CS205 C/C++ Programming Lab Assignment 2

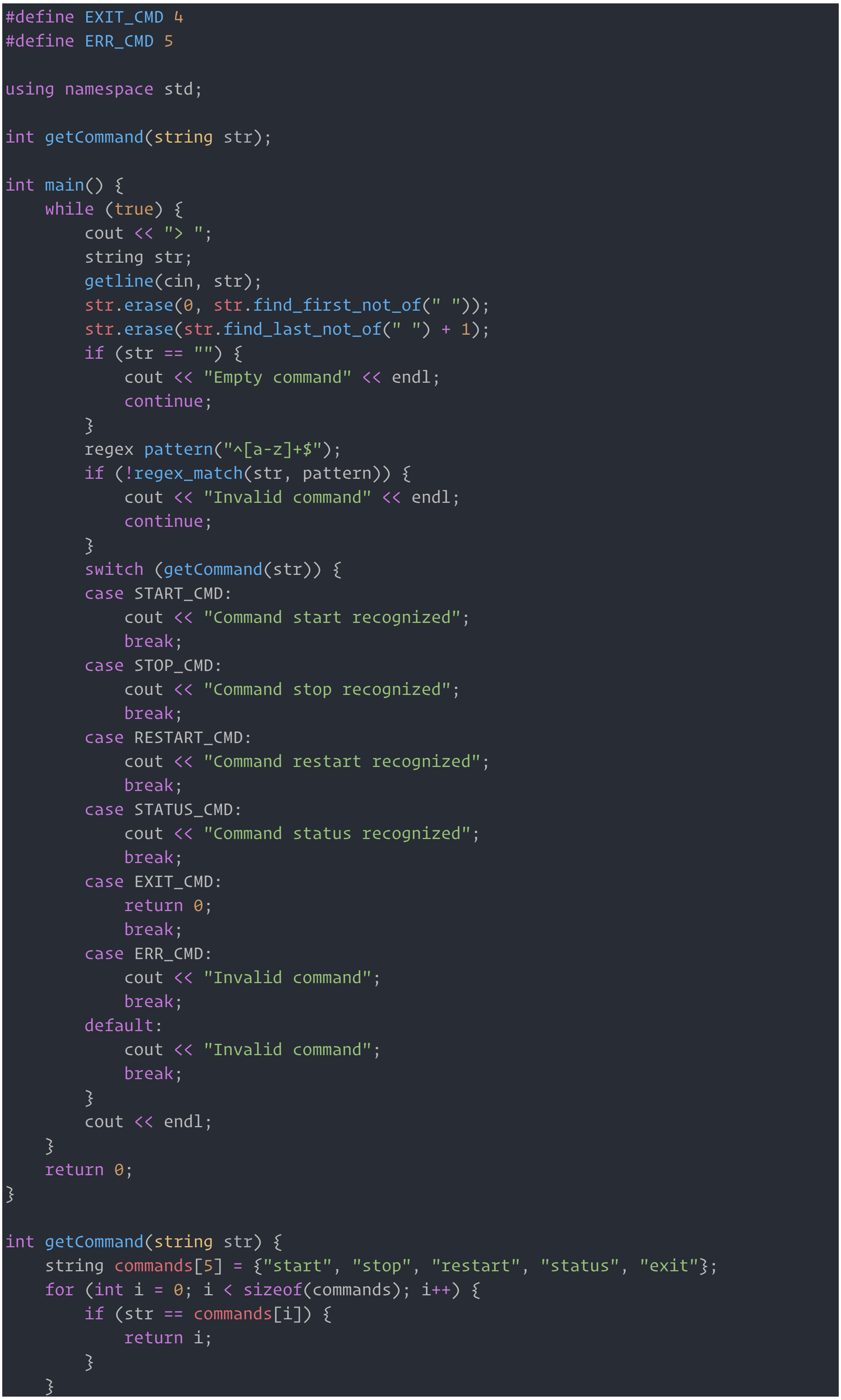
Name: 钟兆玮 (Zhaowei Zhong)
SID: 11611722

Part 1 - Analysis

You have seen during the lecture that "switch" doesn't allow using strings.
It's very frequent to have programs that control other programs that run as services (we'll talk about programs that run as services and how a program can control another program towards the end of the course). For instance, there is a famous Web server program called "apache" and when you want to control it you start another program called "apache_ctl".
This program (just like a console) displays a prompt such as "> " and then expects you to type a command, for instance "start" to start the server, "stop" to stop it, "restart" to stop it first and then start it, "reload" to make it read again a configuration file that was modified, etc. Of course there is also an "exit" command to quit apache_ctl.
You are asked to write a program that will accept the following commands: start, stop, restart, status, exit (to quit it) and, when a command other than "exit" is recognized, will simply display "command <name here> recognized". It must also say "Invalid command" if the command isn't recognized.
You are asked to test the command in a switch statement, not an if ... else if ... else if ... structure.
For this, you'll have an array of strings containing the commands, you'll search it, and if you find the command you'll return its index in the array. The index is an integer and can be used for "switch".
For legibility, you'll associate a symbol to each index.
For instance if you have
char *commands[] = {"start", "stop", ... };
You can have
#define START_CMD   0
#define STOP_CMD   1
and use in the switch:
case START_CMD: ...

