Project 2 - An Essay Project

**CHANG HUANG**

**Global scope** A global name is one that is declared outside of any class, function, or namespace. However, in C++ even these names exist with an implicit global namespace. The scope of global names extends from the point of declaration to the end of the file in which they are declared. For global names, visibility is also governed by the rules of [linkage](https://learn.microsoft.com/en-us/cpp/cpp/program-and-linkage-cpp?view=msvc-170) which determine whether the name is visible in other files in the program.

**Function scope** A [label](https://learn.microsoft.com/en-us/cpp/cpp/labeled-statements?view=msvc-170) has function scope, which means it is visible throughout a function body even before its point of declaration. Function scope makes it possible to write statements like goto cleanup before the cleanup label is declared.

A stack frame is a frame of data that gets pushed onto the stack. In the case of a call stack, a stack frame would represent a function call and its argument data.

If I remember correctly, the function return address is pushed onto the stack first, then the arguments and space for local variables. Together, they make the "frame," although this is likely architecture-dependent. The processor knows how many bytes are in each frame and moves the stack pointer accordingly as frames are pushed and popped off the stack.

A default argument is a value provided in a function declaration that is automatically assigned by the compiler if the calling function doesn’t provide a value for the argument. In case any value is passed, the default value is overridden.

Pass-by-reference means to pass the reference of an argument in the calling function to the corresponding formal parameter of the called function. The called function can modify the value of the argument by using its reference passed in.

The scope resolution operator can be used as both unary and binary. You can use the unary scope operator if a namespace scope or global scope name is hidden by a particular declaration of an equivalent name during a block or class.

Function overloading is a feature of object-oriented programming where two or more functions can have the same name but different parameters. When a function name is overloaded with different jobs it is called Function Overloading. In Function Overloading “Function” name should be the same and the arguments should be different. Function overloading can be considered as an example of a [polymorphism](https://www.geeksforgeeks.org/polymorphism-in-c/) feature in C++.

A template is a simple yet very powerful tool in C++. The simple idea is to pass data type as a parameter so that we don’t need to write the same code for different data types. For example, a software company may need to sort() for different data types. Rather than writing and maintaining multiple codes, we can write one sort() and pass data type as a parameter.

C++ adds two new keywords to support templates: ‘template’ and ‘typename’. The second keyword can always be replaced by the keyword ‘class

1.Time Complexity: Finding the Time complexity of Recursion is more difficult than that of Iteration:

Recursion: Time complexity of recursion can be found by finding the value of the nth recursive call in terms of the previous calls. Thus, finding the destination case in terms of the base case, and solving in terms of the base case gives us an idea of the time complexity of recursive equations. Please see Solving Recurrences for more details.

Iteration: Time complexity of iteration can be found by finding the number of cycles being repeated inside the loop.

2.Usage: Usage of either of these techniques is a trade-off between time complexity and size of code. If time complexity is the point of focus, and number of recursive calls would be large, it is better to use iteration. However, if time complexity is not an issue and shortness of code is, recursion would be the way to go.

Recursion: Recursion involves calling the same function again, and hence, has a very small length of code. However, as we saw in the analysis, the time complexity of recursion can get to be exponential when there are a considerable number of recursive calls. Hence, usage of recursion is advantageous in shorter code, but higher time complexity.

Iteration: Iteration is repetition of a block of code. This involves a larger size of code, but the time complexity is generally lesser than it is for recursion.

3.Overhead: Recursion has a large amount of Overhead as compared to Iteration.

Recursion: Recursion has the overhead of repeated function calls, that is due to repetitive calling of the same function, the time complexity of the code increases manyfold.

Iteration: Iteration does not involve any such overhead.

4.Infinite Repetition: Infinite Repetition in recursion can lead to CPU crash but in iteration, it will stop when memory is exhausted.

Recursion: In Recursion, Infinite recursive calls may occur due to some mistake in specifying the base condition, which on never becoming false, keeps calling the function, which may lead to system CPU crash.

Iteration: Infinite iteration due to mistake in iterator assignment or increment, or in the terminating condition, will lead to infinite loops, which may or may not lead to system errors, but will surely stop program execution any further.

Smart pointers are defined in the std namespace in the <memory> header file. They are crucial to the RAII or Resource Acquisition Is Initialization programming idiom. The main goal of this idiom is to ensure that resource acquisition occurs at the same time that the object is initialized, so that all resources for the object are created and made ready in one line of code. In practical terms, the main principle of RAII is to give ownership of any heap-allocated resource—for example, dynamically-allocated memory or system object handles—to a stack-allocated object whose destructor contains the code to delete or free the resource and also any associated cleanup code.

uses of static in variable declaration, in function

declaration, in class variable/member function declaration

Range-based for loop in C++ is added since C++ 11. It executes a for loop over a range. Used as a more readable equivalent to the traditional for loop operating over a range of values, such as all elements in a container.The auto keyword is **a simple way to declare a variable that has a complicated type**. For example, you can use auto to declare a variable where the initialization expression involves templates, pointers to functions, or pointers to members.

The principle of least privilege is **the idea that at any user, program, or process should have only the bare minimum privileges necessary to perform its function**.