Week 3

CS 150 – C++ Programming I
In-Person Lecture 3

A Little Review

- We'll start with a little review of weeks 1 and 2 before we dive into today's exams, which will take the last 3 hours.
  - To review the material, I'm going to ask you questions
  - You are going to confer with your "group" (those in your row)
  - You'll answer using a "clicker" program
- Log into the desktop computers
- Double-click the file in Q:\faculty5\sgilbert\cs150\TTAFT
- Log in with your Canvas ID (eg. sgilbert) and your student ID (eg. C01234567), just like the Homework Console

A Little String Review

- Construct: `string s("Hello World");`
  - `string dashes(50, '-'); // 50 dashes`
  - `string ell("hello", 1, 3); // ell`
  - `string raw = R"("\hello\")"; // "\hello\"`
- Input: token and line, `>>`, `getline(cin, line)`
- Member functions & operators: `size()`, `at()`, `front()`, `back()`, `substr()`, `find()`, `[]`, `+`, `+=`
- Loops: `for(auto e : str) …`

Question

```
string s1 = string('*', 10);
string s2{10, '*'};
string s3{10, "*"};
string s4(10, '*');
```

- Which initializes a **string** object to 10 asterisks?
  - A. **s1**
  - B. **s2**
  - C. **s3**
  - D. **s4**
  - E. None of these

Question

```
string s1 = string();
string s2;
string s3 = "";
string s4();
```

- Which does NOT create a **string** object?
  - A. **s1**
  - B. **s2**
  - C. **s3**
  - D. **s4**
  - E. None of these (that is ALL create a string object)

Question

```
string s1 = "abc";
string s2 = "xyz";
string s3 = s1 + '-' + s2;
cout << s3 << endl;
```

- What prints?
  - A. **abc**
  - B. **xyz**
  - C. **abc-xyz**
  - D. **xyz-abc**
  - E. Compile error: illegal concatenation

Question

```
string s1;
s1 = "abc" + "-" + "xyz";
cout << s1 << endl;
```

- What prints?
  - A. **abc**
  - B. **xyz**
  - C. **abc-xyz**
  - D. **xyz-abc**
  - E. Compile error: illegal concatenation

Question

```
string s1 = "horse", s2 = "cart";
if (s1 < s2) { cout << s2; }
else { cout << s1; }
cout << endl;
```

- What is the output?
  - A. horse
  - B. cart
  - C. horsecart
  - D. Compile error: using < with string
  - E. Compile error: illegal braces

Question: What is stored in x?

- ```
  string s = "abcdefg";
  auto x = s.substr(0, 1) + s.back();
  ```

  - A. `"ag"`
  - B. `'ag'`
  - C. `200`
  - D. Does not compile
  - E. Something else
  ```

Question: What is stored in x?

```
string s = "abcdefg";
auto x = s.back() + s.front();
```

- A. "ag"
- B. "ga"
- C. 200
- D. Does not compile
- E. Something else

Loop Concept Review

- Strategy for building correct loops
  - a) loop bounds b) bounds precondition
    c) advance the loop d) goal precondition
    e) loop operation f) postcondition
- Necessary & intentional bounds, loop guards
- Sequence point: when all side-effects are evaluated
  - Full expression (semicolon or a condition)
  - `&&`, `||`, `?:` and comma operator
  - Variable declarations (`int a = 3, b = a`)

Question

- Six loop-building steps covered in the Course Reader are:
  - 1) loop bounds, 2) bounds precondition, 3) advancing the loop, 4) goal precondition, 5) goal operation, 6) postcondition
- Which is represented by the highlighted line(s):
  - A. 1
  - B. 2
  - C. 3
  - D. 4
  - E. 5

```
Given: the variable str is a string
Create the counter variable, initialized to 0
Create the variable current-character as a character
Place the first character in str into current-character
While the current-character is not a period
{
    Add one to (or increment) the counter variable
    Store the next character from str in current-character
}
Add one to the counter to account for the period.
The variable counter contains the goal
```

Question

- Six loop-building steps covered in the Course Reader are:
  - 1) loop bounds, 2) bounds precondition, 3) advancing the loop, 4) goal precondition, 5) goal operation, 6) postcondition
- Which is represented by the highlighted line(s):
  - A. 1
  - B. 2
  - C. 3
  - D. 4
  - E. 5

