Program Control Structures (Matlab)

Miguel A. Figueroa-Villanueva

Department of Electrical and Computer Engineering
University of Puerto Rico - Mayagüez Campus

September 2, 2009
Outline

1. **Program Control Structures**
   - Sequence Structure
   - Selection Structure
   - Repetition Structure

2. **Top-Down Stepwise Refinement**
   - Introduction
   - Example 1
   - Example 2

3. **Summary**
Bohm and Jacopini (1966) work led to demonstrate that all algorithms can be expressed in a structured program with only three control structures:

- the sequence structure,
- the selection structure,
- and the repetition structure.

These structures will be represented using flowcharts with two connectors; one entry and one exit connector.

Programs will be built by creating multiple combinations of these single-entry/single-exit control statements.

These modules can be combined in two forms:

- control-statement stacking,
- and control-statement nesting.

That is, all programs will be built using only three types of control statements combined in only two ways.
Sequence Structure

- Built into structured languages, such as C.
  - That is, no special keyword required.
- Program instructions are executed sequentially, in the order they appear. This is important for program clarity.
- The *infamous* goto statement led to many problems in the past due to its arbitrary *transfer of control*.
- What value will the variable total have at the end of the execution of the program in the Figure?

```
START

counter = 0;
total = 10;
goto B;

A:
total = total + counter;

B:
counter = counter + 1;
goto A;

END
```

**goto example**
Selection Structure

- The selection structure is used to choose from alternative courses of action.
- Also called conditional or branching structure.
- Selection structures can be classified into three categories:
  - single-selection statement
  - double-selection statement
  - multiple-selection statement
- Matlab has three types: `if`, `if...else`, and `switch`.
if - single-selection statement

**definition**
Structure that will execute the enclosed *statement or set of statements* if and only if *expression* is true.

**pseudocode**
```
if expression
    statement;
end
```

**flowchart**
```
expression
    true
        statement;
    false
```

M. A. Figueroa-Villanueva (UPRM - ECE)
if - single-selection statement

example

problem

Print *PASSED* if grade is greater or equal to 65.
if - single-selection statement

solution

```
if grade >= 65
    fprintf('PASSED');
end
```

Pseudocode:
```
if grade >= 65
    fprintf('PASSED');
end
```

Flowchart:
```
grade >= 65
true
Print PASSED;
false
```

M. A. Figueroa-Villanueva (UPRM - ECE)
if..else - double-selection statement

definition
Structure that will execute the eclosed statement_a if expression is true, otherwise it will execute the eclosed statement_b.

pseudocode
```
if expression
    statement_a;
else
    statement_b;
end
```

flowchart
The flowchart represents the flow of execution:
- If expression is true, execute statement_a.
- If expression is false, execute statement_b.
- Return to the start of the structure.
if...else - double-selection statement

eexample

problem

Print *PASSED* if grade is greater or equal to 65, otherwise print *FAILED*. 
if..else - double-selection statement

solution

```
if grade >= 65
    fprintf('PASSED');
else
    fprintf('FAILED');
end
```

Pseudocode:
```plaintext
if grade >= 65
    Print PASSED;
else
    Print FAILED;
end
```

Flowchart:
- Start
- Decision: grade >= 65
  - True: Print PASSED;
  - False: Print FAILED;
- End
if..else - multiple-selection statement

- It is also possible to create a multiple-selection structure using **nested** if..else structures.

```
A   B   C   D
```

```
C   D
```
if..else - multiple-selection statement

example

problem
Print the grade letter of a student based on the point percentage.
if..else - multiple-selection statement

documentation

pseudocode

```matlab
if grade >= 90
    printf('A');
else
    if grade >= 80
        printf('B');
    else
        if grade >= 65
            printf('C');
        else
            printf('F');
    end
end
end
```

flowchart

```
if grade >= 90
    Print A;
else
    if grade >= 80
        Print B;
    else
        if grade >= 65
            Print C;
        else
            Print FAILED;
    end
end
end
```
if..else - multiple-selection statement

- The following structure can be used to organize the multiple-selection structure in the previous example.
- These are equivalent for the compiler, however it is more readable and compact.

```matlab
if grade >= 90
    fprintf('A');
elseif grade >= 80
    fprintf('B');
elseif (grade >= 65)
    fprintf('C');
else
    fprintf('F');
end
```
The repetition structure is used to repeat a series of instructions multiple times.
Also called iterative structure or loops.
There are two common iterative structures:
- counter-controlled repetition
- sentinel-controlled repetition
Matlab has two types: `while` and `for`.

**example**

```matlab
while (there are more items in my shopping list)
    purchase next item and cross it off my list;
end
```
### while - repetition statement

**Definition**

Structure that will execute the enclosed `statement while expression` is true. When the `expression` is false it will stop the execution of the enclosed `statement`.

