDING

PROTECTED VIEW Be careful—files from the Internet can contain viruses. Unless you need to edit, it's safer to stay in Protected View. Enable Editing

AND relationship

```
graph TD; A{IF INVCD = "A"} -- N --> B[MOVE "BAD CODE" TO MSG]; A -- Y --> C{IF AMT > 5000}; C -- N --> D[MOVE "PROBLEM" TO MSG]; C -- Y --> E[MOVE "OKAY" TO MSG]; B --> F(( )); D --> F; E --> F; F --> G(( ));
```

```
if invcd = "A"
  if amt > 5000
    move "OKAY" to msg
  else
    move "PROBLEM" to msg
  end if
else
  move "BAD CODE" to msg
end if
```

not samp
no compound

In this example, the no on both conditions calls for processing, but different messages are moved. Therefore you must code the if statements separately and you may not use a compound if.

SLIDE 8 OF 20 COMMENTS 107%

iffloppseudo-1.ppt [Protected View] - PowerPoint

FILE HOME INSERT DESIGN TRANSITIONS ANIMATIONS SLIDE SHOW REVIEW VIEW ADD-INS STORYBOARDING

PROTECTED VIEW Be careful—files from the Internet can contain viruses. Unless you need to edit, it's safer to stay in Protected View. Enable Editing

OR relationship

```
graph TD; A{IF INVCD = "A"} -- N --> B{IF AMT > 5000}; A -- Y --> C[MOVE "OKAY" TO MSG]; B -- N --> D(( )); B -- Y --> E[MOVE "OKAY" TO MSG]; D --> F(( )); E --> F; F --> G[ ];
```

Now I am starting to look at OR relationships. In this example I want to do the same thing if either of the conditions is true so I can use a compound.

```
if invcd = "A" or amt > 5000
  move "OKAY to msg"
end if
```

```
if invcd = "A"
  move "OKAY" to msg
else
  if amt > 5000
    move "OKAY" to msg
  end if
end if
```

Because the yes to both conditions call for the same processing, I can use a compound.

SLIDE 10 OF 20

COMMENTS

12:51 PM 9/30/2014

iffloppseudo-1.ppt [Protected View] - PowerPoint

FILE HOME INSERT DESIGN TRANSITIONS ANIMATIONS SLIDE SHOW REVIEW VIEW ADD-INS STORYBOARDING

PROTECTED VIEW Be careful—files from the Internet can contain viruses. Unless you need to edit, it's safer to stay in Protected View. Enable Editing

6

7

8

9

10

11

12

OR relationship

```
graph TD; A{IF INVCD = "A"} -- N --> B{IF AMT > 5000}; A -- Y --> C[MOVE "OKAY" TO MSG]; B -- N --> D(( )); B -- Y --> E[MOVE "OKAY" TO MSG]; C --> F(( )); E --> F; D --> F; F --> G(( ))
```

```
if invcd = "A" or amt > 5000  
  move "OKAY" to msg  
end if
```

```
if invcd = "A"  
  move "OKAY" to msg  
else  
  if amt > 5000  
    move "OKAY" to msg  
  end if  
end if
```

Because the yes to both conditions call for the same processing, I can use a compound.

SLIDE 10 OF 20

COMMENTS

107%

iffloppseudo-1.ppt [Protected View] - PowerPoint

FILE HOME INSERT DESIGN TRANSITIONS ANIMATIONS SLIDE SHOW REVIEW VIEW ADD-INS STORYBOARDING

PROTECTED VIEW Be careful—files from the Internet can contain viruses. Unless you need to edit, it's safer to stay in Protected View. Enable Editing

6

7

8

9

10

11

12

OR relationship

```
graph TD; A{IF INVC D = "A"} -- N --> B{IF AMT > 5000}; A -- Y --> C[MOVE "OKAY" TO MSG]; B -- N --> D(( )); B -- Y --> E[MOVE "OKAY" TO MSG]; C --> F(( )); E --> F; D --> F; F --> G(( ))
```

```
if invcd = "A" or amt > 5000
  move "OKAY" to msg
end if
```

```
if invcd = "A"
  move "OKAY" to msg
else
  if amt > 5000
    move "OKAY" to msg
  end if
end if
```

Because the yes to both conditions call for the same processing, I can use a compound.