```
Given: the variable str is a string
Create the counter variable, initialized to 0
Create the variable current-character as a character
Place the first character in str into current-character
While the current-character is not a period
{
    Add one to (or increment) the counter variable
    Store the next character from str in current-character
}
Add one to the counter to account for the period.
The variable counter contains the goal
```

Question

- Six loop-building steps covered in the Course Reader are:
  - 1) loop bounds, 2) bounds precondition, 3) advancing the loop, 4) goal precondition, 5) goal operation, 6) postcondition
- Which is represented by the highlighted line(s):
  - A. 1
  - B. 2
  - C. 3
  - D. 4
  - E. 5

```
Given: the variable str is a string
Create the counter variable, initialized to 0
Create the variable current-character as a character
Place the first character in str into current-character
While the current-character is not a period
{
    Add one to (or increment) the counter variable
    Store the next character from str in current-character
}
Add one to the counter to account for the period.
The variable counter contains the goal
```

Question

- Six loop-building steps covered in the Course Reader are:
  - 1) loop bounds, 2) bounds precondition, 3) advancing the loop, 4) goal precondition, 5) goal operation, 6) postcondition
- Which is represented by the highlighted line(s):
  - A. 1
  - B. 2
  - C. 3
  - D. 4
  - E. 5

```
Given: the variable str is a string
Create the counter variable, initialized to 0
Create the variable current-character as a character
Place the first character in str into current-character
While the current-character is not a period
{
    Add one to (or increment) the counter variable
    Store the next character from str in current-character
}
Add one to the counter to account for the period.
The variable counter contains the goal
```

Question

- The highlighted line is:
  - A. a loop guard
  - B. a necessary condition
  - C. an intentional condition
  - D. a boundary condition
  - E. None of these

```
Given: the variable str is a string (may be empty)
Create the counter variable, initialized to -1
If the variable str has any characters then
{
    Set counter to 0
    Create the variable current-character as a character
    Place the first character in str into current-character
    While more-characters and current-character not a period
    {
        Add one to (or increment) the counter variable
        Store the next character from str in current-character
    }
    If current-character is a period then
        Add one to the counter to account for the period.
    Else
       Set counter to -2
}
If counter is -1 the string was empty
Else if counter is -2 there was no period
Else the variable counter contains the goal
```

Question

- Here is a limit loop Which statement advances the loop?
  - A. `while (b != 0)`
  - B. `int t = b`
  - C. `b = a % b`
  - D. `a = t`
  - E. None of these

```
int gcd(int a, int b) {
    while (b != 0) {
        int t = b;
        b = a % b;
        a = t;
    }
    return a;
}
```

Question

- What is the value of x after this loop completes?
  - A. Compiler error.
  - B. Infinite loop, x never becomes >= 0.
  - C. -1
  - D. -4
  - E. 0

```
int x = 4;
do {
    x -= 5;
    x++;
} while (x >= 0);
```

Question

```
int val = 1;
while (val++ < 5)
    cout << val << " ";
```

- What prints? (Assume all includes, etc.)
  - A. 2 3 4 5
  - B. 1 2 3 4
  - C. 1 2 3 4 5
  - D. Syntax error
  - E. Endless Loop

Question

```
int val = 1;
while (val++ < 5);
    cout << val << " ";
```

- What prints? (Assume all includes, etc.)
  - A. 2 3 4 5
  - B. 5
  - C. 6
  - D. Syntax error
  - E. Endless Loop

Question

```
int val = 1;
while (val < 5);
    cout << val++ << " ";
```

- What prints? (Assume all includes, etc.)
  - A. 2
  - B. 1
  - C. 5
  - D. 1 2 3 4
  - E. Endless Loop

Question

```
int val = 1;
while (val < 5)
    cout << val++ << " ";
```

- What prints? (Assume all includes, etc.)
  - A. 2 3 4 5
  - B. 1 2 3 4
  - C. 1 2 3 4 5
  - D. Syntax error
  - E. Endless Loop

Question

```
int i = 5;
while(--i) cout << i;
```

- What prints here?
  - A. Compiler error. `--i` not a Boolean expression.
  - B. Infinite loop
  - C. 54321
  - D. 43210
  - E. 4321

Question

