A Pretty-printing / Automatic Style Program

Lab 7: https://maryash.github.io/135/labs/lab_07.html

- Sadab Hafiz

Auto-formatter

The goal of this lab is to create a program that will take a .cpp file and apply correct formatting.

```
                int main(){
            // Hi, I'm a program!
int x = 1;
    for(int i = 0; i < 10; i++) {
cout << i;
          cout << endl;
 }
    }
```

```
int main(){
    // Hi, I'm a program!
    int x = 1;
    for(int i = 0; i < 10; i++) {
        cout << i;
        cout << endl;
    }
}
```

The first step in our program is to remove all the leading indents from each line. We will implement a function to do that.

```
                int main(){
            // Hi, I'm a program!
int x = 1;
    for(int i = 0; i < 10; i++) {
cout << i;
            cout << endl;
 }
    }
```

```
$ ./unindent < bad-code.cpp
int main(){
// Hi, I'm a program!
int x = 1;
for(int i = 0; i < 10; i++) {
cout << i;
cout << endl;
}
}
```

Input redirection from text files

When using standard input (cin), it is possible to redirect input from a text file. In other words, instead of typing lines of input in the terminal, you can get those inputs from a text file.

```
$ ./unindent < bad-code.cpp ←
int main(){
// Hi, I'm a program!
int x = 1;
for(int i = 0; i < 10; i++) {
cout << i;
cout << endl;
}
}
```

← The first line here shows how you can redirect input of your program from a text file using < with your executable.

In this case, if you use cin, or getline(cin, str), the input for the executable `./unindent` will be obtained from the file called `bad-code.cpp`.

Removing the leading spaces

Implement the function: string removeLeadingSpaces(string line);

This function takes a single string and removes all the leading spaces. For example, calling `removeLeadingSpaces(“ Hello”);` will return `Hello`

We will create a main function that gets the data using getline and outputs the result of calling removeLeadingSpaces function on each line that we read

The input in this case will come from the text file. Create a text file with badly indented code to test your program

Main function pseudocode

For our main function, we need to use a while-loop that will keep getting the user input and print the output of calling the removeLeadingSpaces function on each line of input. Follow this pseudocode:

```
// everything in green is pseudocode
int main(){
      . . . initialize an empty string which will be used to get user input
      while(getting a single line of user input using getline){
            . . . print the result of calling `removeLeadingSpaces` on user input
      }
      return 0;
}
```

Removing spaces from each line

You can find the index of first non-space character and get the rest of the string starting from that index

In order to find the first non-space character, consider using the `isspace()` function from `<cctype>` library. This function was introduced in the last lab

In order to get the rest of the string, use the `substr()` function from `<string>` library. If you're not sure how to use this function, you can also use a for-loop

Indenting unindented lines

If we want to correctly indent each line, we need to count how many { and } are there on each line. For this lab, assume that all { increases the number of indents and all } decreases the number of indents.

```
0   int main(){
1   // Hi, I'm a program!
1   int x = 1;
1   for(int i = 0; i < 10; i++) {
2   cout << i;
2   cout << endl;
2   }
1   }
```

Initialize an integer variable that starts at 0. Increment the variable based on the number of { in the current line and decrement it based on the number if } in the current line.

Printing this variable followed by the lines would result in the output to the left.

Counting curly braces

You can write a function to count both open and closed curly braces:
int countChar (string line, char c) ;

This function will count how many times parameter `c` happens in the string parameter `line`. Sounds like a for-loop with an if-statement.

Call this function with **{** to get the number of open curly braces. For example:
`countChar(“ {{Hello ”, ‘ { ’) == 2`

Similarly, get the number of closed curly braces: `countChar(“ }}Hello ”, ‘ } ’) == 2`

Updating main function

```
// everything in green is pseudocode

int main(){

      string input = “”;

      int tabs = 0;

      while ( getting a single line of user input using getline and storing in `input`){

            cout << tab << “ ” << removeLeadingSpaces(input) <<endl;

            . . . increment diff by the number of “{” in user input

            . . . decrement diff by the number of “}” in user input

      }

      return 0;

}
```

Replacing the number with indents

Now that we have the number of indents, we will replace the number with tabs for indentation. A tab character in c++ is `\t`.

```
// everything in green is pseudocode
int main(){
      string input = “”;
      int tabs = 0;
      while ( getting a single line of user input using getline and storing in `input`){
            . . . print `tabs` amount of `\t` using a for-loop
            cout << removeLeadingSpaces(input) <<endl;
            . . . increment diff by the number of “{” in user input
            . . . decrement diff by the number of “}” in user input
      }
      return 0;
}
```

Final Changes

If you did everything correctly so far, you will see the following output:

```
int main(){
    // Hi, I'm a program!
    int x = 1;
    for(int i = 0; i < 10; i++) {
        cout << i;
        cout << endl;
        } // <-- closing for loop
    }     // <-- closing main
```

Based on the output on the left, you can see that the last two lines are not good. They should have one less indent.

To fix that, we will use a conditional to check if the first character of the current line is }. If it is, we will print one less tab before printing the line.

Needless to say, this program will not work in certain scenarios

Final Changes

```
// everything in green is pseudocode

string input = “”;

int tabs = 0;

while ( getting a single line of user input using getline and storing in `input`){

      . . . if the first non-space character of `input` is `}`, decrease `tabs` by 1

      . . . print `tabs` amount of `\t` using a for-loop

      cout << removeLeadingSpaces(input) << endl;

      . . . increment diff by the number of “{” in user input

      . . . decrement diff by the number of “}” in user input

      // undo the effect of the conditional earlier

      . . . if the first non-space character of `input` is `}`, increment `tabs` by 1

}
```