SLIDE 10 OF 20

COMMENTS

107%

ifflowpseudo-1.ppt [Protected View] - PowerPoint

FILE HOME INSERT DESIGN TRANSITIONS ANIMATIONS SLIDE SHOW REVIEW VIEW ADD-INS STORYBOARDING

PROTECTED VIEW Be careful—files from the Internet can contain viruses. Unless you need to edit, it's safer to stay in Protected View. Enable Editing

OR relationship

```
graph TD; D1{IF INVCD = "A"} -- Y --> P1[MOVE "OKAY" TO MSG]; D1 -- N --> D2{IF AMT > 5000}; D2 -- Y --> P2[MOVE "OKAY" TO MSG]; D2 -- N --> P3[MOVE "PROBLEM" TO MSG]; P1 --> J1(( )); P2 --> J1; P3 --> J2(( )); J1 --> J2;
```

```
if invcd = "A"
  move "OKAY" to msg
else
  if amt > 5000
    move "OKAY" to msg
  else
    move "PROBLEM" to msg
  end if
end if
```

This shows a message for failing to meet either of the two conditions in the and relationship. As you will see on the next slide, this could have been handled with a compound.

SLIDE 11 OF 20

COMMENTS

12:54 PM 9/30/2014

PowerPoint interface showing a slide titled "OR relationship". The slide contains a flowchart, two code snippets, and a handwritten note.

Flowchart:

```
graph TD
    D1{IF INVCD = "A"}
    D2{IF AMT > 5000}
    P1[MOVE "PROBLEM" TO MSG]
    P2[MOVE "OKAY" TO MSG]
    P3[MOVE "OKAY" TO MSG]
    J(( ))
    K(( ))

    D1 -- N --> D2
    D1 -- Y --> P3
    D2 -- N --> P1
    D2 -- Y --> P2
    P1 --> J
    P2 --> K
    P3 --> K
    J --> J
    K --> J
```

Code Snippets:

```
if invcd = "A" or amt > 5000
  move "OKAY to msg"
else
  move "PROBLEM" to msg
end if
```

```
if invcd = "A"
  move "OKAY" to msg
else
  if amt > 5000
    move "OKAY" to msg
  else
    move "PROBLEM" to msg
  end if
end if
```

Handwritten Note: "Since they are the same, I can do a compound." (with a blue circle around the two "MOVE 'OKAY' TO MSG" boxes in the flowchart)

Handwritten Note: "same" (with a blue circle around the two "MOVE 'OKAY' TO MSG" boxes in the flowchart)

Handwritten Note: "Since they are the same, I can do a compound." (with a blue circle around the two "MOVE 'OKAY' TO MSG" boxes in the flowchart)

ifflowpseudo-1.ppt [Protected View] - PowerPoint

FILE HOME INSERT DESIGN TRANSITIONS ANIMATIONS SLIDE SHOW REVIEW VIEW ADD-INS STORYBOARDING

PROTECTED VIEW Be careful—files from the Internet can contain viruses. Unless you need to edit, it's safer to stay in Protected View. Enable Editing

OR relationship

```
graph TD; D1{IF INVCD = "A"} -- Y --> A1[MOVE "CODE - OKAY" TO MSG]; D1 -- N --> J1(( )); D2{IF AMT > 5000} -- Y --> A2[MOVE "AMT - OKAY" TO MSG]; D2 -- N --> A3[MOVE "PROBLEM" TO MSG]; A1 --> J2(( )); A2 --> J2; A3 --> J2; J1 --> J2; J2 --> End(( ));
```

Here the actions are different depending on which condition gets the yes so I cannot do a compound.

```
if invcd = "A"
  move "CODE - OKAY" to msg
else
  if amt > 5000
    move "AMT - OKAY" to msg
  else
    move "PROBLEM" to msg
  end if
end if
```

In this example, I am moving a different message for the yes on the invcd question and a different message for the yes on the amt question. Since the yes processing is different we cannot use compound. Compound requires that if I get a yes to either question, the processing is the same.