```
int i = 5;
while(i) cout << i--;
```

- What prints here?
  - A. Compiler error. `i` not a Boolean expression.
  - B. Infinite loop
  - C. 54321
  - D. 543210
  - E. 43210

Function Review

- Calling a function; arguments & returned values
  - `auto a = std::sqrt(7.5);`
- Declaration, interface or prototype
  - `double sqrt(double); // semicolon`
- Definition or implementation
  - `// No semicolon, name required`
    `double sqrt(double a) { return a / 42; }`
- Separate compilation: client, `.h`, `.cpp`, `makefile` (variables, dependencies, targets & actions)

Question

- Given this prototype and variable definition, which of these function calls are legal (that is, they will compile)?
  - A. `f(2);`
  - B. `f(&a);`
  - C. `f(a);`
  - D. All of these
  - E. None of these

```
void f(int& n);
int a = 7;
```

Question

- Given this prototype and variable definition, which of these function calls are illegal (that is, they will NOT compile)?
  - A. `f(2);`
  - B. `f(&a);`
  - C. `f(a);`
  - D. All of these
  - E. None of these

```
void f(float n);
int a = 7;
```

Question

- Given this prototype and variable definition, which of these function calls are legal (that is, they will compile)?
  - A. `f(2);`
  - B. `f(&a);`
  - C. `f(a);`
  - D. All of these
  - E. None of these

```
void f(float& n);
int a = 7;
```

Question

- What kind of error is this?

```
ex1.cpp:18: undefined reference to `f()'
```

  - A. Syntax error (wrong "grammar")
  - B. Type error (wrong assignment or initialization)
  - C. Compiler error (something missing at this step)
  - D. Linker error (something missing at this step)
  - E. Runtime error (breaks when running)

Question

- What kind of error is this?

```
ex1.cpp:17:5: error: use of undeclared identifier 'func'
```

  - A. Syntax error (wrong "grammar")
  - B. Type error (wrong assignment or initialization)
  - C. Compiler error (something missing at this step)
  - D. Linker error (something missing at this step)
  - E. Runtime error (breaks when running)

Question

- What kind of error is this?

```
throwing an instance of 'std::out_of_range'
```

  - A. Syntax error (wrong "grammar")
  - B. Type error (wrong assignment or initialization)
  - C. Compiler error (something missing at this step)
  - D. Linker error (something wrong at this step)
  - E. Runtime error (breaks when running)

Question

- What kind of error is this?

```
error: assigning to 'int' from incompatible type 'D'
```

  - A. Syntax error (wrong "grammar")
  - B. Type error (wrong assignment or initialization)
  - C. Compiler error (something missing at this step)
  - D. Linker error (something wrong at this step)
  - E. Runtime error (breaks when running)

Question

```
double discount(double& cost)
{
    return (0.9 * cost);
}
int main ()
{
    cout << discount(49.95) << endl;
}
```

- What is the syntax error with this code segment? Which line will the error point to?
  - A. The parameter type should not be a reference (#1)
  - B. The argument to discount must be a variable (#7)
  - C. The discount function should return a reference (&) (#1)
  - D. The discount function should return void (#1)
  - E. There is no error. The code is correct.

Question

