

ASCII Drawing Using Loops and Functions

In this project, you will implement a program that interprets a simple drawing language to create ASCII art. Before fancy computer displays existed, enterprising computer users generated art by printing sequences of ASCII characters to a line printer. The art form continues today, and many examples can be found at the following URL:

http://dir.yahoo.com/Arts/Visual_Arts/Computer_Generated/ASCII_Art/

The goal of this assignment is to become familiar with functions with C++. You should practice top down design skills, including the creation of a structure chart.

The input file:

The input file for this program is named "Commands.txt". The contents and interpretation are the same as program 6. A sample input file follows.

```
; CS 1044 Spring 2001
; Project 7 Test File 1
; Basic Cat
space 1
print 1 /
print 1 \
print 1 _
print 1 /
print 1 \
newline
print 1 (
space 1
print 1 o
print 1 .
print 1 o
space 1
print 1 )
newline
space 1
print 1 >
space 1
print 1 ^
space 1
print 1 <
newline
quit
```

The output file:

The output file is named "Drawing.txt". An output file, which corresponds to the given input file, is shown below. The format and contents are the same as project 6 with the exception of the project number in the header.

```
Programmer: Craig Struble
CS 1044 Project 7 Spring 2001

/\_/\
( o.o )
> ^ <
```

Modular Design:

You must use modular design for this assignment and incorporate C++ functions into your implementation. For each of the `print`, `space`, and `newline` commands, you must use a function to implement their behavior. You must also include a function that reads a command from the input file and returns an integer value representing the command that was read. The integer values must be identified with named constants. Consider using a `switch` statement to select which function to execute inside of your `while` loop body.

Documentation and other requirements:

You must meet the following requirements (in addition to designing and implementing a program that merely produces correct output):

- Your program, at a minimum, must contain the 4 functions described in **Modular Design**, in addition to `main()`.
- You must make appropriate use of C++ parameter-passing mechanisms.
- One of your functions must be a value-returning function.
- You must include a top-down design of your program in comments following your pledge statement. See Appendices 3 and 4 in the course notepack for design examples.
- Document each function with a function documentation header as described in the notes
- Define your functions after your main program
- Use function prototypes appropriately
- Write a header comment with your identification information, the required pledge statement (below), and a brief description of what the program does.
- Write a comment explaining the purpose of every variable and named constant you use.
- Write comments describing what most of the statements in your program do.
- Use descriptive identifiers for variables and for constants.
- Use named constants instead of “magic values” whenever it is appropriate.
- Use both a `while` loop and a `for` loop (appropriately) in your implementation.
- Do **NOT** use `break`, `continue`, `return`, or `goto` statements to exit a loop prematurely.
- Do **NOT** use arrays.
- Do **NOT** use global variables. Global constants are OK.

Submitting your program:

You will submit this assignment to the Curator System (read the *Student Guide*), and it will be graded automatically. Instructions for submitting, and a description of how the grading is done, are contained in the *Student Guide*.

You will be allowed up to five submissions for this assignment. Use them wisely. Test your program thoroughly before submitting it. Make sure that your program produces correct results for every sample input file posted on the course website. If you do not get a perfect score, analyze the problem carefully and test your fix with the input file returned as part of the Curator e-mail message, before submitting again. The highest score you achieve will be counted.

The *Student Guide* can be found at: <http://ei.cs.vt.edu/~eags/Curator.html>

The submission client can be found at: <http://spasm.cs.vt.edu:8080/curator/>

Evaluation:

Your submitted program will be assigned a score based upon the runtime testing performed by the Curator System. We may very well evaluate your submission of this program for documentation style, and to see whether you followed the requirements given in this specification. Therefore, you should compare your comments to those given in the programs for projects 1 and 2, as well as consider the scoring of your fourth project. The programs serve as useful a guide to acceptable documentation style at this point in the course.

If your program is evaluated for documentation and requirements, your instructor will specify how that score will be counted.

Pledge:

Each of your program submissions must be pledged to conform to the Honor Code requirements for this course. Specifically, you **must** include the following pledge statement in the header comment for your program:

```
// On my honor:  
//  
// - I have not discussed the C++ language code in my program with  
//   anyone other than my instructor or the teaching assistants  
//   assigned to this course.  
//  
// - I have not used C++ language code obtained from another student,  
//   or any other unauthorized source, either modified or unmodified.  
//  
// - If any C++ language code or documentation used in my program  
//   was obtained from another source, such as a text book or course  
//   notes, that has been clearly noted with a proper citation in  
//   the comments of my program.  
//  
// - I have not designed this program in such a way as to defeat or  
//   interfere with the normal operation of the Curator System.  
//  
// <Student Name>
```

Failure to include this pledge in a submission is a violation of the Honor Code.