SLIDE 13 OF 20

ifflowpseudo-1.ppt [Protected View] - PowerPoint

FILE HOME INSERT DESIGN TRANSITIONS ANIMATIONS SLIDE SHOW REVIEW VIEW ADD-INS STORYBOARDING

PROTECTED VIEW Be careful—files from the Internet can contain viruses. Unless you need to edit, it's safer to stay in Protected View. Enable Editing

cond1 AND (cond 2 OR cond3)

```
if invcd = "A"
  if amtfst > 500
    move "OKAY" to msg
  else
    if amtsnd > 200
      move "OKAY" to msg
    end if
  end if
end if
```

This is the situation where one thing (invcd = A) has to be true and either of two other things. Since the actions are the same I can do a compound. See next slide.

```
graph TD
    Start(( )) --> D1{IF INVCD = "A"}
    D1 -- N --> M1(( ))
    D1 -- Y --> D2{IF AMTFST > 500}
    D2 -- N --> D3{IF AMTSND > 200}
    D2 -- Y --> A1[MOVE "OKAY" TO MSG]
    D3 -- Y --> A2[MOVE "OKAY" TO MSG]
    D3 -- N --> M1
    A1 --> M2(( ))
    A2 --> M2
    M1 --> M2
    M2 --> D1
```

SLIDE 14 OF 20

COMMENTS

1:00 PM 9/30/2014

ifflowpseudo-1.ppt [Protected View] - PowerPoint

FILE HOME INSERT DESIGN TRANSITIONS ANIMATIONS SLIDE SHOW REVIEW VIEW ADD-INS STORYBOARDING

PROTECTED VIEW Be careful—files from the Internet can contain viruses. Unless you need to edit, it's safer to stay in Protected View. Enable Editing

cond1 AND (cond 2 OR cond3)

```
if invcd = "A" and (amtfst > 500 or amtsnd > 200)
  move "OKAY" to msg
end if
```

either

Note the use of parenthesis here. In boolean logic AND gets resolved before OR. That means if we did not have the parenthesis, it would treat this like invcd = A and amtfst > 500 or just amtsnd > 200.

Continue below

We want the logic to be invcd = A and either amtfst > 500 or amtsnd > 200. To do this, we need to change the order of operation so that the two things in the or relationship are grouped together. We do this by using parenthesis.

```
graph TD
    D1{IF INVCD = "A"}
    D2{IF AMTFST > 500}
    D3{IF AMTSND > 200}
    P1[MOVE "OKAY" TO MSG]
    P2[MOVE "OKAY" TO MSG]
    P3[MOVE "OKAY" TO MSG]
    C1(( ))
    C2(( ))
    C3(( ))
    C4(( ))

    D1 -- N --> C1
    D1 -- Y --> D2
    D2 -- N --> C1
    D2 -- Y --> P1
    C1 --> D3
    D3 -- N --> C2
    D3 -- Y --> P2
    C2 --> C3
    P1 --> C4
    P2 --> C4
    P3 --> C4
    C3 --> C4
    C4 --> C5[Continue below]
```

The condition1 and either condition2 or condition3 can be written in compound because I want to do the same processing no matter as long as condition1 is true and either condition2 or condition3 are true.

SLIDE 15 OF 20 COMMENTS 107%

iflowpseudo-1.ppt [Protected View] - PowerPoint

FILE HOME INSERT DESIGN TRANSITIONS ANIMATIONS SLIDE SHOW REVIEW VIEW ADD-INS STORYBOARDING

PROTECTED VIEW Be careful—files from the Internet can contain viruses. Unless you need to edit, it's safer to stay in Protected View. Enable Editing

cond1 AND (cond 2 OR cond3)

```
if invcd = "A" and (amtfst > 500 or amtsnd > 200)
  move "OKAY" to msg
end if
```

either

Note the use of parenthesis here. In boolean logic AND gets resolved before OR. That means if we did not have the parenthesis, it would treat this like invcd = A and amtfst > 500 or just amtsnd > 200.

