Branching constructs review

- What are branching constructs? What type of branching constructs we studied?
- what is nested if?
- What is multiway if? How does multiway if relate to nested if?
- What is a switch statement? Is it better than multiway if?
- What does break inside switch do?
- What is conditional assignment? what construct can be used instead?
- What are named constants? Why are they needed?
- What is a block? What is special about declaring a variable inside a block?

Iterative constructs

- Key Points
  - Make sure there is a statement that will eventually nullify the iteration criterion (i.e., the loop must stop)
  - Make sure that initialization of any loop counters or iterators is properly performed
  - Have a clear purpose for the loop
    - Document the purpose of the loop and how the body of the loop advances the purpose of the loop

Iterative Constructs

- Provide
  - Ability to control how many times a statement list is executed
- Three constructs
  - while statement
  - for statement
  - do-while statement

The While Statement

- Syntax
  
while (Expression) 
Action

- Semantics
  - If Expression is true then execute Action
  - Repeat this process until Expression evaluates to false
  - Action is either a single statement or a group of statements within braces

The Do-While Statement

- Syntax
  
do Action 
while (Expression)

- Semantics
  - Execute Action
  - if Expression is true then execute Action again
  - Repeat this process until Expression evaluates to false
  - Action is either a single statement or a group of statements within braces
The `for` statement

- **Syntax**
  ```
  for (ForInit; ForExpr; PostExpr)
  Action
  ```

- **Semantics**
  - Execute `ForInit` statement
  - While `ForExpr` is true
    - Execute `Action`
    - Execute `PostExpr`
  - Example
    ```
    for (int i = 0; i < 20; ++i) {
      cout << "i is " << i << endl;
    }
    ```

**break** and **continue** with iterative constructs

- **break** - exits innermost loop
  ```
  while(true) {
    int i;
    cin >> i;
    if (i < 0) {
      cout << "found negative number\n";
      break;
    }
  }
  ```

- **continue** - skip the remaining statements and start a new iteration (evaluate expression)
  ```
  int pos_num=0;
  for (int i = 0; i < 20; ++i) {
    cin >> i;
    if(i < 0) continue;
    pos_num++;
  }
  ```