**Pseudocode**

```plaintext
while expression
    statement;
end
```

**Flowchart**

[Flowchart showing the while loop structure]
while - repetition statement

example

problem

Determine the first power of 2 larger than 1000.
**while - repetition statement**

**solution**

**pseudocode**

```plaintext
product = 2;
while product <= 1000
    product = product * 2;
end
```

**flowchart**

- `product = 2;`
- `product <= 1000`
- `true: product = product * 2;`
- `false: product = product * 2;`
while - counter-controlled repetition

- Loop repeated until a counter variable reaches a certain value.
- If we need $n$ iterations, the counter will count $n$ times.
while - counter-controlled repetition

example

problem

A class of ten students took a quiz. The grades (integers in the range 0 to 100) for this quiz are available to you. Determine the class average on the quiz.
**while - counter-controlled repetition**

**solution**

### Pseudocode

```plaintext
total = 0;
counter = 0;

while counter < 10
    Read grade;
    total = total + grade;
    counter = counter + 1;
end

average = total / counter;
Print average;
```

### Flowchart

```

```

Read grade;
total = total + grade;
counter = counter + 1;
```

```

```

counter < 10

```

true

```

```

false

average = total / counter;
Print average;
```
**while** - counter-controlled repetition

**manual variable inspection**

Create a table to inspect the algorithm functionality for a given instance. This will help verify whether your algorithm is correct.

**pseudocode**

```plaintext
total = 0;
counter = 0;

while counter < 10
    Read grade;
    total = total + grade;
    counter = counter + 1;
end

average = total / counter;
Print average;
```

**table**

For the instance where we have the following grades: 98, 76, 71, 87, 83, 90, 57, 79, 82, 94.

<table>
<thead>
<tr>
<th>iteration</th>
<th>counter</th>
<th>grade</th>
<th>total</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>-</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>98</td>
<td>98</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>76</td>
<td>174</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>71</td>
<td>245</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>87</td>
<td>332</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>83</td>
<td>415</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>90</td>
<td>505</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>57</td>
<td>562</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>79</td>
<td>641</td>
</tr>
<tr>
<td>9</td>
<td>9</td>
<td>82</td>
<td>723</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>94</td>
<td><strong>817</strong></td>
</tr>
</tbody>
</table>

average = 817 / 10 = 81.7
while - sentinel-controlled repetition

- Loop repeated until a variable is given a **sentinel value**.
- The sentinel value is a special value that indicates the *end of data entry*.
- Also known as, **signal value** or **flag value**.
- The sentinel-controlled repetition is used when we need to iterate for an indefinite amount of times (i.e., **the number of iterations are not known before the loop begins execution**).
while - sentinel-controlled repetition

example

problem

Develop a class averaging program that will process an arbitrary number of grades each time the program is run.
**while - sentinel-controlled repetition**

**solution**

**pseudocode**

```plaintext
total = 0;
counter = 0;

Read grade;
while grade not equal −1
    total = total + grade;
    counter = counter + 1;
    Read grade;
end

if counter not equal 0
    average = total / counter;
    Print average;
else
    Print "No grades entered."
end
```

**flowchart**

```
  total = 0;
counter = 0;
Read grade;

grade != -1

true

total = total + grade;
counter = counter + 1;
Read grade;

false

counter != 0

true

average = total / counter;
Print average;

false

Print "No grades entered."
```

M. A. Figueroa-Villanueva (UPRM - ECE)
while - sentinel-controlled repetition
manual variable inspection

**pseudocode**

```
total = 0;
counter = 0;

Read grade;
while grade not equal -1
    total = total + grade;
counter = counter + 1;
    Read grade;
end

if counter not equal 0
    average = total / counter;
    Print average;
else
    Print "No grades entered."
end
```

**table**

For the instance where we have the following grades: 98, 76, 71, 87.

```
<table>
<thead>
<tr>
<th>iteration</th>
<th>counter</th>
<th>grade</th>
<th>total</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>98</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>76</td>
<td>98</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>71</td>
<td>174</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>87</td>
<td>245</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>-1</td>
<td>332</td>
</tr>
</tbody>
</table>

average = 332 / 4 = 83
```
Top-Down, Stepwise Refinement

- Approach algorithm design from the **top-down** (i.e., *from the general to the specific*).
- Design in several steps, with each step being a **refinement** of the previous one.
A college has a list of test results (1 = pass, 2 = fail) for 10 students.

Write a program that analyzes the results (i.e., prints the number of passes and failures) and if the passes are greater than 8, print “Raise Tuition”.
specifications

- The program must process 10 test results, hence a *counter-controlled loop* is required.
- Two variables can be used to count **passes** and **failures**, respectively.
- Each input result is a number, either a 1 or a 2.
Top-Down, Stepwise Refinement

solution

top level outline

Analyze exam results and decide if tuition should be raised.

first refinement

- Initialize variables.
- Input the ten quiz grades and count passes and failures.
- Print a summary of the exam results and decide if tuition should be raised.
Top-Down, Stepwise Refinement

solution

refinement - initialize variables

