COSC 2P95 – Lab Exercise 2 – Section 01 – Functions and Arrays

Lab Exercises are uploaded via the Sakai page.
Since there's both a source file and four screen shots (or text dumps), package it all into a .zip file before uploading.

Mathematical functions are often continuous, with a literally infinite number of intermediate values between any pair of positions within the domain. Whether to render visually, or analyze its shape, it's sometimes necessary to *discretize* the function.
Discretization is merely the process of substituting discrete values into a function, to take samples at known points along its axes. It converts an infinitely-continuous function into a finite number of values.

Both this lab exercise and next week's will both focus on discretization. We'll start off simple this week, and then get into a more interesting example next week.

All you need to do is to:

- Present the user with a menu:
  - Have them enter 1–4 to select one of the four functions defined below
  - Have them enter 0 to quit
  - The program continues looping until they quit
- The min and max x and y coordinates will be clamped (always from -4..6 for x, and -12..5 for y)
- The user will be asked to select a number of graduations
  - This provides the level of precision
  - The higher the value, the more intermediate values to be calculated between e.g. -4 and 6
- The values will be loaded into an array
  - Yes, this part is mandatory
- The user will then be asked to choose between displaying a 'bitmap' or values
  - Choosing 0 will display the array's values as O for positive values and X for negative values
  - Choosing 1 will display all of the array's values as numbers; one line per row
- It does not matter if your X and/or Y are flipped relative the sample plots linked below
- All menu items must be displayed through standard error (i.e. use cerr instead of cout)
- All displayed array values (whether O/X or values) must go to standard output (use cout)

Let's show the functions by going to Wolfram Alpha!

https://www.wolframalpha.com/input/?i=plot+sin+x+cos+y,+x%3D-4..6,+y%3D-12..5

https://www.wolframalpha.com/input/?i=plot+(sin(x)%2Bcos^2(1%2F2y)-x%2Fy),+x%3D-4..6,+y%3D-12..5

https://www.wolframalpha.com/input/?i=plot+%28%281%2F2*sin%28x%29%29%2B%281%2F2*cos%28y%29%29%29,+x%3D-4..6,+y%3D-12..5

https://www.wolframalpha.com/input/?i=plot+%28%281%2F2*sin%28x%29%29%2B%28x*cos%283y%29%29%29,+x%3D-4..6,+y%3D-12..5

Tips:

- Now that we're transitioning over from C to C++, using variable-sized 2D arrays as parameters is actually more trouble than it's worth. Just use a 1D array of precision×precision elements. The mapping to treat it as a 2D array (e.g. the ability to still access 'rows') is nearly trivial, so long as you remember how large a row is
- The third function, displayed as a 'bitmap', will look like a checkerboard. Try starting there
- Remember that, if you want to have a loop extend over a defined range, you're starting at the smallest possible value, and then extending to that smallest possible value plus ~100% of the difference between the two bounds
  - If rounding errors cut off a single column of values, don't worry about it. So long as you get something that 'looks' like the expected patterns, that's fine
- To get sin and cos, just `#include` the `cmath` header
- Refer to the links for the *actual* formulae

Submission is simple:

- Take either a screenshot or text dump for each of the four functions, using the 'bitmap' output
- Also include your source code
- Package into a .zip file
- Submit via Sakai as usual

Sample execution:

```
$ ./e
Select your function:
1. sin(x)cos(y)
2. sin(x)+cos^2(y/2)-x/y
3. 1/2 sin(x) + 1/2 cos(y)
4. 1/2 sin(x) + xcos(3y)
0. Quit
2
Number of graduations per axis: 40
(0) Bitmap or (1) Values? 0
0000000XXXXXX000000000000000000000000000
000000XXXXXXX000000000000000000000000000
00000XXXXXXXXX00000000000000000000000000
0000XXXXXXXXXXX00000000000000000XXXXXX00
000XXXXXXXXXXXXX000000000000000XXXXXXX00
000XXXXXXXXXXXXX000000000000000XXXXXXXX0
00XXXXXXXXXXXXXX000000000000000XXXXXXXX0
000XXXXXXXXXXXXX000000000000000XXXXXXXX0
000XXXXXXXXXXXXX0000000000000000XXXXXX00
0000XXXXXXXXXXX000000000000000000XXXX000
00000XXXXXXXXX00000000000000000000000000
00000XXXXXXXXX00000000000000000000000000
000000XXXXXXX000000000000000000000000000
0000000XXXXXX000000000000000000000000000
000000XXXXXXX000000000000000000000000000
000000XXXXXXX000000000000000000000000000
00000XXXXXXXXX00000000000000000000000000
000XXXXXXXXXXXX0000000000000000000000000
00XXXXXXXXXXXXX0000000000000000000000000
XXXXXXXXXXXXXXXX000000000000000000000000
XXXXXXXXXXXXXXXX000000000000000000000000
XXXXXXXXXXXXXXXX000000000000000000000000
XXXXXXXXXXXXXXXX000000000000000000000000
XXXXXXXXXXXXXXXX000000000000000000000000
XXXXXXXXXXXXXXX0000000000000000000000000
XXXXXXXXXXXXXXX0000000000000000000000000
XXXXXXXXXXXXXXX0000000000000000000000000
XXXXXXXXXXXXXXX0000000000000000000000000
XXXXXXXXXXXXXXXX000000000000000000000000
000000000000000000XXXXXXXXXXXXXXXXXXXXXX
0000000000000000000000XXXXXXXXXXXXXXXXXX
000000000000000000000000XXXXXXXXXXXXXXXX
0000000000000000000000000XXXXXXXXXXXXXXX
00000000000XXX00000000000XXXXXXXXXXXXXXX
000000000XXXXXXX000000000XXXXXXXXXXXXXXX
00000000XXXXXXXX0000000000XXXXXXXXXXXXXX
0000000XXXXXXXXX0000000000XXXXXXXXXXXXXX
0000000XXXXXXXXX00000000000XXXXXXXXXXXXX
0000000XXXXXXXX000000000000XXXXXXXXXXXXX
00000000XXXXXX00000000000000XXXXXXXXXXXX
Select your function:
1. sin(x)cos(y)
2. sin(x)+cos^2(y/2)-x/y
3. 1/2 sin(x) + 1/2 cos(y)
4. 1/2 sin(x) + xcos(3y)
0. Quit
0
```