**Class**: user-defined type, create object structure **Linkage**: external-extern(connect var across scope in different modules), internal-static(same but within its own modules) and non-existent(no linkage exists) **Storage Duration**: automatic(from declare to end of scope), static(entire lifetime of program), dynamic(create using new and end when deallocate with delete[]), thread-thread\_local(last lifetime of thread) **Memory Segment**: code(store program instructions), data(data survives lifetime of program), stack(store local data statically allocated), heap(stores local dynamically allocated) **Pre-processing**->**Complication-**>**Linking** **Pointer**: hold address in memory, required \*, generic(void) can hold address of any type, different type passing value caused error **References**: required &, similar to pointers **lvalue**: has identity, modifiable (if not const), refer to an object(ex: int x or const int x = 5) **rvalue**: opposite of lvalue, temp object **Copy Constructors**: copying current object to newly created object with the same type **Assignment Operator**: defines logic of copying data from constructor **Move Operators**: use rvalues and make shallow copy (only value) over current object **Enumerated Types**: composed of symbolic constants (class::type), scoped value created using class declaration, underlying type is integer **Inheritance**: Derived Class: status Base Class (Base own Derived) **Pure Virtual**: declared in base class as virtual something = 0; but will be implemented later by another subclass (derived class) **glvalue**: has identity, evaluates object or function **rvalue**: movable value, temporary object/subobject, value not associated with object **prvalue**: has no identity, movable **Recursion**: call the function itself within the body function **Lambda Expression**: passing a function as an object **Catching Exceptions**: can catch using multiple handlers, each handler can handle multiple exceptions **noexcept**: optimize machine code, exception is handled internally, terminates if uncaught exception thrown **Normal Exits**: destroy object with thread storage duration, current thread, static storage, close all C streams **Containers**: shells of data structure, manages the memory, provides member functions to access elements **Iterators**: data structure traversal, various means of accessing elements, **Algorithms**: services on data using iterators and functors  **Sequential Containers**: array(storage of fixed size)->vector(storage of variable size)->deque(variable size, double ended queue)->forward\_list->list





