There is a plp_3.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

1. **Write a piece of code that creates a variable of each of these common data types and follows the naming conventions of C++(see Discussion Questions Part 1a):**
    a. Int:
    ```
    int my_integer = 7;
    ```
    b. String:
    ```
    std::string my_string = "My String";
    ```
    c. Floating-point number:
    ```
    float my_float = 2.5;
    ```
    d. Boolean:
    ```
    bool my_bool = false;
    ```
    e. array/list:
    ```
    std::vector<int> my_vector(7, 100);
    ```
    f. hash/dictionary:
    ```
    std::map<int, int> my_map;
    ```

2. **In your code, experiment with doing different things with the data types:**
    a. **Can you add ints and floats? If you do, is the resulting variable an int(narrowing conversion) or a float(widening conversion)?**
    You can add integers and floating-point numbers together in C++, however, the resulting variable depends entirely on the data type that is holding the result of the addition. If you store the result in an integer data type it will be a narrowing conversion. However, if you store the result in a floating-point data type it will be a widening conversion.

    b. **Can you put different data types in the same array or list?**
    It is not possible in C++ to put different data types in the same vector. This is due to the fact that when a vector is declared it is assigned the data type it will be holding. Thus, it can only hold that one data type.

    c. **Can one data type be converted to another (int to float, string to int, etc.)?**
    This depends entirely on the data type that you are trying to convert. For example, you can convert an int to a floating-point number, and vice versa, by using type-casting (see Discussion Questions Part 6). However, if you want to convert a string to an int you must import the string library and use the stoi (string to integer) method(see Discussion Questions Part 4). There is also the stof method, which does the same thing as stoi but with the floating-point number data type. Additionally, to convert an int or float to a string you could use the to_string method in the same string library.

    Resources:

http://www.cplusplus.com/doc/tutorial/typecasting/
http://www.cplusplus.com/reference/string/to_string/
http://www.cplusplus.com/reference/string/stoi/
http://www.cplusplus.com/reference/string/stof/

Discussion Questions

1. **What are the naming requirements for variables in your language?**
   There are no strict requirements for naming variables in C++, which means the compiler will not enforce variable names. However, there are some standards that have been developed by the programming community. When naming variables it is best to use a name that is descriptive. This will ensure that anyone reading your code will be able to understand what the code does. In general, abbreviations are disapproved. However, it is acceptable to use abbreviations that are commonly used such as num for number or i for incrementor. Additionally, variable names should be all lowercase. It is acceptable to use an underscore to separate words. If the variable is a class data member a trailing underscore should also be included in the name.
   a. **What about naming conventions? Are they enforced by the compiler/interpreter, or are they just standards in the community?**
      As stated above, naming conventions are not enforced by the compiler but are instead standards in the programming community. To begin, most everything follows the descriptive and abbreviation rules mentioned above. However, there are many more guidelines that extend to a variety of different program aspects. First, the name of the file itself follows the same conventions as variable names, with the addition that dashes are also permitted to separate the words along with the underscores. Next, Camel Case should be used to name types. Camel Case is when a capital letter is used for the first letter in each word. Data structs also follow the same guidelines as variables. When defining a constant, Camel Case should be used and a lowercase k should precede the whole name. Additionally, function names also use Camel Case. However, if the are accessors or mutators they must follow the conventions of variable names. Next, when defining namespaces, only lowercase should be used. Lastly, enumerators should follow the same naming conventions as constants.

   Reference:
   https://google.github.io/styleguide/cppguide.html#Naming

2. **Is your language statically or dynamically typed?**
   C++ is considered to be both a statically and dynamically typed language. However, it appears that it is more widely used as the former. A statically typed language requires that all of the data types of the program's variables be declared before the variables are used. In addition, in statically typed languages, type checking is done when the program is compiled. A dynamically typed language does not require that data types be declared before using the variable, and type checking is done at runtime.

   References:
   http://www.cplusplus.com/info/description/
   https://docs.oracle.com/cd/E57471_01/bigData.100/extensions_bdd/src/cext_transform_typing.html

3. **Is your language strongly or weakly typed?**
   C++ is a strongly typed language. This means that if a data type is specified, the data that is assigned to it must match that data type. This is due to the fact that the compiler will consider the data that is held to be of that specified data type, but if it is not the correct one an error will be generated. The purpose of these kinds of languages is to minimize errors, however, it becomes the burden of the programmer to ensure that the code they have typed is correct.

   References:
   http://www.cplusplus.com/info/description/
   https://www.programiz.com/terms/s/strongly-typed

4. **If you put this line, x = "5" + 6 (or something similar), in a program and try to print x, what does it do? If it does not compile, why? Is there something you can do to make it compile?**
   The code that I wrote to test this is:

```
int x;
x = "5" + 6;
std::cout << x;
```

   This code will not compile, which is due to the fact that C++ is a strongly typed language(see Discussion Questions Part 3). Since the variable x is of an integer data type and the code is trying to add a string to an integer, the compiler will not allow it. To fix this issue the code can instead be written like this:

```
int x = std::stoi("5") + 6;
std::cout << x;
```

   This code uses the stoi method from the string library(see Code Related Questions Part 2c) to convert the string into an integer, which then allowed the equation to be executed

and stored in the integer variable x.

5. **Describe the limitations (or lack thereof) of your programming language as they relate to the coding portion of the assignment (adding ints and floats, storing different types in lists, etc.). Are there other restrictions or pitfalls that the documentation mentions that you need to be aware of?**
   It is clear to see that C++ has a couple of limitations, which became evident in the code related section of this paper. First, vectors, which are equivalent to arrays in other languages, can only hold one data type. Second, the fact that standard type-casting cannot be done with strings is a little tedious since the string library must be imported and additional methods must be used. However, none of these things are great enough to completely deter the use of the language.

6. **How do type conversions work in your language? Are they narrowing or widening, and do they work by default or do they have to be declared by the programmer?**
   Type conversions can be done by using type-casting, which is the process of converting an expression that is of one data type to another data type. This can be done multiple ways in C++ depending on what data types are being converted and supports both narrowing and widening conversions(see Code Related Questions Part 2a and Part 2c for additional information).

   Source:
   http://www.cplusplus.com/doc/oldtutorial/typecasting/