# Introduction To Computer Programming C++
# Mr. Clausen
# Program C9A, C9B, C9C, C9G

**Program 9A: Numbers, Squares, and Cubes 25 points**

Write a program that uses a **"for loop"** that will ask the user for an integer between 1 and a user specified limit (or upper limit - I will test your program using the number 25) and print a list of these numbers, their perfect squares and their prefect cubes. Save the program as LastNameFirstNameP9A.cpp in your "S:" directory. To see a model for this program, look at the source codes for the programs "books.cpp" and "pizza.cpp" in the network directory titled: IntroCompProgFiles. Look in the folder, Other C++ Resources and the folder Other Textbook Examples.
As you type all your programs this year, be sure not to type past the 80-column line in Borland C++ 5.02 for Windows. If you have any statements longer than 80 columns, press the return key to "wrap" the statement around to the next line.

1) Type comments at the beginning of the program to display your name and other information just like those used for program 1A. **Make sure to change the program name and program description in these comments, so that the program number, name, and description say what is listed above.**

2) Include <iostream.h> so we can use the **cout** and **cin** commands. Also include <conio.h> so you can use **getch**() to leave your output displayed on the screen until the user presses any key to continue. Include <iomanip.h> so we can use the **setw** command. To raise one integer to another integer power you will need to add: #include <math.h>.

3) There are no constants necessary for this program.

4) Inside the **int** main() function, on the first line below the left curly bracket that begins the main function, declare variables of type **int** named upper_limit, number, square, and cube. When using the variable counter in the for loop, declare this variable in the first part of the for loop as an integer.

5) In all our programs, follow the Input, Calculations, and Output organization of your program. Make sure that you include the following comment lines in the **int** main ( ) portion of your program (each comment followed by the appropriate source code).

6) After the variable declarations (before the input section) use **cout** statements to display your name and period output just like those used for program 1A **Make sure to change the program name and program description in these cout statements.** Start these commands with the following statement:
//------------------------------Display My Information-----------------------

7) For the Input section, ask the user to enter an integer that represents the upper limit. Assign the user's input to the variable upper_limit. Make your program user friendly by prompting them for this value. Start these commands with the following statement:
//------------------------------------Input----------------------------------

8) We are going to combine the calculations and output sections of your program into one part. It should consist of the following comment line, a heading for each column for number, square, and cube, and commands to display the number, and its square and cube. Start these commands with the following statement:

```
//--------------------------Calculations and Output--------------------------
```

Create a "heading" for these numbers and have the numbers listed in three columns on the screen. To format the numbers nicely use the setw command to set a width for each of the columns. You also need to right justify these numbers. This will require that you add #include <iomanip.h> to your preprocessor directives and use the following commands: cout << setiosflags (ios::right) ; and cout<< setw(15); To raise one integer to another integer power you will need to add: #include <math.h>; Make sure that you use assignment statements to calculate the squares and cubes. To square a number, you can multiply the number by itself, for example, square = number * number; You can also use the pow function to square a number. The command would look like this: square = pow(number, 2);. To cube a number, you can multiply the number by itself three times, for example, cube = number * number * number; You can also use the pow function to cube a number. The command would look like this: cube = pow(number, 3);. All of these commands along with cout statements need to be inside the body of the for loop.

Your output should look like this:

```
Number        Square          Cube
= = = = = = = = = = = = = = = = = = = = =
      1             1             1
      2             4             8
      3             9            27
```

…until the user's upper limit.

When you are finished with your program, have tested it thoroughly to make sure that your calculations are correct, and are sure that you don't need to make any changes, then save your program in the "W" network mapping, and the Program 9A folder.

**Program 9B: Menu Choice Celsius To Fahrenheit Or Visa Versa Part 2 (10 points)**

1. You are going to add some lines of code to your program to convert a temperature from Celsius to Fahrenheit or from Fahrenheit to Celsius (This was program 8B Menu Choice.). Save the program as LastNameFirstNameP9B.cpp in your "S:" directory.

2. Add a **do…while loop** to your **int** main function that repeats the entire program until the user enters a lowercase 'q' or uppercase 'Q'. The reserved word 'do' should appear after all of the variable declarations and initializations, and the output of your name. The while statement will be near the end of the **int** main function and needs to have a **compound Boolean expression** to consider both cases of upper case Q, and lower case q. **Be sure to save your program before you run it, as you might end up with an infinite loop if your logic is not correct.**

When you are finished with your program, have tested it thoroughly to make sure that your calculations are correct, and are sure that you don't need to make any changes, then save your program in the "W" network mapping, and the Program 9B folder.

