

Dot Patterns Laboratory

Overview

In this laboratory, you will draw a series of 16 dot patterns using nested `for` loops. You may think of the dot patterns as *geometric puzzles* to be solved using loop structures. The 16 puzzles are printed immediately after the description of the exercise.

Project Files

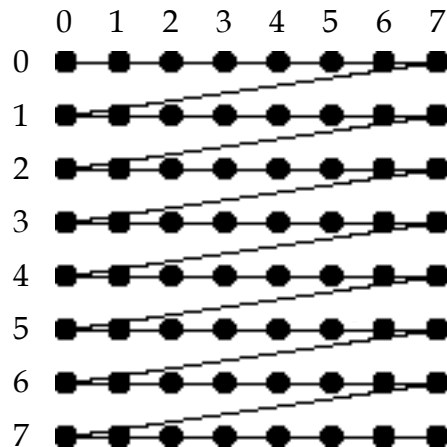
To start on this exercise, copy from the web the self extracting archive `DotPatterns.exe` to a suitable folder on your local machine, double-click this archive, and save the items in the archive to the subfolder suggested (`DotPatternsProject`). If you want a sample solution, then download and expand the archive `DotPatternsX.exe`. This lab document will be available online separately in both Word and PDF formats.

Open the `DotPatternsProject` folder and double click the project file `Shell.dsw`. You will see the main program `Shell.cpp` in the file list and also the header files from the Core Tools library. The source files for the Core Tools are in a subfolder of the project folder but are NOT used explicitly. The reason is that the Core Tools are already compiled and are saved in two files within the Debug subfolder: `winCore.dll` and `winCore.lib`. *Do NOT under any circumstances change the Core Tools files and/or attempt to compile them. The reason is that an entirely different project is needed to build and compile a dynamic-link-library.*

You should only modify the main program source file `Shell.cpp`.

Activities

If you run the sample solution program, `DotPatternsSol.exe`, you will see Pattern 1 drawn in the drawing window. The dot pattern should look like the picture below (except that we have reduced the size and have added row and column labels).



By inspecting procedure `DrawPattern` in the source file `shell.cpp` you will see that the shape of Pattern 1 is controlled by the following *nested loop structure* found as the first block of the `switch` statement in procedure `DrawPattern`:

```
switch (pattern) {
  case 1:
    for (row = 0; row < 8; row++)
      for (col = 0; col < 8; col++)
        BigDot.Draw(row, col);
    break;
```

As this nested loop executes, procedure `BigDot.Draw(row, col)` automatically connects each dot to the next dot that is drawn. You may ask: *Why are most of the connections horizontal lines?* The answer lies in the order in which the loops are nested. The row loop is the outer loop and the column loop is the inner loop. This means that, for each value of the row index, an entire cycle of the column loop must be executed. Therefore, for each row, the inner column loop draws a dot in column 0, then in column 1, and so on through column 7. This explains why the dots in each row are connected by a sequence of horizontal lines. When a given row is finished the last dot drawn is in column 7. The next row begins with a dot in column 0. This explains the diagonal lines from the upper right to the lower left which connect successive rows.

The direct purpose of this lab is to develop your ability to design nested loops. The dot patterns represent typical patterns of traversal through the cells of a 2-dimensional table structure. By doing the exercises you will enhance your ability to translate from a diagram that describes what the nested loops should do into the source code that will do the specified task.

The indirect purpose of the lab is to introduce classes and objects. The procedure `BigDot.Draw(row, col)` arranges the details of drawing the dots and making the lines which connect successive dots. This procedure is in fact an example of an object called `BigDot` using one of its member functions `Draw` to perform an action. You may program this exercise simply by following the pattern shown above but over time we want you to understand precisely how `BigDot.Draw` works.

A printed copy of the startup program `shell.cpp` immediately follows the sheet with the 16 dot patterns. You should look at this program in more detail.

Your task in this exercise is to draw the remaining 15 dot puzzles using appropriate loop structures. Each puzzle requires giving a new twist to the loop structures. For example, in puzzles 3 to 6, you must figure out how to replace the constants 0 and 8 in the inner loop by *expressions* which depend on the row index. In some of the later puzzles, you will need a *sequence* of loop constructs since a single nested loop structure will not be sufficient. In a few cases, you will need an extra loop variable besides `row` and `col`.

You should insert the code for each of the remaining 15 patterns as blocks in the `switch` statement of procedure `DrawPattern`. See the location in the file marked with the comment:

```
// enter the code for the remaining 15 patterns here
```

When your solutions are entered in the `switch` statement, the driver routine `UserLoop` will enable you to interactively test all of the patterns.

Educational Goals:

1. Loop constructs can be confusing when expressed in the symbols of a programming language such as C++. This laboratory should help you learn how to *visualize* what loop constructs are doing and how to *create* loops with specified properties.
2. The regular arrangement of dots in a rectangular grid imitates what people do when information is organized in a table. You may think of each dot as representing a data location and the loops as *visiting all or some of the data locations in a particular order*. The concept of visiting data in a particular order will arise often in computer science.
3. The concept of table is formalized by computer scientists as the *array* and by mathematicians as the *vector* or *matrix*. Hence this exercise on control structures will later be relevant as you learn more about the structures into which data is organized.
4. The source code for the project uses one class, `DrawDots`, one object of this class, `BigDot`, and one featured method, `Draw`. We intend this exercise to permit you to use a class and an object *before* we discuss the details of how to design classes.
5. Finding solutions to the loop puzzles should also improve your problem solving skills. We hope you have fun solving the puzzles.