The C++ Cheetsheet:

C++ Specific Concepts:

A

- **`auto`:** A keyword that tells the compiler to automatically deduce the type of a variable from its initializer. It makes code cleaner and less repetitive.
- **Algorithms (STL):** A collection of functions in the Standard Template Library for performing common operations on sequences of objects, like `std::sort`, `std::find`, and `std::accumulate`.

C

- **`class`:** The fundamental building block of Object-Oriented Programming (OOP). It bundles data (member variables) and functions (member methods) that operate on that data.
- **`const`:** A qualifier that specifies a variable's value cannot be changed. Using it correctly (`const` correctness) improves performance and makes code safer and easier to understand.
- **Constructor:** A special member function of a class that is automatically called when an object of that class is created. It initializes the object's data.
- **Container (STL):** An object that holds a collection of other objects (elements). Key examples are `std::vector`, `std::map`, and `std::unordered_map`.

D

- **Destructor:** A special member function that is automatically called when an object is destroyed. It is used to release resources (like memory) that the object acquired.
- **`delete`:** The operator used to deallocate memory that was allocated from the heap with the `new` operator. Using smart pointers is now preferred over manual `delete`.

E

- **Encapsulation:** The OOP principle of bundling data and the methods that operate on that data within a single unit (a class) and restricting direct access to some of the object's components.

H

- **Header File (`.h` or `.hpp`):** A file containing C++ declarations (like function prototypes and class definitions) intended to be shared across multiple source files using the `#include` directive.
- **Heap:** The region of your computer's memory where dynamically allocated memory resides (i.e., memory allocated with `new`). It is larger but slower to access than the stack.

I

- **Inheritance:** The OOP mechanism by which a new class (derived class) can acquire the properties and behaviors of an existing class (base class).
- **`inline`:** A hint to the compiler to insert the code of a function directly at the call site instead of performing a function call, potentially improving performance for very short functions.
- **Iterator:** An object that enables traversal through the elements of a container (like a `std::vector`). It acts like a generalized pointer.

L

- **Lambda Expression:** A concise way to define an anonymous function object right at the location where it is invoked or passed as an argument. Heavily used in modern C++.

M

- `map` **(STL):** A standard container that stores key-value pairs, keeping the keys sorted. It provides fast lookup times (logarithmic).
- **Move Semantics (`&&`):** A C++11 feature that allows resources to be "moved" from one object to another instead of being copied, providing a major performance boost by avoiding expensive allocations.

N

- `namespace`**:** A feature used to organize code into logical groups and to prevent name collisions that can occur when your code includes multiple libraries.
- `new`**:** The operator used to dynamically allocate memory on the heap. It returns a pointer to the allocated memory.

O

- **Object-Oriented Programming (OOP):** A programming paradigm based on the concept of "objects," which can contain data and code. Its core principles are encapsulation, inheritance, and polymorphism.
- **Operator Overloading:** The ability to provide a special meaning for an operator (like `+`, `-`, or `*`) for a user-defined type (a class). For example, defining addition for two matrix objects.

P

- **Pointer (`*`):** A variable that stores the memory address of another variable. Essential for dynamic memory allocation and high-performance programming.
- **Polymorphism:** The OOP principle that allows you to treat objects of different derived classes as objects of a common base class, most often achieved through virtual functions. It means "many forms."

R

- **RAII (Resource Acquisition Is Initialization):** A core C++ programming idiom where resource management (memory, files, locks) is tied to the lifetime of an object. The resource is acquired in the constructor and released in the destructor. This is the principle that makes smart pointers work.
- **Reference (`&`):** An alias for an already existing variable. It's often used for passing function arguments efficiently to avoid making a copy of the data.
- **Range-based for loop:** A modern C++ feature that provides a simpler, cleaner syntax for iterating over all the elements of a container.

S

- **Smart Pointers:** A C++11 feature. Objects that act like pointers but automatically manage the memory they point to, deallocating it when it's no longer needed. They help prevent memory leaks. The main types are `std::unique_ptr`, `std::shared_ptr`, and `std::weak_ptr`.
- **Stack:** The region of memory where local variables are stored. It is very fast but limited in size. Memory is automatically allocated and deallocated as functions are called and return.

- **`static`:** A keyword with multiple meanings, but commonly used to create a class member that is shared by all instances of the class, rather than each instance having its own copy.
- **STL (Standard Template Library):** The heart of modern C++. A library of template-based containers, algorithms, and iterators that provides common programming data structures and functions.
- **`struct`:** Almost identical to a `class`, but its members are `public` by default, whereas a `class`'s members are `private` by default. Often used for simple data aggregation.

T

- **Template:** A powerful C++ feature that allows you to write generic code that works with any data type. For example, you can write a single sorting function template that works for vectors of integers, doubles, or custom objects. This is key for writing high-performance, reusable libraries.

U

- **`unordered_map` (STL):** A standard container that stores key-value pairs using a hash table. It provides, on average, faster lookups (constant time) than `std::map` but does not keep the elements sorted.

V

- **`vector` (STL):** The most commonly used container. A dynamic, contiguous array that can grow or shrink in size as needed.
- **`virtual` Function:** In OOP, a member function in a base class that you expect to be redefined in derived classes. It is the mechanism that enables polymorphism at runtime.