

CSE1301
Computer Programming:
Lecture 19
Flowcharts and Debugging

1

Topics

- The Software Development Cycle
- Flowcharts
 - Selection
 - Sequence
 - Iteration
- How to diagnose errors in your program?
 - Methods for debugging
 - Methods for testing

2

Components of the Software Development Process

- Define the problem clearly
- Analyze the problem
- Design an algorithm
 - top-down design
- Code (Implement) the algorithm
- Test the code
- Document the system

3

Development Cycle

```
graph LR; Analysis[Analysis] --> Design[Design]; Design --> Implement[Implement]; Implement --> Test[Test]; Test --> Implement;
```

4

Debugging and Testing

- Debugging: the process of finding and correcting errors (a.k.a “bugs”)
- Testing: executing the program on a test data set

5

Types of Errors

- syntactic: how instructions are written
 - Example 1:**

```
while (i=0; i < 5; i++)  
{  
    printf("%d\n", i);  
}
```
- semantic: what they represent
 - Example 2:**

```
n=0; ...; n=6; ...  
for (i=n; i != 5; i++)  
{  
    printf("%d\n", i);  
}
```
 - Example 3:**

```
if (choice = 'Q')  
{  
    break;  
}
```

6

Flowcharts

- Represent flow of control of algorithms:
 - sequences
 - selection
 - iteration
- Useful for:
 - Finding semantic errors
 - Determining test data set

7

Sequence (revision)

- Series of instructions to be carried out in a fixed sequential order
- Example 1:
 - Step A:** *input number*
 - Step B:** *add 1 to number*
 - Step C:** *output number*

8

Flowchart: Sequence

- Represented by concatenating instructions (usually vertically)

9

Sequence (cont)

Example 2:

- Step A:** *input number*
- Step B:**
 - if number is negative, then add -1 to number*
 - else add 1 to number*
- Step C:** *output number*

10

Flowchart: Selection

```

Step A
if ( condition C1 )
{
  <sequence S1>
}
else
{
  <sequence S2>
}
Step C
    
```

11

Example: Algorithm to Flowchart

```

input number
if number is negative,
  then add -1 to number
else add 1 to number
output number
    
```

12

Flowchart: Iteration (while loop)

```

while ( condition C1 )
{
  <sequence S1>
}
    
```

13

Flowchart: Iteration (for loop)

```

for ( init ; condition C1 ; increment )
{
  <sequence S1>
}
    
```

14

How to choose which iteration?

- Do you know exactly how many times the loop will execute?
 - If yes, then use FOR
- Is it possible the sequence may never be executed?
 - If yes, then use WHILE

15

Example: Code to Flowchart (Spot the error!)

```

for ( i=0; i<10; i++ )
{
  scanf("%d\n", &x);
  if ( x < 0 )
  {
    break;
  }
}
    
```

16

Example: Code to Flowchart (correct version)

```

for ( i=0; i<10; i++ )
{
  scanf("%d\n", &x);
  if ( x < 0 )
  {
    break;
  }
}
    
```

17

Algorithm to Flowchart Example: Calculating Mean

```

input totalNumbers
set sum to 0
set count to 0
while (count < totalNumbers)
{
  input nextNum
  add nextNum to sum
  add 1 to count
}
output "Sum was" sum
output "Mean was" sum/count
    
```

18

Algorithm to Flowchart Example: Calculating Mean (cont)

```

input totalNumbers
set sum to 0
set count to 0
while (count < totalNumbers)
{
  input nextNum
  add nextNum to sum
  add 1 to count
}
output "Sum was" sum
output "Mean was" sum/count
    
```

19

Algorithm to Flowchart Example: Calculating Mean (cont)

```

input totalNumbers
set sum to 0
set count to 0
while (count < totalNumbers)
{
  input nextNum
  add nextNum to sum
  add 1 to count
}
output "Sum was" sum
output "Mean was" sum/count
    
```

20

Algorithm to Flowchart Exercise: Calculating Mean (cont)

```

input totalNumbers
set sum to 0
set count to 0
while (count < totalNumbers)
{
  input nextNum
  add nextNum to sum
  add 1 to count
}
output "Sum was" sum
output "Mean was" sum/count
    
```

22

Algorithm to Flowchart Exercise: Calculating Mean (cont)

