# MECH 215

Lecture 2

Computers and Programming Basics

# Programming

- There a number of ways to use assignment operations.
- The most common operators for mathematical relationships are addition +, subtraction -, multiplication * and division /.
- There is the same precedence in operation as with normal math, it is often a good idea to use brackets to insure that the operations occur in the order that you want.
- X=(3+x)*4

# Programming

- In the case of logical operations there is:
  - AND true if all are true else false
  - OR true if at least one is true
  - NOT true becomes false and false becomes true
- There are auto increment features
- i=i+1 can be written as i++
- J=j+5 can be written as j +=5
- Similarly for multiply and divide.
- Note that i++ adds after (post increment) while ++i adds before (pre increment)

# Programming

- Input and output are handled by the cin and cout objects.
- cout feeds a value from a memory location to the screen and cin feed a value from the keyboard into a memory location.
- In addition to performing operations to manipulate data you may also compare data.
- This will be used when we look at control and program flow.

# Programming

- The first control statement is the if statement.
- If something is true do an activity.
- There is one if for each condition but you may perform multiple operations for each if.
- If (condition)
- {
- Statement 1;
- Statement 2;
- ….
- };

# Programming

- If there are only two mutually exclusive events (basically a yes no question) then there is also a if-else statement.
- If the first condition applies do the instructions under the if, otherwise do the situations under the else.
- This form of control is easy to use but you must make sure that you test the behavior properly.
- There is also the possibility to use
- If(cond)
- Else if (cond)
- Else if (cond)
- Else
- Structures as well.

# Programming

- There is also the switch and case statement.
- This will look at a list of possible input values (like a menu) and will perform a set of tasks based on which input matches the value on the list.
- Use the Break command to isolate the options.
- Acts like a function switch (input_variable)
- {
- Case ‘first choice’:
- do something
- Break;
- Case ‘second choix’:
- do another thing
- Break;
- ….
- Default:
- Do default operations
- };

# Programming

- The case statement can be replaced with if, else if and else statements with proper conditions.
- Typically the switch and case statements are used for menus or other finite number of choice control structures.
- The if statement is more general and is typically the favorite choice for general control.

# Programming

- Looping is the  process of repeating a set of instructions until a certain condition is met.
- There are while loop, for loop as well as do until loops.
- While (condition)
- {
- Instruction 1
- Instruction 2
- Instruction 3
- }

# Programming

- Do
- {
- Instruction 1
- Instruction 2
- Instruction 3
- } while (condition)
- Unlike the while loop which tests the condition before doing the instructions in the loop (PRE-TEST) the do while loop tests after all of the instructions have been performed (POST-TEST).

# Programming

- The condition portion may have more than one condition.
- For example if the loop will work as long as x is greater than 10 *or* less than 3 we could write:
- While ((x>10)||(x<3))
- If both conditions must be true (AND) we could write:
- While (((x>10)&&(x<3))
- If the condition was x must not equal 12 we could write:
- While (x!=12)
- There are many operators and possible instructions, it is a good idea to read the text to become more familiar with them.

# Programming

- For (initialization; test; update)
- {
- Instruction 1
- Instruction 2
- Instruction 3
- }
- Note that the for loop will test if the condition is true, if yes it will perform the instructions within the loop, then update the variable.

# Programming

- Loops can be nested,
- This means you can have a loop within a loop (over even more).
- The order of the loops will determine which instruction occurs first. To illustrate let us look at a pseudo code loop.
- For i=1:10
- For j=1:10
- x=x+j
- end j loop
- Y=y+i
- End I loop

# Programming

- The loop would start with i=1 and when it encounters the j loop it will perform the j loop for j=1 to 10 (assuming that when j=1 the loop ends) and then continue to the next step.
- Assuming that before the loops x=0 and y=0, the first iteration would end with i=1, j=10 and x=1+2+3+4+5+6+7+8+9 and y=1.
- The next iteration would be i=2, j=10 and x=2*(1+2+3+4+5+6+7+8+9) and y=1+2.

# Programming

- The use of loops requires some careful thought when selecting conditions to end the loop. Do not re-use variables or indices unless you are sure that the loop operation (especially the ending conditions) will not be adversely affected.
- The variables used in a loop may be defined prior to the loop and this is often quite helpful. It is easy to edit code where all of the int, bool, float, etc.. Statements are in the same place.

# Programming

- Structure is a method of linking multiple data elements to one type or class.
- Struct name_of_stuct
- {
- Int integer_variable_1, int_variable_2;
- Bool bool_var_1;
- Float float_variable_1.5;
- Double double_precision_floating_point_var_1.50;
- Char you_get_the_idea;
- };

# Programming

- The struct instruction defines what are the parts that are used to make a particular structure.
- When a structure is defined
- Name_of_struct Concordia_student_1
- Then the code can put values into the memory locations reserved for this structure. For example if I wanted to put the value “10” into the integer_variable_1 portion of the Concordia_student_1 structure I would write:
- Concordia_student_1.integer_variable_1=10;

# Programming

- Structures are very helpful when making changes to an existing code. This will allow you to add or modify the structure in one place and the code will use this as a template for all of the structures in the code.
- Note that the period is used to access a particular part of a particular structure. In the case of nested structures we could write.
- My_struct_1.first_part.second_part.third_part.last_part=10

# Programming

- This would access the structure my_struct_1. Then it would access first_part, this is also a structure which has a portion called second_part. We access second part to found out it is also a structure with a part called third_part. When we access the third part we can access the last portion of memory which is last_part. When then placed within that location the value 10.
- This may seem a bit involved when we start but with practice it can be quite helpful to work with.

# Programming

- I have placed some codes and examples for you to work with.
- I would suggest that you look at the codes and become familiar with how they operate, how to write you won codes and how to modify these codes for your own needs.

# Next Lecture

- In the next lecture we will discuss step-wise refinement of code.