Triangle Pattern

Exercises

1. Triangle Printing Program:

You can only use

- cout << endl;
- cout << "*";
- cout << " ";

Enter the size of the pattern, where size is the maximum number of asterisks in a line. For example, size of 5 looks like

Pattern of asterisks

Print out the following pattern with size 5, where size is the maximum of asterisks in a line.

```
*****
****
***
**
*
```

Pattern of Asterisk

another triangle pattern of size 5

```
*****
 ****
  ***
   **
    *
```

Pattern of asterisks

```
*****
****
***
**
*
```

(1) What different from one line to the next?

Print 5 asterisks in the first line.

Print 4 asterisks in the second line.

Print 3 asterisks in the third line.

Print 2 asterisks in the fourth line.

Print 1 asterisk in the fifth line.

(2) When do we stop?

Pattern of asterisks

```
*****
****
***
**
*
```

(1) What different from one line to the next?

Print 5 asterisks in the first line.

Print 4 asterisks in the second line.

Print 3 asterisks in the third line.

Print 2 asterisks in the fourth line.

Print 1 asterisk in the fifth line.

Q: What variable(s) are used to trace the above change?

Initialize related variable

```
*****
****
***
**
*
```

```
//initialize the variable.
int num_asterisks = 5;
```

What to do in each row?

Cursor is here

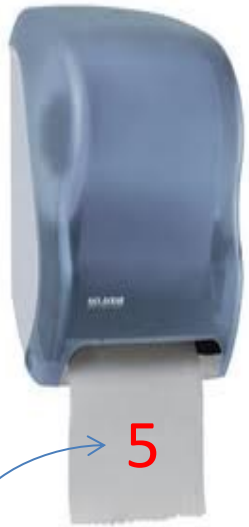
*****

****

***

**

*



int num_asterisks = 5;

print num_asterisks *s; //use num_asterisks

print a new line;

How to prepare to move to each row?

Cursor is here

*****

****

***

**

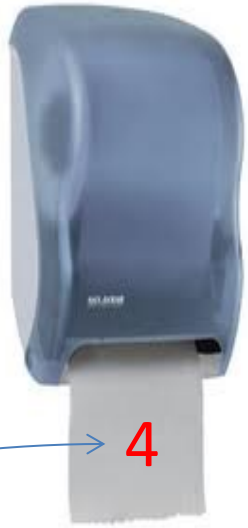
*

int num_asterisks = 5;

Print num_asterisks *s;

Print a new line;

num_asterisks--; //# of *'s to print in coming row



A structure w/o repetition statement

Cursor is here

*****

****

***

**

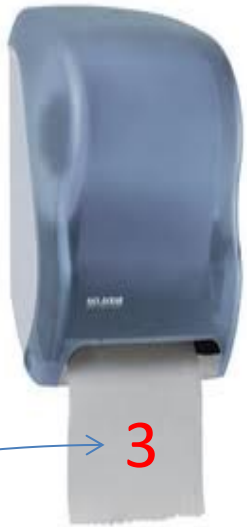
*

int num_asterisks = 5;

Print num_asterisks *s;
Print a new line.
num_asterisks--;

Print num_asterisks *s;
Print a new line;
num_asterisks--;

....



Rewrite in repetition statement

```
*****
****
***
**
*
```

```
int num_asterisks = 5;

while ( num_asterisks > 0 ) {
        print num_asterisks *s;
        print a new line;
        num_asterisks--;
}
```

How to print num_asterisks asterisk

```
*****
****
***
**
*
```

```
int num_asterisks = 5;

while ( num_asterisks > 0 ) {
        Print num_asterisks *s;
        print a new line.
        num_asterisks--;
}
```

How to print num_asterisks asterisk

~~Print num_asterisks asterisks;~~

Say it in Java!

Related: print 5 asterisks using a repetition statement.

```
for (int i = 0; i < 5; i++)
    cout << "*";
```

How to print num_asterisks asterisk

Print num_asterisks asterisk;

Related: print 6 asterisks using a repetition statement.

```
for (int i = 0; i < 6; i++)
    cout << "*";
```

How to print num_asterisks asterisk

print num_asterisks *;

```
for (int i = 0; i < num_asterisks; i++)
    cout << "*";
```

Pattern of asterisks

```
*****
****
***
**
*
```

```
for (int i = 0; i < num_asterisks; i++)
    cout << "*";
```

```
int num_asterisks = 5;

while ( num_asterisks > 0) {
        Print num_asterisks asterisk;
        print a new line;
        num_asterisks--;
}
```