Continue below

We want the logic to be invcd = A and either amtfst > 500 or amtsnd > 200. To do this, we need to change the order of operation so that the two things in the or relationship are grouped together. We do this by using parenthesis.

```
graph TD
    Start(( )) --> D1{IF INVCD = "A"}
    D1 -- N --> M1(( ))
    D1 -- Y --> D2{IF AMTFST > 500}
    D2 -- N --> M2(( ))
    D2 -- Y --> P1[MOVE "OKAY" TO MSG]
    P1 --> M3(( ))
    D3{IF AMTSND > 200}
    D3 -- N --> M4(( ))
    D3 -- Y --> P2[MOVE "OKAY" TO MSG]
    P2 --> M5(( ))
    M1 --> M6(( ))
    M2 --> M6
    M3 --> M6
    M4 --> M6
    M5 --> M6
    M6 --> End[Continue below]
```

The condition1 and either condition2 or condition3 can be written in compound because I want to do the same processing no matter as long as condition1 is true and either condition2 or condition3 are true.

SLIDE 15 OF 20

COMMENTS

107%

iffloppseudo-1.ppt [Protected View] - PowerPoint

FILE HOME INSERT DESIGN TRANSITIONS ANIMATIONS SLIDE SHOW REVIEW VIEW ADD-INS STORYBOARDING

PROTECTED VIEW Be careful—files from the Internet can contain viruses. Unless you need to edit, it's safer to stay in Protected View. Enable Editing

cond1 AND (cond 2 OR cond3)

```
if invcd = "A"
  if amtfst > 500 or amtsnd > 200
    move "OKAY" to msg
  end if
end if
```

Another way to code it using two if statements.

```
graph TD
    Start(( )) --> D1{IF INVCD = "A"}
    D1 -- N --> C1(( ))
    D1 -- Y --> D2{IF AMTFST > 500}
    D2 -- Y --> A1[MOVE "OKAY" TO MSG]
    D2 -- N --> C1
    C1 --> D3{IF AMTSND > 200}
    D3 -- Y --> A2[MOVE "OKAY" TO MSG]
    D3 -- N --> C1
    A1 --> C2(( ))
    A2 --> C2
    C1 --> C2
    C2 --> End(( ))
```

SLIDE 16 OF 20

COMMENTS

1:03 PM 9/30/2014

iflowpseudo-1.ppt [Protected View] - PowerPoint

FILE HOME INSERT DESIGN TRANSITIONS ANIMATIONS SLIDE SHOW REVIEW VIEW ADD-INS STORYBOARDING

PROTECTED VIEW Be careful—files from the Internet can contain viruses. Unless you need to edit, it's safer to stay in Protected View. Enable Editing

cond1 AND cond 2 OR cond3

```
if invcd = "A" and amtfst >500 or amtsnd > 200
  move "OKAY" to msg
end if
```

If I leave out the parenthesis the logic will look like this - NOT THE RESULTS I WAS LOOKING FOR

```
graph TD
    Start(( )) --> D1{IF INVCD = "A"}
    D1 -- Y --> D2{IF AMTFST > 500}
    D1 -- N --> D3{IF AMTSND > 200}
    D2 -- Y --> M1[MOVE "OKAY" TO MSG]
    D2 -- N --> D4{IF AMTSND > 200}
    D3 -- Y --> M2[MOVE "OKAY" TO MSG]
    D3 -- N --> M3(( ))
    D4 -- Y --> M4[MOVE "OKAY" TO MSG]
    D4 -- N --> M3
    M1 --> M3
    M2 --> M3
    M3 --> M5(( ))
    M4 --> M5
    M5 --> End(( ))
```

Now we are going to look at how this decision would look if we did not use parenthesis when writing the compound. If condition1 and condition 2 are not true then I need to ask condition3. In other words if either invcd is not equal to A or amtfst is not greater than 500, I need to ask about amtsnd >200. Therefore it

