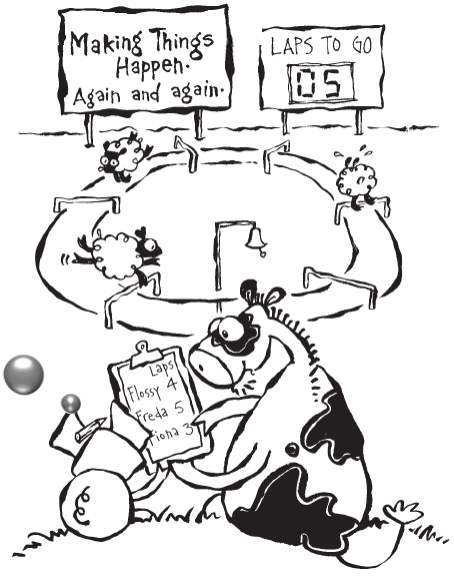
Topic 1 – CS1A Review – P4 - Repetition

Chapter 6 in the shrinkwrap

Repetition Structures

What if I want some instructions to run over and over again?



Repetition Structures

Repetition
➔When a set of instructions need to be executed more than 1 time

- Run a select set instructions repeatedly
  - until some condition is false
- Conditions again are based on a Boolean Expression
- The computer evaluates a Boolean Expression and executes the code until that condition is FALSE

3 Basic Repetition Structures

- For Loop
  - Part of a program is executed a given number of times.
- While Loop
  - Part of a program is executed while some condition is true: *While some condition is true execute these instructions*
- Do While Loop
  - Part of a program is executed at least one time and then repeats until some condition is false.

For right now we will focus on the For Loop

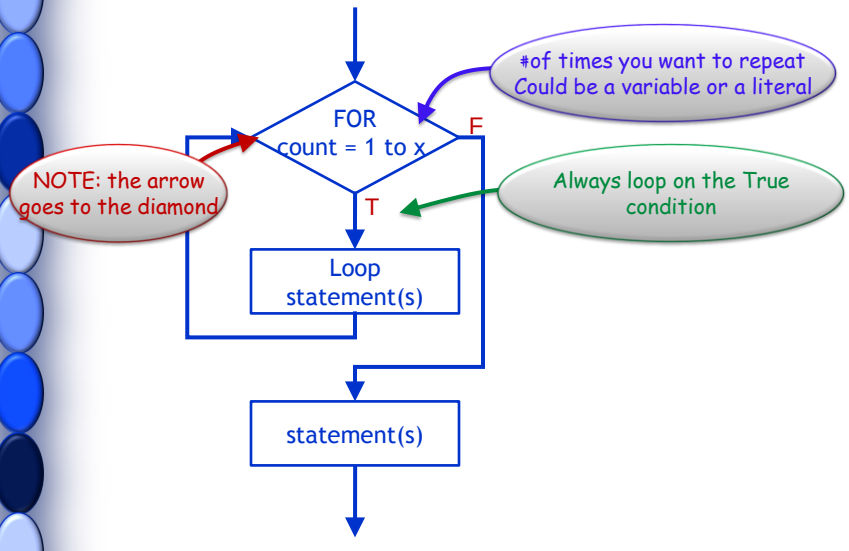
The Loop Control Variable (LCV)

- The LCV is what controls when our loop will execute and when it will exit
- FOR ANY LOOP WE MUST!!
  1 - Initialize the LCV
  2 - Compare (or check) the LCV
  (in some conditional statement)
  3 - Change the LCV

WARNING:
You must change the LCV or else your loop will run forever!
That is called an *infinite loop*.

of 27

Flowchart: For loop



Coding a For Loop

- We use the for loop when we know ahead of time how many times we need to repeat a code segment
- update LCV is the amount the LCV should be modified at each iteration

Syntax:

```
for (initialization; condition; update LCV )
{
   loop statements;
}
```


For Loop Example

initialization condition Update

```
for(count = 1; count <= 3; count = count + 1)
{
   cout << "Enter Name: ";
   cin.getline(username,25);

   cout << "Enter Age: ";
   cin  >> age;
   cin.ignore(10000, '\n');

   cout  >> "Press enter to continue";
   cin.ignore(10000, '\n');
}
```

get the \n off the buffer

flush the input buffer

Why not use cin.get?

