Control Flow

Objectives & Outcomes

In this lesson, students will learn about control flow in C++ programs. In particular, students will learn about:

- The different types of control flow statements in C++
- How to use **if/else**, **for**, and while loops for different tasks
- How to use **break** and **continue** statements

By the end of this lesson, students should be able to write programs that use various types of control flow statements to achieve different tasks.

Introduction

Control flow refers to the order in which the instructions in a program are executed. In C++, there are several ways to control the flow of execution, including:

- **Sequential**: instructions are executed one after the other in the order they are written
- **Selection**: instructions are executed based on whether a certain condition is true or false
- **Repetition**: instructions are executed multiple times based on whether a certain condition is true or false

In this lesson, we will focus on selection and repetition, which are the two most common types of control flow statements.

If/Else Statements

An if/else statement is used to execute a certain set of instructions only if a certain condition is true. For example, consider the following code:

```
int x = 5;
if (x > 10) {
   cout << "x is greater than 10" << endl;
}
else {
   cout << "x is less than or equal to 10" << endl;
}
```

In this code, we first declare a variable x and set it equal to 5. We then use an if/else statement to print out a message depending on the value of x. If x is greater than 10, the message "x is greater than 10" will be printed. If x is less than or equal to 10, the message "x is less than or equal to 10" will be printed.

If/else statements can be used to execute different sets of instructions based on different conditions. For example, consider the following code:

```
int x = 5;
if (x > 10) {
   cout << "x is greater than 10" << endl;
}
else if (x == 10) {
   cout << "x is equal to 10" << endl;
}
else {
   cout << "x is less than 10" << endl;
}
```

In this code, we first declare a variable x and set it equal to 5. We then use an if/else statement to print out a message depending on the value of x. If x is greater than 10, the message "x is greater than 10" will be printed. If x is equal

to 10, the message "x is equal to 10" will be printed. If x is less than 10, the message "x is less than 10" will be printed.

For Loops

A for loop is used to execute a certain set of instructions multiple times. The number of times the instructions are executed is determined by a loop control variable. For example, consider the following code:

```
for (int i=0; i<5; i++) {
   cout << i << endl;
}
```

In this code, we use a for loop to print out the numbers 0 through 4. The loop control variable is i, which is first initialized to 0. The instructions in the for loop are then executed 5 times. On the first iteration, i has a value of 0 and the message "0" is printed. On the second iteration, i has a value of 1 and the message "1" is printed. This continues until the fifth iteration, where i has a value of 4 and the message "4" is printed.

While Loops

A while loop is similar to a for loop, except the number of times the instructions are executed is determined by a condition. For example, consider the following code:

```
int i = 0;
while (i<5) {
   cout << i << endl;
   i++;
}
```

In this code, we use a while loop to print out the numbers 0 through 4. The loop control variable is i, which is first initialized to 0. The instructions in the while loop are then executed 5 times. On the first iteration, i has a value of 0 and the message "0" is printed. On the second iteration, i has a value of 1 and the

message "1" is printed. This continues until the fifth iteration, where i has a value of 4 and the message "4" is printed.

Loop Control Statements

There are two special statements that can be used inside loops: break and continue.

Break

The break statement can be used to force a loop to terminate early. For example, consider the following code:

```
for (int i=0; i<5; i++) {
   if (i==3) {
      break;
   }
   cout << i << endl;
}
```

In this code, we use a for loop to print out the numbers 0 through 4. The loop control variable is i, which is first initialized to 0. The instructions in the for loop are then executed 5 times. On the first iteration, i has a value of 0 and the message "0" is printed. On the second iteration, i has a value of 1 and the message "1" is printed. On the third iteration, i has a value of 2 and the message "2" is printed. However, on the fourth iteration, i has a value of 3. Because 3 is equal to 3, the break statement is executed and the loop is terminated early. As a result, the message "4" is never printed.

Continue

The continue statement can be used to skip the rest of the instructions in a loop and go directly to the next iteration. For example, consider the following code:

```
for (int i=0; i<5; i++) {
   if (i==3) {
      continue;
   }
   cout << i << endl;
}
```

In this code, we use a for loop to print out the numbers 0 through 4. The loop control variable is i, which is first initialized to 0. The instructions in the for loop are then executed 5 times. On the first iteration, i has a value of 0 and the message "0" is printed. On the second iteration, i has a value of 1 and the message "1" is printed. On the third iteration, i has a value of 2 and the message "2" is printed. However, on the fourth iteration, i has a value of 3. Because 3 is equal to 3, the continue statement is executed and the rest of the instructions in the loop are skipped. As a result, the message "4" is never printed.