Activity #7: While Loops
Recorder's Report

Manager: Reader:

Recorder: Driver:

Date: Score: Satisfactory / Not Satisfactory

Record your team's answers to the key questions (marked with ) below.

a) Model 1, Question #4

b) Model 2, Question #11.f

c) Model 3, Question #15

Activity #7: While Loops

In this course, you will work in teams of 3–4 students to learn new concepts. This activity will introduce you to loops and the syntax of a while loop in C++.

Content Learning Objectives

After completing this activity, students should be able to:

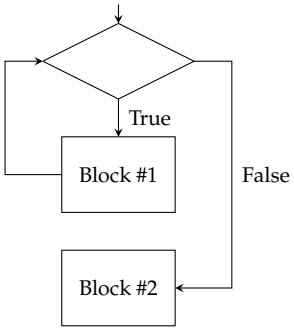
- Explain the three parts of a loop.
- Explain the syntax of a `while` loop in C++.
- Explain **sentinel-controlled** and **counter-controlled** loops.
- Explain **assignment operators**.

Process Skill Goals

During the activity, students should make progress toward:

- Write code that includes **sentinel-controlled** and **counter-controlled** loops.
- Write code that uses **assignment operators**.

Model 1 A Diagram and A C++ Program




```
1  #include <iostream>
2  #include <string>
3
4  using namespace std;
5
6  int main() {
7    // print a person's name 10 times
8    string name;
9    cout << "Enter your name: ";
10   cin >> name;
11   int x = 0;
12   while (x < 10) {
13     cout << name << endl;
14     x = x + 1;
15   }
16   cout << "Nice to meet you!" << endl;
17 }
18
```

Refer to Model 1 above as your team develops consensus answers to the questions below.

Questions (10 min) Start time:

1. Which lines of code go with the following shapes on the provided diagram?

a) The diamond

b) The "Block #1" rectangle

c) The "Block #2" rectangle

2. Every loop structure requires three different actions. Identify the line in the C++ program above that corresponds to each of these actions.

a) **Initialize** a variable to control the number of loop iterations.

b) **Test** a condition to determine if we should keep looping.

c) **Update** the variable involved in the test condition.

3. The complete program is in `activity07a.cpp`. Run the program and answer the following questions.

a) What would the model program do differently if line 11 was: `int x = 1;`?

b) What would the model program do differently if line 12 was: `while (x <= 10)`?

c) What would the model program do differently if line 14 was: `x = x + 2;`?

4. Change one or more of lines 11, 12, and 14 to make the program print the name 23 times.

Model 2 Another C++ Program

```
1  #include <iostream>
2  #include <iomanip>
3
4  using namespace std;
5
6  int main() {
7    int number;
8    cout << "Enter a positive integer: ";
9    cin >> number;
10   int x = 1;
11   while (x <= number) {
12     if (x % 10 == 0) {
13       cout << setw(2) << x << endl;
14     } else {
15       cout << setw(2) << x << " ";
16     }
17     x = x + 1;
18   }
19   cout << endl;
20 }
21
```

Refer to Model 2 above as your team develops consensus answers to the questions below.

Questions (25 min) Start time:

5. Explain what each line or range of lines in this program does.

a) Lines 8-9:

b) Line 10:

c) Line 11:

d) Lines 12-16:

e) Line 17:

6. The complete program can be found in `activity07b.cpp`. Run it and answer the following questions.

a) What is the output when you enter the number 5? How many times did the loop execute?

b) What is the output when you enter the number 25? How many times did the loop execute?

c) Unlike the loop in model 1, this loop prints the counter. How might this make testing easier?

7. The following code snippet should print the numbers from 1 to 10, but it doesn't print anything. Correct the problem. Replace the contents of `activity07b.cpp` to help in your debugging.

```
int number = 12;
while (number <= 10) {
  cout << number << endl;
  number = number + 1;
}

```

8. Enter and execute the code below, then answer the questions that follow.

a) Describe the output.

```
int number = 0;
while (number <= 10) {
  cout << number << endl;
  number = number - 1;
}

```

b) Does the program end? Why or why not?

9. The following step-by-step instructions will assist you in creating a program that prompts the user for a number between 1 and 10 (inclusively). As long as the number is out of range, it re-prompts the user for a valid number.

a) Write code to prompt the user for a number between 1 and 10 and store it in a variable.

b) Write a **Boolean expression** that is true exactly when the variable is **not** between 1 and 10.

c) Use the Boolean expression you just created to write a **while loop** that executes when the user input is out of range. Leave the body of the while loop empty for now.

d) Write code to prompt the user to re-enter a valid number and store it in the same variable.

e) Finally, write code that prints out a message telling the user they entered a valid number.

f) Now put all of the pieces together and replace the code in `activity07b.cpp`. Does your code work properly? If not, correct it and test again.

10. A *counter-controlled loop* is one for which the number of times the loop will execute is known ahead of time. Which of the loops in this model (questions 5, 7, 8, and 9) are counter-controlled?

11. Enter and execute the following code in the file `activity07b.cpp`. Remember to `#include <string>` at the top of your file.

```
#include <iostream>

using namespace std;

int main() {
  string word;
  char doAgain = 'y';
  while (doAgain == 'y') {
    cout << "Enter a word: ";
    cin >> word;
    cout << "The first letter is " << word.at(0) << endl;
    cout << "Type 'y' to enter another word, anything else to quit. ";
    cin >> doAgain;
  }
  cout << "Done!" << endl;
}

```

a) What does this program do?

b) What is the variable name used to store the user's input?

c) What does `word.at(0)` represent?

d) What happens if you change 0 to 1 in `word.at(0)`?

e) When will the program end?

f) A *sentinel-controlled loop* is one in which the loop body repeats until the user enters a particular value or values. Why is this an example of a *sentinel-controlled loop*?

Model 3 Assignment Operators

Initial `x` Value	Assignment Operator	Final `x` Value
2	`x += 1;`	3
2	`x -= 1;`	1
4	`x *= 2;`	8
4	`x /= 2;`	2
7	`x %= 4;`	3

Refer to Model 3 above as your team develops consensus answers to the questions below.

Questions (15 min) Start time:

12. The code `x += 5;` is equivalent to which of the following lines of code?

x = 5; `x = x + 5;`

x = y + 5; y = x + 5;

13. An *assignment operator* provides a concise way of creating assignment statements when the variable on the left-hand side (LHS) will also appear on the right-hand side (RHS). In your own words, describe what each of the following assignment operators does.

a) `+=`

b) `-=`

c) `*=`

d) `/=`

e) `%=`

14. The table below is similar to that seen in Model 3. Fill in the missing pieces with appropriate values or assignment operator statements. Assume that `x` is an integer variable.

Operator	Initial `x` Value	Statement	Final `x` Value
`+=`	6		8
`+=`	5	`x += 1;`	
`-=`		`x -= 3;`	6
`*=`	4		4
`/=`	23		2

15. Is the assignment operator `23 += total` valid? Why or why not?

16. The following code snippet should print the numbers beginning with 100 and counting down to 1. However, it is missing a line of code. Add the missing code using an assignment operator. Indicate the line number at which the code should be inserted.

```
1  int countdown = 100;
2  while (countdown > 0) {
3    cout << countdown << endl;
4  }
5
```

17. Use assignment operators to write loops to do each of the following. You can test your code in the `activity07c.cpp` file.

a) A loop that prints out all multiples of three between 0 and 100. (i.e. 0, 3, 6, 9, . . .)

b) A loop that prints all powers of 2 (i.e. 2^n) from 1 up to 100.