```
double discount(double& cost)
{
    return (0.9 * cost);
}
int main ()
{
    double price = 49.95;
    cout << discount(price) << endl;
}
```

- What is the logical error with this code segment?
  - A. The parameter type should not be a reference
  - B. The argument to discount cannot be a variable
  - C. The discount function should return a reference (&)
  - D. The discount function should return void
  - E. There is no error. The code is correct.

Lesson 3A Preview - Data Flows & Control

- Overloading and Default Arguments
  - Overloading - different functions can have the same name
  - Default arguments - a function may have mandatory and optional parameters (and be called in different ways)
- Data Flows: input, output, and input-output parameters
  - Output parameter - no value going in, filled inside function
  - Input parameter - a copy of the value is passed
  - Input-output parameter - parameter changed inside function
  - Memorize data-flow checklists
- Control structures: `switch`, conditional operator, `do-while`

Lesson 3B Preview - Recursion

- A recursive function is one that calls itself
  - Must have a "qualification" called a base case
  - Each recursive call must simplify the original problem
  - Examples: elevator, factorial, Fibonacci
- Naive recursion may be inefficient
  - A function may unnecessarily call a function (fibonacci)
  - A function my unnecessarily allocate memory (palindrome)
  - Solve by writing a helper function and convert original into a recursive wrapper function
- Recursive checklists and a Code-Step-by-Step example

Lesson 3C Preview - Streams & Filters

- The standard streams and the `cat` filter program
  - Redirection: input, output, error and piping
  - Common built-in Unix filter programs
- Types of filter programs
  - Process filters modify the contents of the stream
  - State filters monitor the stream and act on state changes
- Writing your first filter program using data loops
  - Passing streams as arguments and using auxiliary functions
- The stream class hierarchy, UML and file I/O

Lesson 3D Preview - Files & String Streams

- Applied stream processing
  - Patterns for processing lines and tokens
  - Completing the `searchFile()` function
  - Validating input data and handling input errors
- Introducing String Streams
  - Output string streams and the `format()` function
  - Input string streams and the `inputStats()` function
  - Using the `fail()` and `eof()` member functions

LEC-4A Preview-Errors & Assertions

- Using the C++11 string conversion functions
  - Using string streams to "back port" to C++98
- Using the C++ Preprocessor
  - Creating `#defined` constants
  - Using `#if`, `#ifndef` and `#endif` for conditional compilation
- How to handle different kinds of errors
  - Illuminating programming errors with `assert()`
  - Using completion codes to signal an error
  - Using the stream error flags to signal errors

LEC-4B Preview-Exceptions & Templates

- Using C++ exceptions for error handling
  - Using `throw` to signal that an error has occurred
  - What kinds of things can you (and should you) `throw`?
  - Using `try` and `catch` blocks to handle thrown exceptions
  - Using multiple `catch` blocks to handle different errors
- Using templates to write generic functions
  - Instantiating template functions (explicit vs. implicit)
  - Template argument deduction
  - Function templates with multiple type parameters
  - Inferred return types and template overloading

LEC-4C Preview-User-Defined Data Types

- Structured types and the structure definition
  - Creating and initializing structure variables
  - Accessing structure members and aggregate operations
  - Passing structures as arguments to functions
  - Returning structures from functions and structured bindings
- User-defined enumerated or scalar types
  - Scoped (new) and plain (old) enumerated types
  - Enumerated variables and casting to the underlying type
  - Enumerated input and output

LEC-4D Preview-Vectors & Algorithms

- Using the *std::vector* library type
  - Using different constructors to create `vector` variables
  - Retrieving the `vector` `size()` and the `size_t` type
  - Range-checked and unsafe element access
  - Growing and shrinking a `vector`
  - Using `vector` with different kinds of loops
- Common *vector* algorithms
  - Counting for a condition and accumulation algorithms
  - Find the extreme values in a `vector`
  - Modifying algorithms: filling, shifting, shuffling and sorting

Week 3 Homework Preview

- Week 3 HW due by next Tuesday by 1pm (July 4th, No class)
  - H09 - Overloading and data flows: the `read()` procedure
  - H10 - Practicing Recursion - `find()` and `stringCleaning()`
  - H11 - Veni Vidi Vici - writing a process filter for encryption
  - H12 - Writing the `strip()` state filter
  - H13 - Data Sets - processing streams using tokens
  - H14 - A Console Input Library using input string streams
  - H15 - The `tokenStats()` function using string streams

Week 4 Homework Preview

- Week 4 HW due by 1pm July 11th
  - H16 - Templates & conditional compilation: `to_string()`
  - H17 - Structures - defining the `Point` type
  - H18 - Nested structures - defining the `Triangle` type
  - H19 - Streams and `vector` - Spell Checker Part I
  - H20 - Streams and `vector` - Spell Checker Part 2
  - H21 - `vector` Algorithms: the sorted merge

Programming Exam 3 and Midterm Exam #1

- Now – Programming Exam #3
  - I will collect your cellphones, watches & electronics
  - Place all books, backpacks, notes at front or back of the room
  - Move to your assigned seat; do not log in
  - I will start PE03 on your computer
  - Log in using your Homework Console credentials
  - When you are done, submit the exam and leave
- Come back by 4pm when Midterm Exam #1 will start