Part 2 - Code

```
#include <iostream>
#include <string>
#include <regex>

#define START_CMD 0
#define STOP_CMD 1
#define RESTART_CMD 2
#define STATUS_CMD 3
```



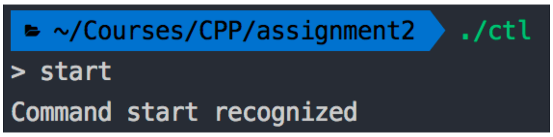
```
#define EXIT_CMD 4
#define ERR_CMD 5

using namespace std;

int getCommand(string str);

int main() {
    while (true) {
        cout << "> ";
        string str;
        getline(cin, str);
        str.erase(0, str.find_first_not_of(" "));
        str.erase(str.find_last_not_of(" ") + 1);
        if (str == "") {
            cout << "Empty command" << endl;
            continue;
        }
        regex pattern("^[a-z]+$");
        if (!regex_match(str, pattern)) {
            cout << "Invalid command" << endl;
            continue;
        }
        switch (getCommand(str)) {
        case START_CMD:
            cout << "Command start recognized";
            break;
        case STOP_CMD:
            cout << "Command stop recognized";
            break;
        case RESTART_CMD:
            cout << "Command restart recognized";
            break;
        case STATUS_CMD:
            cout << "Command status recognized";
            break;
        case EXIT_CMD:
            return 0;
            break;
        case ERR_CMD:
            cout << "Invalid command";
            break;
        default:
            cout << "Invalid command";
            break;
        }
        cout << endl;
    }
    return 0;
}

int getCommand(string str) {
    string commands[5] = {"start", "stop", "restart", "status", "exit"};
    for (int i = 0; i < sizeof(commands); i++) {
        if (str == commands[i]) {
            return i;
        }
    }
    return 5;
}
```

Part 3 - Result & Verification

Test Case #1: Normal Case: start

Input:

start

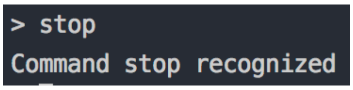


```
~/Courses/CPP/assignment2  ./ctl
> start
Command start recognized
```

Test Case #2: Normal Case: stop

Input:

stop

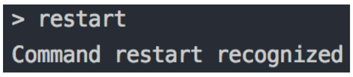


```
> stop
Command stop recognized
```

Test Case #3: Normal Case: restart

Input:

restart

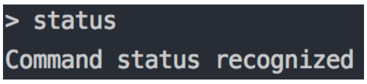


```
> restart
Command restart recognized
```

Test Case #4: Normal Case: status

Input:

status

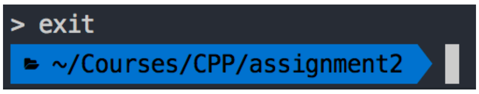


```
> status
Command status recognized
```

Test Case #5: Normal Case: exit

Input:

exit

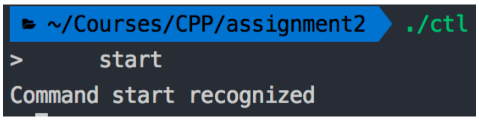


```
> exit
~/Courses/CPP/assignment2
```

Test Case #6: Command with blanks

Input:

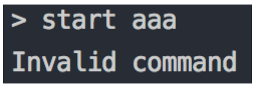
      start



```
~/Courses/CPP/assignment2  ./ctl
>      start
Command start recognized
```

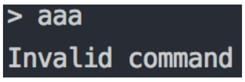
Test Case #7: Invalid Command

Input:
start aaa



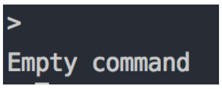
```
> start aaa
Invalid command
```

Test Case #8: Invalid Command
Input:
aaa



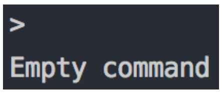
```
> aaa
Invalid command
```

Test Case #9: Blank input
Input:



```
>
Empty command
```

Test Case #10: No input (Enter)
Input:



```
>
Empty command
```

Part 4 - Difficulties & Solutions

The `switch` does not accept string, so I use array finding to converse them into pre-defined int first, then determine the command using `switch` statements.