What is structure?
==============

1. A structure is a user-defined data type in C/C++.
2. A structure creates a data type that can be used to group items of possibly different types into a single type.

3. The ‘struct’ keyword is used to create a structure. The general syntax to create a structure is as shown below:

```
struct structureName{
    member1;
    member2;
    member3;
    .
    .
    .
    memberN;
};
```

Structures in C++ can contain two types of members:

Data Member: These members are normal C++ variables. We can create a structure with variables of different data types in C++.

Member Functions: These members are normal C++ functions. Along with variables, we can also include functions inside a structure declaration.

How to Declare the Structure Variable ?
==================================
A structure variable can either be declared with structure declaration or as a separate declaration like basic types.

```
// A variable declaration with structure declaration.
struct Point
{
int x, y;
} p1; // The variable p1 is declared with structure 'Point'
//=====================================================
struct Point
{
int x, y;
} p2,p3,p4; // We can declare the multiple variable
//=====================================================
struct Point
{
int x, y;
};
int main()
{
struct Point p1; // The variable p1 is declared like a normal variable
}
```
==

Example:

```
#include <iostream>
using namespace std;
struct Name{
    string fname;
    string lname;
}n1;

int main()
{
    struct Name n2;
    n1.fname="vikas";
    n1.lname="singh";
    n2.fname="ravi";
    n2.lname="singh";
    // Structure members can also be initialized using curly braces ‘{}’.
    struct Name n3={"prashant","singh"};

    // Structure members are accessed using dot (.) operator.
    cout<<n1.fname<<" "<<n1.lname<<endl;//vikas singh
    cout<<n2.fname<<" "<<n2.lname<<endl;//ravi singh
    cout<<n3.fname<<" "<<n3.lname<<endl;//prashant singh
    return 0;
}
```
==

Array of Structure
==============
```
#include <iostream>
using namespace std;
struct Data{
    int val1;
    int val2;
};
int main()
{
    Data arr[10];//Array of the struct type

    for(int i=0;i<10;i++)
    {
        arr[i].val1=i;
        arr[i].val2=i*10;
    }
    for(int i=0;i<10;i++)
    {
        cout<<arr[i].val1<<"-"<<arr[i].val2<<endl;
    }
    return 0;
}
```
Output
```
0-0
1-10
..
9-90
```
==

What is a structure pointer?

=======================

Like primitive types, we can have pointer to a structure.
If we have a pointer to structure, members are accessed using the arrow (->) operator instead of the dot (.) operator.

```
#include <iostream>
using namespace std;
struct Data{
    int val1;
    int val2;
};
```

```
int main()
{
    Data d1={10,20};
    Data *p=&d1;
    cout<<p->val1<<"-"<<p->val2<<endl;//10-20
    return 0;
}
```
===

Self Referential structure

=====================

1. Self Referential structures are those structures that have one or more pointers which point to the same type of structure, as their member.
2. In other words, structures pointing to the same type of structures are self-referential in nature
3. Types of Self Referential Structures
    1. Self Referential Structure with Single Link ⇒ These structures can have only one self-pointer
    2. Self Referential Structure with Multiple Links ⇒ These structures can have multiple self-pointer

Self Referential Structure with Single Link
