Printing Shapes

Topics: nested for-loops, patterns, and shapes
Lab 4: https://maryash.github.io/135/labs/lab_04.html

- Sadab Hafiz

Nested for-loop

In order to print a basic rectangular shape, we can use a nested for-loop. You may have seen this already in cs127. Here's how to make a 5x5 rectangular shape:

```
# include <iostream>

using namespace std;

int main(){

     for(int i = 0; i < 5; i++){                    // for-loop for rows

          for(int j = 0; j < 5; j++){               // for-loop for columns

               cout << "*";                         // print stars (5 times total)

          }

          cout << endl;                             // move to next row after printing 5 stars

     }

     return 0;

}
```

Nested for-loop (continued)

The outer loop variable(i) corresponds to the current row and the inner loop variable(j) corresponds to the current column. If you are trying to create a particular shape, look for patterns in the numbers (i,j). For a basic rectangular shape, we print a star in every combination of i and j:

		for (int j = 0; j < 5; j++)					
	i , j	j = 0	j = 1	j = 2	j = 3	j = 4	Output:
for (int i = 0; i < 5; i++)	i = 0	0,0	0,1	0,2	0,3	0,4	*****
	i = 1	1,0	1,1	1,2	1,3	1,4	*****
	i = 2	2,0	2,1	2,2	2,3	2,4	*****
	i = 3	3,0	3,1	3,2	3,3	3,4	*****
	i = 4	4,0	4,1	4,1	4,3	4,4	*****

Checkerboard Shape

Let's try to create a checkerboard shape within a 5x5 square. Based on the table below, observe the relationship between the i and j values. When do we print a star and when do we print a space?

		for (int j = 0; j < 5; j++)					
	i , j	j = 0	j = 1	j = 2	j = 3	j = 4	Output:
for (int i = 0; i < 5; i++)	i = 0	0,0	0,1	0,2	0,3	0,4	* * *
	i = 1	1,0	1,1	1,2	1,3	1,4	* *
	i = 2	2,0	2,1	2,2	2,3	2,4	* * *
	i = 3	3,0	3,1	3,2	3,3	3,4	* *
	i = 4	4,0	4,1	4,1	4,3	4,4	* * *

Checkerboard Shape (continued)

Stars are printed for the following combinations of (i,j):

0,0	0,2	0,4	1,1	1,3	2,0	2,2	2,4	3,1	3,3	4,0	...

Spaces are printed for the following combinations of (i,j):

0,1	0,3	1,0	1,2	1,4	2,1	2,3	3,0	3,2	3,4	4,1	...

Pattern: Stars are only printed when both i and j are either even or odd. Spaces are printed when one is even and the other is odd.

This is where the modulo(%) operator comes in handy. If both i and j are even, i%2 and j%2 will be 0. If both i and j are odd, i%2 and j%2 will be 1.

In other words, print a star(“*”) if (i%2 == j%2). For anything else, print a space (“ ”).

Checkerboard Shape (another approach)

What if we add i and j?

Stars are printed for the following combinations of (i,j):

i,j	0,0	0,2	0,4	1,1	1,3	2,0	2,2	2,4	3,1	3,3	4,0	...
i+j	0	2	4	2	4	2	4	6	4	6	4	...

Spaces are printed for the following combinations of (i,j):

i,j	0,1	0,3	1,0	1,2	1,4	2,1	2,3	3,0	3,2	3,4	4,1	...
i+j	1	3	1	3	5	3	5	3	5	7	5	...

Do you see the pattern?

Cross Shape

Similar to the checkerboard problem, can you see a pattern for the cross shape? Observe the i and j when the stars are being printed.

		for (int j = 0; j < 5; j++)					
	i , j	j = 0	j = 1	j = 2	j = 3	j = 4	Output:
for (int i = 0; i < 5; i++)	i = 0	0,0	0,1	0,2	0,3	0,4	* *
	i = 1	1,0	1,1	1,2	1,3	1,4	* *
	i = 2	2,0	2,1	2,2	2,3	2,4	*
	i = 3	3,0	3,1	3,2	3,3	3,4	* *
	i = 4	4,0	4,1	4,1	4,3	4,4	* *

Cross Shape (continued)

Stars are printed for the following combinations of (i,j):

0,0	0,4	1,1	1,3	2,2	3,1	3,3	4,0	4,4	...

