Shyan Yen

PLP4: Control Flow

1. What types of conditional statements are available in your language? (e.g.: if/else, if/then/else, if/elseif/else). Does it allow for statements other than “if” (e.g. Perl has an “unless” statement, which does the opposite of “if”!)

* if-else
* Nested if-else
* if-else-if
* else-if

Sources: <http://ecomputernotes.com/cpp/control-structures/conditional-statements>

<https://www.tutorialspoint.com/cplusplus/cpp_if_else_statement.htm>

2. Does your language use short-circuit evaluation? If so, make sure that your code includes an example.

C++ does use short-circuit evaluation using the && operator. It will check if the first

boolean is false, and it it is then it will not check the second boolean.

3. How does your programming language deal with the “dangling else” problem?

C++ deals with the dangling else problem by adding scope operators so that the if

statements are enclosed with separate braces and the else statement is left on the outside

of the braces to be executed.

Source: <https://www.youtube.com/watch?v=WN-3dSCO7Hc>

4. Does your language include multiple types of loops (while, do/while, for, foreach)? If so, what are they and how do they differ from each other?

C++ offers the while, do/while, and for loop. The while loop executes the statement

within its braces repeatedly based on a condition within the parenthesis while the

condition is true. The do/while loop is similar to the while loop except that the do/while is guaranteed to execute at least once. In the for loop, initialization happens once and the loop is controlled based on if the condition is true or false. The statements will execute depending on whether or not the condition holds true. After each execution, the loop counter is updated by 1 each time. While loops are more flexible allowing for substituting variables while for loops will let you know how many times you iterated through the loop as well as having an initializer and incrementer.

Source: <https://www.tutorialspoint.com/cplusplus/cpp_do_while_loop.htm>

<https://beginnersbook.com/2017/08/cpp-for-loop/>

<https://www.codecademy.com/en/forum_questions/510e3c1a3011b8fa25005255>

5. Can you use break or continue statements (or something similar) to control iteration?

C++ allows for both break and continue statements to control iteration.

6. If your language supports switch or case statements, do you have to use “break” to get out of them? Can you use “continue” to have all of them evaluated?

Yes you have to use break to get out of a case statement.

7. Is there anything special in terms of control flow that your language does that isn't addressed in this assignment? If so, what is it and how does it work? Include an example of it in your code.

C++ has a basic control flow statement called halt. It is accomplished by using the exit() function which is defined in the cstdlib header. It takes an integer parameter that is

returned as an exit code similar to the return value of main(). C++ includes various jump

keywords such as goto, break, and continue. C++ has exceptions that offer a mechanism

that handles errors that occur in a function. If an error occurs in a function that the function cannot handle, this will trigger an exception which makes the CPU to jump to the nearest block of code that handles the type of exception.

Source: <https://www.learncpp.com/cpp-tutorial/51-control-flow-introduction/>