Analysis of C and C++

Programming Languages 1st Homework

Bedirhan Ömer Aksoy

200104004074

C and C++

C

C is a general purpose programming language. It was developed in 1972 by Dennis M. Ritchie at the Bell Telephone Laboratories to develop the UNIX operating system. It is one of the most popular languages, because it is an essential language in computer science. C is a procedural programming language. It means that C is a language which has a set of functions, instructions and statements that must be executed in a defined order to perform a job or program.

Every C program has tokens; such as keywords (while, switch, break), variables, constants, punctuators ([, {, :, ;) and operators (!, *, ||, /). Semicolons are used to express the end statements in C language. The semicolon expresses that the statement has been terminated and the following statements are new statements. It prevents ambiguity and disorder while writing the code. C has several advantages and disadvantages.

C language offers plenty of advantages that makes it a primary building block for major programming languages. It is a fast and efficient language because it’s one of the closest programming languages to assembly. C provides programmers with a rich set of built-in and user defined functions with libraries.

C language, while widely used and respected, has its share of disadvantages. One significant weakness is its insufficiency of object-oriented programming (OOP) concepts. It can limit the structured organization and modularity of code. C don't enforce strict type checking and it leads to potential type-related errors that may not be noticed until runtime. C lacks checks for runtime errors, which can make debugging more challenging.

C++

C++ is an object oriented programming language which gives a clear structure to programs and allows code to be reused, lowering development costs. Danish computer scientist Bjarne Stroustrup began developing the new programming language shortly after joining the technical staff at Bell Laboratories in 1979. It is portable and usable to develop programs which can be adapt easily to multiple platforms.

The main aim of C++ language is to add the concept of object orientation to C programming. Object Oriented Programming is a model which provides many concepts, like inheritance, data binding and polymorphism. One of the main concepts of object oriented programming is classes. It is a user defined data type which holds its own data members and member functions. These features can define the properties and behavior of the objects of a class.

Both C and C++ have a similar syntax and code structure. They share the same basic syntax. Nearly all of C's operators and keywords are also present in C++ and do the same thing. But C++ has a slightly extended grammar than C. Same concepts of stack, heap, file-scope and static variables are present in both languages.

C++ is often called as a superset of C. Beside its similarities, there are some differences between these languages. C does not support polymorphism, encapsulation and inheritance which is because C supports procedural programming and doesn't support object oriented programming.

Approaches and Design Philosophies

C and C++ are very similar languages but they have some exact pattern and design philosophies which makes them different. Here are some of these main similarities and differences about their paradigms and design philosophies.

Similarities

- **Procedural Foundation:** Both C and C++ languages have strong basis in procedural programming. They support functions and structures which makes them usable for low level system programming and algorithm implementation.

- **Efficiency:** Both languages are designed to be efficient and gives user the control over memory and hardware. They offer users to optimize code for performance.

- **Portability:** The code written in C and C++ can often be compiled and run on various platforms with minimal modifications.

- **Strong Typing:** Both C and C++ employ strong typing. It means that users must specify the variable type when declaring the variables. This helps to catch the errors related with type at compile time.

Differences

- **Object Oriented Programming (OOP):** The main difference between these two related languages is C++'s strong emphasis on object oriented programming. It offers classes and objects which leads to creation of reusable, modular code. C is a procedural language and it doesn't support OOP.

- **Standard Template Library (STL):** C++ includes the Standard Template Library and it provides a wide set of templated classes and algorithms for common data structures and algorithms. C doesn't have an equivalent standard library for these data structures and algorithms.
- **Memory Management:** C and C++ makes memory management different. C relies on manual memory management with using functions like malloc() and free() functions. C++ offers the concept of constructors and destructors which makes memory management more convenient and safer.

- **Compatibility:** C has a high degree of compatibility with other programming languages and systems. C++ introduces features that can make it less compatible with pure C code and can require more careful integration.

- **Philosophy:** The design philosophy of C is building things as simply as possible, without any unnecessary thing. It is portable and it is a very close language to assembly. It is relatively easy to write a compiler for. The philosophy of C++ is to offer a powerful and flexible language which allows high performance, efficient and generic software development.

