Midterm Topics and Advice For Studying

October 24, 2025

Midterm topics

I'll go through the sections in the textbook one by one saying how important they are.

- Section 4 discusses errors.
  - **Build Errors**
    - Using a variable before its declaration will result in a *build error*.
    - Redefining a variable in the same scope causes a *build error*.
    - If a variable's type is a `const` fundamental type, it must be initialized when it is defined.
    - Assigning a new value to a `const` variable after initialization will result in a *build error*.
    - Forgetting to include header files (e.g., `#include <iostream>`) or missing `using namespace std;` will *not* appear in the exam.
  - **Runtime Errors**
    - If `s` is a `std::string` and `pos >= s.length()`, then calling `s.at(pos)` produces a *runtime error*.
  - **Undefined Behavior**
    - Using a variable before it is initialized leads to *undefined behavior*.
    - Using `++i` twice within a single expression leads to *undefined behavior*.
    - Integer division by `0` leads to *undefined behavior*.
    - If `s` is a `std::string` and `pos > s.length()`, then accessing `s[pos]` leads to *undefined behavior*.
- Section 5 was where we really got going.
  - Variables and their values: defining variables and assigning values.
  - Naming conventions for variables.
  - Arithmetic operators: `+`, `-`, `*`, `/`.
  - **Increment and decrement operators: `++`, `--`.**
  - **Integer division: understand how integer division truncates results.**
  - **Remainder operator: `%`.**

- Mathematical functions: must use `#include <cmath>`; know common math functions and how to write mathematical formulas in C++.
- Constants: using the `const` keyword to define immutable variables.
- Boolean type (`bool`) and character type (`char`).
- **Type casting: implicit conversions and explicit casting using `static_cast`.**
- Understand the cause of *round-off errors* in floating-point arithmetic.

- Most of Section 6 needs to be known.
  - Strings as sequences of characters.
  - **Common string member functions: `length()` and `substr()`.**
  - **Accessing individual characters in a string using the subscript operator `[]` and `.at()` function.**

- Almost all of section 7 needs to be known inside out. ~~Exceptions to this are...~~
  - Know how to use `cin` to receive input, and how to use `cout` to output;
  - Input buffer ,cin.ignore(), cin.get(), cin.peak()

- Almost all of section 8 needs to be known inside out.
  - `if-else` **Statements and the Conditional Operator**
    - Understand the basic syntax of `if`, `if-else`, and `if-elseif-else` chains.
    - Relational operators: `<`, `>`, `==`, `!=`, `<=`, `>=`.
    - Understand how comparisons and logical conditions evaluate to `true` or `false`.
    - Pay attention to the order of conditions in an `if-elseif-else` chain; once a condition is true, later branches are skipped.
  - `while` **Loop**
    - Understand the mechanism and syntax of a `while` loop.
    - Variables defined inside the loop are not accessible outside the loop.
    - Avoid common logic or infinite loop errors (e.g., forgetting to update the loop variable).
    - Be able to use *hand-tracing* to analyze and predict loop behavior.
  - `for` **Loop**
    - Understand the mechanism and syntax of a `for` loop.
    - Variables (including the loop counter) defined inside the loop are not accessible outside the loop.
    - Pay attention to loop bounds and off-by-one errors.
  - `do-while` **Loop**
    - Understand the mechanism and syntax of a `do-while` loop.
    - Remember that a `do-while` loop always executes its body at least once.
  - **Nested Loops**

    - Be able to analyze and trace nested loops to determine their behavior and output step by step.

- `break` **and** `continue`
    - The `break` statement exits the *current* loop immediately without executing the remaining statements in the loop body.
    - The `continue` statement skips the rest of the current iteration and proceeds