- Modify the flowchart to add an extra check so that the mean is output only when count is positive

22

Use of Flowcharts

- Pseudo-code \leftrightarrow flowchart
- Flowchart \leftrightarrow code

23

Debugging Basics

- Know the (C) language well

Examples:

```

float x, y, z = 3.5;
printf("%d\n", &num);
scanf("%f", x);
scanf("%s", &name);
if (i < N);
{
  scanf("%d\n", &i);
}
    
```

24

Debugging Basics (cont)

- Pay attention to compiler error and warning messages

Examples:

```
if (ch = 'Q')
{
    break;
}
```

“Possible incorrect assignment”

```
int N;
...
scanf("%d", N);
```

“N” might be used uninitialized in this function

Tracing

- Trace execution of a program:
 - location in the program
 - status/contents of variables
- Tools:
 - programming environment
 - E.g., “step”, “breakpoints”, “watch”
 - debugging statements
 - E.g., output values of variables, markers at specific locations, etc

Example: Debugging Statements

```
...
for (i=0; i<N; i++)
{
    scanf("%s", name);
}
}
```

Example: Debugging Statements (cont)

```
const int debugging = 1;
...
for (i=0; i<N; i++)
{
    scanf("%s", name);

    if (debugging)
    {
        printf("for: i=%d, name=%s\n", i, name);
    }
}
```

TIP: make debugging statements conditional on a boolean variable

Example: Debugging Statements (alternative)

```
#define DEBUG 1
int main()
...
for (i=0; i<N; i++)
{
    scanf("%s", name);

    #if DEBUG
        printf("for: i=%d, name=%s\n", i, name);
    #endif
}
}
```

Testing Techniques

- Test data set should “fully” test the program
- All logical paths of the program should be traversed (i.e., every line of code should be executed at least once)
- Use the design represented by the flowchart

TIP: build your programs incrementally, testing small components as you go along

Example: BestMark

Problem:

- Write a program which reads a list of marks, and prints out the best mark
- Example:
 - Input: 18 56 65 96 24 30
 - Output: Best mark is 96

31

Example: BestMark (cont)

Algorithm

```
set bestMark to 0
loop
{
  input mark
  if (end of input)
  then exit loop
}
```

```
output "Best mark is ", bestMark
```

32

Example: BestMark (cont)

Algorithm

```
set bestMark to 0
loop
{
  input mark
  if (end of input)
  then exit loop

  if (mark > bestMark)
  then
  {
    set bestMark to mark
  }
}
```

```
output "Best mark is ", bestMark
```

33

Example: BestMark (cont)

How do I validate the input?



Algorithm

```
set bestMark to 0
loop
{
  input mark
  if (end of input)
  then exit loop

  if (mark > bestMark)
  then
  {
    set bestMark to mark
  }
}
```

```
output "Best mark is ", bestMark
```

34

Classes of Test Data

- Valid data
- Valid boundary data
- Special or unusual cases
- Invalid data

35

Test Data: Valid Data

- Reasonable data for the problem
- Example: BestMark
 - What is the test out of?
 - If mark is out of 100, valid test data is 75, 65, 55

36

Test Data: Valid Boundary Data

- Data with extreme values
 - Example: BestMark
 - minimum of 0
 - maximum of 100
- Test selection conditions
- Test iteration exit conditions
- Test first and last elements of an array

37

Test Data: Special Cases

- Example: BestMark
 - What if someone is absent or the mark is withheld (special consideration)?

```

input markEntered
if (markEntered is "Abs" or "WH")
{
    output "No mark for this student"
    set mark to 0
}
else
{
    set mark to numerical value of markEntered
}
    
```

38

Test Data: Invalid Data

- Invalid data is
 - of an incorrect type, or
 - outside the expected range
- Use features of the programming language to ensure correct data type
 - Example: BestMark
 - mark can be restricted to an integer

```

int mark;
scanf("%d", &mark);
    
```

39

Test Data: Invalid Data (cont)

```

input markEntered
...
set mark to numerical value of markEntered
if (cannot get number from markEntered)
{
    output "Invalid input"
}
    
```

```

...
if ((mark < 0) or (mark > 100))
{
    output "Mark has to be between 0 and 100"
}
    
```