SLIDE 17 OF 20

COMMENTS

1:05 PM 9/30/2014

ifflowpseudo-1.ppt [Protected View] - PowerPoint

FILE HOME INSERT DESIGN TRANSITIONS ANIMATIONS SLIDE SHOW REVIEW VIEW ADD-INS STORYBOARDING

PROTECTED VIEW Be careful—files from the Internet can contain viruses. Unless you need to edit, it's safer to stay in Protected View. Enable Editing

cond3 OR cond1 AND cond2

```
graph TD; A[IF AMTSND > 200] -- Y --> B[MOVE "OKAY" TO MSG]; A -- N --> C(( )); B --> D(( )); C --> E[IF INVCD = "A"]; E -- Y --> F[MOVE "OKAY" TO MSG]; E -- N --> G(( )); F --> D; G --> H(( )); H --> I[IF AMTFST > 500]; I -- Y --> J[MOVE "OKAY" TO MSG]; I -- N --> D; J --> D; D --> K(( )); K --> L[ ];
```

A simplified version of the previous slide but still NOT THE RESULTS I WAS LOOKING FOR

```
if amtsnd > 200 or invcd = "A" and amtfst > 500  
  move "OKAY" to msg  
end if
```

This shows asking condition3 which stood alone first and then asking for condition1 and condition2.

SLIDE 18 OF 20

COMMENTS

1:07 PM 9/30/2014

iflowpseudo-1.ppt [Protected View] - PowerPoint

FILE HOME INSERT DESIGN TRANSITIONS ANIMATIONS SLIDE SHOW REVIEW VIEW ADD-INS STORYBOARDING

PROTECTED VIEW Be careful—files from the Internet can contain viruses. Unless you need to edit, it's safer to stay in Protected View. Enable Editing

cond1 AND (cond 2 OR cond3)

```
if invcd = "A"
  if amtfst > 500
    move ">500 - OKAY" to msg
  else
    if amtsnd > 200
      move ">200 -OKAY" to msg
    else
      move "PROBLEM" to msg
    end if
  end if
end if
```

```
graph TD
    Start(( )) --> D1{IF INVCD = "A"}
    D1 -- N --> D2{IF AMTSND > 200}
    D1 -- Y --> D3{IF AMTFST > 500}
    D2 -- N --> M1[MOVE "PROBLEM" TO MSG]
    D2 -- Y --> M2[MOVE ">200 -OKAY" TO MSG]
    D3 -- Y --> M3[MOVE ">500 - OKAY" TO MSG]
    M1 --> J1(( ))
    M2 --> J1
    M3 --> J2(( ))
    D3 -- N --> J1
    J1 --> J2
    J2 --> End(( ))
```

I have now added processing if the answer to amtsnd > 200 is no and changed the processing on the okay moves. Note that this means I can no longer process this as a compound

SLIDE 19 OF 20

COMMENTS

1:09 PM 9/30/2014

Microsoft PowerPoint interface showing a slide with a flowchart and code. The slide title is "cond1 AND (cond 2 OR cond3)".

Flowchart Logic:

- Decision: IF INVCD = "A"
 - Y: Decision: IF AMTFST > 500
 - Y: MOVE ">500 - OKAY" TO MSG
 - N: Decision: IF AMTSND > 200
 - Y: MOVE ">200 - OKAY" TO MSG
 - N: MOVE "PROBLEM" TO MSG
 - N: MOVE "CODE PROBLEM" TO MSG

Code Snippet:

```
if invcd = "A"
  if amtfst > 500
    move ">500 - OKAY" to msg
  else
    if amtsnd > 200
      move ">200 -OKAY" to msg
    else
      move "PROBLEM" to msg
    end if
  end if
else
  move "CODE PROBLEM" to msg
end if
```

Note the layout is not great for the code problem message, but I had a slide layout problem.

