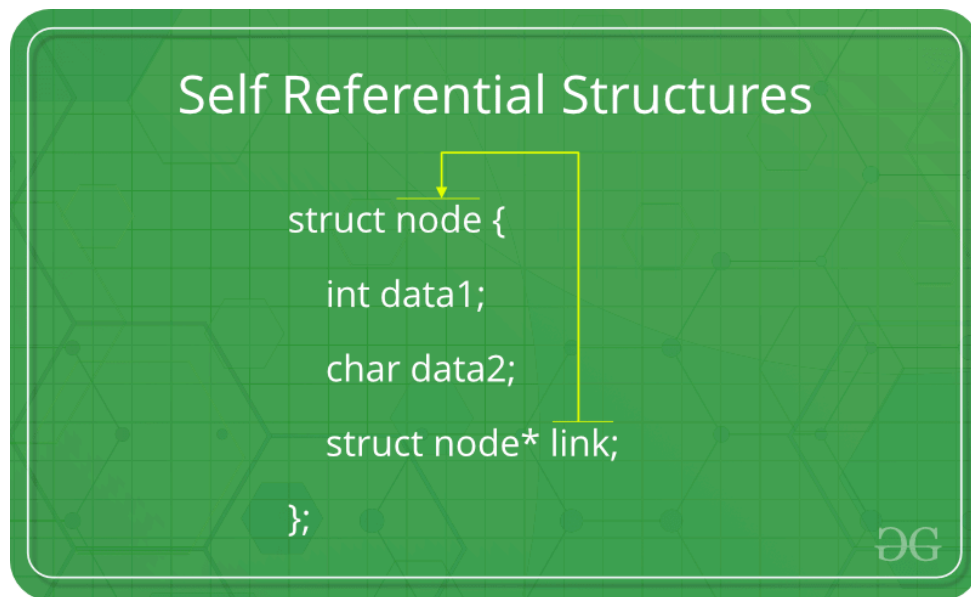


Self Referential Structures

Last Updated : 11 Jul, 2025

Self Referential structures are those [structures](#) that have one or more pointers which point to the same type of structure, as their member.



In other words, structures pointing to the same type of structures are self-referential in nature

Example:

CPP Java Python JavaScript

```
1 struct node {  
2     int data1;  
3     char data2;  
4     struct node* link;  
5 };  
6  
7 int main()  
8 {  
9     struct node ch;
```

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our [Cookie Policy](#) & [Privacy Policy](#).

Got It !

```
11 }
```



No Output

Output

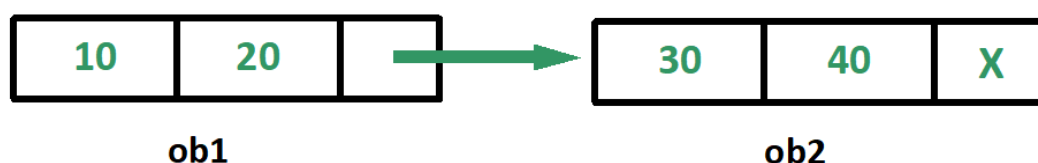
In the above example 'link' is a pointer to a structure of type 'node'. Hence, the structure 'node' is a self-referential structure with 'link' as the referencing pointer.

An important point to consider is that the pointer should be initialized properly before accessing, as by default it contains garbage value.

Types of Self Referential Structures

1. Self Referential Structure with Single Link
2. Self Referential Structure with Multiple Links

Self Referential Structure with Single Link: These structures can have only one self-pointer as their member. The following example will show us how to connect the objects of a self-referential structure with the single link and access the corresponding data members. The connection formed is shown in the following figure.



