Solution: Self-Reference

```
#include <iostream>
#include <simpio.h>
#include "console.h"
#include "Stack.h"
#include "map.h"
#include "filelib.h"
using namespace std;

/* Function Prototypes */
string funcName(string& input, int i);
void mapFuncToLineNum(string & input, Map<string, int> & funcLines);
string readFromFile(string filename);
void printMap(Map<string, int> & funcLines);

/* Constants
 * ----------
 * Using the ASCII representation of characters is actually really BAD style.
 * However, if our code contains curly brackets within quotation marks, it
 * would not be able to run on itself. So using the ASCII representation
 * is a hack around it. Don't copy at home!
 */
const int OPENINGBRACKET  = 123;
const int CLOSINGBRACKET = 125;

/* void mapFuncToLineNum(string & input, Map<string, int> & funcLines)
 * ---------------------------------------------------------------------
 * Processes the input string that contains code of a program.
 * Stores the function names and their corresponding line numbers into a map.
 * Uses a stack to keep track of opening and closing curly brackets.
 * The input string and the map are passed by reference since they are large
 * objects that are expensive to copy.
 */
void mapFuncToLineNum(string & input, Map<string, int> & funcLines) {
    Stack<char> bracketsStack;
    int lineCount = 1; // Since we start on the line 1.
    string curFuncName = "";
    bool lookingForNextFunction = true;
    for (int i = 0; i < input.length(); i++) {
        char cur = input[i];
        if (cur == OPENINGBRACKET) {
            if (lookingForNextFunction) {
                curFuncName = funcName(input, i);
                funcLines[curFuncName] = lineCount;
                lookingForNextFunction = false;
            }
            bracketsStack.push(cur);
        } else if (cur == CLOSINGBRACKET) {
            bracketsStack.pop(); // Pop the corresponding opening bracket
            if (bracketsStack.isEmpty()) {
                // We reached the end of the previous function, so we can
                // start looking for the next one.
                lookingForNextFunction = true;
            }
        } else if (cur == '\n') lineCount++;
    }
}
```

```
/* string funcName(string & input, int i)
 * --------------------------------------
 * Returns the function name of the parsed symbol by traversing the string
 * backwards until the beginning of the line by checking for the previous
 * newline character or the beginning of the file.
 * At the beginning of this function, i is the index of the opening curly
 * bracket. It is important to pass this parameter by value and not by reference.
 * Credit: This function was written by SL Cristian Cibils.
 */
string funcName(string & input, int i) {
    string name = "";
    i--;
    while (i >= 0) { // if i is 0, we hit the beginning of the file
        if (input[i] == '\n') break; // if we found the previous newline
        name = input[i] + name;
        i--;
    }
    return name;
}

/* string readFromFile(string filename)
 * ------------------------------------
 * Opens a file and reads its contents into a string that it returns.
 * Credit: This function was written by SL Cristian Cibils.
 */
string readFromFile(string filename) {
    ifstream in;
    if (!openFile(in, filename)) {
        error("Cannot read file!");
    }
    string file_contents = "";
    string line;
    while (getline(in, line)) {
        file_contents += line + "\n";
    }
    in.close();
    return file_contents;
}

/* void printMap(Map<string, int> & funcLines)
 * -------------------------------------------
 * Prints out the contents of the map from function name to line number.
 */
void printMap(Map<string, int> & funcLines) {
    for (string key : funcLines.keys()) {
       cout << key << " : " << funcLines[key] << endl;
    }
}

/* main function */
int main() {
    string input = readFromFile("../compile_simple/src/compile_simple.cpp");
    Map<string, int> funcLines;
    mapFuncToLineNum(input, funcLines);
    printMap(funcLines);
    return 0;
}
```