**Program 9C Arithmetic Series 25 points**

Write a program to add all the numbers from 1 to a user specified limit (I will test your program using the number 100) and display the sum. The output of the program should be the numbers 1 through the upper limit in the first column, and the running total of the numbers in the second column. Although this could be written with a "for loop", **I am requiring that you to use a "count controlled while loop" instead**.

Save the program as LastNameFirstNameP9C.cpp in your "S:" directory. To see a model for this program, look at the source codes for the programs "books.cpp" and "pizza.cpp" in the network directory titled: IntroCompProgFiles. Look in the folder, Other C++ Resources and the folder Other Textbook Examples. **I also recommend that you look carefully at slide number 37 in the lecture notes for this program.**
As you type all your programs this year, be sure not to type past the 80-column line in Borland C++ 5.02 for Windows. If you have any statements longer than 80 columns, press the return key to "wrap" the statement around to the next line.

1) Type comments at the beginning of the program to display your name and other information just like those used for program 1A. **Make sure to change the program name and program description in these comments, so that the program number, name, and description say what is listed above.**

2) Include <iostream.h> (so you can use the cout and cin commands), and inlcude <conio.h> so you can use getch() to leave your output displayed on the screen until the user presses any key to continue. Include <iomanip.h> so we can use the **setw** command.

3) There are no constants to declare.

4) Inside the **int** main() function, on the first line below the left curly bracket that begins the main function, declare variables of type **int** named upper_limit, counter, and sum. Don't forget to initialize all of these variables to zero.

5) In all our programs, follow the Input, Calculations, and Output organization of your program. Make sure that you include the following comment lines in the **int** main ( ) portion of your program (each comment followed by the appropriate source code).

6) After the variable declarations (before the input section) use **cout** statements to display your name and period output just like those used for program 1A **Make sure to change the program name and program description in these cout statements.** Start these commands with the following statement:
//-------------------------------Display My Information-----------------------

7) For the Input section, ask the user to enter an integer that represents the upper limit. Assign the user's input to the variable upper_limit. Make your program user friendly by prompting them for this value. Start

these commands with the following statement:
//-----------------------------------Input----------------------------------

8)  We are going to combine the calculations and output sections of your program into one part.  It should consist of the following comment line, a heading for each column for number, and sum, and commands to display the number, and its sum.  Start these commands with the following statement:
//--------------------------Calculations and Output--------------------------

Create a "heading" for these numbers and have the numbers listed in two columns on the screen.  To format the numbers nicely use the setw command to set a width for each of the columns.  You also need to right justify these numbers.  This will require that you add #include <iomanip.h> to your preprocessor directives and use the following commands: cout << setiosflags (ios::right) ; and cout<< setw(10);

The output should look like this:

```
Number        Sum
= = = = = = = = = = = = =
     1          1
     2          3
     3          6
     4         10
```

etc… until the user entered upper limit.

When you are finished with your program, have tested it thoroughly to make sure that your calculations are correct, and are sure that you don't need to make any changes, then save your program in the "W" network mapping, and the Program 9C folder.

**Program 9G   program First Animation    25 points + up to 5 possible extra credit points**

For this program, don't draw any pictures in the background of the graphics screen.  Keep the background a solid color (it can be black or any other solid background color).

Your job is to animate an object (a circle or rectangle, etc.) across the graphics screen.  If you wish to earn the 5 extra credit points, then you must use a shape other than those built in to Borland.  For the extra credit, draw a  simple "object" and keep it relatively small (no more than 50 by 50 pixels) like a small boat, car, plane, etc.

**For animation, draw the picture in a foreground color, then draw the same picture in the background color to erase it.  DO NOT USE cleardevice( ) to erase your animated shape!**

Your animation must be a combination horizontal and vertical movement or, a diagonal line animation from one corner of the screen to another corner.

Use any type of loop that is appropriate to your animation.

For those of you that are in Algebra, remember the formula to calculate the equation of a line between any 2 points on the graphics screen:

$$y - y_1 = \frac{y_2 - y_1}{x_2 - x_1}(x - x_1)$$

**<u>Note</u>**:
When you use initgraph and tell the compiler where Borland's bgi module is **use:** "c:\\bc5\\bgi" an absolute directory reference, instead of relative directory reference (what the book uses… "c:..\\bgi") I will deduct points off of anyone who does not make this change. This will work much better in our networked environment, because it is an absolute reference to the location of the bgi. The notation c: .. Simply moves up a level in your current directory. Our programs are on the network, and the absolute reference works best (especially for me when I need to grade your programs: get the hint?).

**Please use this graphics mode:**
int graphdriver = VGA, graphmode= VGAHI;

When you are finished with your program, have tested it thoroughly to make sure that your calculations are correct, and are sure that you don't need to make any changes, then save your program in the "W" network mapping, and the Program 9G folder.