Sabancı University

Faculty of Engineering and Natural Sciences
CS204 Advanced Programming
Spring 2022

Homework 4 – C++ Program enclosing symbol check with stacks

Due: 13/04/2022, Wednesday, 21:00

PLEASE NOTE:

Your program should be a robust one such that you have to consider all relevant programmer mistakes and extreme cases; you are expected to take actions accordingly!

You can NOT collaborate with your friends and discuss solutions. You have to write down the code on your own. Plagiarism will not be tolerated!

Introduction

In this homework, you will implement a program which analyzes whether a given .cpp file (C++ source code) contains enclosing symbol errors or not. In other words, the program will check whether all curly brackets, { }, square brackets, [ ], and regular parenthesis, ( ), are opened and closed correctly or not. To achieve this, the program must use a *stack* data structure. The details of the methods, algorithms and the data structure to be used will be explained in the subsequent sections of this homework specification.

Program Flow

At first, the program asks for the name of the .cpp file to be analyzed. You should check whether the file has been opened successfully or not. In case of a file open failure, ask for a different file name until it is opened successfully. While reading the content of the .cpp file, your program is going to print some information regarding the enclosing symbols, i.e. bracket characters, { } [ ] ( ), in the file. Namely, it will print the following information for each symbol:

1. the symbol itself,
2. the line number in which it occurs, and
3. if the symbol is a closing one, i.e. } or] or), the line number of its matching opening symbol.

As soon as a syntax error related to enclosing symbols is detected, your program will display an appropriate error message and will stop without checking the rest of the file. If processing the input file is complete and no errors are found, you will display a general success message. See the sample runs about these messages, but before that, please read on the document to comprehend the rules that you will need to check in your program.

Line numbering starts with 1.

Data Structure to be Used

In this homework you are going to use *STACKS ONLY*. Stacks will be used to keep track of enclosing symbol information of the input .cpp file. The stack code files that we provide together with the homework document, **`DynStackHW4.cpp`** and **`DynStackHW4.h`**, must be used in your program and **you are not allowed to modify their content (but you need to change the file names to follow the file naming convention)**. Your main program must be written as another cpp file.

You cannot use any other multiple data containers, such as vectors, built-in arrays, dynamic arrays, other linked lists, etc., in this homework.

The stack class, namely `DynStack`, provided in the homework package is a specialized stack class that stores information regarding enclosing symbols. Each node in the stack will contain information about a single symbol as well as the line number in which that symbol is located. `DynStack` uses the `StackNode` type of struct as shown below:

```
struct StackNode {
  char symbol; // enclosing symbol
  int line_number; // line where symbol is seen
  StackNode* next;
};
```

Enclosing Symbol Rules

Once you've opened the input .cpp file, you will create an empty `DynStack` object and will begin processing the file one character at a time. Your program must check that the input .cpp file abides by the following rules:

1. Any opening symbol, `{ ( [`, must have a matching closing symbol, `} ] )`, that appears somewhere after the opening one. However, there could be some other codes between them.
2. A particular closing symbol closes the last seen but unclosed matching opening symbol (see the example in item 3 below).
3. Once you have an opening symbol, there could be other opening ones coming after (same type or not). Here, the rule is this: all open ones, which are opened afterward, must be closed before closing a previous one. For example, consider the following code:

```
Line 1   int main(){
Line 2       x(10, {20});
Line 3       y(5);
Line 4   }
```

   Here, the open curly bracket on Line 1 (the red one) is closed at the end on Line 4. Meanwhile, there are several other parenthesis and brackets opened and closed. You can follow the matchings through colors.
4. There should not be any unclosed open symbols left when you reach the end of the file.

