* **Structures**
  + A structure is a collection of one or more variables, possibly of different types, grouped together under a single name for convenient handling.
  + Structures help to organize complicated data, particularly in large programs, because they permit a group of related variables to be treated as a unit instead of as separate entities.
  + The keyword struct introduces a structure declaration, which is a list of declarations enclosed in braces.
  + Structures are just like classes in other languages. Where all the parameters that need to be accessed through the object of a class. Similarly, the parameters in the structure accessed by the variable with which the structure is defined.
  + Arrays of structures: Each element of an array is a structure.
  + Structures can be accessed using pointer variables. To access members of the structure by using the pointer then we use a **->** symbol.
  + Structures cannot be defined inside another structure.
* **Union**
  + A union is a variable that may hold {at different times> objects of different types and sizes, with the compiler keeping track of size and alignment requirements.
  + Unions provide a way to manipulate different kinds of data in a single area of storage, without embedding any machine-dependent information in the program.
  + Unions are just like structures. The memory allocated to a structure is the sum all the data types that are present in the structure but in case of union, the memory would be the memory of the largest data type on the union.
  + In structure, if we change the value of one attribute then that would not affect the values of other parameters but when the value of one parameter is changed in the union it may affect the value of other parameters.
  + Just like structures, we can access the members of the union using the **arrow(->).**
  + Unions can be useful in many situations where we want to use the same memory for two or more members.
* **Typedef and Define**
  + Typedef is used to give a datatype a name.
  + Using typedef we can create our own data types that can be used in the program.
  + #define is used to define a particular value or a variable as an alias.
  + typedef is limited to giving symbolic names to types only, whereas #define can be used to define an alias for values as well, e.g., you can define 1 as ONE, 3.14 as PI, etc.
  + typedef interpretation is performed by the compiler where #define statements are performed by the preprocessor.
  + #define should not be terminated with a semicolon, but typedef should be terminated with a semicolon.