40

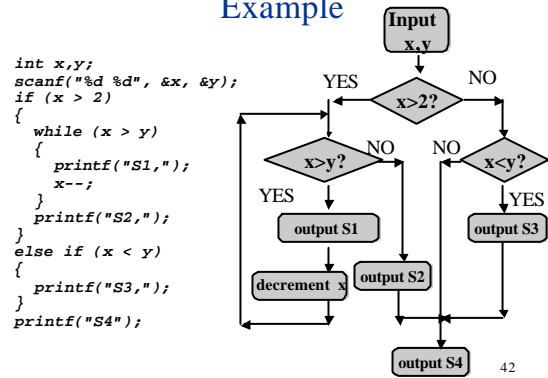
Algorithm: BestMark

```

set bestMark to 0
loop
{ input markEntered
  if (end of input)
  break loop
  if ( markEntered is "Abs" or "WH" )
  { output "No mark for this student"
  }
  else
  { set mark to numerical value of markEntered
    if (cannot get number from markEntered)
    { output "Invalid input"
    }
    else if ((mark < 0) or (mark > 100))
    { output "Mark has to be between 0 and 100"
    }
    else /* valid input! */
    { if (mark > bestMark)
      set bestMark to mark
    }
  }
}
output "Best mark is ", bestMark
    
```

best1

Example



42

Example (cont)

```

int x,y;
scanf("%d %d", &x, &y);
if (x > 2)
{
    while (x > y)
    {
        printf("S1,");
        x--;
    }
    printf("S2,");
}
else if (x < y)
{
    printf("S3,");
}
printf("S4");
    
```

43

Example (cont)

```

int x,y;
scanf("%d %d", &x, &y);
if (x > 2)
{
    while (x > y)
    {
        printf("S1,");
        x--;
    }
    printf("S2,");
}
else if (x < y)
{
    printf("S3,");
}
printf("S4");
    
```

44

Example (cont)

```

int x,y;
scanf("%d %d", &x, &y);
if (x > 2)
{
    while (x > y)
    {
        printf("S1,");
        x--;
    }
    printf("S2,");
}
else if (x < y)
{
    printf("S3,");
}
printf("S4");
    
```

45

Example (cont)

```

int x,y;
scanf("%d %d", &x, &y);
if (x > 2)
{
    while (x > y)
    {
        printf("S1,");
        x--;
    }
    printf("S2,");
}
else if (x < y)
{
    printf("S3,");
}
printf("S4");
    
```

46

Example (cont): Valid Data

- Which lines of code indicate what is valid data?


```

int x,y;
scanf("%d %d\n", &x, &y);
            
```
- Valid data is any integer:
 - positive,
 - negative, or
 - zero

47

Example (cont): Test data for all logical paths

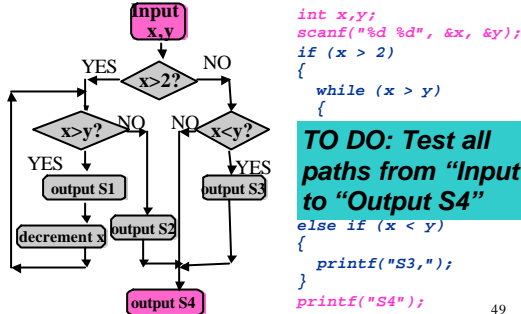
- What is done for every input?
- What does this say about the output?
 - S4 must be output at the end every time

```

int x,y;
scanf("%d %d", &x, &y);
if (x > 2)
{
    while (x > y)
    {
        printf("S1,");
        x--;
    }
    printf("S2,");
}
else if (x < y)
{
    printf("S3,");
}
printf("S4");
    
```

