There is a plp_4.cpp file also included in this repository that contains code that tests all of this information. It may help to look at this file while reading through the document.

Code Related Questions

How are the following control statements written in your language? If your language does not support these control statements specifically, try to find a way to emulate the behavior.

1. **A one-condition if/else stataement(i.e. "If x == true")**

```
if (x == 5) { std::cout << "x is 5" <<std::endl; }
else { std::cout << "x is not equal to 5" << std::endl; }
```

2. **A multi-condition if/else statement(i.e. "If x > 0 && y < 10")**

```
if (x == 5 && y == 4) { std::cout << "x is 5 and y is 4" << std::endl; }
else { std::cout << "x is not 5 or/and y is not 4" << std::endl; }
```

3. **Different kinds of loops:**
   a. **While**

```
while (x < 10) {
      std::cout << x << std::endl;
      x++;
}
```

   b. **Do while**

```
do {
      std::cout << x << std::endl;
      x++;
      y++;
} while (y < 10);
```

   c. **For**

```
for (int i = 0; i < 10; i++) {
      std::cout << i << std::endl;
}
```

   d. **Foreach**

   There is no foreach loop in C++, however, the same effect can be attained by coding the following:

```
std::vector<int> my_vector{ 1,2,3,4,5 };
for (int a : my_vector) {
      std::cout << a << std::endl;
}
```

4. **A switch-case statement**

```
int z = 2;
switch (z) {
      case 1:
            std::cout << "z is 1" << std::endl;
            break;
      case 2:
            std::cout << "z is 2" << std::endl;
            break;
}
```

5. **Break statement**

```
for (int i = 0; i < 10; i++) {
      if (i == 5) {
            break;
      }
      std::cout << i << std::endl;
}
```

6. **Continue statement**

```
for (int i = 0; i < 10; i++) {
      if (i == 5) {
            continue;
      }
      std::cout << i << std::endl;
}
```

Discussion Questions

1. **What types of conditional statements are available in your language (if/else, if/then/else, if/elseif/else)?**
   C++ allows a programmer to use if/else if/ else conditional statements. The if keyword can be used to evaluate a condition, and, if it is fulfilled, execute a designated statement or statement block. There is an optional choice to add else if and else statements. The difference between the two is that else if evaluates a condition, much like if, while the else statement says “if all other conditions have failed, do this” (See Code Related Questions Parts 1 and 2 for examples).

2. **Does your language use short-circuit evaluation? If so, make sure that your code includes an example.**
   Short-circuit evaluation simply means that if the compiler is evaluating a statement and the first one evaluates to false, it won’t bother checking the other statement. This is because no matter what the second statement evaluates to the condition won’t pass. C++ does implement short-circuit evaluation.

Resource: https://www.interviewcake.com/concept/java/short-circuit-evaluation
https://stackoverflow.com/questions/5211961/how-does-c-handle-short-circuit-evaluation

3. **How does your programming language deal with the ”dangling else” problem?**
   The dangling else problem refers to the way that a compiler reads nested if-else statements. Let us suppose that we have two if statements after each other with an else at the end. To the eye, it may appear that the else belongs to either one depending on how it is indented in the program, however, in C++ indentation does not matter. This means that the compiler may process those statements in a way that it was not intended to be processed. The way that C++ fixes this issue is by ensuring that the compiler will always associate an else statement with the nearest preceding if statement, so long as running it does not cause a syntax error. You can use curly braces to induce syntax errors.

   Reference:
   http://www.mathcs.emory.edu/~cheung/Courses/561/Syllabus/2-C/dangling-else.html

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++ has a few different kinds of loops, which are as follows: while, do-while, for, and a range-based for loop. A while loop will check a condition before starting the iteration, and will only enter if it passes. A do-while loop is similar to a while loop, however, instead of checking the condition before the code enters the loop, it checks it after. This ensures that no matter what the condition is, one iteration of the loop will always be performed. On the other hand, a for loop will only execute a certain number of times, unlike a while loop which operates until the condition is false. Additionally, for loops allow the programmer to initialize the starting number, set the condition on how many iterations to perform, and set the incrementation all in one statement. While there is no explicit foreach loop in C++, the same effect can be attained(see Code Related Questions 3d).

5. **Can you use break or continue statements (or something similar) to exit loops?**
   Break statements can be used to exit a loop. This can be implemented by checking for condition using an if statement within the block of the loop, and if comes back as true use a break statement to exit the loop.

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?**
   For switch statements in C++, you must use break to get out of a particular case. If this keyword is not implemented at the end of each case, all of the statements and cases that follow it will be implemented. This will continue until the program reaches another break statement or the end of the switch block. Additionally, continue statements can only be used in loops.

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? Make sure to include an example of it in your code as well.**
   There are control flow statements that have not been explicitly defined up to this point. These would be the range-based for loop and the goto statement. The range-based for loop is how foreach loops can be implemented in C++(see Discussion Questions 3d). It essentially iterates through each element in a given data type, such as a string or vector. The goto statement can be used to cause the code to jump to another place in the program, which is defined by a label. However, it is important to note that this is considered a low-level feature and is not used as much in high-level programming.

Reference:
http://www.cplusplus.com/doc/tutorial/control/