Windows taskbar at the bottom shows: SLIDE 20 OF 20, COMMENTS, 1:10 PM, 9/30/2014.

structureintro-1.ppt [Protected View] - PowerPoint

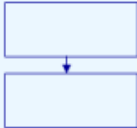
FILE HOME INSERT DESIGN TRANSITIONS ANIMATIONS SLIDE SHOW REVIEW VIEW ADD-INS STORYBOARDING

PROTECTED VIEW Be careful—files from the Internet can contain viruses. Unless you need to edit, it's safer to stay in Protected View. Enable Editing

1 Logic Structures - Focus on Sequencing
2 **Logic Structures - Focus on Sequencing**
3
4
5
6
7

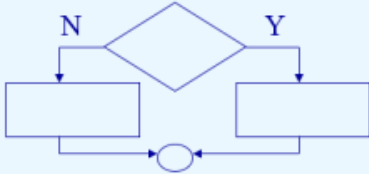
Programming logic involves three structures:

sequence structure

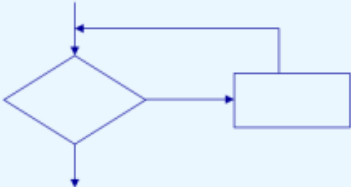


To be considered a programming language, the language must support all of these structures.

selection structure (conditions)



loop structure (iteration)



The three structures are all that is logically needed to write a procedural or structured program. All programming commands can be incorporated into these three basic structures.

SLIDE 2 OF 7 COMMENTS 1:11 PM 9/30/2014 107%

structureintro-1.ppt [Compatibility Mode] - PowerPoint

FILE HOME INSERT DESIGN TRANSITIONS ANIMATIONS SLIDE SHOW REVIEW VIEW ADD-INS STORYBOARDING

Clipboard Slides

Font Paragraph Drawing Editing

DO WHILE LOOP

```
graph TD; Start(( )) --> Condition{condition}; Condition -- Y --> Processing[processing]; Processing --> Condition; Condition -- N --> End(( ));
```

In this logic if the condition gets a no it is possible to drop out without ever processing.

possible
never
Process

DO WHILE LOOP:
The while loop shown here tests a condition to see if the processing should be done.
If the answer to the condition is YES, the processing box shown is executed.
If the answer to the condition is NO, the processing box shown is not executed.

This shows the DO while loop which is frequently used in programming to cause processing to be repeated until a specific condition is met. When the condition is met the processing will not be executed and control will drop out of the loop.

SLIDE 3 OF 7

NOTES COMMENTS

103%

structureintro-1.ppt [Compatibility Mode] - PowerPoint

FILE HOME INSERT DESIGN TRANSITIONS ANIMATIONS SLIDE SHOW REVIEW VIEW ADD-INS STORYBOARDING

Clipboard Slides Font Paragraph Drawing Editing

Since I process before I test, I will always do the processing once.

DO UNTIL LOOP

```
graph TD; A[processing] --> B{condition}; B -- Y --> A; B -- N --> C[ ];
```

DO UNTIL LOOP:
The while loop shown here executes the processing once and then tests a condition to see if the processing should be done. This means that the loop will always be executed once since it is executed before checking is done. Once the processing has been done once, further processing is determined by the answer to the condition.
If the condition gets a YES, then the processing is executed again.
If the condition gets a NO, then the processing is not executed again.

In this structure, the processing is done once and then the condition is checked to see if it should be done again. In this structure, the processing will always be done at least once since the check is after the completion of the processing.

SLIDE 4 OF 7

NOTES COMMENTS

1:16 PM 9/30/2014

DO WHILE LOOP

```
graph TD; Start(( )) --> Condition{condition}; Condition -- Y --> Processing[processing]; Processing --> Condition; Condition -- N --> Exit(( ));
```

DO WHILE LOOP:
The while loop shown here tests a condition to see if the processing should be done.
If the answer to the condition is YES, the processing box shown is executed.
If the answer to the condition is NO, the processing box shown is not executed.

This shows the DO while loop which is frequently used in programming to cause processing to be repeated until a specific condition is met. When the condition is met the processing will not be executed and control will drop out of the loop.