Keyword and Syntax Differences

Keyword Differences

Programming languages, such as C++ and C, have similar syntax and many of their keywords are the same. Both languages have similar code structure. But there are some keywords, a total of 30 keywords, in c++ that are not found in C. These are some of these keywords with their usage:

- **class** = Class in C++ is the main thing that provides the object-oriented programming. It is a user defined data type that contains its own data members and member functions. These are accessible and usable by created instances of that class.
- **friend** = Friend keyword can be used with classes or functions. These functions and classes which are declared as a friend can access private and protected members of other classes.
- **operator =** C language contains operators too, but this keyword provides users with special meaning to already existing operators in C++. This is called operator overloading and it is a compile time polymorphism.
- **namespace** = Namespace keyword provides the space that users can declare or define identifiers, such as variables, methods and classes in C++.
- **using** = Using keyword in C++ allows users to determine the use of a specific namespace. It provides flexibility to developers when working on big programs or libraries where there may be many different namespaces in use.
- **new** = New keyword in C++ is an operator. If sufficient memory is available on heap, it allocates the required bytes of memory and returns the address of this memory address to the pointer variable. In C, if users want to make dynamic allocations, **malloc()** and **calloc()** functions can be used.

- **delete =** It is an operator that is used to delete arrays and objects which are dynamically created by the previously described **new** operator. In C, if the user wants to delete the related memory block, **free()** function can be used.

Syntax Differences

C and C++ languages have very similar syntax because C++ is a (nearly) superset of C. Every keyword and syntax rule from C also exists in C++. But C++ has more features than C and these features require new syntax rules. These are some syntax rules of C++ with their usages which are don't exists in C:

- **Classes and Objects =** In C++, users can define classes and create their objects from them as described in the previous part. C language doesn't have this feature

```
// C++
class Fruits{
public:
        int seedCount;
}
Fruits apple;
```

- **Function Overloading =** C++ allows users to define functions with the names which are already other defined functions names with different parameter lists. C doesn't support this feature.

```
// C++
int count(int treeCount, int averageCount);
double count(int farmerCount, double pickRate);
```

- **Templates =** Templates enables users to create a function template that its functionality can be adapted to more than one type or class without repeating the entire code for each type in C++.

```
// C++
template <typename T>
T min(T a, T b) {
        return (a < b) ? a : b;
}
```

- **Reference Variables =** A reference variable in C++ is another name for a pre-existing variable. It seems similar to pointers but it has some different properties. A reference must be initialized when it is created and when it is initialized to an object, it cannot be changed to refer to another object.

```
// C++
int pickles = 7;
int &countOfPickles = pickles;
```

- **Exception Handling =** Exception handling in C++ allows users to handle and fix runtime errors and it enables more reliable programs. It contains try, catch and throw keywords.

```
// C++
try {
        if (x < 4) {
                throw x;
        }
}
catch ( int x) {
        cout <<  “Exception Detected” << endl;
}
```

Punctuation Similarities and Differences

C and C++ share many punctuation symbols due to C++ being a developed version of the C programming language. This similarity allows C++ to inherit and extend the foundational punctuation rules of C. Developers who are already familiar with C can easily transition to C++ since they can apply their knowledge when working in C++. Here are some common punctuation symbols that both languages use:

- **Semicolon(‘;’):** In C and C++, semicolon is used to terminate statements.
- **Comma(‘,’):** Used to separate items in a list of arguments in function calls.
- **Arrow(‘->’):** Used to access through pointers to structs.
- **Question Mark and Colon(‘?’ and ‘:’):** Used in ternary conditional operator.
- **Curly Braces(‘{‘ and ‘}’):** Used to define blocks of code, such as function bodies, loops, and conditional statements in both languages.
- **Operators:** C and C++ share a common set of operators for arithmetic (+, -, /, %), comparison (== , !=, <, >, <=, >=), logical (&&, ||, !) and assignment (=).