Pattern of asterisks

```
*****
****
***
**
*
```

```
for (int i = 0; i < num_asterisks; i++)
    cout << "*";
```

int num_asterisks = 5;

while (num_asterisks > 0) {

~~Print num_asterisks asterisk;~~

~~print a new line.~~

```
cout << endl;
```

num_asterisks--;

}

Generalize to any size

```
*****
****
***
**
*
```

Enter size from console.

```
for (int i = 0; i < num_asterisks; i+
    cout << "*";
```

size

int num_asterisks = ~~5~~;

while (num_asterisks > 0) {

~~Print num_asterisks *;~~

~~print a new line;~~

cout << endl;

num_asterisks--;

}

Warning: size ≠ num_asterisks

- size is the maximum number of asterisks in all rows. It will not change from row to row.
- num_asterisks is the number of asterisks in each row. It will change from row to row.
- We may use size to initialize num_asterisks or in condition deciding whether we finish drawing the pattern.

```
*****
****
***
**
*
```

Triangle of asterisks: nested-loop

```
int num_asterisks = 5;
while (num_asterisks > 0) {
        //print num_asterisks *s

        for (int i = 0; i < num_asterisks; i++)
            cout << "*";

        //print a new line
        cout << endl;

        num_asterisks--;
}
```

Code to print triangle pattern

```
cout << "enter size: ";
int size;
cin >> size;

int num_asterisks = size;
while (num_asterisks > 0)
{
    //print * for num_asterisks times
    for (int i = 0; i < num_asterisks; i++)
        cout << "*";

    cout << endl; //print a new line
    num_asterisks--;
}
```

```
***
**
*
```

Code to print triangle pattern (simplified)

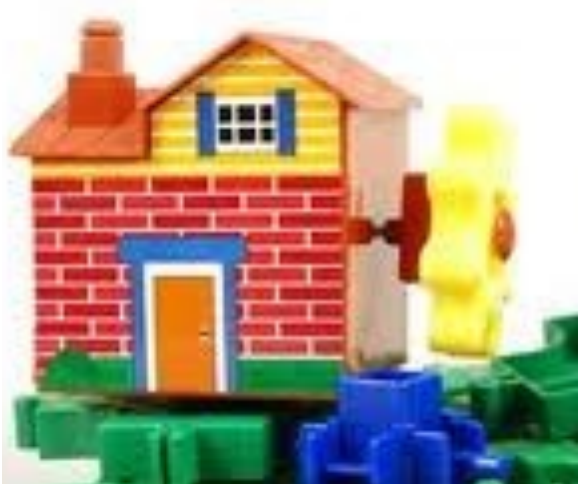
```
***
**
*
```

```
cout << "enter size: ";
int size;
cin >> size;

//simplified version: use two for-statements
for (int num_asterisks = size;
     num_asterisks > 0; num_asterisks--)
{
    //print * for num_asterisks times
    for (int i = 0; i < num_asterisks; i++)
        cout << "*";

    cout << endl;
}
```

Building block: do something for x times

Print out * for num_asterisks times

```
for (int i = 0; i < num_asterisks; i++)
    cout << "*";
```



do something for x times

```
for (int i = 0; i < x; i++)
    do something
```

Method to work pattern of asterisks

(1) What **changes** happen from one row to the next?

(2) What **variables** to describe those changes?

(3) What are **initial values** of those variables?

(4) What do we do in each row, especially, how to **use** those variables in each row?

(5) How to **update** those variables to prepare for the next row?

(6) When shall we **stop**?

What changes happen from one row to the next?

```
  *
 **
***
****
*****
```

Number of preceding spaces (spaces before the first *) is different from row to row.

Use variables to describe them.

Print out **some** preceding spaces;

Print out **several** asterisks;

Print out one new line.

What variables to describe those changes?

```
    *
   **
  ***
 ****
*****
```

num_prec_spaces: number of preceding spaces

num_asterisks: number of asterisks

What are the initial values of the variables?

```
   *
  **
 ***
****
*****
```

num_prec_spaces = 4;

num_asterisks = 1;

How do the variables changes?

```
    *
   **
  ***
 ****
*****
```

num_prec_spaces	num_asterisks
4	1
3	2
2	3
1	4
0	5

What do we do in each row?

```
    *
   **
  ***
 ****
*****
```

What do we do each row?

Print num_prec_spaces spaces;
Print num_asterisks *;
Print a new line.

