CS 6015: Software Engineering

Spring 2024

Lecture 5: How to design programs using C++

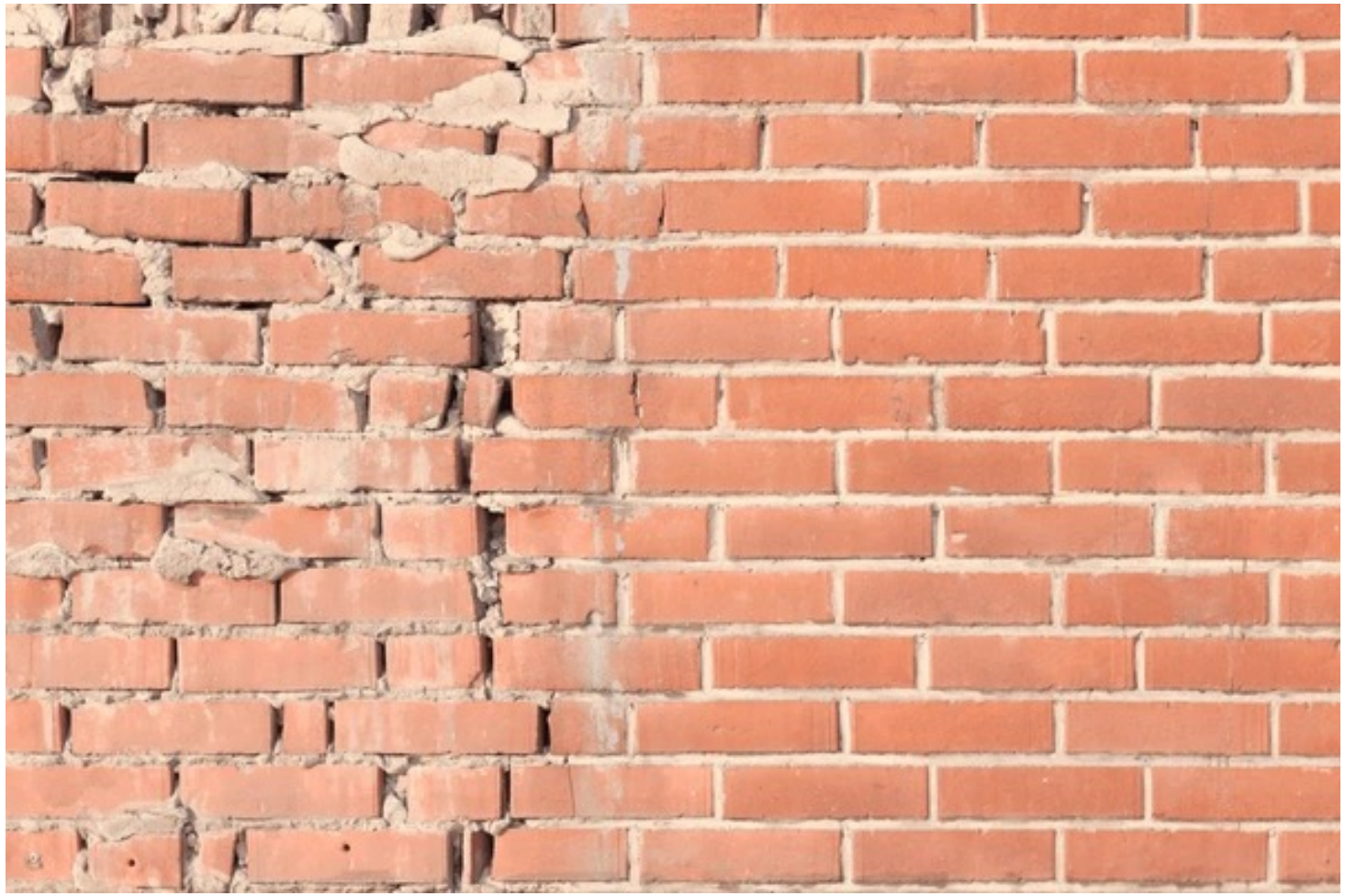
Last Week

- Version/Source Control
- Testing / Catch2 framework

This Week

- How to design a program in C++
- Debugging (Lab 2)
- Code review today
- Assignment 3 released – due next Tuesday
  - Did you invite the TAs to your GitHub Repos?
  - Does your repo contain .o files, xcodeproject files etc…??

Programming vs Good Programming



One way for writing programs

- Start by writing the code
- Do experiments here and there
- Stop when the results look decent

- What if we need to maintain such programs?
  - Read all the code
  - Understand the operations
  - Consumes lot of time

- Called garage programming and might not succeed on many occasions

Better approach for writing programs

- Write a description about the program
  - Explains what the program does, input it expects and what it produces
- Translate the problem statement to a program
  - Any small change to the problem statement is easy to translate to a small change to the program
- This is the process of designing programs

From problems to programs

- How to transform a problem statement to a program?
  - Determine what can be relevant and what can be ignored
  - Determine what the program consumes and produces
  - Define operations and how to relate input to output
- How to deal with operations?
  - Find out if the language provides these operations or you must develop functions to implement them.
- Once you have a program, check if it performs the intended computations.

The design process

- Express how to **represent** data
  - Example: Representing data for a program that transforms Celsius temp to Fahrenheit

The design process

- Express how to **represent** data
  - Use numbers to represent temperature
  - What is the temperature range?
  - Do we have decimals?
  - Use float

The design process

- Write down a statement of **purpose**, a signature and a header.

The design process

- Write down a statement of **purpose** and a signature

```
// Converts Celsius temperature to Fahrenheit
// Takes the temperature as input and returns converted value
float celsius_to_fahrenheit(float n)
{
}
```

The design process

- Provide some **examples** that illustrates the purpose and the signature

The design process

- Provide some **examples** that illustrates the purpose statement and the signature
  - Converts Celsius to Fahrenheit
  - given: 0, expect: 32;
  - given: 100, expect: 212;

  - `CHECK(celsius_to_fahrenheit(0) == 32 );`
  - `CHECK(celsius_to_fahrenheit(100) == 212 );`
  - `CHECK(celsius_to_fahrenheit(-40) == -40 );`

  - `CHECK(celsius_to_fahrenheit(-274) == ?? ); // min -273, then update purpose`

The design process

- Provide a **template** that replaces the function’s body. It contains mainly the parameters.

The design process

- Provide a **template** that replaces the function's body. It contains mainly the parameters.

```
float celsius_to_fahrenheit(float n)
{
    ... n ...
}
```

The design process

- Time to code. Replace the **body** of the function with an expression that computes what the purpose statement promises

The design process

- Time to code. Replace the **body** of the function with an expression that computes what the purpose statement promises

```
float celsius_to_fahrenheit(float n)
{
    return (n * 9.0) / 5.0 + 32;
}
```

The design process

- Test the function on the examples provided before

  - `CHECK(celsius_to_fahrenheit(0) == 32 );`
  - `CHECK(celsius_to_fahrenheit(100) == 212 );`
  - `CHECK(celsius_to_fahrenheit(-40) == -40 );`

- What if test fails?

The design process

- Test the function on the examples provided before

  - `CHECK(celsius_to_fahrenheit(0) == 32 );`
  - `CHECK(celsius_to_fahrenheit(100) == 212 );`
  - `CHECK(celsius_to_fahrenheit(-40) == -40 );`

- What if test fails?
  - Incorrect example samples provided before
  - Error in the function
  - Both

The design process – Summary

- Determine the **representation**
- Write a **purpose** statement
- Write **examples**
- Create a **template** for the implementation
- Finish **body** implementation case-by-case
- Run **tests** (Fix if tests fail)