SLIDE 3 OF 7

NOTES COMMENTS

1:16 PM 9/30/2014

structureintro-1.ppt [Compatibility Mode] - PowerPoint

FILE HOME INSERT DESIGN TRANSITIONS ANIMATIONS SLIDE SHOW REVIEW VIEW ADD-INS STORYBOARDING

Clipboard Slides Font Paragraph Drawing Editing

DO WHILE LOOP

This example shows a do while loop where I am reading records from a file (getting input from a file). I want the processing to continue as long as there are records on the file. To do this, I am going to use an initializing read. I read the initial record and then I process a loop until the end of file (EOF) has been reached. To make this work, I always read or input another record at the end of the loop.

```

graph TD
    Start([Start]) --> Read1[/Initializing Read/]
    Read1 --> NotEOF{Not EOF}
    NotEOF -- Y --> Process[Process]
    Process --> Read2[/Read/]
    Read2 --> NotEOF
    NotEOF -- N --> Stop([Stop])
  
```

The initializing or priming read takes in the first records. All other records are read with the read inside the loop. Notice that it is done just before I go back to check. So if the read is not successful, I will not do anything before I check and then leave the loop and stop.

Pseudocode:

```

start
input/read record
do while not EOF
  process
  input/read record
end while loop
stop
  
```

Note that I use the term initializing read, priming read is another term that can be used. When you are inputting a record you can use the word read, the word input, the word get or any word that implies retrieving a record from the file. In these

SLIDE 5 OF 7

NOTES COMMENTS

103%

1:19 PM 9/30/2014

structureintro.ppt [Protected View] - PowerPoint

FILE HOME INSERT DESIGN TRANSITIONS ANIMATIONS SLIDE SHOW REVIEW VIEW ADD-INS STORYBOARDING

PROTECTED VIEW Be careful—files from the Internet can contain viruses. Unless you need to edit, it's safer to stay in Protected View. Enable Editing

1 Logic Structures - Focus on Looping
2 Logic Structures - Focus on Looping
3 Logic Structures - Focus on Looping
4 Logic Structures - Focus on Looping
5 Logic Structures - Focus on Looping
6 Logic Structures - Focus on Looping
7 Logic Structures - Focus on Looping

Here doing the loop is controlled by a counter.

This example shows a do while loop controlled by a counter. I have determined that I want to continue the looping process while the counter is less than the stop point. When this condition is no longer true, I will exit the loop

DO WHILE LOOP

Pseudocode:
counter = startPoint
do while counter < stopPoint
 process
 increment counter
end while loop

Handwritten notes:
A box containing '1 + 2 3 4' with 'Counter' written below it.
A list of 'process' written three times, with '1', '2', and '3' written to the right of each.
Two boxes containing '1' and '4', with 'Start point' and 'Stop point' written below them.

The looping structure where I want to use the **while loop** to do something a certain number of times, requires the following:

- **initialize the counter outside the loop to a specific start point**
- **test the counter to determine whether or not to enter the loop**
- **increment the counter inside the loop**

Be sure to read this list.

In this slide, I am showing the specifics of the do while structure. I am using a counter to determine how many times I want to do a loop. I set the counter to the start point outside the loop. I then test the counter against the stop point to determine if I should process. If I process, I increment the counter before looping

SLIDE 6 OF 7 COMMENTS 107%

structureintro.ppt [Protected View] - PowerPoint

FILE HOME INSERT DESIGN TRANSITIONS ANIMATIONS SLIDE SHOW REVIEW VIEW ADD-INS STORYBOARDING

PROTECTED VIEW Be careful—files from the Internet can contain viruses. Unless you need to edit, it's safer to stay in Protected View. Enable Editing

DO WHILE LOOP

This example shows a do while loop controlled by a counter. I have determined that I want to continue the looping process while the counter is less than the stop point. When this condition is no longer true, I will exit the loop

Handwritten: +2 3 4 Counter

Pseudocode:
 counter = startPoint
 do while counter < stopPoint
 process
 increment counter
 end while loop

Handwritten: process 1, process 2, process 3

Handwritten: Start point 1, Stop point 4

The looping structure where I want to use the **while loop** to do something a certain number of times, requires the following:

- **initialize the counter outside the loop to a specific start point**
- **test the counter to determine whether or not to enter the loop**
- **increment the counter inside the loop**

In this slide, I am showing the specifics of the do while structure. I am using a counter to determine how many times I want to do a loop. I set the counter to the start point outside the loop. I then test the counter against the stop point to determine if I should process. If I process, I increment the counter before looping

