**built-in types** - the type is called built-in when the compiler knows how to represent objects of this type and which operations can be applied to it without declarations written by the programmer

**class** - a user-defined type that can contain function members, data members and member types (nested classes)

**const** - used to indicate that a function can be called for the const object

**constructor** - a member function with the same name as class used to initialize the object

**destructor** - an operation that is implicitly invoked when the object is destroyed. Often, it releases resources.

**enum** - **enumeration -** is a simple user-defined type that holds set of symbol constants as its values

**enumerator** - symbol constant, a value of enum

**helper function** - a function that can be defined outside the class and therefore doesn’t interrupt with the data of the class

**implementation** - a part of the class that is not available directly to the user

**in-class initializer** - an initializer specified as a part of member’s declaration

**inlining** - a function that is defined within the class. The code of it is being generated every time the compiler sees the point of call.

**interface** - a part of class declaration used to specify how this class can be used

**invariant** - a statement that has to be true at the given point of the program

**representation** - A “type” knows how to represent data needed in the object

**struct** - are primarily used for the structures where the members can take any value, thus struct doesn’t support invariants

**structure** - A particular way of organizing data in computer so it can be used efficiently

**user-defined types** - not build-in types that are being implemented by the programmer

**valid state** - state is a value, and if the value is valid it called valid state