CS 161, Lecture 7: Loops and Error Handling – 26 January 2018

```
% phd.m
%
% author: Cecilia
% date: 09/08/05

load THESIS_TOPIC

while (funding==true)
    data = run_experiment(THESIS_TOPIC);
    GOOD_ENOUGH = query(advisor);
    if (data > GOOD_ENOUGH)
        graduate();
        break
    else
        THESIS_TOPIC = new();
        years_in_gradschool += 1;
    end
end
```

Match

- Choices: for loop, while loop, do while loop
- Scenarios:
  - Given a record of students' grades, calculate the final grades
  - Create a list of groceries by asking the user for items to be on the list
  - Calculate the average of a list of numbers
  - Search a file for the first 'a' character, return the location
  - Until there is a winner, play the game

for each student → calcgrade

do while

for loop

while loop

do while loop

Extra Looping Details: Scope

- Loops (and if statements) assume the first line after them is in their scope

```
for (int i=0; i<5; i++)
        cout << i << endl;

for (int i=0; i<5; i++) {
        cout << i << endl;
}
```

Extra Looping Details: Scope

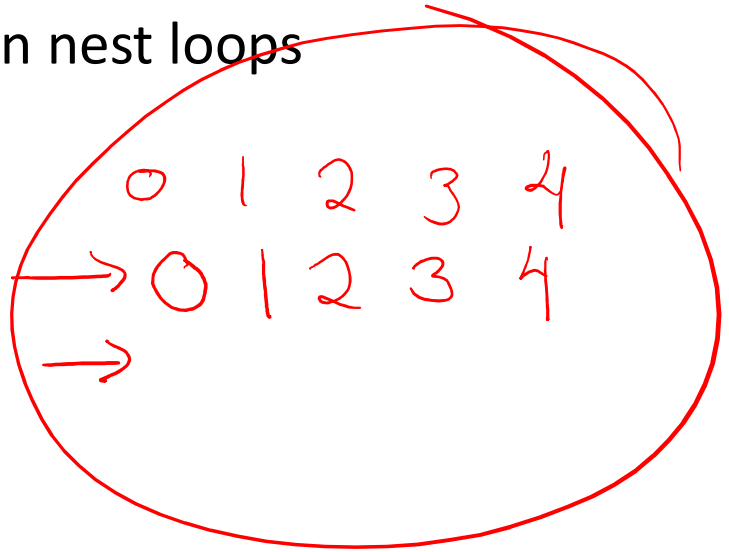
- The names of variable can be the same but their memory address are different

```
int i = 0;
for (int i = 0; i < 5; i++) {
        cout << i << endl;
}
cout << i << endl;
```

Extra Looping Details: Nesting

- Just like with conditionals, we can nest loops

```
for (int i=0; i<5; i++) {
        for (int j = 0; j<5; j++) {
                cout << j;
        }
        cout << endl;
}
```



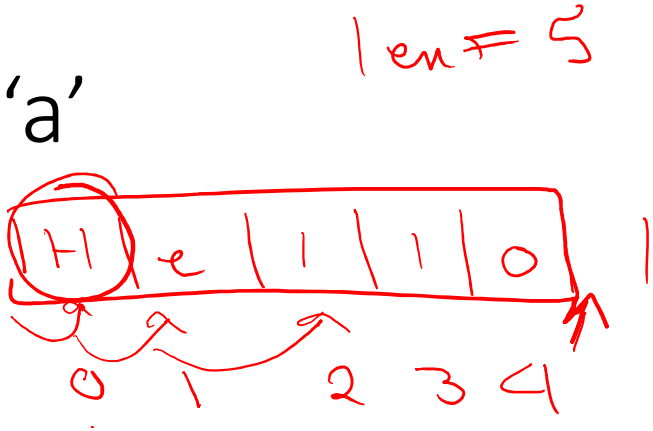
Extra Looping Details: Terms

- **Break:** used with switch and loops, breaks out of the closest associated case or loop (for, while or do while). This can only occur in a loop or a case.
- **Return:** leave the current function, which exits the program when in main() function. You can put this anywhere inside any function.
- **exit():** exit the entire program no matter where this is encountered. You can put this anywhere inside any function, so long as you include <cstdlib>

