Sure! Here's an expanded and updated list of topics, organized in a logical order for a beginner to pro-level learning journey:

1.      Introduction to C++

         Overview of C++ and its history

         Installing a C++ compiler and development environment

         Basic structure of a C++ program

         Hello World program

2.      Variables and Data Types

         Fundamental data types (int, float, double, char, bool)

         Declaration and initialization of variables

         Constants and literals

         Scope and lifetime of variables

         Type casting and conversions

3.      Operators and Expressions

         Arithmetic operators

         Comparison operators

         Logical operators

         Bitwise operators

         Operator precedence and associativity

         Control structures (if-else, switch-case, loops)

4.      Functions and Program Structure

         Functions and function prototypes

         Passing arguments by value and by reference

         Function overloading

         Recursion

         Header files and namespaces

         Program structure and organization

5.      Arrays and Strings

         Arrays and multidimensional arrays

         Strings and string operations

         Pointers and dynamic memory allocation

         Pointers and arrays

6.      Object-Oriented Programming

         Classes and objects

         Constructors and destructors

         Encapsulation and data hiding

         Inheritance and polymorphism

         Virtual functions and abstract classes

         Templates and generic programming

7.      Input and Output

         Input and output streams

         File input and output

         Manipulators and formatting

         Exception handling

8.      Standard Template Library

         Containers (vector, list, stack, queue, set, map)

         Algorithms (sorting, searching, etc.)

         Iterators and algorithms

9.      Memory Management

         Memory allocation and deallocation

         Smart pointers

         Memory leaks and memory corruption

10. Multithreading and Concurrency

         Threads and thread management

         Synchronization and mutual exclusion

         Deadlocks and livelocks

         Parallel programming and data race conditions

11. Networking and Sockets

         Client-server architecture and communication

         Sockets and socket programming

         HTTP and web services

         Network security and encryption

12. Graphics and User Interfaces

         Graphics and rendering

         User interface design and widgets

         GUI toolkits and frameworks

         Game development and 3D graphics

13. Advanced Topics

         C++11/14/17/20 features and standards

         Performance optimization techniques

         Code profiling and debugging

         Software design patterns and architectures

         Embedded systems programming

14. Practice and Projects

         Practice coding exercises and challenges

         Work on small programming projects to apply your knowledge and develop your skills

         Participate in coding contests and hackathons

Remember that the order and topics covered can vary based on your individual learning needs and goals. This is just a general guideline for a beginner to pro-level C++ learning plan. Start with the basics, and gradually build up your knowledge and skills by practicing and building projects. Good luck!

Learning C++ from beginner to pro level can be a challenging task, but it's definitely possible with dedication and persistence. Here are some steps you can take to start learning C++:

1.      Learn the basics: Start by learning the basics of C++, such as variables, data types, operators, control structures, functions, and arrays. There are many online resources available for learning the basics of C++, including websites, tutorials, and online courses.

2.      Practice programming: Once you have learned the basics of C++, start practicing programming by writing small programs. Practice is essential for learning any programming language, as it helps you to apply what you have learned and develop your programming skills.

3.      Learn object-oriented programming: C++ is an object-oriented programming language, so it's important to learn the principles of object-oriented programming (OOP) such as classes, objects, inheritance, and polymorphism.

4.      Read books: There are many books available on C++ that can help you learn the language in depth. Some recommended books include "C++ Primer" by Stanley B. Lippman, "The C++ Programming Language" by Bjarne Stroustrup, and "Effective C++" by Scott Meyers.

5.      Practice advanced programming concepts: After you have learned the basics of C++ and OOP, you can start learning more advanced programming concepts such as templates, exceptions, multithreading, and networking.

6.      Work on projects: Working on projects is a great way to apply what you have learned and gain practical experience in C++. Try building small projects, such as games, utilities, or web applications, to improve your programming skills.

7.      Join online communities: Join online communities such as forums, chat rooms, and social media groups to connect with other C++ developers and learn from their experiences. You can also get feedback on your projects and ask questions about C++.

Remember that learning C++ takes time and practice. Be patient, persistent, and keep learning new things to improve your programming skills. Good luck!