For this project, my goal was to find a way to test if each language performs short-circuit evaluation. The approach I took was to use AND in each language to evaluate a statement between a variable A, and a function called Evaluate which just prints a confirmation message and returns 1. To do this, I set A = 1, that way it’s easy to force a true or false condition. Then I tested each combination of True and False by always putting the A condition before calling the Evaluate function after the logical operator. Therefore, if the language does short-circuit evaluation, it will only print out the result of the operation. Otherwise it prints out the confirmation message indicating the function ran and will also output the result of the operation.

The results of these tests show that each language does in fact use short-circuit evaluation. However, I had to take special caution with Ada. This language has two operators AND and AND THEN. This is tricky because these both evaluate differently. AND will always evaluate both conditions, therefore triggering the extra output from the function. AND THEN will cause Ada to perform short-circuit evaluation. These tests are included in the results below.

As a side note for Shell, instead of using a function for the second condition, I used an echo. An echo evaluates as True, so I only tested False and True, and True and True. This still gave me a correct short circuit evaluation, and one that was not.

|  |  |  |
| --- | --- | --- |
| **Language** | **And Short Circuit** | **And Then Short Circuit** |
| **Ada** | **NO** | **YES** |
| **Shell** | **Yes** | **n/a** |
| **PHP** | **Yes** | **n/a** |
| **Perl** | **Yes** | **n/a** |

**Ada**

|  |  |
| --- | --- |
| INPUT | RESULT |
| False and True | Evaluation confirmation and False |
| True and False | Evaluation confirmation and False |
| True and True | Evaluation confirmation and True |
| False and False | Evaluation confirmation and False |
| False and then True | Short Circuit False |
| True and then False | Evaluation confirmation and False |
| True and then True | Evaluation confirmation and True |
| False and then False | Short Circuit False |

--Joseph Camacho-Terrazas

--9/6/2020

--Input: None

--Output: The result of each evaluation, should be either T or F and sometimes an evaluation statement

--Precondition: None

--Postcondition: Print the results from tests and a confirmation message if 2nd condition is reached in comparison statements.

with Ada.Text\_IO; use Ada.Text\_IO;

with Ada.Integer\_Text\_IO; use Ada.Integer\_Text\_IO;

procedure program2 is

    A : Integer;

    --This function will print out a statement if it's evaluated

    --It returns 1 so that it can be evaluated against a 1 or a 0

    function Evaluate return Integer is

    begin

        Put\_Line("Condition has been evaluated");

        return 1;

    end Evaluate;

begin

    --Set A to 1 so that we can test our logical operations

    A := 1;

    --For AND, I am testing both AND and AND THEN because Ada will always evaluate both statements for AND and short circuit for AND THEN

    --Begin testing AND statements

    Put\_Line("===Testing AND with function as second condition===");

    --Test F and T

    if A = 0 and Evaluate = 1 then

        Put\_Line("True");

        New\_Line;

    else

        Put\_Line("False");

        New\_Line;

    end if;

    --Test T and F

    if A = 1 and Evaluate = 0 then

        Put\_Line("True");

        New\_Line;

    else

        Put\_Line("False");

        New\_Line;

    end if;

    --Test T and T

    if A = 1 and Evaluate = 1 then

        Put\_Line("True");

        New\_Line;

    else

        Put\_Line("False");

        New\_Line;

    end if;

    --Test F and F

    if A = 0 and Evaluate = 0 then

        Put\_Line("True");

        New\_Line;

    else

        Put\_Line("False");

        New\_Line;

    end if;

    --Testing with function as the first condition just to verify that the function does work properly

    Put\_Line("===Testing AND with function as first condition===");

    --Test F and T

    if Evaluate = 0 and A = 1 then

        Put\_Line("True");

        New\_Line;

    else

        Put\_Line("False");

        New\_Line;

    end if;

    --Test T and F

    if Evaluate = 1 and A = 0 then

        Put\_Line("True");

        New\_Line;

    else

        Put\_Line("False");

        New\_Line;

    end if;

    --Test T and T

    if Evaluate = 1 and A = 1 then

        Put\_Line("True");

        New\_Line;

    else

        Put\_Line("False");

        New\_Line;

    end if;

    --Test F and F

    if Evaluate = 0 and A = 0 then

        Put\_Line("True");

        New\_Line;

    else

        Put\_Line("False");

        New\_Line;

    end if;

    --Begin testing AND THEN statements

    Put\_Line("===Testing AND THEN with function as second condition===");

    --Test F and then T

    if A = 0 and then Evaluate = 1 then

        Put\_Line("True");

        New\_Line;

    else

        Put\_Line("False");

        New\_Line;

    end if;

    --Test T and then F

    if A = 1 and then Evaluate = 0 then

        Put\_Line("True");

        New\_Line;

    else

        Put\_Line("False");

        New\_Line;

    end if;

    --Test T and then T

    if A = 1 and then Evaluate = 1 then

        Put\_Line("True");

        New\_Line;

    else

        Put\_Line("False");

        New\_Line;

    end if;

