Integer Plot Function – Documentation

Csc 600-01

Keawa Rozet

Code available at: <https://github.com/krozet/Integer-Plot-Function.git>

Implementation

My method for printing out the integers involves getting the input from the user by use of command line arguments. While my method consists of 4 for loops, it isn’t nearly as criminal as it first appears. Two of imbedded for loops are necessary to iteratate over the 7x7 sized numbers, and the remaining two for loops are used to move from argument to argument, and to move from number to number within each argument.

I utilize a three dimentional array in my solution because I believe it offers the most intuitive means of retrieving all ten 7x7 numbers. On line 133 the heart of the program comes through in a condense but necessary way. In the NUMBERS[][][] array, the first dimention is used to select which 7x7 number to use. The index (argv[arg][k]-'0') converts each char [k] in the current argument [arg] to an int by subracting the ASCII value ‘0’.

Each row is written with all of the columns of each 7x7 number outputted one after another. For this method, the nested for loops were entirely necessary.

I decided to use this method because it was quick and easy to understand. It would have been simple to over complicate it by using a separate class to hold the numbers, or perhaps even by concatenating each row of every number to one another into a new array and then outputting that, but I felt all of that was unnecessary for the problem. This is an inexpensive solution.

Source Code

/\*

Keawa Rozet

CSc 600-01 - Programming Languages

Homework #2

Integer Plot Function

\*/

#include <iostream>

#include <stdio.h>

#include <cstring>

#define NUM\_DIM 7

#define SINGLE\_DIGITS 10

const char NUMBERS[SINGLE\_DIGITS][NUM\_DIM][NUM\_DIM] = {

{

{' ', ' ', '@', '@', '@', '@', ' '},

{' ', '@', '@', ' ', ' ', '@', '@'},

{' ', '@', '@', ' ', ' ', '@', '@'},

{' ', '@', '@', ' ', ' ', '@', '@'},

{' ', '@', '@', ' ', ' ', '@', '@'},

{' ', '@', '@', ' ', ' ', '@', '@'},

{' ', ' ', '@', '@', '@', '@', ' '}

},

{

{' ', ' ', ' ', '@', '@', ' ', ' '},

{' ', ' ', '@', '@', '@', ' ', ' '},

{' ', ' ', ' ', '@', '@', ' ', ' '},

{' ', ' ', ' ', '@', '@', ' ', ' '},

{' ', ' ', ' ', '@', '@', ' ', ' '},

{' ', ' ', ' ', '@', '@', ' ', ' '},

{' ', '@', '@', '@', '@', '@', '@'}

},

{

{' ', ' ', ' ', '@', '@', '@', ' '},

{' ', ' ', '@', '@', ' ', '@', '@'},

{' ', ' ', ' ', ' ', '@', '@', ' '},

{' ', ' ', ' ', '@', '@', ' ', ' '},

{' ', ' ', '@', '@', ' ', ' ', ' '},

{' ', '@', '@', ' ', ' ', ' ', ' '},

{' ', '@', '@', '@', '@', '@', '@'}

},

{

{' ', '@', '@', '@', '@', '@', '@'},

{' ', '@', '@', ' ', ' ', '@', ' '},

{' ', ' ', ' ', ' ', '@', ' ', ' '},

{' ', ' ', ' ', '@', '@', '@', ' '},

{' ', ' ', ' ', ' ', ' ', ' ', '@'},

{' ', '@', '@', ' ', ' ', ' ', '@'},

{' ', '@', '@', '@', '@', '@', ' '}

},

{

{' ', ' ', ' ', '@', '@', '@', ' '},

{' ', ' ', '@', ' ', '@', '@', ' '},

{' ', '@', ' ', ' ', '@', '@', ' '},

{' ', '@', '@', '@', '@', '@', ' '},

{' ', ' ', ' ', ' ', '@', '@', ' '},

{' ', ' ', ' ', ' ', '@', '@', ' '},

{' ', ' ', ' ', ' ', '@', '@', ' '}

},

{

{' ', '@', '@', '@', '@', '@', '@'},

{' ', '@', '@', ' ', ' ', ' ', ' '},

{' ', '@', '@', ' ', ' ', ' ', ' '},

{' ', '@', '@', '@', '@', '@', '@'},

{' ', ' ', ' ', ' ', ' ', '@', '@'},

{' ', ' ', ' ', ' ', ' ', '@', '@'},

{' ', '@', '@', '@', '@', '@', '@'}

},

{

{' ', '@', '@', '@', '@', '@', '@'},

{' ', '@', '@', ' ', ' ', ' ', ' '},

{' ', '@', '@', ' ', ' ', ' ', ' '},

{' ', '@', '@', '@', '@', '@', '@'},

{' ', '@', '@', ' ', ' ', '@', '@'},

{' ', '@', '@', ' ', ' ', '@', '@'},

{' ', '@', '@', '@', '@', '@', '@'}

},

{

{' ', '@', '@', '@', '@', '@', '@'},

{' ', ' ', ' ', ' ', ' ', '@', '@'},

{' ', ' ', ' ', ' ', '@', '@', ' '},

{' ', ' ', ' ', '@', '@', ' ', ' '},

{' ', ' ', '@', '@', ' ', ' ', ' '},

{' ', '@', '@', ' ', ' ', ' ', ' '},

{' ', '@', '@', ' ', ' ', ' ', ' '}

},

{

{' ', '@', '@', '@', '@', '@', '@'},

{' ', '@', '@', ' ', ' ', '@', '@'},

{' ', '@', '@', ' ', ' ', '@', '@'},

{' ', '@', '@', '@', '@', '@', '@'},

{' ', '@', '@', ' ', ' ', '@', '@'},

{' ', '@', '@', ' ', ' ', '@', '@'},

{' ', '@', '@', '@', '@', '@', '@'}

},

{

{' ', '@', '@', '@', '@', '@', '@'},

{' ', '@', '@', ' ', ' ', '@', '@'},

{' ', '@', '@', ' ', ' ', '@', '@'},

{' ', '@', '@', '@', '@', '@', '@'},

{' ', ' ', ' ', ' ', ' ', '@', '@'},

{' ', ' ', ' ', ' ', ' ', '@', '@'},

{' ', '@', '@', '@', '@', '@', '@'}

}

};

int main(int argc, char\*\* argv) {

int size;

