C++ Programming – Lecture 1

C++ Basics

- Every C program is a valid C++ program too
- Different versions of C++ are C++98, C++11, C++14 and C++17
- C++ has 47 keywords and 51 operators

C++ Types

- C++ pre-defined or Primary types:

  char - signed, unsigned
  int - signed, unsigned, short, long
  real - float, double, long double
- C++ user-defined or Secondary or Derived types

  Pointer
  Array, String
  Structure
  Union
  Enum
  Class
- Purpose of different user-defined types

  Pointer : To access/store value at a memory location
  Array, String : To store collection of similar elements
  Structure : To store collection of dissimilar elements
  Union : To share same memory locations
  Enum : To enumerate numerical data conveniently
  Class : To combine data and functions that operate on them together

Structures

- Structure is a collection of dissimilar (usually) elements stored in adjacent locations
- Terminology :

  struct employee { char name ; int age ; float salary ; } ;
  struct employee e1, e2, e[10] ;

  struct - Keyword employee - Structure name / tag
  name, age, salary - Structure elements / Structure members
  e1, e2 - Structure variables e[ ] - Array of structures
- Structure elements are stored in adjacent memory locations

- Size of structure variable = sum of sizes of structure elements
- 2 ways to copy structure elements :

  ```
  struct  emp  e1 = { "Rahul", 23, 4000.50 } ;
  struct  emp  e2,  e3 ;
  e2.n = e1.n ;  e2.a = e1.a ;        e2.s = e1.s ;  → Piecemeal copying
  e3 = e1 ;   → Copying at one shot
  ```

- Structures can be nested :

  ```
  struct  address { char  city[ 20 ] ; long  int  pin ; } ;
  struct emp { char n[ 20 ] ;  int age ;  struct address  a ;  float  s ; } ;
  struct emp e ;
  ```

  To access city and pin we should use e.a.city and e.a.pin
- To access structure elements using structure variable, use . operator as in

  ```
  struct emp e ;  cout <<  e.name << e.age << e.sal ;
  ```

- To access structure elements using structure pointer, use -> operator as in

  ```
  struct emp e ;  struct emp *p ;
  p = &e ;
  cout << p->name << p->age << p->sal ;
  ```

Unions

- Size of union variable is size of biggest element of the union. Elements are accessed using .
- Utility of union - Permits access to same memory locations in multiple ways
- Usage :

  ```
  union a
  {
      int  i ;  char  ch[ 4 ] ;
  } ;
  union a z ;
  z.i = 512 ;
  cout << z.i << z.ch[0] << z.ch[1]  << z.ch[2] << z.ch[3] ;
  ```

- If a number is ABCD then in little endian architecture it is stored as DCBA
- Little Endian - Low byte is stored first. Big Endian - High byte is stored first. Endianness is a matter of convenience. So both are good

Enums

- Often we are required to handle an ordered listing of items. Example, colors like red, green, blue or marital status like married, unmarried or divorced. Instead of handling these as integers, enums are a better way.
- Usage of enums :

```
enum color { red, green, blue } ;
enum  color  windowcolor, buttoncolor ;
windowcolor = green ; buttoncolor = blue ;
cout << windowcolor << buttoncolor ;
```

- While defining structure, union or enum variable there is no need to use the keywords struct, union or enum

Programming Paradigms

- Programming paradigm indicates how a program is organized
- Structured programming is based on interaction of functions
- OO programming is based on interaction of objects

Classes and Objects

- A class is a user-defined type on the basis of which objects are created
- Classes indicate how the objects created from them would look like
- An object contains specific data values and functions that can access or and/or manipulate them
- Data members and member functions are encapsulated in an object
- Data hiding means denying direct access to data from outside the object
- Data values in objects are often called instance data or state of the object
- In principle every object has instance data and member functions
- In practice each object has instance data, whereas member functions are shared amongst objects
- Sharing is justified as from one object to another member functions are going to remain same

Defining Classes, Creating objects

- public members of a class are accessible from outside the class
- private members of a class are NOT accessible from outside class

- Within a class any member can access any other member
- By default class members are private. By default structure members are public
- Usually data in a class is kept private and the data is accessed / manipulated through public member functions of the class
- Public member functions of a class can be accessed using the . operator through the syntax object.function( )

cout and cin

- cout is an object of ostream class. It is used for sending output to screen
- endl is used send '\n' to the screen
- << is an insertion operator
- cin is an object of istream class. It is used for receiving input from keyboard
- >> is an extraction operator
- <<, >> can be cascaded
- ostream and istream classes are declared in iostream header file
- cout and cin objects are defined in a namespace called std
- cin and cout are better than printf( ) and scanf( ) as there is no need to remember and use the format specifiers

Namespaces

- Namespace is a container for related classes and objects
- To use cout and cin, istream file must be included and a using namesapce statement should be used at the beginning of the program
- If using namespace statement is not used then cout and cin must be prefixed with std::