    --Test F and then F

    if A = 0 and then Evaluate = 0 then

        Put\_Line("True");

        New\_Line;

    else

        Put\_Line("False");

        New\_Line;

    end if;

    Put\_Line("===Testing AND THEN with function as first condition===");

    --Test F and then T

    if Evaluate = 0 and then A =1 then

        Put\_Line("True");

        New\_Line;

    else

        Put\_Line("False");

        New\_Line;

    end if;

    --Test T and then F

    if Evaluate = 1 and then A = 0 then

        Put\_Line("True");

        New\_Line;

    else

        Put\_Line("False");

        New\_Line;

    end if;

    --Test T and then T

    if Evaluate = 1 and then A = 1 then

        Put\_Line("True");

        New\_Line;

    else

        Put\_Line("False");

        New\_Line;

    end if;

    --Test F and then F

    if Evaluate = 0 and then A = 0 then

        Put\_Line("True");

        New\_Line;

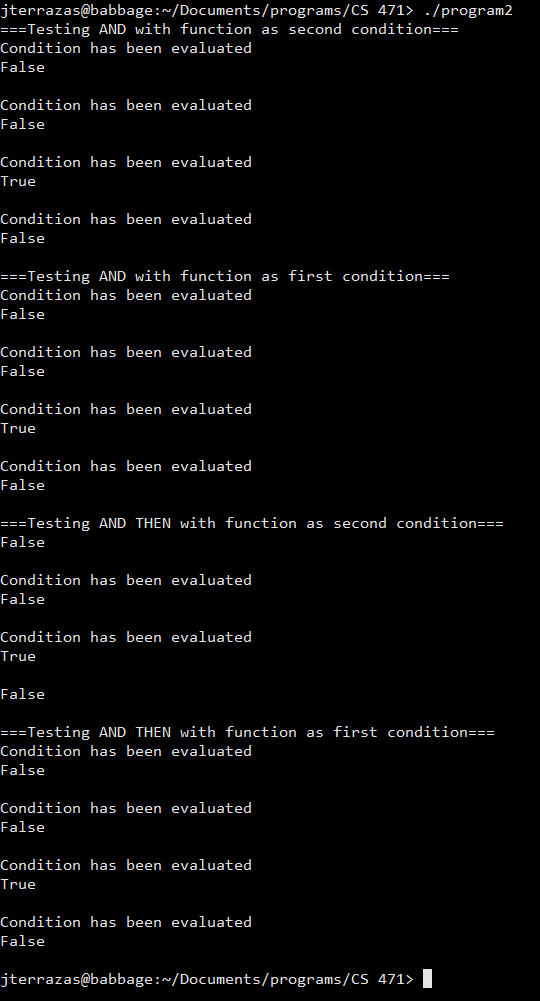
    else

        Put\_Line("False");

        New\_Line;

    end if;

end program2;



**Shell**

|  |  |
| --- | --- |
| INPUT | RESULT |
| False and True | Short Circuit False |
| True and True | Evaluation confirmation and False |

#!/bin/sh

#Joseph Camacho-Terrazas

#09/6/2020

#Input: None

#Output: Results of each tests

#Preconditions: None

#Postconditions: Print the results from tests and a confirmation message if 2nd condition is reached in comparison statements.

#Declare variable a for evaluation

a=1

#Testing AND logical operators

echo "===Testing AND with echo as the second condition==="

#test F and T

if [[ $a == 0 ]] && echo "Condition has been evaluated"

then

    echo "True"

else

    echo "False"

fi

echo

#test T and T

if [[ $a == 1 ]] && echo "Condition has been evaluated"

then

    echo "True"

else

    echo "False"

fi

echo

#Testing with function as the first condition just to verify that the function does work properly

echo "===Testing AND with echo as the first condition==="

#test F and T

if  echo "Condition has been evaluated" && [[ $a == 0 ]]

then

    echo "True"

else

    echo "False"

fi

echo

#test T and T

if echo "Condition has been evaluated"  && [[ $a == 1 ]]

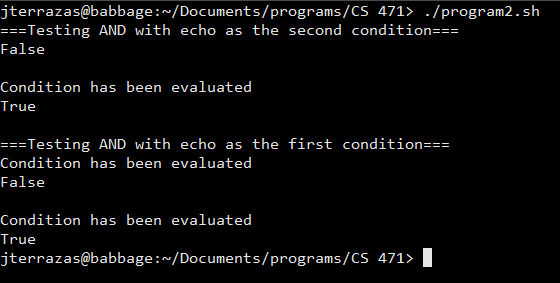
then

    echo "True"

else

    echo "False"

fi



**PHP**

|  |  |
| --- | --- |
| INPUT | RESULT |
| False and True | Short Circuit False |
| True and False | Evaluation confirmation and False |
| True and True | Evaluation confirmation and True |
| False and False | Short Circuit False |

<?php

//Joseph Camacho-Terrazas

//9/6/2020

//Input: None

//Output: Results of each test

//Preconditions: None

//Postconditions: Print the results from tests and a confirmation message if 2nd condition is reached in comparison statements.