Error Handling

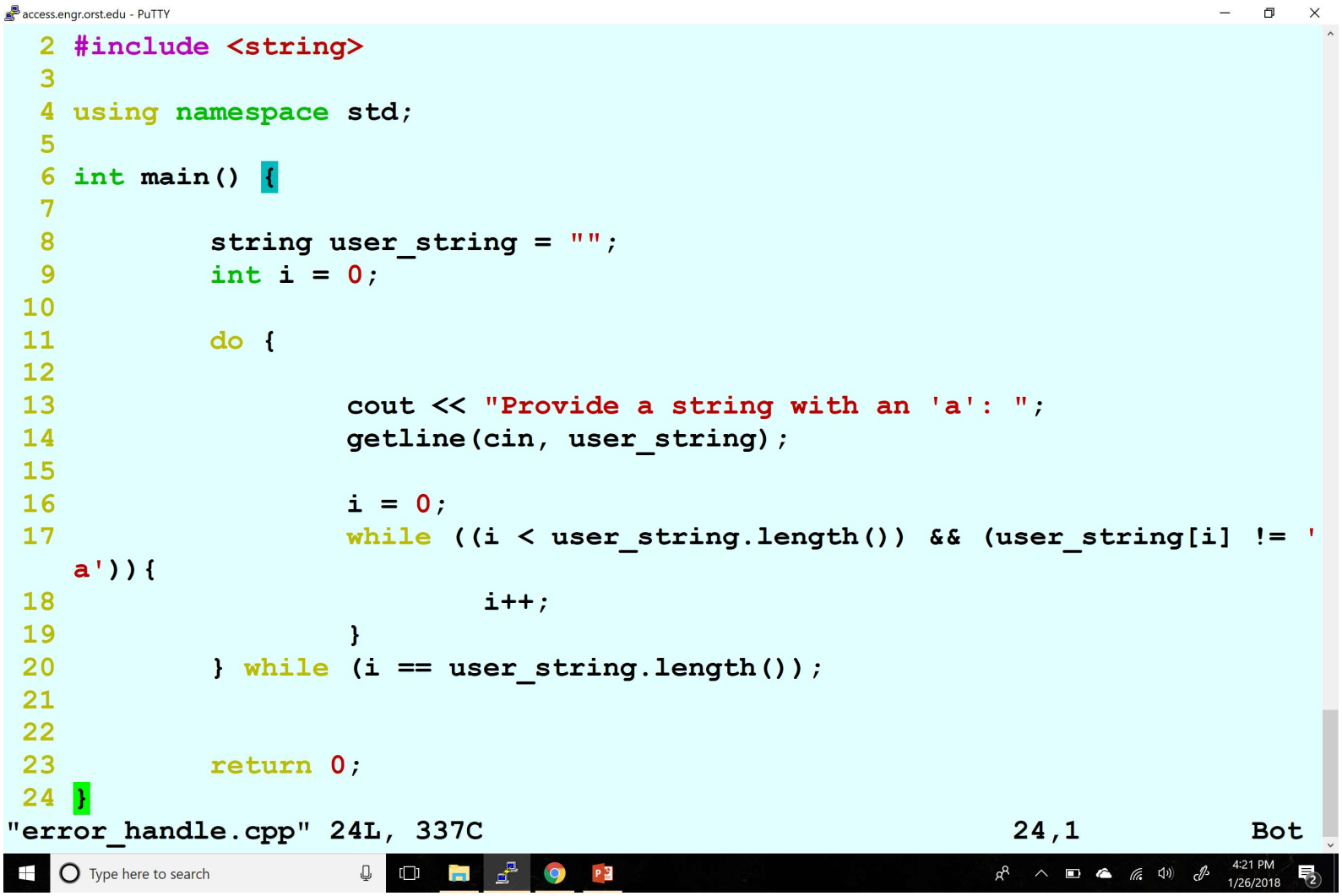
- Catching and recovering from mistakes users may make
- Typically will take input as a string
- Examples of errors:
  - Wrong data type
  - Not following instructions (negative for positive, etc)
    - Too much data
    - More than one datatype
    - No data
  - Max num possible reached

Example: Must have an ‘a’



- Input: string from the user
- Constraint: must have an ‘a’
- Output: reprompts if input doesn’t have an ‘a’, proceeds otherwise

do once
while the character is not 'a'
keep looping


access.engr.orst.edu - PuTTY
#include <string>
using namespace std;
int main() {
string user_string = "";
int i = 0;
do {
cout << "Provide a string with an 'a': ";
getline(cin, user_string);
i = 0;
while ((i < user_string.length()) && (user_string[i] != 'a')){
i++;
}
} while (i == user_string.length());
return 0;
}
"error_handle.cpp" 24L, 337C
24,1
Bot
Type here to search
4:21 PM
1/26/2018

Exercise: check_length

- Input: int number of the correct length of a string, string user provides
- Outputs: if the string is the correct length
- Constraints: you can’t use the built in length() function
- Design

Feedback

- **https://tinyurl.com/y8wyx3te**