SLIDE 6 OF 7 COMMENTS 107%

structureintro-1.ppt [Compatibility Mode] - PowerPoint

FILE HOME INSERT DESIGN TRANSITIONS ANIMATIONS SLIDE SHOW REVIEW VIEW ADD-INS STORYBOARDING

Clipboard Font Paragraph Drawing Editing

DO WHILE LOOP

This example shows a do while loop where I am reading records from a file (getting input from a file). I want the processing to continue as long as there are records on the file. To do this, I am going to use an initializing read. I read the initial record and then I process a loop until the end of file (EOF) has been reached. To make this work, I always read or input another record at the end of the loop.

```
graph TD; Start([Start]) --> Read1[/Initializing Read/]; Read1 --> NotEOF{Not EOF}; NotEOF -- Y --> Process[Process]; Process --> Read2[/Read/]; Read2 --> NotEOF; NotEOF -- N --> Stop([Stop]);
```

Here EOF controls ending the program - when all the records have been processed then the program ends.
With the counter, the counter controls the end and when it reaches the number you want it stops.

Pseudocode:

```
start  
input/read record  
do while not EOF  
  process  
  input/read record  
end while loop  
stop
```

Note that I use the term initializing read, priming read is another term that can be used.
When you are inputting a record you can use the word read, the word input, the word get or any word that implies retrieving a record from the file. In these

SLIDE 5 OF 7

NOTES COMMENTS

1:27 PM 9/30/2014

Initialize

- Initial read of a file/table
- Set counter
- Or there are other ways

Condition to end loop

Change

- read another
- Change counter
- Or...

structureintro-1.ppt [Compatibility Mode] - PowerPoint

FILE HOME INSERT DESIGN TRANSITIONS ANIMATIONS SLIDE SHOW REVIEW VIEW ADD-INS STORYBOARDING

Clipboard Paste New Slide Section Slides

Font Paragraph Drawing Editing

Logical Structures - Basic on looping

2

3

4

5

6

7

DO UNTIL LOOP

This example shows a do until loop controlled by a counter. I have determined that I always want to process once and that I want to continue the looping process until the counter equals the stop point. When this condition is true, I will exit the loop

Pseudocode:
 counter = startPoint
 do
 process
 increment counter
 until counter = stopPoint

```

    graph TD
      A[counter = start point] --> B[process]
      B --> C[increment counter]
      C --> D{counter not = stop point}
      D -- Y --> B
      D -- N --> E[ ]
  
```

The looping structure where I want to use the **until loop** to do something a certain number of times, requires the following:

- **initialize the counter outside the loop to a specific start point**
- **process**
- **increment the counter inside the loop**
- **test the counter to determine whether or not to loop again**

In the do until loop, I will always process at least once. I think it is clearer when writing the pseudocode, to show the until condition at the bottom of the loop rather than at the top.

SLIDE 7 OF 7

NOTES COMMENTS

100%

1:30 PM
9/30/2014

FILE HOME INSERT DESIGN TRANSITIONS ANIMATIONS SLIDE SHOW REVIEW VIEW ADD-INS STORYBOARDING Grocer, Priscilla

PROTECTED VIEW Be careful—files from the Internet can contain viruses. Unless you need to edit, it's safer to stay in Protected View. Enable Editing

DO UNTIL LOOP

This example shows a do until loop controlled by a counter. I have determined that I always want to process once and that I want to continue the looping process until the counter equals the stop point. When this condition is true, I will exit the loop

```
Pseudocode:  
counter = startPoint  
do  
  process  
  increment counter  
until counter = stopPoint
```

Handwritten notes: A box containing '+2 3 4' with 'Counted' written below it. Below the flowchart, 'Process 1', 'Process 2', and 'Process 3' are written. To the right, two boxes contain '1' and '4', labeled 'Start point' and 'Stop point' respectively.

- initialize the counter outside the loop to a specific start point
- process
- increment the counter inside the loop
- test the counter to determine whether or not to loop again

In the do until loop, I will always process at least once. I think it is clearer when writing the pseudocode, to show the until condition at the bottom of the loop rather than at the top.

SLIDE 7 OF 7 COMMENTS 100%