//Create Evaluate function to print a confirmation message if it's evaluated.

//Returns true for easy comparisons

function Evaluate(){

    echo "Condition has been evaluated" .PHP\_EOL;

    return True;

}

//Testing "AND" operators

echo "===Testing AND with function as second condition===" .PHP\_EOL;

//Test F and T

if (False && Evaluate()) {

    echo "True" .PHP\_EOL;

} else {

    echo "False" .PHP\_EOL;

}

echo "" .PHP\_EOL;

//Test T and F

if (True && Evaluate() == False) {

    echo "True" .PHP\_EOL;

} else {

    echo "False" .PHP\_EOL;

}

echo "" .PHP\_EOL;

//Test T and T

if (True && Evaluate()) {

    echo "True" .PHP\_EOL;

} else {

    echo "False" .PHP\_EOL;

}

echo "" .PHP\_EOL;

//Test F and F

if (False && Evaluate() == False) {

    echo "True" .PHP\_EOL;

} else {

    echo "False" .PHP\_EOL;

}

echo "" .PHP\_EOL;

//Testing with function as the first condition just to verify that the function does work properly

echo "===Testing AND with function as first condition===" .PHP\_EOL;

//Test F and T

if (Evaluate() == False && True) {

    echo "True" .PHP\_EOL;

} else {

    echo "False" .PHP\_EOL;

}

echo "" .PHP\_EOL;

//Test T and F

if (Evaluate() && False) {

    echo "True" .PHP\_EOL;

} else {

    echo "False" .PHP\_EOL;

}

echo "" .PHP\_EOL;

//Test T and T

if (Evaluate() && True) {

    echo "True" .PHP\_EOL;

} else {

    echo "False" .PHP\_EOL;

}

echo "" .PHP\_EOL;

//Test F and F

if (Evaluate() && False) {

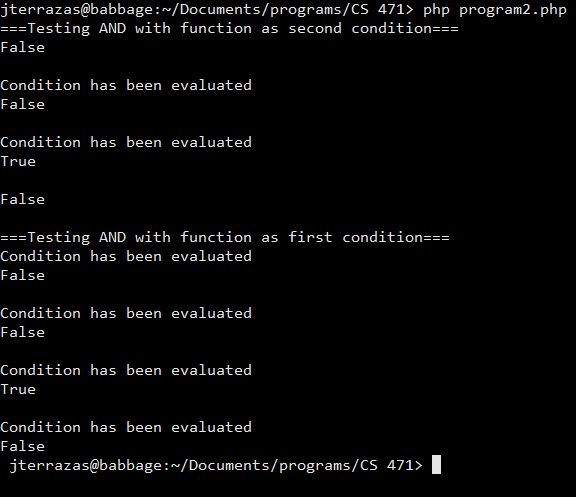
    echo "True" .PHP\_EOL;

} else {

    echo "False" .PHP\_EOL;

}

?>



**Perl**

|  |  |
| --- | --- |
| INPUT | RESULT |
| False and True | Short Circuit False |
| True and False | Evaluation confirmation and False |
| True and True | Evaluation confirmation and True |
| False and False | Short Circuit False |

#Joseph Camacho-Terrazas

#09/6/2020

#Input: None

#Output: Result of each test

#Preconditions: None

#Postconditions: Print the results from tests and a confirmation message if 2nd condition is reached in comparison statements.

#!/usr/bin/perl

use warnings;

#Create function Evaluate that prints a confirmation message upon evaluation

#Returns 1 for easy comparisons

sub Evaluate {

    printf "Condition has been evaluated \n";

    return 1;

}

#Declare variable a for comparisons

$a = 1;

#Testing "AND" logical operators

printf "===Testing AND with function as second condition===\n";

#Test F and T

if ($a == 0 && Evaluate() == 1) {

    printf "True\n";

} else {

    printf "False\n"

}

printf "\n";

#Test T and F

if ($a == 1 && Evaluate() == 0) {

    printf "True\n";

} else {

    printf "False\n";

}

printf "\n";

#Test T and T

if ($a == 1 && Evaluate() == 1) {

    printf "True\n";

} else {

    printf "False\n";

}

printf "\n";

#Test F and F

if ($a == 0 && Evaluate() == 0) {

    printf "True\n";

} else {

    printf "False\n";

}

printf "\n";

#Testing with function as the first condition just to verify that the function does work properly

printf "===Testing AND with function as first condition===\n";

#Test F and T

if (Evaluate() == 0 && $a == 1) {

    printf "True\n";

} else {

    printf "False\n"

}

printf "\n";

#Test T and F

if (Evaluate() == 1 && $a == 0) {

    printf "True\n";

} else {

    printf "False\n";

}

printf "\n";

#Test T and T

if (Evaluate() == 1 && $a == 1) {

    printf "True\n";

} else {

    printf "False\n";

}

printf "\n";

#Test F and F

if (Evaluate() == 0 && $a == 0) {

    printf "True\n";

} else {

    printf "False\n";

}

