COMP2001/COMP2401

Introduction to System Programming

Summer 2012, Assignment 3
Deadline : June14, 2012, at 19:00 PM, submitted via WebCT

Objectives:

You will write a program in C under Linux to create a system similar to Assignment 2 for generating text-art works **more efficiently**. The goal is to be familiar with pointers, file input/output, and program organization. By completing this assignment, you will be able to

- manipulate pointers;
- use linked lists as data structures;
- read and write files;
- use command line arguments;
- organize your program.

You **MUST** submit your source code (should be multiple .c files and .h files) and any other files (readme.txt and Makefile) required to compile and run your program through WebCT.

All submitted files should be packaged as **one single .tar or .zip file**. TAs should be able to compile and run your program under our SCS Linux machines. [Total 15 marks]

Description:

You are going to create a new interactive system to display text-based art works by placing character-based shapes at specified locations, which is the same as assignment 2. BUT there are some constraints to transform the old codes into a more efficient program.

(1) The array of 'allShapes' used in main function is replaced by a linked list. You should implement a data structure for your linked list to store all of shapes. It might mean you should modify according functions for printMap or other related functions because of using linked lists.

(2) The initial 6 different shapes should be loaded by reading a text file called 'init.txt'.

(3) 'init.txt' is loaded by using command line arguments when you run your program. For example, if your executable file is called TESTART, you should use 'TESTART init.txt' to load your initial shapes.

(4) The new interactive system has five options in the menu and allows users to select [0 - 4] different options:

- select 0: print out the space map displaying all those shapes,
  similarly as found in the sample output shown in assignment 2;
- select 1: add a shape;
- select 2: delete a shape;
- select 3: save all shapes into a text called 'shapes.txt';

- select 4: exit system.

(5) 'shapes.txt' is formatted as the 'init.txt' file.

Requirements:

1. You must define a data structure for your linked list to store all of your shapes.
2. There are **four types** of shapes constructed with capital letters ('A', 'C, 'D', and 'G') in this program: ALINE, a vertical line of three 'A' characters; CCROSS, a cross with two perpendicular lines intersected with each other at the middle point (there are 7 'C' characters on each line); DSQUARE, a square of 2 by 2 'D' characters; and GRECTANGLE, a rectangle of 8 by 2 'G' characters. It is the same as assignment 2.
3. You must use enumerated data type called **PatternType** to name the different shape types, which is the same as assignment 2.
4. You have to organize your source codes into **multiple** .c files or .h files to make your program clearly read.
5. You are **not allowed** to use global variables in this assignment.
6. You should provide **Makefile** and **readme.txt** file to guide TAs to compile and run your program.
7. No memory leak.

Marking scheme:

1. **Submission**
   - You must follow all the instructions **exactly**, or you will lose marks
2. **Deductions**
   - 2 marks if the assignment is marked **Late** in WebCT (submitted between 19:00 and 19:30 PM)
   - 15 marks if the assignment is marked **Missed** in WebCT (submitted after 19:30 PM)
   - 3 marks if the code does not compile, if any submitted files are missing or corrupt or in the wrong format, or if the program consistently crashes
   - 1 mark for missing comments or other bad style (non-standard indentation, improper funct/var names, etc)
   - 2 mark if using global variables
   - 4 marks if not loading the initial shapes through a text file called 'init.txt' by using command line arguments
   - 1 mark if using a single .c file without using multiple .c files to organize your program
   - 2 marks for memory leak
   - 1 mark if not providing readme.txt and Makefile
3. **Bonus Marks**
   - Up to 3 extra marks are available for fun and creative additional features.