Conditional Statements & Loops

ELEC1201 Lecture 4

4 October 2013

Last edit 1 October 2014

Enrico Costanza and Jeff Reeve
Textbook

- The material covered today is in chapter 5
- Please go through chapters 3, 4 and 5 before the lab
Credits + feedback

- The quizzes in these slides are from http://www.cprogramming.com/
- The rest of the material is new, so feedback is welcome
Quiz to check your understanding

- Which of the following shows the correct syntax for an if statement?
  - A. `if` expression
  - B. `if { expression }
  - C. `if ( expression )`
  - D. `expression if`
Quiz to check your understanding

• What will be printed on screen from the following code?

```c
if ( (1 && (4 & 1)) )
{
    printf("true");
}
else
{
    printf("false");
}
```

– A. true
– B. false
Let's Look Again at If-else

- General definition:
  ```java
  if (<condition>)
  {
      <statements executed if condition is true>
  }
  else
  {
      <statements executed if condition is false>
  }
  ```
Nested if-else statements

- Adding to the earlier examples:

```c
if (temperature <= 0.0)
{
    printf("it's freezing!\n");
}
else
{
    if (temperature > 28.0)
    {
        printf("it's quite hot!\n");
    }
    else
    {
        printf("don't know what to say..\n");
    }
}
```
Nested if-else statements (cont.)

- Nested statements can quickly become confusing and hard to read
- Make sure you use {} and indentation appropriately!
- When we work with integer values, in some situations there is a good alternative to nested if-else statements
The switch Statement

• General structure:

```java
switch(<test-value>){
    case <match-value>:
        <statements to execute when matched>
        break;
    case <match-value>:
        <statements to execute when matched>
        break;
    default:
        <statements to execute when no match found>
}
```
The switch Statement (cont.)

- Elements in gray are optional:

```java
switch(<test-value>){
    case <match-value>:
        <statements to execute when matched>
        break;
    case <match-value>:
        <statements to execute when matched>
        break;
    default:
        <statements to execute when no match found>
}
```
The switch Statement (cont.)

- There may not be a default case
- If break is not present the other cases will be also executed
Switch-case example 1

- Simple example with integers:

```c
switch(integerVariable){
    case 1:
        printf("One\n");
        break;
    case 2:
        printf("Two\n");
        break;
    default:
        printf("It was neither one nor two\n");
}
```
Switch-case example 2

- In this case the first case does not have a break statement:

```c
switch(integerVariable){
    case 1:
        printf("One\n");
    case 2:
        printf("Two\n");
    break;
    default:
        printf("It was neither one nor two\n");
}
```
Switch-case example 3

• In C char variables are also integer numbers!

• Simple example with char:
  ```c
  switch(charVariable){
    case 'a':
      printf("It's a!\n");
    break;
    case 'b':
      printf("It's b!\n");
    break;
    default:
      printf("It is neither a nor b\n");
  }
  ```

• This would not work with strings!
Quiz to check your understanding

- What follows the case statement?
  - A. :
  - B. ;
  - C. -
  - D. A newline
Quiz to check your understanding

• What is required to avoid falling through from one case to the next?
  – A. end;
  – B. break;
  – C. stop;
  – D. A semicolon.
Quiz to check your understanding

• What is the result of the following code?

```c
int x=0;
switch(x)
{
    case 1: printf( "One" );
    case 0: printf( "Zero" );
    case 2: printf( "Hello World" );
}
```

– A. One
– B. Zero
– C. Hello World
– D. ZeroHello World
Ternary Operator

- A similar behavior to the if-else statements can be achieved using the ternary operator

- Its anatomy is:
  \[
  (\langle \text{condition} \rangle) \ ? \ \langle \text{statement executed if condition is true} \rangle \ : \ \langle \text{statements executed if condition is false} \rangle;
  \]

(all just one line)
Ternary Operator

- For example:
  
  ```
  (temperature <= 0.0) ? printf("freezing") : printf("not freezing");
  ```

- This is roughly equivalent to:
  ```
  if (temperature <= 0.0)
  {
      printf("freezing");
  }
  else
  {
      printf("not freezing");
  }
  ```
Ternary Operator

- Ternary operator expressions can be embedded in other statements, e.g.:
  ```
  int var = (temperature <= 0.0) ? 0 : 1;
  ```

- Remember: try to make your code as easy to read as possible!

- Use the ternary operator only if it makes the code more readable!
Loops

- Loops are another key features of programming
- A loop is a piece of code that is repeated N times depending on some condition
- In C we have 3 loop structures:
  - for
  - while
  - do..while
The for Loop

- The anatomy of the for loop is:
  ```
  for (<initial statement>; <test expression>; <cycle statement>)
  {
      <statements to be executed each cycle>
  }
  ```

- The *initial statement* is executed *before the beginning of the loop* – this is *always* executed once.