3 Basic Repetition Structures

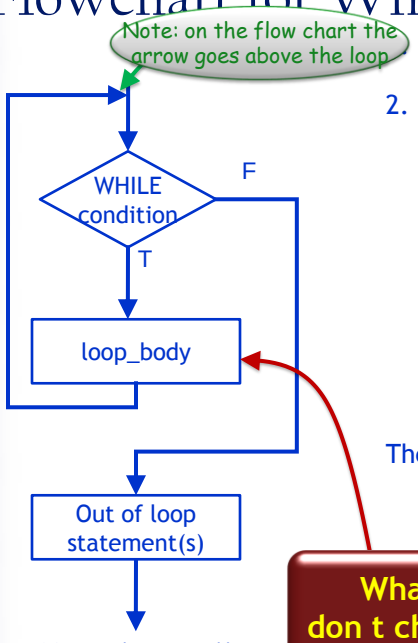
- For Loop
  - Part of a program is executed a given number of times.
- While Loop
  - Part of a program is executed while some condition is true: *While some condition is true execute these instructions*
- Do While Loop
  - Part of a program is executed at least one time and then repeats until some condition is false.

Now we will move onto the While Loop

While Loop

- What if we don't know how many times we need to run our loop?
- We use the while loop
  - → the LCV is modified dynamically within the loop
- The LCV needs to be initialized before entering the loop
- The condition is tested at the top of the loop → making it a pre-test loop
  - if the condition evaluates to TRUE
    - the loop is entered
  - if the condition evaluates to FALSE
    - the loop is bypassed
- The LCV MUST be updated within the loop
  - typically just before the end of the loop
  - otherwise you will run into an infinite loop situation

Flowchart for While Loop



First it evaluates the condition.

2. → If the condition is TRUE it executes the loop_body and returns to step 1

→If the condition is FALSE it exits the loop and does not execute the loop_body, it moves on to the statement immediately following the loop

The loop_body will continue to be executed as long as the condition is true

What will happen if we don t change the LCV in here?

Coding a While Loop

- Event-controlled loop – a loop that terminates based on a condition and a *sentinel value*
  - *this loop executes an unspecified number of times*
- Does a while loop always have to execute?

Syntax:

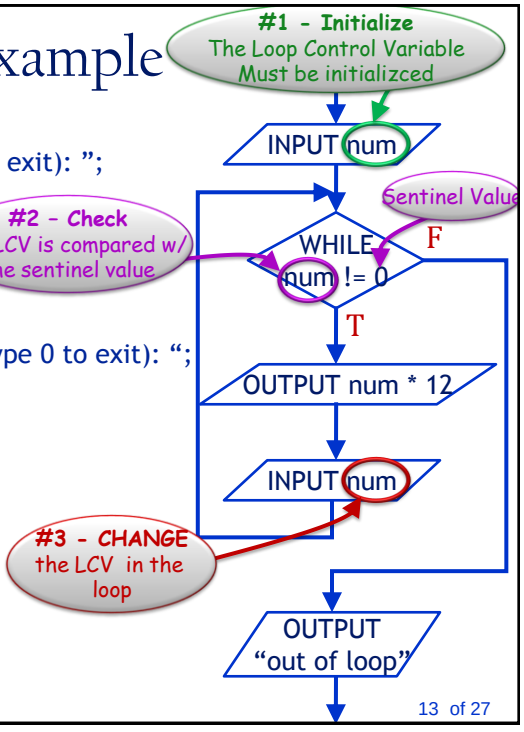
```
initialize the LCV;
while (condition)
{
   loop statements;
   change the LCV;
}
```

While loop Example

```
cout << "Enter an integer (type 0 to exit): ";
cin   >> num;
while (num != 0)
{
   cout << num * 12 << endl;
   cout << "Enter another integer (type 0 to exit): ";
   cin   >> num;
}
cout << "Out of loop" << endl;
```

What does this do?

WARNING:
You must change the LCV so that you don't have an infinite loop!



