The **if-else** Statement

- Form: `if (boolean-expression) statement_1 else statement_2`
- If `boolean-expression` is true, `statement_1` is executed; otherwise, `statement_2` is
- Either statement may be compound
- Either statement may be an `if` or `if-else` statement
- If expression is not of type `bool`, it is converted to `bool`
The **switch** Statement

- Form: `{ switch (integer-expression) {
case integer-value: action1
  case integer-value: action2
...
default: default-action
}

- If `expression = value_k`, `action_k` executed
- Use **break** to exit **switch** statement
- **default** optional; `default-action` executed if no `case` applies

---

**Fall-through**

```cpp
switch(x) {
  case 0:
    std::cout << "x is zero\n";
  case 1:
    std::cout << "x is one\n";
  default:
    std::cout << "x is something else\n";
}
```

- If `x==0`, then all 3 cases will apply
- Should use **break** in **cases** 0 and 1
- Otherwise, will "fall through" to next case and execute its action too
Several **cases** at Once

```cpp
switch(x) {
    case 0:
    case 1:
    case 2:
        std::cout << "x is 0, 1 or 2\n"; break;
    default:
        std::cout << "x is 3 or more\n";
}
```

- Actions can be empty, allowing multiple **cases** to use same action
- Not all values need **case**; if none apply, and no **default**, nothing happens

The **do-while** Loop

- Form: 
  ```cpp
do statement while (boolean-expr)
```
- **statement** executed at least once
- Usually a compound statement
- If **boolean-expr** is true, loop will continue; otherwise, it will terminate
- If **boolean-expr** is the value **true**, loop will run forever, so a **break** statement should be used inside the loop
The **while** Loop

- Form:
  ```
  while (boolean-expr) statement
  ```
- Usually a compound statement
- Iteration will continue as long as `boolean-expr` is true
- *Statement* will not execute at all, if `boolean-expr` is immediately **false**
- Previous cautionary note about infinite loop applies here

The **for** Loop

- Best for iterating fixed number of times
- Form:
  ```
  for (expr$_1$; boolean-expr; expr$_2$) stmt
  ```
- `expr$_1$` evaluated before loop begins
- `boolean-expr` evaluated before each iteration; if **false**, loop terminates
- `stmt` executed during each iteration
- `expr$_2$` evaluated after each iteration
- Typical loop that executes `n` times:
  ```
  for (i = 0; i < n; i++) stmt
  ```
Example: Gauss-Seidel

- This program uses Gauss-Seidel iteration to numerically solve a two-point boundary-value problem
- Illustrates while and for loops, the const qualifier, arrays, math functions, typical code for scientific computing
- Also available in FORTRAN 90 (Linux archive only), use gfortran command to compile

break, continue and goto

- The break; statement will cause an immediate exit from an enclosing for, do, while or switch statement
- The continue; statement will interrupt an iteration of a for, do or while loop and begin the next iteration
  - for loop: expr_2 is evaluated
  - do loop: boolean-expr is evaluated
- goto label; jumps to the statement with given label, which is an identifier
Example: Tokenizing Input

- Reads from standard input, recognizes certain C++ tokens (e.g. operators, literals, punctuation, etc.) and prints a list of them (preview of Project 2!)
- Example of first stage of compiling: interpreting text as language constructs
- Illustrates: \texttt{switch} statements, "fall-through", \texttt{while} loops, \texttt{break} statements, other aspects of flow control, strings, standard input (\texttt{cin})

Next Time

- Namespaces (such as \texttt{std})
- More on the C++ Standard Library
- Your questions on Project 1
  - Don't procrastinate!
  - Don't hesitate to ask for help!