Classification of Languages

1. *Procedure Oriented Programming Language*
   FORTRON is 1st POP Language
   e.g. : C, FORTRON & PASCAL

2. *Object Oriented Programming Language*
   Simula is 1st OOP Language in 1960
   e.g.: C++, Smalltalk, Java and C# . Smalltalk is only language which is purely OOP Language. C++ is not purely object oriented programming language. it is also called as partial object oriented programming language

3. *Object Based Programming Language*
   Ada is 1st object based language.
   e.g.: visual basic, Ada &Modula-2

4. Rule Based Programming Language
 e.g.: PROLOG and LISP

Any New language is basically designed for two reasons
1. To overcome or avoid limitations of previous language
2.To provide new features
Advantages of C Language
1. C is portable
2. C is efficient
– can interact with hardware efficiently
3. C is flexible
– we can cerate application software or system software
4. C is freely available
–wide variety of compliers are available

- **C is said to be procedure oriented, structured programming language.**
- **When program becomes complex, understating and maintaining such programs is very difficult.**

- ***Limitations of C Programming with respect to C++***
- **Language don’t provide security for data.**
- **Using functions we can achieve code re-usability, but re-usability is limited. The programs are not extendable.**
- **We can not write function inside structure**

So “Bjarne Stroustrup” designed a new language c with classes in 1979 on DEC PDP11 machine. Restructure by ANSI in 1983.

in C++ 63 Keywords are available.
(unmanaged c++)

- ***Procedure oriented***
- **Emphasis on steps or algorithm**
- **Programs are divided into small code units i.e. functions**
- **Most functions share global data & can modify it**
- **Data move from function to function**
- **Top-down approach**

- ***Object Oriented***
- **Emphasis on data of the program**
- **Programs are divide into small data units i.e. classes**
- **Data is hidden & not accessible outside class**
- **Objects communicate with each other**
- **Bottom- up approach**

Sunbeam Infotech

Variable declaration

- In C, variable should be declared at the start of the block.
- This restriction is removed in C++. We can declare the variables anywhere in function.

Data types

- **C++ supports all data types provided by C language. i.e. int, float, char, double, long int, unsigned int, etc.**
- **C++ add two more data types:**
- **1. *bool* :- it can take *true* or *false* value. It takes one byte in memory.**
- **2. wchar_t :- it can store 16 bit character. It takes 2 bytes in memory.**

Comments in C++

- In C, comments are written as
- /*This is comment*/
- In C++, we can use above style. In addition C++ provides one more way for writing comments.
- //This is comment
- The second style is preferred for single line comments.

Structure

- **Structure is a collection of similar or dissimilar data. It is used to bind logically related data into a single unit.**
- **This data can be modified by any function to which the structure is passed**
- **Thus there is no security provided for the data within a structure.**
- **This concept is modified by C++ to bind data as well as functions.**

Diff Between struct in c & c++

- structure in c
- We can’t write function inside structure
- At the time of creating variable of structure writing struct keyword is compulsory

  eg. struct time t;
- By default all the members are accessible outside structure. C lang does not have a concept of access specifies
- If we want to call any function on structure variable

  struct time t1;

  input(&t1); print(t1);

- structure in c++
- We can write function inside structure
- At the time of creating object of structure writing struct keyword is optional
- eg.   time t;
- By default all members of struct in c++ are public (we can make them private)
- If we want to call member function on object.
-        time t1;
-        t1.input(); t1.print();

struct time {
        int hr, min, sec;
};
void input(struct time *p)
 {
printf(“Enter Hr Min Sec:”);
scanf(“%d%d%d”, &p→hr, &p→min,
        &p→sec);
}
 	
struct time t;
input(&t);
	struct time
	 {
	        int hr, min, sec;
	        void input()
	        {
	        printf(“Enter Hr Min Sec::”);
	        scanf(“%d%d%d”,&this→hr,
	        &this→min, &this→sec);
	        }
	};
	time t;
	t.input();

Demo structure in c

```c
#include<stdio.h>
#pragma pack(1) // slack bytes
struct student
{
	int rollno;
	char name[10];
	float per;
};
void accept_stud_info(struct student* s);
void display_stud_info(const struct student *s);
int main(void)
{
	struct student s1;
	printf("\n enter student info::");
	accept_stud_info(&s1);
	printf("student info :: \n");
	display_stud_info(&s1);
	return 0;
}
```

Demo structure in c

```
void accept_stud_info(struct student *s)
{
    printf("\n enter rollno::");
    scanf("%d", &s->rollno);
    printf("\n enter name::");
    scanf("%s", s->name);
    printf("\n enter per::");
    scanf("%f", &s->per);
    return;
}
void display_stud_info(const struct student *s)
{   // s->per=0;  s is constant
    printf("\n rollno  name  per \n");
    printf("%-5d%-10s%6.2f", s->rollno, s->name, s->per);
    printf("\n%-5d%-10s%6.2f",(*s).rollno, (*s).name,(*s).per);
    return;
}
```

Demo structure in cpp

```
#include<stdio.h>
#pragma pack(1) // slack bytes
struct student
{
    private:  // variable // data member  // field
        int rollno;
        char name[10];
        float per;

    public:
       void accept_stud_info()
       {
              printf("\n enter rollno::");
              scanf("%d", &rollno);
              printf("\n enter name::");
              scanf("%s", name);
              printf("\n enter per::");
              scanf("%f", &per);
              return;
       }
```

Demo structure in cpp

```
    void display_stud_info()
    {
        printf("\n rollno  name  per \n");
        printf("%-5d%-10s%6.2f", rollno, name, per);
        printf("\n\n\n");
       return;
    }
};
int main(void)
{
    student s1;//struct student s1;
    printf("\n enter student info::");
    s1.accept_stud_info();  //accept_stud_info(&s1);
    //s1.per=45;
    printf("student info :: \n");
    s1.display_stud_info(); //display_stud_info(&s1);
    return 0;
}
```

Access specifiers

- **By default all members in structure are accessible everywhere in the program by dot(.) or arrow(→) operators.**

- **But such access can be restricted by applying access specifiers**

  - **private: Accessible only within the struct**
  - **public: Accessible within & outside struct**