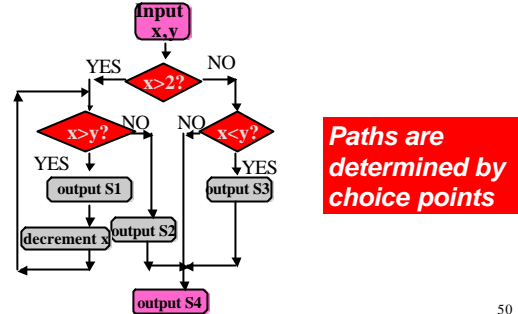
48

Example (cont): Test data for all logical paths



49

Example (cont): Choice Points



50

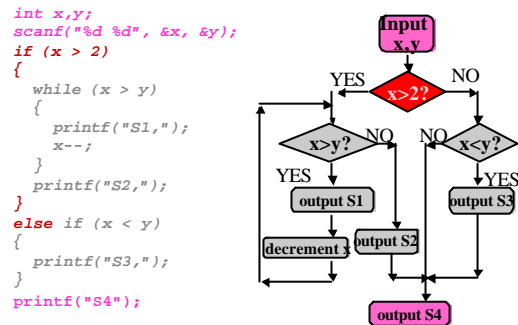
Example (cont): Choice Points

- What are the highest level choice points?

```
int x,y;
scanf("%d %d", &x, &y);
if (x > 2)
{
    while (x > y)
    {
        printf("S1,");
        x--;
    }
    printf("S2,");
}
else if (x < y)
{
    printf("S3,");
}
printf("S4");
```

51

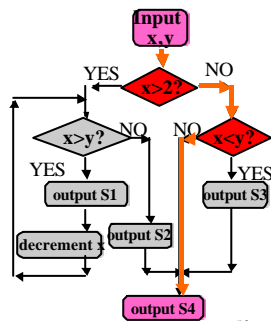
Example (cont): Choice Points



52

Example (cont): Choice Points

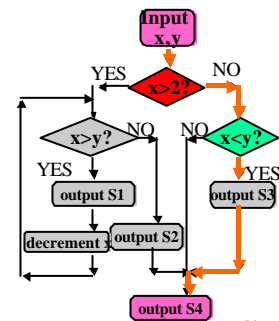
Test data Case 1:
NOT (x>2), NOT (x<y)
Specific Values:
x==2, y == 2
Output: S4



53

Example (cont): Choice Points

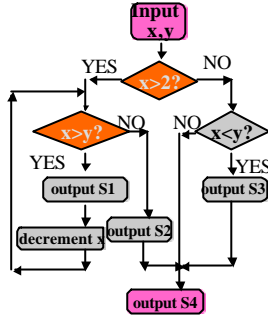
Test data Case 2:
NOT (x>2), x<y
Specific Values:
x==2, y == 3
Output: S3, S4



54

Example (cont): Choice Points

```
int x,y;
scanf("%d %d", &x, &y);
if (x > 2)
{
    while (x > y)
    {
        printf("S1,");
        x--;
    }
    printf("S2,");
}
else if (x < y)
{
    printf("S3,");
}
printf("S4");
```



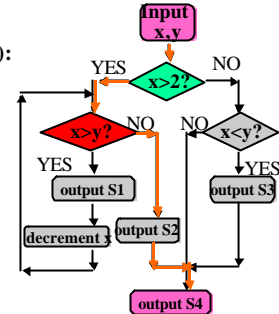
55

Example (cont): Choice Points

Test data Case 3
(Loop body not executed):
x > 2, NOT(x > y)

Specific Values:
x==3, y == 4

Output: S2, S4



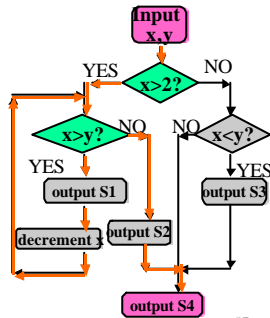
56

Example (cont): Choice Points

Test data Case 4
(Loop body executed):
x > 2, x > y

Specific Values:
x==5, y == 4

Output: S1, S2, S4



57

Notes on Loop Tests

- Is it possible that a loop never terminates?
– only if the algorithm is incorrect
- Example:

 <pre>while (x > y) { printf("S1,"); x++; }</pre> 	<pre>while (x > y) { printf("S1,"); x--; }</pre>
--	---

58

Exercise: Changing the Algorithm

```
/* Step 1 */
while (x > 0)
{
    /* Step 2 */
    if (y == 2)
    {
        /* Step 2a */
    }
    else
    {
        /* Step 2b */
    }
    /* Step 3 */
}
/* Step 4 */
```

- Provide a set of test data:
 - valid
 - valid boundary
 - invalid
- How would you ensure that the loop *always* terminates?

59

Summary

- Testing is an important part of the software development process
- Considering all the test data cases can lead to a change in the algorithm
- Flowcharts can be used to design the test data set

60