CMPSC 16
Problem Solving with Computers I
Spring 2014

Instructor: Lucas Bang

Lecture 5: Introduction to C: More Control Flow
How are the subtasks in an algorithm combined?

- They can be combined as a **sequence**: 
  - first execute task1, then execute task2
- They can be combined using **selection**: 
  - check a condition, if the condition is true execute task1, if not execute task 2
- They can be combined using **repetition** (aka **iteration**): 
  - repeat task1 until a condition becomes true

- Structured programming languages (like C) have language constructs for composing tasks using sequence, selection and repetition
  - We call these **control structures** since they control the execution order of the statements
C has 7 basic control structures

• 1st is trivial: sequence structure: statements listed sequentially are executed sequentially
  
  a = 4;
  b = 5;

• 3 choices of selection structures:
  - if
  - if/else
  - switch

• 3 choices of repetition structures:
  - while
  - for
  - do/while
Basic Operation | Pseudocode Notation | Flowchart Symbol
--- | --- | ---
Input | read radius | read radius

Computation | set area to $\pi \cdot \text{radius}^2$ | area $= \pi \cdot \text{radius}^2$

Output | print radius, area | print radius, area

Comparisons | if radius $< 0$ then ... | is radius $< 0$?

Beginning of algorithm | main: | start main

End of algorithm | | stop main
Sequence in flowchart
if selection structure as a flowchart
if/else Selection Structure
Practice True/False questions

• Say int x=2, y=8, z=17;

  – What is (x < y || z > x && y > z)?

  – What is (x < y - z)?

  – What is (x + y > y + z / y)?

• What is (--z == x * y++)?

  – And after that statement executes, what is (z > y * x)?
Practice True/False questions

• Say \( \text{int } x=2, \ y=8, \ z=17; \)
  
  – What is \((x < y || z > x && y > z)\)?  
    \[ \text{true} \]
  
  – What is \((x < y - z)\)?  
    \[ \text{false} \]
  
  – What is \((x + y > y + z / y)\)?  
    \[ \text{false} \]

• What is \((-z == x * y++)\)?  
  
  – And after that statement executes, what is \((z > y * x)\)?  
    \[ \text{false} \]
Updated precedence summary

<table>
<thead>
<tr>
<th>Precedence</th>
<th>Operation</th>
<th>Associativity</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>( )</td>
<td>Innermost first</td>
</tr>
<tr>
<td>2</td>
<td>++ -- + - ! (type)</td>
<td>Right to left (unary)</td>
</tr>
<tr>
<td>3</td>
<td>* / %</td>
<td>Left to right</td>
</tr>
<tr>
<td>4</td>
<td>+ -</td>
<td>Left to right</td>
</tr>
<tr>
<td>5</td>
<td>&lt; &lt;= &gt; &gt;=</td>
<td>Left to right</td>
</tr>
<tr>
<td>6</td>
<td>== !=</td>
<td>Left to right</td>
</tr>
<tr>
<td>7</td>
<td>&amp;&amp;</td>
<td>Left to right</td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>= += -= *= /= %=</td>
<td>Right to left</td>
</tr>
</tbody>
</table>
More Boolean expressions

• Note a difference from math descriptions:
  – In math: \((0 < \text{value} < 10)\)
    • that expression is always true in C no matter what value is
  – In C must be: \((0 < \text{value} \&\& \text{value} < 10)\)

• De Morgan’s Law – to simplify boolean expressions
  – Let A and B represent boolean values
    1. \(! (A \&\& B)\) is the same as \(!A \mid \mid !B\)
    2. \(! (A \mid \mid B)\) is the same as \(!A \&\& !B\)
switch selection structure
switch statement syntax

switch (controlling expression) {
    case constant expression:
        statements;
        break; /* otherwise “fall through” */
    case constant expression:
        statements;
        break;
    case constant expression:
        statements;
        break;
    ...
    default:
        statements without a condition;
}

– See .../demos/gradepoints.c
Nesting & indenting

- No such thing as multiple else blocks – others actually nested inside else block
  - e.g.,
    
    ```java
    if (grade >= 90)
      message = "Excellent";
    else
      if (grade >= 60)
        message = "Pass";
      else
        message = "Fail";
    
    - Gets messy, so usually else/if on same line:
      
      ```java
      else if (grade >= 60) ...```
Nesting/indenting (cont.)

• Critical to test relations in the correct order
  – Sometimes means stating the *negative condition*

• Also watch out for “dangling else” problems
  ```
  if (first-level condition)
    if (second-level condition)
      do something;
    else (what level?) …
  ```

  – last else goes with the closest if
while Iteration Structure (Loop)
while statement syntax

while (boolean expression) 
operation; /* or a block { } */

• E.g., do something 10 times:
  int counter = 0;       /* initialize */
  while (counter != 10) { /* compare to limit */
    ...
    ++counter; /* increment (or change) */
  }

  e.g., whiletab.c (4th Ed. Program chapter3_2, p. 103)
    • Note: better structure type for “counter-controlled” loop soon
  • while is best type when termination is uncertain
    – e.g., processing the characters or “tokens” in a string
    – e.g., reading unlimited lines of data from a file
for Iteration Structure

initialize

?  T  increment

F
for statement syntax

• More natural than while to implement counter-controlled loops:

```c
int c;
for (c = 0; c < 10; c++)
    /* statement; or {...} */
```

• Example: fortab.c (p. 106)
• Note: header requires three fields (i.e., always two “;”), but okay to leave blank or even use , to do multiple things
do/while Iteration Structure
do/while

do {
    ... /* loop body */
} while (boolean expression);

• Example: dotab.c (p. 104)
• Notes:
  – Always executes at least once
    • e.g., good for user input checking
  – Don’t forget the semicolon at the end
break, continue statements

• Sanctioned ways to loosen the structures
  – Without using the dreaded goto statement!
  – Usually as a result of some condition:
    • e.g., if (/* a good reason */) break;
• break – says exit the structure now
  – Control diverts to 1\textsuperscript{st} statement after closing `}`
    • Already saw how it skips remaining cases in switch structure
• continue – says terminate this iteration of loop
  – Repetition structures only – restarts at next iteration
Review: 7 basic control structures in C

- **Sequence**
- **Selection**
- **Iteration**

- Increment
- Initialize
- Break
7 basic control structures in C

- Sequence structure: statements listed sequentially are executed sequentially
  
  \[ a = 4; \]
  \[ b = 5; \]

- 3 selection structures:
  - if
  - if/else
  - switch

- 3 repetition (loop) structures:
  - while
  - for
  - do/while