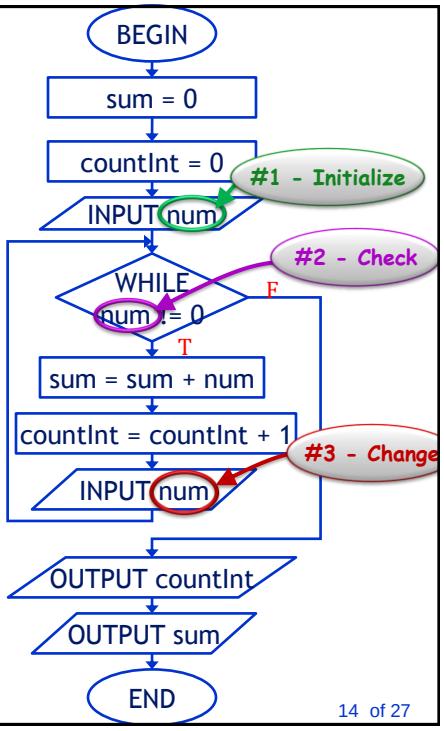
Example #2 – While loop

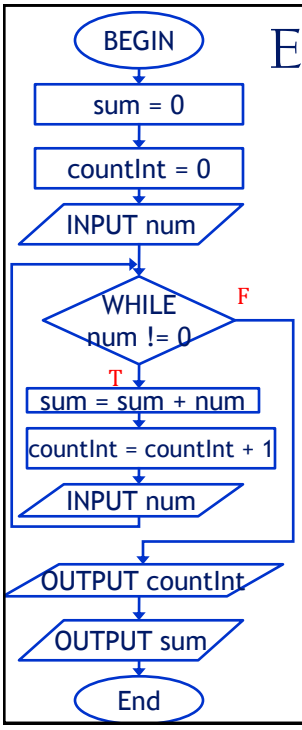
Example

Write a program that will accept a set of integers from a user, sum the set of integers & count how many integers were input. It will stop accepting integers when a 0 is reached.

Output the total number of integers given as input & the sum

First, let's draw the flowchart.





Example #2 - Code

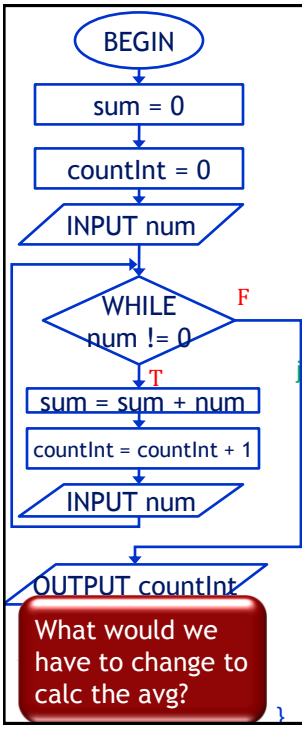
```
#include <iostream>
using namespace std;

int main()
{
```

declaration section with data table

```
    int num;        // IN - integer value
    int sum;        // CALC & OUT -
                    //        - running total of integers
    int countInts;  // CALC & OUT -
                    //        - total number of inputs

     // INITIALIZATION SECTION
     //       Irunning total (sum)
     //       & counter (countInts)
    sum       = 0;
    countInts = 0;
```


```
cout << "Enter an integer (type 0 to exit): ";
cin  >> num;
```

INITIALIZE the LCV by reading the 1st Input

```
// PROCESSING –
//    This loop calculates running total of integers input &
//     counts the # of inputs
while (num != 0)
```

CHECK the LCV against the SENTINEL VALUE

```
{
   // PROCESSING –
   //     Add integer (num) to the running total (sum)
   //     before reading a new value  &
   //     Increment the counter
   sum = sum + num;
   countInts = countInts + 1;

   cout << "Enter another integer  (type 0 to exit):  ";
   cin  >> num;
```

CHANGE the LCV by reading the next Input

```
}

// OUTPUT - counter (countInts) and running total (sum)
cout << "\n\nThe total number of integers was ";
cout << countInts;
cout << "\nThe sum of the integers is " << sum << endl;
}
```

```
int num1, num2, num3;
num1 = 2;
num2 = num1 * 2;
num3 = num1 + num2;
while (num1 <= 15)
{
   cout << num1 << " " << num2 << " " << num3 << endl;
   if (num1 < 6)
   {
      num1 = num1 + num2;
   }
   else
   {
      num1 = num1 + num3;
   }
   num2 = num2 + num3;
   cout << num3 << " " << num2 << " " << num1 << endl;
}
cout << "Out of loop ";
```

num1 num2 num3

What will the output be for this code segment?