//move through each command line argument

for (int arg = 1; arg < argc; arg++) {

//gets the length of each argument

size = std::strlen(argv[arg]);

//positions rows

for (int i = 0; i < NUM\_DIM; i++) {

//checks each char in the argument

for (int k = 0; k < size; k++) {

//positions column for each number

for (int j = 0; j <NUM\_DIM; j++) {

//(argv[arg][k]-'0') converts each argument char to an int

//this int determins which number will be written in the NUMBERS //array

//the entire row is written for each number before moving to the //next row

std::cout << NUMBERS[(argv[arg][k]-'0')][i][j];

}

}

std::cout << std::endl;

}

}

return 0;

}

Example Build and Execution through Terminal

> g++ -o pl int-plot.cpp

> ./pl 1 12 123 1234 12345 123456 1234567 12345678 123456789 1234567890

@@

@@@

@@

@@

@@

@@

@@@@@@

@@ @@@

@@@ @@ @@

@@ @@

@@ @@

@@ @@

@@ @@

@@@@@@ @@@@@@

@@ @@@ @@@@@@

@@@ @@ @@ @@ @

@@ @@ @

@@ @@ @@@

@@ @@ @

@@ @@ @@ @

@@@@@@ @@@@@@ @@@@@

@@ @@@ @@@@@@ @@@

@@@ @@ @@ @@ @ @ @@

@@ @@ @ @ @@

@@ @@ @@@ @@@@@

@@ @@ @ @@

@@ @@ @@ @ @@

@@@@@@ @@@@@@ @@@@@ @@

@@ @@@ @@@@@@ @@@ @@@@@@

@@@ @@ @@ @@ @ @ @@ @@

@@ @@ @ @ @@ @@

@@ @@ @@@ @@@@@ @@@@@@

@@ @@ @ @@ @@

@@ @@ @@ @ @@ @@

@@@@@@ @@@@@@ @@@@@ @@ @@@@@@

@@ @@@ @@@@@@ @@@ @@@@@@ @@@@@@

@@@ @@ @@ @@ @ @ @@ @@ @@

@@ @@ @ @ @@ @@ @@

@@ @@ @@@ @@@@@ @@@@@@ @@@@@@

@@ @@ @ @@ @@ @@ @@

@@ @@ @@ @ @@ @@ @@ @@

@@@@@@ @@@@@@ @@@@@ @@ @@@@@@ @@@@@@

@@ @@@ @@@@@@ @@@ @@@@@@ @@@@@@ @@@@@@

@@@ @@ @@ @@ @ @ @@ @@ @@ @@

@@ @@ @ @ @@ @@ @@ @@

@@ @@ @@@ @@@@@ @@@@@@ @@@@@@ @@

@@ @@ @ @@ @@ @@ @@ @@

@@ @@ @@ @ @@ @@ @@ @@ @@

@@@@@@ @@@@@@ @@@@@ @@ @@@@@@ @@@@@@ @@

@@ @@@ @@@@@@ @@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@

@@@ @@ @@ @@ @ @ @@ @@ @@ @@ @@ @@

@@ @@ @ @ @@ @@ @@ @@ @@ @@

@@ @@ @@@ @@@@@ @@@@@@ @@@@@@ @@ @@@@@@

@@ @@ @ @@ @@ @@ @@ @@ @@ @@

@@ @@ @@ @ @@ @@ @@ @@ @@ @@ @@

@@@@@@ @@@@@@ @@@@@ @@ @@@@@@ @@@@@@ @@ @@@@@@

@@ @@@ @@@@@@ @@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@

@@@ @@ @@ @@ @ @ @@ @@ @@ @@ @@ @@ @@ @@

@@ @@ @ @ @@ @@ @@ @@ @@ @@ @@ @@

@@ @@ @@@ @@@@@ @@@@@@ @@@@@@ @@ @@@@@@ @@@@@@

@@ @@ @ @@ @@ @@ @@ @@ @@ @@ @@

@@ @@ @@ @ @@ @@ @@ @@ @@ @@ @@ @@

@@@@@@ @@@@@@ @@@@@ @@ @@@@@@ @@@@@@ @@ @@@@@@ @@@@@@

@@ @@@ @@@@@@ @@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@@@ @@@@

@@@ @@ @@ @@ @ @ @@ @@ @@ @@ @@ @@ @@ @@ @@ @@

@@ @@ @ @ @@ @@ @@ @@ @@ @@ @@ @@ @@ @@

@@ @@ @@@ @@@@@ @@@@@@ @@@@@@ @@ @@@@@@ @@@@@@ @@ @@

@@ @@ @ @@ @@ @@ @@ @@ @@ @@ @@ @@ @@

@@ @@ @@ @ @@ @@ @@ @@ @@ @@ @@ @@ @@ @@

@@@@@@ @@@@@@ @@@@@ @@ @@@@@@ @@@@@@ @@ @@@@@@ @@@@@@ @@@@