While these two languages share many same punctuation and syntax rules, there are some differences. These differences are essentially because of the additional features and capabilities added in C++. Here are some of these key punctuation differences between C and C++:

- **Standard Header Files Including:** In C, standard library headers are typically included using the <header.h> syntax, such as <stdio.h>.

- **Standard Input/Output:** C uses functions like ‘printf’ and ‘scanf’ for formatted input / output. C++ uses the stream insertion (‘<<’) and extraction (‘>>’) operators for input / output with objects like ‘cin and ‘cout’.

Semantic Similarities and Differences

C and C++ are related programming languages. Because of this, they share many semantic common features but C++ offers additional features and concepts. Here are some of the main semantic similarities and differences between C and C++.

- **Syntax**: C and C++ shares similar syntax. Both languages use curly braces to include code blocks “{ }”, semicolons to terminate the statements “;” and many common operators which are mentioned in previous parts.

- **Data Types:** Both languages use main data types like int, float, char and struct.

- **Functions:** In C and C++, functions are called and defined with similar way. They can accept parameters and return values. But C++ allows function overloading and users can have multiple functions with same name but different parameters.

- **Arrays:** Arrays in C and C++ are used with same syntax for declaration and indexing. C++ adds the capability to use structures which are similar to arrays called “vectors” from the STL.

- **Control Structures:** In both languages, control structures, such as if, else, for, while and switch, are common.

- **Input / Output:** C and C++ provide similar input/output functions like printf(), scanf() in C and cout, cin in C++ for console input and output.

- **Memory Management:** Both C and C++ gives control of memory allocation and deallocation to users. In C, users can allocate and deallocate memory with functions like malloc() and free(), in C++ with new and delete.

- **Classes and Objects:** C++ supports classes and objects. It allows programmers to create and use object oriented programming elements like polymorphism, inheritance and encapsulation.

C and C++ languages share a common base, but C++ extends the language with features on object oriented programming and additional libraries.

Efficiency of Languages

C and C++ known for their high performance capabilities and making a balance between low level control and high level abstraction. They offer developers an environment to create optimized codes for a wide range of applications. When it comes to comparing the efficiency of these languages, many similarities and some differences can be list here:

- **Execution Speed:** C and C++ can be equally efficient in raw execution speed. Because they both compile to machine code and the generated code can highly optimized.

- **Overhead:** C++ comes with some overhead due to features of object oriented programming tools. While this overhead is generally minimal, sometimes it can affect performance.

- **Memory Management:** C and C++ gives users a big control over memory. In C, users manage memory manually with functions like malloc() and free(). It can lead to memory leaks and errors if programmers don't use it carefully. In C++, users can allocate and deallocate memory with new and delete keywords easily without considering any issues, which can help in more efficient memory management.

- **Libraries and Frameworks:** C++ offers a wide range of libraries and frameworks such as STL. They can make development more efficient, but using these features can occur some runtime overhead. C provides fewer built-in libraries. It can lead to more manual coding but potentially less overhead.

- **Development Time:** C++ has more features and higher level language than C and it can make development faster in some cases. But if these features don't used efficiently, they can lead to overhead and complexity.

If users require low-level control and maximum efficiency, both languages gives similar performances. The efficiency of code written in both languages mostly depends on the qualification of the programmer and the design of the algorithms.

Learning Curves

C, often considered as the mother of all programming languages and it has a relatively easy learning curve. It is a compact programming language with a small amount of concepts. That makes it relatively easy for beginners to understand. The syntax is straightforward and it focuses on imperative programming. C doesn't have many high level abstractions and features. It might require users to write more code for hard tasks.

On the other hand, C++ offers a harder learning curve. It was built upon C and includes the principles of object oriented programming (OOP). While it inherits C syntax and concepts, it also offers a wide range of features like classes, templates and a complex standard library. These additional complexities can make C++ more challenging for starters, specifically who are not already familiar with the C.

In summary, the learning curve comparison about C and C++ mostly depends on the users past experience. For users who don’t have any programming experience before, C’s simplicity can offer an easier entry language and for users with a strong history in C or users who experienced in complex software projects, C++’s learning curve can be more smooth than the others. The choice between C and C++ should be selected by project requirements, personal preferences and development goals.