

Goals for This Lecture:

- Understand C++ loop structures
- Understand C++ conditional structures

If-else constructs

- C++ has an if-else conditional construct similar to the if-then-else structure of FORTRAN
- Form:

```
if( boolean_expression )
{
    statement block 1;
    ...
}
else
{
    alternative statement block;
    ...
}
```

- If the statement block consists of a single statement the curly brackets may be omitted
- Note that there is no semi-colon after the `if` or `else`
- The `else` clause may be omitted
- If-else constructs can be nested just as in FORTRAN

Multiway if-else constructs

- The multiway if-else conditional construct mimics the elseif clause in FORTRAN
- Form:

```
if( boolean_expression1)
{
    statement block 1;
    ...
}
else if ( boolean_expression2)
{
    statement block 2;
    ...
}
else
{
    alternative statement block;
    ...
}
```

The Switch Statement

- The C++ switch construct is somewhat similar to the FORTRAN case construct
- Form:

```
switch( controlling_expression)
{
  case constant_1:
    statement block 1;
    ...
    break;
  case constant_2:
    statement block 2;
    ...
    break;
  default:
    alternative statement block;
    ...
}
```

- The controlling expression must be of type int, bool, char, or an enum constant
- The **break** statements are needed to exit each branch of the switch
- This means that you can have multiple case labels for each branch

The while construct

- C++ has a while construct that is like the FORTRAN DO WHILE construct

- Form:

```
while ( boolean_expression )
{
    statement block;
    ...
}
```

- C++ loops can be nested just like FORTRAN loops
- The C++ **break** statement acts similarly to the FORTRAN **exit** statement
 - It causes execution to transfer to the first executable statement outside the innermost loop
- The C++ **continue** statement acts similarly to the FORTRAN **cycle** statement
 - It causes execution to transfer to the beginning of the loop

The do-while construct

- C++ has a do-while construct that has no exact analog in FORTRAN

- Form:

```
do {  
    statement block;  
    ...  
} while ( boolean_expression );
```

- Unlike the while loop, the do-while loop evaluates the boolean expression at the end of each loop
- This means the loop always executes at least once

The for construct

- C++ has a for construct that is analogous to indexed DO loop of FORTRAN

- Form:

```
for (initialization_action; boolean_expression; update_action)  
{  
    statement block;  
    ...  
}
```

- Execution continues as long as the boolean expression is true
- Example:

```
for (i=1; i <= 10; i++)  
{  
    cout << " i = " << i << endl;  
    ...  
}
```

The comma operator

- In C++ the comma operator allows execution of a list of statements and returns the value of the last statement.

- Example:

```
z = (x=2, y=x+3.14);
```

- Example:

```
for (sum = 0, i=1; i <= 10; i++)  
  {  
    sum = sum+i*i;  
    cout << " sum = " << sum << endl;  
    ...  
  }
```

Assignment

- Do the self-test exercises in chapter 2 of Savitch
- Read sections 3.1-3.2 of Savitch