

Classes: A First Look

```
#include <iostream.h>
```

```
#define SIZE 10
```

```
// Declare a stack class for characters
```

```
class stack {
```

```
    char stck[SIZE]; // holds the stack
```

```
    int tos;          // index of top-of-stack
```

```
public:
```

```
    void init();          // initialize stack
```

```
    void push(char ch); // push character on stack
```

```
    char pop();          // pop character from stack
```

```
}
```

// Initialize the stack

```
void stack::init() { tos = 0; }
```

// Push a character.

```
void stack::push(char ch) {  
    if (tos==SIZE) { cout << "Stack if full"; return; }  
    stck[tos] = ch;  
    tos++; }
```

// Pop a character

```
char stack::pop() {  
    if (tos==0) { cout << "Stack is empty";  
                return 0; // return null on empty stack  
            }  
    tos--; return stck[tos]; }
```

```
main() {  
    stack s1, s2; // create two stacks  
    int i;  
    // initialize the stacks  
    s1.init();  
    s2.init();  
  
    s1.push('a');      s2.push('x');  
    s1.push('b');      s2.push('y');  
    s1.push('c');      s2.push('z');  
  
    for (i=0; i<3; i++) cout << "Pop s1: " << s1.pop() << "\n";  
    for (i=0; i<3; i++) cout << "Pop s2: " << s2.pop() << "\n";  
  
    return 0;  
}
```

This page intentionally left blank



HW #3 (Equilateral Triangle)

- Upon completing this assignment, you should be able to implement a simple class, as well as gain a better understanding of the building and use of classes and objects.
- An equilateral triangle等邊三角形 is a triangle whose sides are equal.
- You are to write a class called **Triangle**, using **C++** (including filenames **triangle.h** and **triangle.cpp**), or **Java** or **Python** if you prefer, that will allow the creation and handling of equilateral triangles, whose sides are integers in the ranger 1-39.

HW #3 (2)

- The single constructor for the Triangle class should have 3 parameters: an integer size (required), which is the length of a side; a border character (optional, with a default of '#'); and a fill character (optional, with a default of '*'). If the size provided is less than 1, set the size to 1. If the size provided is greater than 39, set the size to 39. The class will need to provide internal storage for any member data must be kept track of.

HW #3 (3)

- There should be member functions **GetSize**, **Perimeter**, and **Area**, which will return the size of a side, the perimeter of the Triangle, and the area of the Triangle, respectively. The first 2 should return integer results. The **Area** function should return its result as a double.
- There should be member functions **Grow** and **Shrink**, which will increase or decrease (respectively) the size of the Triangle's sides by 1, unless this would cause the size to go out of bounds (out of the 1-39 range); in the latter case, **Grow** and **Shrink** should make no change to the size.

HW #3 (4)

- There should be member functions **SetBorder** and **SetFill**, which each allows a new border or fill character (respectively) to be passed in as a parameter. The characters that should be allowed for the border or fill characters are characters from the '!' (ascii 33) up through the '~' (ascii 126). If an attempt is made to set the border or fill characters to anything outside the allowable range, the function should set the border or fill back to its original default (the ones listed for the constructor – the border default is '#' and the fill default is '*').

HW #3 (5)

- There should be a member function called **Draw** that will display a picture of the Triangle on the screen. You may assume that the cursor is already at the beginning of a line when the function begins, and you should make sure that you leave the cursor on the line following the picture afterwards (i.e., print a new line after the last line of the Triangle). Use the border character to draw the border of the Triangle, and use the fill character to draw the internal characters.

HW #3 (6)

- Separate the characters on a line in the picture by a single space to make the Triangle look more proportional (to approximate the look of an equilateral triangle). You may not use formatting function like **setw** to draw the Triangle. This must be handled with loops. (You will only print out the newline, spaces, the border character, and maybe the fill character on any given line.)
- Provide a member function called **Summary** that displays all information about a Triangle: its size, perimeter, area, and a picture of what it looks like. When displaying the area (decimal data), always show exactly 2 decimal places. Your output should be in the exact same format as sample run below.

