Lab 9

Nested Loops

Learning Objectives

The objective of this lab is to understand more about performing iterations in C. We have worked with the while loop, the do-while loop and the for loop. Now we illustrate the use of nested loops. Though in general one can inter-change the types of loops nested (i.e. one can have an inner while-loop and an outer for-loop), in this lab we use for loops nested within another for loop.

Basic Lab Instructions!

- Talk to your classmates for help.
- You may want to bring your textbook to future labs to look up syntax and examples.
- Stuck? Confused? Have a question? Ask a TA/Lab Engineer for help, or look at the book or past lecture slides.
- Complete as many problems as you can within the allotted time. You don't need to keep working on these exercises after you leave the lab.
- Before you leave today, make sure to check in with one of the Lab Engineers/TAs in the lab to get credit for your work.

Lab Tasks

Creating a New Project

After completing this section you will have created a project for today's lab.

1. Open the solution CS110Labs you created in the lab last week.
2. Add a new project called Lab08 within the solution CS110Labs. Set the project as startup project (right-click the project in Solution Explorer). STOP HERE! start with the lab tasks, and refer to the following steps accordingly.
3. Click File in the top menu bar of Visual Studio, and then select the Add New Item option (Add Existing Item option depending on the lab task).
4. Select C++ File
5. Type in the code listed.
6. Compile (Build Lab08) the program and execute it (Execute without Debugging).

Task #1: Nesting Loops

In programming, the term "nested" usually means to put something inside the same thing. "Nested loops" would be two loops with one inside the other one. If you do it right, then means the inner loop will repeat all its iterations every time the outer loop does one more iteration.

Start by opening NestingLoops.c with the following code in your project, and get it to compile.

```
1   // this is #1 - I'll call it "CN"
2   for ( char c='A'; c <= 'E'; c++ )
3   {
4     for ( int n=1; n <= 3; n++ )
5     {
6       printf( "%c %d\n", c, n );
7     }
8   }
9
10  printf("\n\n");
11
12  // this is #2 - I'll call it "AB"
13  for ( int a=1; a <= 3; a++ )
14  {
15    for ( int b=1; b <= 3; b++ )
16    {
17      printf( "%d-%d ", a, b );
18    }
19    /* your code comes here!!! */
20  }
```

What You Should See

```
1  A 1
2  A 2
3  A 3
4  B 1
5  B 2
6  B 3
7  C 1
8  C 2
9  C 3
10 D 1
11 D 2
12 D 3
13 E 1
14 E 2
15 A 3
16
17
18 1-1 1-2 1-3 2-1 2-2 2-3 3-1 3-2 3-3
```

Task #2: What You Should Do on Your Own

1. Look at the first set of nested loops ("CN"). Which variable changes faster? Is it the variable controlled by the outer loop (c) or the variable controlled by the inner loop (n)? Answer in a comment.

2. Change the order of the loops so that the "c" loop is on the inside and the "n" loop is on the outside. How does the output change?

3. Look at the second set of nested loops ("AB"). Add newline (\n) at the end of printf() statement. How does the output change? (Then change it back.)

4. Add a printf("\n") statement after the close brace of the inner loop (the "b" loop), but still inside the outer loop. How does the output change?

Task #3: Odometer Loops

Open the following code (OdometerLoops.c), and get it to compile.

```
  for ( int thous=0; thous<10; thous++ )
  {
    for ( int hund=0; hund<10; hund++ )
    {
      for ( int tens=0; tens<10; tens++ )
      {
        for ( int ones=0; ones<10; ones++ )
        {
          printf( " %d%d%d%d\r", thous, hund, tens, ones );
          delay( 10 );
        }
      }
    }
  }
```

What You Should See

```
 9999
```

What You Should Do on Your Own

1. Delete all the open braces and close braces from all the outer for loops. Leave the curly braces that belong to the innermost loop (the "ones" loop). Does it still work? Answer in a comment.

2. Change all the loops so that they count from 0 to 7 instead of 0 to 9. This will display numbers in "octal" (base 8) instead of "decimal" (base 10).

3. Change the code so that the human gets to type in a number for the base, and your odometer counts up to that instead of 8.