When would we use them?

- As a counter
  - Count the # of inputs
- Running totals
  - Sum a # of inputs
- When you don't know how many times you need to loop

Example

Sum a number of integers. Output the sum and the total number of integers given as input.

3 Basic Repetition Structures

- For Loop
  - Part of a program is executed a given number of times.
- While Loop
  - Part of a program is executed while some condition is true: *While some condition is true execute these instructions*
- Do While Loop
  - Part of a program is executed at least one time and then repeats until some condition is false.

Now we will move onto the Do While Loop

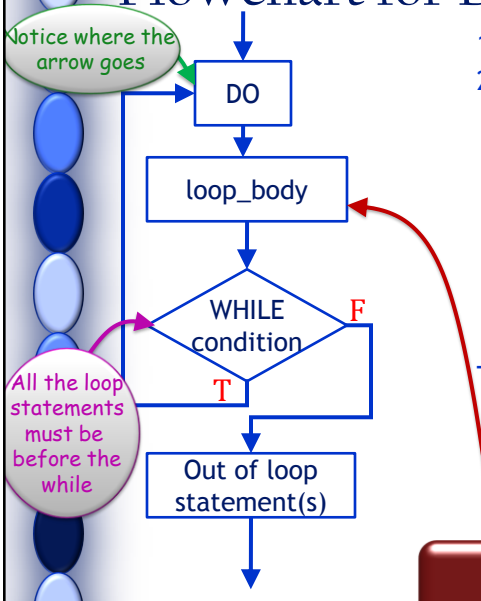
Do While Loop

- The LCV can be initialized in the loop

The loop statements are executed BEFORE the condition is tested

- The condition is tested at the bottom of the loop → making it a post-test loop
  - if the condition evaluates to TRUE
    - the loop statements are executed again
  - if the condition evaluates to FALSE
    - the loop exits (the statements are run at least 1X)
- The LCV MUST be updated within the loop
  - typically at the beginning of the loop
  - otherwise you will run into an infinite loop situation

Flowchart for Do While Loop



1. First, it executes the loop_body.
2. → If the condition is TRUE it executes the loop_body until the condition is false.

   → If the condition is FALSE it exits the loop and executes the statement immediately following the loop.

The loop_body will ALWAYS be executed at least ONCE and then continue to be executed as long as the condition is true.

What will happen if we don t change the LCV in here?

Coding a Do While Loop

- Event-controlled loop – a loop that terminates based on a condition and a *sentinel value*
  - *this loop executes an unspecified number of times*
- Do the statements in a Do While loop ever have to execute?
- For this loop the LCV can be initialized before or within the loop

Syntax:

```
do
{
    change the LCV;
    loop statements;
}
while (condition);
```

Example: Do While

```
int weight;
do
{
     cout << "Enter your weight: ";
     cin   >> weight;
     if (weight < 100)
     {
        cout << "You must weigh at least 100 lbs"
        cout << "\n\t to use this program.\n";
     }
}
while (weight < 100);

cout << "Out of loop with a valid weight of
cout << weight << " lbs.";
```

#1 - Initialize

#3 -Change

#2 - Check LCV

What will the flowchart look like?

When would we use them

- When we know we have to run the code at least 1 time
- Great for checking for valid inputs

Which loop should I use?

- Which loop should be used if we want the loop body to execute a specific number of times?
- Which loop should be used if we want the loop to be controlled an event rather than a counter and the body may or may not be executed?
- Which loop should be used if we want the loop to be controlled an event rather than a counter and the body must be executed at least one time?

Which loop should I use?

- Write a code segment that sums a group of positive integers.
- Write a code segment that sums 10 numbers.
- Write a code segment that validates that a user has entered a value between 1 & 5

Loops – Common Errors

1 – Not initializing the Loop Control Variable

2 – Not updating the LCV within a while or do while loop

3 – A loop that is not properly written can run forever this is called an infinite loop

→ make sure that the condition that ends the loop can and will happen

4 – Off by 1 errors (common in for loops)

How many times will these run?

```
for (i = 0; i <= 3; i = i + 1)
for (i = 1; i < 3; i = i + 1)
for (i = 1; i <= 3; i = i + 1)
for (i = 1; i >= 1;  i = i + 1)
```