Pattern 1: First pattern is easy to spot. Observe combinations such as (0,0), (1,1), (2,2) etc. A star("*") is printed when i is equal to j.

Pattern 2: Similar to checkerboard, let's add the remaining pairs:

i,j	0,4	1,3	2,2	3,1	4,0
i+j	0+4 = 4	1+3 = 4	2+2 = 4	3+1 = 4	4+0 = 4

The column size is 5 which makes the last index of columns equal to 4.

Thus, combining the patterns, we print a star("*") `if (i==j || (i+j) == col_size-1)`

Cross Shape (another approach)

Stars are printed for the following combinations of (i,j):

0,0	0,4	1,1	1,3	2,2	3,1	3,3	4,0	4,4	...

Pattern 2: Observe these combinations: (0,4) , (1,3) , (2,2) , (3,1) , (4,0). We know that the outer for loop increments i by one which is why i goes from 0 to 4 naturally. However, the j in this pattern seems to be decreasing by 1 during each iteration. We also know that the column size is 5. The last index of j is one less than the column size. By subtracting each i from column size minus one, we can get the new j:

col_size - 1 - i = j	5 - 1 - 0 = 4	5 - 1 - 1 = 3	5 - 1 - 2 = 2	5 - 1 - 3 = 1	5 - 1 - 4 = 0

Thus, combining the patterns, we print a star(“*”) if (i==j || (col_size - 1 - i) == j)

Upside-down trapezoid(ish) shape

The following is a 4x7 upside-down trapezoid(?) shape:

		for (int j = 0; j < stars; j++)							
	i , j	j = 0	j = 1	j = 2	j = 3	j = 4	j = 5	j = 6	Output:
for (int i = 0; i < 4; i++)	i = 0	0,0	0,1	0,2	0,3	0,4	0,5	0,6	*******
	i = 1	1,0	1,1	1,2	1,3	1,4	1,5	1,6	*****
	i = 2	2,0	2,1	2,2	2,3	2,4	2,5	2,6	***
	i = 3	3,0	3,1	3,2	3,3	3,4	3,5	3,6	*

Upside-down trapezoid (impossible shape)

Certain shapes are impossible to make. For example (5x7):

		for (int j = 0; j < stars; j++)							
	i , j	j = 0	j = 1	j = 2	j = 3	j = 4	j = 5	j = 6	Output:
for (int i = 0; i < 4; i++)	i = 0	0,0	0,1	0,2	0,3	0,4	0,5	0,6	*******
	i = 1	1,0	1,1	1,2	1,3	1,4	1,5	1,6	*****
	i = 2	2,0	2,1	2,2	2,3	2,4	2,5	2,6	***
	i = 3	3,0	3,1	3,2	3,3	3,4	3,5	3,6	*
	i = 4	4,0	4,1	4,2	4,3	4,4	4,5	4,6	???

Upside-down trapezoid shape (continued)

In each row, we have to print some spaces followed by some stars. For the 4x7 shape, notice how the number of stars and spaces change on each row:

row (i)	# of stars	# of spaces before/after
0	7	0
1	5	1
2	3	2
3	1	3

Thus, we can have two for-loops for each row. One to print spaces and another to print stars. The number of spaces and stars can be tracked using int variables.

Upside-down trapezoid shape (continued)

Using two variables and for-loops, you can do the following (4x7):

```
int stars = 7;

int spaces = 0;

for(int i = 0; i < 4; i++){                    // for-loop for rows

      … for-loop to print spaces amount of “ ”

      … for-loop to print stars amount of “*”

      cout << endl;                            // move to next row

      stars -= 2;                              // print two less stars in next row

      spaces ++;                               // print one more space in next row

}
```