HW #3 (7)

- Following is a sample driver program that uses objects of type Triangle and illustrates the usage of the member functions.

```
// driver.cpp -- driver program to demonstrate the behavior of  
// the Triangle class
```

```
#include <iostream>  
#include <iomanip>  
#include "triangle.h"
```

```
using namespace std;
```

```
int main()
{
    // create some Triangles
    Triangle t1( -5 ), t2( 7, '^' ), t3( 12, 'X', 'O' ), t4( 50 , '$' , 'o');
    // display original Triangles

    cout << "t1 has size = " << t1.GetSize() << " units.\n";
    t1.Draw();
    cout << "\nt2 has size = " << t2.GetSize() << " units.\n";
    t2.Draw();
    cout << "\nt3 has size = " << t3.GetSize() << " units.\n";
    t3.Draw();
    cout << "\nt4 has size = " << t4.GetSize() << " units.\n";
    t4.Draw();
    cout << '\n';
}
```

```
t1.Shrink(); // demonstrate shrink
t2.Shrink();
t3.Grow(); // and grow
t4.Grow();
cout << "t1 now has size = " << t1.GetSize() << " units.\n";
cout << "t2 now has size = " << t2.GetSize() << " units.\n";
cout << "t3 now has size = " << t3.GetSize() << " units.\n";
cout << "t4 now has size = " << t4.GetSize() << " units.\n";
```

```
// demonstrate perimeter
cout << "t2 has perimeter = " << t2.Perimeter() << " units.\n";
cout << "t3 has perimeter = " << t3.Perimeter() << " units.\n";
// and area
cout << "t2 has area = " << t2.Area() << " square units.\n\n";
cout << "t3 has area = " << t3.Area() << " square units.\n\n";
```

```
t1.Draw();  
t1.Grow();           // show that fill character  
cout << "t1 grows:\n"; // appears only when size  
t1.Draw();           // is at least 3  
t1.Grow();  
cout << "... and grows:\n";  
t1.Draw();  
cout << '\n';
```

```
t1 = t2; // demonstrate the default overload of the  
// assignment operator  
cout << "t1 now has size = " << t1.GetSize() << " units.\n";  
t1.Draw();
```

```
// demonstrate the changing of border and fill characters
t2.SetBorder('@');
t2.SetFill('-');
cout << "t2 now looks like:\n";
t2.Draw();
cout << '\n';
t2.SetBorder('\n'); // illegal border
t2.SetFill('\a');    // illegal fill
cout << "t2 now looks like:\n";
t2.Draw();
cout << '\n';
cout << "\nHere is a summary on t3:\n"; // demonstrate summary
t3.Summary();

return 0;
}
```

HW #3 (8)

- Use the **main function** to test your class. Your class **must** work with this code, **without alteration**. Your class will be tested with a large set of calls than this driver program represents.
- We have also provided the output from the sample execution of the driver program.
- You are encouraged to write your own driver routines to further test the functionality of your class, as well.
- Most questions about the required behavior of the class can be determined by carefully examining the driver program and the sample execution.


```
t1 now has size = 1 units.
```

```
t2 now has size = 6 units.  
t3 now has size = 13 units.  
t4 now has size = 39 units.  
t2 has perimeter = 18 units.  
t3 has perimeter = 39 units.  
t2 has area = 15.5885 square units.
```

```
t3 has area = 73.1791 square units.
```

```
#  
t1 grows:  
#  
# #  
... and grows:  
#  
# #  
# # #
```

```
t1 now has size = 6 units.
```

```
  ^  
  ^ ^  
 ^ * ^  
^ * * ^  
^ * * * ^  
^ ^ ^ ^ ^ ^
```

```
t2 now looks like:
```

```
  @  
  @ @  
 @ - @  
@ - - @  
@ - - - @  
@ @ @ @ @ @
```

```
t2 now looks like:
```

```
  #  
  # #  
 # * #  
# * * #  
# * * * #  
# # # # # #
```

```
Here is a summary on t3:  
Size of triangle's side = 13 units.  
Perimeter of triangle = 39 units.  
Area of triangle = 73.18 units.  
Triangle looks like:
```

```
      X  
      X X  
    X O X  
  X O O X  
X O O O X  
  X O O O O X  
X O O O O O X  
  X O O O O O O X  
X O O O O O O O X  
  X O O O O O O O O X  
X O O O O O O O O O X  
  X O O O O O O O O O O X  
X X X X X X X X X X X X X
```