4. After you've made all the changes, it should look something like this (except that all your numbers will be overwriting each other on the same line instead of printing on separate lines):

```
Which base (2-10): 2
 0000
 0001
 0010
 0011
 0100
 0101
```

Task #4: Basic Nested Loops

Use some simple nested for loops to generate all possible coordinates from (0,0) up to (5,5). Name the program BasicNestedLoops.c. Your output must appear in rows and columns as follows.

```
(0,0) (0,1) (0,2) (0,3) (0,4) (0,5)
(1,0) (1,1) (1,2) (1,3) (1,4) (1,5)
(2,0) (2,1) (2,2) (2,3) (2,4) (2,5)
(3,0) (3,1) (3,2) (3,3) (3,4) (3,5)
(4,0) (4,1) (4,2) (4,3) (4,4) (4,5)
(5,0) (5,1) (5,2) (5,3) (5,4) (5,5)
```

Task #5: Multiplication Table

Use nested for loops to generate a multiplication table, which should go all the way up to 12×9. Name the program MultiplicationTable.c. Your output must appear as follows.

```
 x |  1   2   3   4   5   6   7   8   9
===+=================================
 1 |  1   2   3   4   5   6   7   8   9
 2 |  2   4   6   8  10  12  14  16  18
 3 |  3   6   9  12  15  18  21  24  27
 4 |  4   8  12  16  20  24  28  32  36
 5 |  5  10  15  20  25  30  35  40  45
 6 |  6  12  18  24  30  36  42  48  54
 7 |  7  14  21  28  35  42  49  56  63
 8 |  8  16  24  32  40  48  56  64  72
 9 |  9  18  27  36  45  54  63  72  81
10 | 10  20  30  40  50  60  70  80  90
11 | 11  22  33  44  55  66  77  88  99
12 | 12  24  36  48  60  72  84  96 108
```

Task #6: Number Puzzles I

Use nested for loops to generate a list of all the pairs of positive two digit numbers whose sum is 60, and whose difference is 14. Name the program NumberPuzzles1.c.

Task #7: Getting Individual Digits

Use nested for loops to generate a list of all the positive two digit numbers. Display the numbers, and the sums of their digits. Name the program GettingIndividualDigits.c.

```
10, 1+0 = 1
11, 1+1 = 2
12, 1+2 = 3
13, 1+3 = 4
14, 1+4 = 5
15, 1+5 = 6
16, 1+6 = 7

...

97, 9+7 = 16
98, 9+8 = 17
99, 9+9 = 18
```

Task #8: More Number Puzzles

Use nested for loops to generate a list of all the two digit numbers which are less than or equal to fifty-six, and the sum of whose digits is greater than ten. Name the program NumberPuzzles2.c.

Use another set of nested for loops to find a two-digit number such that the number itself minus the number reversed is equal to the sum of its digits. For example, 72 is not such a number because 72-27 (which is 45) is not the same as the sum of its digits (2+7 = 9).

Finally, put the code for each of the two parts into its own separate function, and have a menu in main() which allows you to choose which of the two sets to find. This main program should keep repeating until you choose to quit (use a do-while loop for this).

```
1) Find two digit numbers <= 56 with sums of digits > 10
2) Find two digit number minus number reversed which equals sum of digits
3) Quit

>1

(numbers go here)

1) Find two digit numbers <= 56 with sums of digits > 10
2) Find two digit number minus number reversed which equals sum of digits
3) Quit

>2

(number goes here)

1) Find two digit numbers <= 56 with sums of digits > 10
2) Find two digit number minus number reversed which equals sum of digits
3) Quit

>3
```

Task #9: Number Puzzles III: Armstrong Numbers

Use nested for loops to find all the three-digit Armstrong numbers. Armstrong numbers are three digit numbers such that the sum of the digits cubed is equal to the number itself. Name the program ArmstrongNumbers.c. For example, 153 is an Armstrong number because $1^3 + 5^3 + 3^3 = 153$. However, 294 is not, because $2^3 + 9^3 + 4^3 = 801$ (not 294)

Bonus: Number Puzzles IV: A New Hope

Use nested for loops to find four positive integers whose sum is 45, and such that the first plus 2, the second minus 2, the third multiplied by 2, and the fourth divided by 2 are all equal. Name the program NumberPuzzles4.c.

The following mathematical statements should be true about these numbers:

$$A + 2 = B - 2 = C \times 2 = D/2$$

and

$$A + B + C + D = 45$$

Hand in

Hand in the word file for this lab at the appropriate location on the blackboard system at LMS. You should hand in a single file named Lab_9_<your reg. No. XXX without angle brackets>.doc(x) that contains the following.

1. All completed C source files representing the work accomplished for this lab: NestingLoops.c; OdometerLoops.c; BasicNestedLoops.c; MultiplicationTable.c; NumberPuzzles1.c; GettingIndividualDigits.c.; NumberPuzzles2.c; ArmstrongNumbers.c; and NumberPuzzles4.c.
2. A paragraph at the end that includes a) a brief explanation of the lab, and b) any comments, or suggestions.

To Receive Credit

1. By showing up on time for lab, working on the lab solution, and staying to the end of the class period, only then you can receive full credit for the lab assignment.
2. Comment your program heavily. Intelligent comments and a clean, readable formatting of your code account for 20% of your grade.
3. In-class lab time is not intended as free time for working on your program assignments. Only if you have completely solved the lab assignment, including all challenges, and have had your work checked off for completeness by your TA/Lab Engineer should you begin the program assignment.