- The *test expression* is used to decide whether the loop is executed (again?) or not.
  - It may be *never* executed!

- The *cycle statement* is executed *at the end of each cycle*. 
The for Loop: Simple Example

- A simple example:
  ```c
  int i; /* we need to declare this variable before the loop */
  for (i=0; i < 4; i++)
  {
    printf("%d\n", i);
  }
  ```
  
- Before the loop starts i is set to zero

- The loop is executed if i is less than 4

- After each cycle the value of i is incremented
The for Loop: Array Example

For loops are very useful to work with arrays:

/* assume we declared int i and an int array arr of size >= 4 */
for (i=0; i < 4; i++)
{
    printf("element %d of the array is %d\n", i, arr[i]);
}
The while Loop

• The while loop is "simpler" in that it only involves the test:

```c
while (<test expression>)
{
    <statements to be executed each cycle>
}
```

• Similar to the for loop note that the cycle may never be executed
Simple while Loop Example

- Let's plug in some values:

```c
int i=0;
while (i < 4)
{
    printf("%d\n", i);
    i++;
}
```
Simple while Loop Example

• In this case the loop will never run!

    int i=5;
    while (i < 4)
    {
        printf("%d\n", i);
        i++;
    }
The do..while Loop

- Similar to the while loop, but the cycle is always executed at least once:

```java
do
{
    <statements to be executed each cycle>
}
while (<test expression>);
```

- The expression is tested after each cycle
Simple do..while Loop Example

• Let's plug in some values:

```c
int i=0;
do
{    printf("%d\n", i);    i++;}
while (i < 4);
```

• This produces the same output as the while loop example
Nested for Loops

- The statements inside a loop can include other loops (and indeed they often do!), for example:

```c
int i, j;
for (i=0; i<4; i++)
{
    for (j=0; j<4; j++)
    {
        printf("i: %d, j: %d\n", i, j);
    }
}
```
Skipping to the Next Loop Cycle

- Sometimes it may be useful to skip the rest of a loop cycle and go straight to the next one, e.g.

```c
int i;
for (i=0; i<10; i++)
{
    if (i%2 == 0)
    {
        continue;
    }
    printf("%d\n", i);
}
```

- However, use it only if it is really necessary! Often it is more clear to modify the loop otherwise.
Breaking Out of Loops

• Sometimes it may be useful to "jump out" of a loop, before the end condition is met, e.g.

```c
int i;
for (i=0; i<4; i++)
{
    if (i==2)
    {
        break; /* this terminates the loop */
    }
}
```

• However, use it only if it is really necessary!
  Often it is more clear to modify the loop test condition
The Infamous goto Statement

• The goto statement allows the code to "jump" to a location identified through a label:

```plaintext
goto <label>;
<statements to be skipped>
<label>: 
<other statements>
```
The Infamous `goto` Statement (cont.)

- Example:

```c
int j;
for (j=0; j<10; j++)
{
    if (j==3)
    {
        goto end;
    }
    printf("j: %d\n", j);
}
end:
printf("end");
```
The Infamous `goto` Statement (cont.)

• Very often the are better (more clear) alternatives to using `goto` statements

• The use of `goto` is generally not advised

• Yet there are good examples of its use, see http://stackoverflow.com/questions/245742/examples-of-good-gotos-in-c-or-c
Quiz to check your understanding

• What is the final value of x when the code
  \texttt{int x; for(x=0; x<10; x++) {}}
  is run?

  – A. 10
  – B. 9
  – C. 0
  – D. 1
Quiz to check your understanding

• When does the code block following `while(x<100)` execute?
  – A. When x is less than one hundred
  – B. When x is greater than one hundred
  – C. When x is equal to one hundred
  – D. While it wishes
Quiz to check your understanding

• Which is not a C language loop structure?
  – A. for
  – B. do while
  – C. while
  – D. repeat until
Quiz to check your understanding

- How many times is a for loop guaranteed to loop?
  - A. 0
  - B. Infinitely
  - C. 1
  - D. Variable
Quiz to check your understanding

• How many times is a do while loop guaranteed to loop?
  – A. 0
  – B. Infinitely
  – C. 1
  – D. Variable
Summary

- Nested `if-else` statements can be sometimes replaced by `switch..case` statements.
- Loops are portions of programs that repeat while a condition is true.
- C offers `for`, `while` and `do..while` loops.
- The `break`, `continue` and `goto` instructions can be used to implement exceptional behaviors: use with care!
- Keep your code easy to read!