

# Lecture 02 Notes

Derek Harter

2015-01-20

## 1 First Session (11 - 11:40)

### 1.1 Memory Concepts, Variables, Representation

- When you declare a variable, it is assigned a location in memory. All references to store or access the named variable are translated by the C compiler using a lookup table.
- The datatype is how pattern of bits in memory are stored and interpreted to mean a number, a character, etc.
- Int32 representation ([http://en.wikipedia.org/wiki/Integer\\_%28computer\\_science%29](http://en.wikipedia.org/wiki/Integer_%28computer_science%29))
- Debug → QuickWatch
- Debug → Windows → Memory
- Representation of integer 1 vs. -1

### 1.2 Arithmetic and Precedence

- Rules of precedence
- Integer division vs. real division
- Modulus operator

### 1.3 Relational Comparisons, and the bool type

- Making decisions, we need be able to perform alternative actions based on making some sort of decision.
- bool should be used instead of an int for boolean flags in a program in modern c.

## **2 Second Session (11:45 - 12:30)**

### **2.1 Algorithm**

An algorithm is a series of unambiguous explicit steps that can be taken to solve a problem. An algorithm is a procedure for solving a problem in terms of:

1. the actions to execute and
2. the order in which the actions execute

### **2.2 Control Structures**

Normally statements in a program execute one after the other in the order in which they are written. This is sequential execution.

- The sequence structure is built into the C language normal operation.  
Question: what/where is the first instruction executed in a C program?
- Selection statements: single if, double if / else, multiple switch
- Repetition statements: for, while (0 or more) and do..while (1 or more)

### **2.3 Multiple Selection with if..else**

- can make 1 of multiple possible decisions using nested if statements.

### **2.4 Basic loop**

- while loop repeats a set of statements 0 or more times, checking for a condition before beginning and before each repetition of the statements in the body of the loop.
- An infinite loop occurs when you don't properly change values so that the condition of the loop is eventually false.

### **2.5 Different types of repetition**

- Our first while loop is a kind of purely logical controlled repetition.  
We keep repeating as long as the condition

is true, and once it becomes false we stop.

- A common variation is, Counter-Controlled repetition, when you know you need to process X specific items (we will see a better way of doing this with a for loop next.)

- Discuss an off-by-one error in a counter-controlled loop.
- Sentential-controlled repetition means we keep repeating until we see some condition. Our first example is really a sentential-controlled loop, though a more typical example is to have some explicit sentential/flag.

## **2.6 Increment, decrement and assignment**

- C provides several assignment operators for abbreviating assignment since it is very common to need to manipulate a value and assign it back to the variable. (`+=`, `-=`, `*=`, etc.)
- increment and decrement are unary operators. We can increment and decrement using pre or post increment, which means either we first increment the value then use that in the expression, or we first use current value in expression then increment after.

# **3 Third Session (12:40 - 1:40)**

## **3.1 for Loop for Counter-Controlled Repetition**

- Counter-controlled loops are so commonly needed (especially when processing arrays of elements later), that C provides a special construct for implementing them.
- Parts of a for loop, initialization, test, manipulation.
- Give examples of counting up and counting down.
- Give example of counting with a step size.

## **3.2 do..while Repetition**

- If we need to ensure some statements are always executed at least once, use do..while construct. Useful so we can avoid duplicating some code.

### 3.3 switch statement

- You may run across the switch statement. Switch provides multiple-selection (like nested if).
- It is limited, can only compare the value of a variable to a constant integral expression (e.g. our example of grades in ranges can't be accomplished using switch, we still need to use nested if for them.)
- switch statement requires use of break statements.
- break and continue statements can be useful sometimes in loops, to avoid some unnecessary repetitions of the loop.
- For example, in the prime number algorithm, we don't need to perform any more loops after we find first negative example.
- Can use a break statement. An equivalent effect can be achieved by using a flag.
- continue can be used to skip any remaining statements in current iteration, and then start immediately with the next iteration of the loop.

### 3.4 Visual Studio Tips

- Turn on expert settings
- Format selection Ctrl-k Ctrl-f