Maybe you have realized, but let us list the possible error cases that might appear:

1. If you encounter a closing symbol and if the last seen and unclosed opening symbol does not match, then this is an error. For example, in the following code, the closing green curly bracket } should pair with the last seen unclosed open symbol, which is the green open parenthesis (after x, but since their types do not match, this is an error.

```
int main(){
    x(10, {20}};
    y(5);
}
```

2. An error occurs if an opening symbol is never closed at the end of the input .cpp file. There is such a case in the sample runs.
3. When a closing symbol is seen and if there is no unclosed opening symbol seen before, then this is an error. For example, consider the following code. Here, the closing curly bracket on Line 3 does not have any opening one since all of the previous opening ones have been previously closed.

```
void foo ()
{}
}
int main(){
    x(10, {20});
    y(5);
}
```

No unclosed opening symbol for this.

Algorithmic Hint: The use of stack is related to checking these rules and error cases. If you keep information regarding the opening symbols on the stack, when you see a closing one, you can check the top of the stack to decide. To do so, `push`, `pop` and `isEmpty` functions of the given `DynStack` class is sufficient, but to be on the safe side we also implemented destructor, deep copy constructor and deep assignment operator as well.

Assumptions

While processing an input .cpp file, please make the following assumptions:

1. There are no comments in the input .cpp file.
2. If there are string or character literals in the input .cpp file, they will not contain enclosing symbols. In other words, strings and characters inside the input .cpp file will not contain any of the symbols `{ } [ ] ( )`

Rules and Submission

General rules and the submission guidelines are the same as previous homework assignments, please refer to them. You have to submit both .cpp and .h files of the stack class (after properly renaming) together with your main program file.

Good Luck!
Albert Levi and Amro Alabsi Aljundi

Sample Runs

Some sample runs are given below, but these are not comprehensive, therefore you have to consider **all possible cases** to get full mark. Inputs are shown in **bold** and *italic*.

The sample input files are provided in the .zip package of this homework.

Sample Run 1:

ex1.cpp

```
#include <iostream>

using namespace std;

int main()
    {
    function((int)10.1, {1,2,3});
    {
        cout << "hello";
    }
    return 0;
}
```

Output:

```
Please enter the file name: ex1.txt
File not found.
Please enter the file name: ex1.c
File not found.
Please enter the file name: ex1.cpp
Found the opening symbol ( at line 5
Found the closing symbol ) at line 5 which closes the opening symbol ( at line 5
Found the opening symbol { at line 6
Found the opening symbol ( at line 7
Found the opening symbol ( at line 7
Found the closing symbol ) at line 7 which closes the opening symbol ( at line 7
Found the opening symbol { at line 7
Found the closing symbol } at line 7 which closes the opening symbol { at line 7
Found the closing symbol ) at line 7 which closes the opening symbol ( at line 7
Found the opening symbol { at line 8
Found the closing symbol } at line 10 which closes the opening symbol { at line 8
Found the closing symbol } at line 12 which closes the opening symbol { at line 6
File processed successfully. No errors were found.
```

Sample Run 2:

ex2.cpp

```
int main(){
    x(10, {20};
    y(5)}
}
```

Output:

```
Please enter the file name: ex2.cpp
Found the opening symbol ( at line 1
Found the closing symbol ) at line 1 which closes the opening symbol ( at line 1
Found the opening symbol { at line 1
Found the opening symbol ( at line 2
Found the opening symbol { at line 2
Found the closing symbol } at line 2 which closes the opening symbol { at line 2
Found the opening symbol ( at line 3
Found the closing symbol ) at line 3 which closes the opening symbol ( at line 3
ERROR!!! Found the closing symbol } at line 3 but the last unclosed opening symbol is ( at line 2
```

Sample Run 3:

ex3.cpp

```
int main(){
    x(10, {20});
    y(5);
    (1+5+(11
```

Output:

```
Please enter the file name: ex3.cpp
Found the opening symbol ( at line 1
Found the closing symbol ) at line 1 which closes the opening symbol ( at line 1
Found the opening symbol { at line 1
Found the opening symbol ( at line 2
Found the opening symbol { at line 2
Found the closing symbol } at line 2 which closes the opening symbol { at line 2
Found the closing symbol ) at line 2 which closes the opening symbol ( at line 2
Found the opening symbol ( at line 3
Found the closing symbol ) at line 3 which closes the opening symbol ( at line 3
Found the opening symbol ( at line 4
Found the opening symbol ( at line 4
ERROR!!! The following opening symbols were not closed:
  Symbol ( at line 4
  Symbol ( at line 4
  Symbol { at line 1
```