```plaintext
passes = 0;
failures = 0;
counter = 0;
```

refinement - input the ten quiz grades and count passes and failures

```plaintext
while counter < 10
    Read grade;
    if student passed
        passes = passes + 1;
    else
        failures = failures + 1;
    end
    counter = counter + 1;
end
```
Top-Down, Stepwise Refinement

solution

refinement - print a summary of the exam results and decide if tuition should be raised

```matlab
Print passes;
Print failures;
if passes > 8
    Print 'Raise Tuition';
end
```
Top-Down, Stepwise Refinement

solution

```
1 passes = 0;
2 failures = 0;
3 counter = 0;
4
5 while counter < 10
6   Read grade;
7       if student passed
8       passes = passes + 1;
9       else
10       failures = failures + 1;
11     end
12     counter = counter + 1;
13 end
```

```
14 Print passes;
15 Print failures;
16 if passes > 8
17     Print 'Raise Tuition';
18 end
```
Top-Down, Stepwise Refinement

**Example 1**

```
counter < 10
true
false

grades = 0;
failures = 0;

Read grade;
```

```
grade == 1
true
false

counter = counter + 1;
```

```
passes = passes + 1;
failures = failures + 1;
```

```
grades > 8
true
false

Print 'Raise Tuition';
```

```
END
```
Develop a program that receives a value $n$ from the user and prints a cube of asterisks of dimension $n$.

For example, if the user enters a value of $n = 5$, the program prints:

```
*****
*   *
*   *
*   *
*   *
*****
```
Top-Down, Stepwise Refinement

solution

**top level outline**

- Get input \( n \) from user.
- Print asterisk square of dimension \( n \).

**refinement - print asterisk square of dimension \( n \)**

- Print \( n \) asterisks in a row and a newline.
- Print \( n-2 \) middle lines (i.e., * *).
- Print \( n \) asterisks in a row and a newline\(^{a} \).

\(^{a}\)Note that this is the same that the first line; reuse the functionality.
Top-Down, Stepwise Refinement

**Example 1**

Print \( n \) asterisks in a row and a newline.

- Print \( n \) asterisks in a row.
- Print newline.

**Example 2**

This requires a counter-controlled loop.

```plaintext
counter = 0;
while counter < n
    Print '*';
    counter = counter + 1;
end
```

Counter-controlled loop: 
```
counter = 0;
while counter < n
    Print '*';
    counter = counter + 1;
end
```
Top-Down, Stepwise Refinement

solution

refinement - print \( n-2 \) middle lines

- Repeat the following \( n-2 \) times:
  - Print asterisk.
  - Print \( n-2 \) spaces.
  - Print asterisk.
  - Print newline.
refinement - print n−2 spaces

This requires a counter-controlled loop similar to the previous one.

counter = 0;
while counter < n - 2
    Print ' ';
    counter = counter + 1;
end

counter < n - 2
false
  
  true
  Print " ";
  counter = counter + 1;
Top-Down, Stepwise Refinement

Solution

refinement - print \( n-2 \) middle lines

This requires a counter-controlled loop as well\(^a\).

\[
\begin{align*}
\text{counter2} & = 0; \\
\text{while} & \ \text{counter2} < n - 2 \\
& \quad \text{// insert previous refinement} \\
& \quad \text{counter2} = \text{counter2} + 1; \\
\text{end}
\end{align*}
\]

\(^a\)Why counter2 and not counter?
Top-Down, Stepwise Refinement
solution - putting it all together

```
final solution

1 Read n;
2 counter = 0;
3 while counter < n
4     Print '∗';
5     counter = counter + 1;
6 end
7 Print '＼n';

8 counter2 = 0;
9 while counter2 < n - 2
10     Print '∗';
11     counter = 0;
12 end
13
14 while counter < n - 2
15     Print ' ′;
16     counter = counter + 1;
17 end
18 Print '＊\n';
19
20     counter2 = counter2 + 1;
21 end
22
23 counter = 0;
24 while counter < n
25     Print '∗';
26     counter = counter + 1;
27 end
28 Print '＊\n';
```
Top-Down, Stepwise Refinement
solution - putting it all together
All structured programs can be expressed with only three control structures:
- the sequence structure,
- the selection structure,
- and the repetition structure.

These **single-entry/single-exit** control structure modules can be combined in two forms: **stacking** and **nesting**.
Summary

- Approach algorithm design from the **top-down** (i.e., *from the general to the specific*).
- Design in several steps, with each step being a **refinement** of the previous one.
- After a solution for a component is worked out, think of it as a **black box** and **reuse it**.