Num_prec_spaces	num_asterisks
4	1
3	2
2	3
1	4
0	5

Update variables for next row

```
    *
   **
  ***
 ****
*****
```

num_prec_spaces	num_asterisks
4	1
3	2
2	3
1	4
0	5

```
num_prec_paces--;
num_asterisks++;
```

When to stop?

```
    *
   **
  ***
 ****
*****
```

num_prec_spaces >= 0

num_prec_spaces	num_asterisks
4	1
3	2
2	3
1	4
0	5

- Use **either** one of these two conditions.
- Variables num_prec_spaces and num_asterisks are not independent: one can be calculated from the other.

num_asterisks <= 5

Pattern of asterisks

Print out the following pattern with a given size, where size is the maximum of asterisks in a line.

The following is of size of 5.

```
    *
   **
  ***
 ****
*****
```

Pattern of asterisks

```
   *
  **
 ***
****
*****
```

```
//initialization
int num_prec_spaces = 4;
int num_asterisks = 1;
while (num_prec_spaces >= 0) {
        //Use variables for current row.
        print num_prec_spaces spaces;
        print num_asterisks *s;
        print out a new line;
        //prepare for the new row
        num_prec_spaces--;
        num_asterisks++;
}
```

As long as something needs to be done more than one time, use while-statement.

Pattern of asterisks

```
   *
  **
 ***
****
*****
```

```
//initialization
int num_prec_spaces = 4;
int num_asterisks = 1;
while ( num_prec_spaces >= 0 ) {
        //Use variables for current row.
        print num_prec_spaces spaces;
        print num_asterisks *s;
        print out a new line;
        //prepare for the new row
        numPrecSpaces--;
        numAsterisks++;
}
```

```
for (int i = 0; i < num_prec_spaces; i++)
    cout << " ";
```

```
for (int i = 0; i <
num_asterisks; i++)
    cout << "*";
```

Pattern of asterisks

```
   *
  **
 ***
****
*****
```

```
//initialization
int num_prec_spaces = 4;
int num_asterisks = 1;
while (num_prec_spaces >= 0 ) {
        //Use variables for current row.
        print num_prec_spaces spaces;
        print num_asterisks *s;
        cout << endl;
        //prepare for the new row
        num_prec_spaces--;
        num_asterisks++;
}
```

```
for (int i = 0; i < num_prec_spaces; i++)
    cout << " ";
```

```
for (int i = 0; i <
num_asterisks; i++)
    cout << "*";
```

Pattern of asterisks: generalize to any size

```
    *
   **
  ***
 ****
*****
```

size-1

```
for (int i = 0; i < num_prec_spaces; i++)
    cout << " ";
```

Input size from console.

```
//initialization
int num_prec_spaces = 4;
int num_asterisks = 1;
while (num_prec_spaces >= 0 ) {
        //Use variables for current row.
        …
        cout << endl;
        //prepare for the new row
        num_prec_spaces--;
        num_asterisks++;
}
```

```
for (int i = 0; i <
num_asterisks; i++)
    cout << "*";
```

Complete code

```
cout << "enter size: ";
int size;
cin >> size;

int num_prec_spaces = size -1;
int num_asterisks = 1;
while (num_asterisks <= size)
{
    //print spaces for num_prec_spaces times
    for (int i = 0; i < num_prec_spaces; i++)
        cout << " ";

    //print * for num_asterisks times
    for (int i = 0; i < num_asterisks; i++)
        cout << "*";

    cout << endl; //print out a new line
    //prepare for the next line
    num_prec_spaces--;
    num_asterisks++;
}
```

```
 *
**
***
```

Complete code (simplified)

```
cout << "enter size: ";
int size;
cin >> size;

int num_asterisks = 1;
while (num_asterisks <= size)
{
    //print spaces for (size - num_asterisks) times
    for (int i = 0; i < (size - num_asterisks); i++)
        cout << " ";

    //print * for num_asterisks times
    for (int i = 0; i < num_asterisks; i++)
        cout << "*";

    cout << endl; //print out a new line
    //prepare for the next line
    num_asterisks++;
}
```

```
  *
 **
***
```

- num_asterisks and num_prec_spaces are not independent.
- num_prec_spaces can be calcuted by size – num_asterisks

Complete code: use two for-statements

```
cout << "enter size: ";
int size;
cin >> size;

for (int num_asterisks = 1;
     num_asterisks <= size; num_asterisks++)
{
    //print spaces for (size - num_asterisks) times
    for (int i = 0; i < (size - num_asterisks); i++)
        cout << " ";

    //print * for num_asterisks times
    for (int i = 0; i < num_asterisks; i++)
        cout << "*";

    cout << endl; //print out a new line
}
```

```
  *
 **
***
```