Sample Run 4:

ex4.cpp

```
int main(){
    x(10, {20});
    y(5);
    (1+5+(11))
}
}
```

Output:

```
Please enter the file name: ex4.cpp
Found the opening symbol ( at line 1
Found the closing symbol ) at line 1 which closes the opening symbol ( at line 1
Found the opening symbol { at line 1
Found the opening symbol ( at line 2
Found the opening symbol { at line 2
Found the closing symbol } at line 2 which closes the opening symbol { at line 2
Found the closing symbol ) at line 2 which closes the opening symbol ( at line 2
Found the opening symbol ( at line 3
Found the closing symbol ) at line 3 which closes the opening symbol ( at line 3
Found the opening symbol ( at line 4
Found the opening symbol ( at line 4
Found the closing symbol ) at line 4 which closes the opening symbol ( at line 4
Found the closing symbol ) at line 4 which closes the opening symbol ( at line 4
Found the closing symbol } at line 5 which closes the opening symbol { at line 1
ERROR!!! Found the closing symbol } but there are no unclosed opening symbols!
```

Sample Run 5:

ex5.cpp

```
void foo(){
    cout << "hello again";
}
int main(){
    x(10, {20});
    y(5);
    (1+5+(11));
}
```

Output:

```
Please enter the file name: ex5.cpp
Found the opening symbol ( at line 1
Found the closing symbol ) at line 1 which closes the opening symbol ( at line 1
Found the opening symbol { at line 1
Found the closing symbol } at line 3 which closes the opening symbol { at line 1
Found the opening symbol ( at line 4
Found the closing symbol ) at line 4 which closes the opening symbol ( at line 4
Found the opening symbol { at line 4
Found the opening symbol ( at line 5
Found the opening symbol { at line 5
Found the closing symbol } at line 5 which closes the opening symbol { at line 5
Found the closing symbol ) at line 5 which closes the opening symbol ( at line 5
Found the opening symbol ( at line 6
Found the closing symbol ) at line 6 which closes the opening symbol ( at line 6
Found the opening symbol ( at line 7
Found the opening symbol ( at line 7
Found the closing symbol ) at line 7 which closes the opening symbol ( at line 7
Found the closing symbol ) at line 7 which closes the opening symbol ( at line 7
Found the closing symbol } at line 8 which closes the opening symbol { at line 4
File processed successfully. No errors were found.
```

Sample Run 6:

ex6.cpp

```
int main(){
    x(10, {20});
    y(5);
    (1+5+(11));
}
}
void who(){
    cout << "hello once more";
}
```

Output:

```
Please enter the file name: ex6.cpp
Found the opening symbol ( at line 1
Found the closing symbol ) at line 1 which closes the opening symbol ( at line 1
Found the opening symbol { at line 1
Found the opening symbol ( at line 2
Found the opening symbol { at line 2
Found the closing symbol } at line 2 which closes the opening symbol { at line 2
Found the closing symbol ) at line 2 which closes the opening symbol ( at line 2
Found the opening symbol ( at line 3
Found the closing symbol ) at line 3 which closes the opening symbol ( at line 3
Found the opening symbol ( at line 4
Found the opening symbol ( at line 4
Found the closing symbol ) at line 4 which closes the opening symbol ( at line 4
Found the closing symbol ) at line 4 which closes the opening symbol ( at line 4
Found the closing symbol } at line 5 which closes the opening symbol { at line 1
ERROR!!! Found the closing symbol } but there are no unclosed opening symbols!
```