Implementation:

C++

Java

Python

C#

JavaScript

```
#include <stdio.h>
```

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our [Cookie Policy](#) & [Privacy Policy](#).

```
char data2;
struct node* link;
};

int main()
{
    struct node ob1; // Node1

    // Initialization
    ob1.link = NULL;
    ob1.data1 = 10;
    ob1.data2 = 20;

    struct node ob2; // Node2

    // Initialization
    ob2.link = NULL;
    ob2.data1 = 30;
    ob2.data2 = 40;

    // Linking ob1 and ob2
    ob1.link = &ob2;

    // Accessing data members of ob2 using ob1
    printf("%d", ob1.link->data1);
    printf("\n%d", ob1.link->data2);
    return 0;
}
```

Output

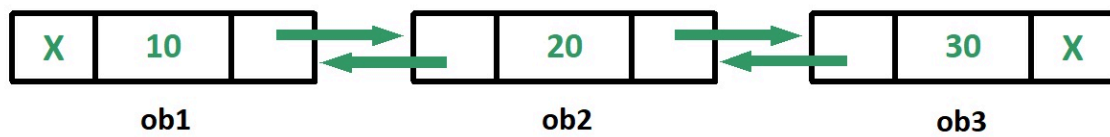
30

40

Self Referential Structure with Multiple Links: Self referential structures with multiple links can have more than one self-pointers. Many complicated data structures can be easily constructed using these structures. Such structures can easily connect to more than one nodes at

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our [Cookie Policy](#) & [Privacy Policy](#).

the following figure.



Implementation:

CPP

Java

Python

JavaScript

```
#include <stdio.h>

struct node {
    int data;
    struct node* prev_link;
    struct node* next_link;
};

int main()
{
    struct node ob1; // Node1

    // Initialization
    ob1.prev_link = NULL;
    ob1.next_link = NULL;
    ob1.data = 10;

    struct node ob2; // Node2

    // Initialization
    ob2.prev_link = NULL;
    ob2.next_link = NULL;
    ob2.data = 20;

    struct node ob3; // Node3

    // Initialization
    ob3.prev_link = NULL;
```

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our [Cookie Policy](#) & [Privacy Policy](#).

```
// Forward links
ob1.next_link = &ob2;
ob2.next_link = &ob3;

// Backward links
ob2.prev_link = &ob1;
ob3.prev_link = &ob2;

// Accessing data of ob1, ob2 and ob3 by ob1
printf("%d\t", ob1.data);
printf("%d\t", ob1.next_link->data);
printf("%d\n", ob1.next_link->next_link->data);

// Accessing data of ob1, ob2 and ob3 by ob2
printf("%d\t", ob2.prev_link->data);
printf("%d\t", ob2.data);
printf("%d\n", ob2.next_link->data);

// Accessing data of ob1, ob2 and ob3 by ob3
printf("%d\t", ob3.prev_link->prev_link->data);
printf("%d\t", ob3.prev_link->data);
printf("%d", ob3.data);
return 0;
}
```

Output

10	20	30
10	20	30
10	20	30

In the above example we can see that 'ob1', 'ob2' and 'ob3' are three objects of the self referential structure 'node'. And they are connected using their links in such a way that any of them can easily access each other's data. This is the beauty of the self referential structures. The connections can be manipulated according to the requirements of the programmer.

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our [Cookie Policy](#) & [Privacy Policy](#).

- [Stacks](#)
- [Queues](#)
- [Trees](#)
- [Graphs](#) etc.

[Comment](#)[More info](#)[Campus Training Program](#)**Corporate & Communications Address:**

A-143, 7th Floor, Sovereign Corporate Tower, Sector- 136, Noida, Uttar Pradesh (201305)

Registered Address:

K 061, Tower K, Gulshan Vivante Apartment, Sector 137, Noida, Gautam Buddh Nagar, Uttar Pradesh, 201305



Advertise with us

Company

[About Us](#)
[Legal](#)
[Privacy Policy](#)
[Careers](#)
[Contact Us](#)

Explore

[POTD](#)
[Job-A-Thon](#)
[Connect](#)
[Community](#)
[Videos](#)

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our [Cookie Policy](#) & [Privacy Policy](#).

DSA
Web Technology
AI, ML & Data Science
DevOps
CS Core Subjects
Interview Preparation

DSA and Placements
Web Development
Data Science
Programming Languages
DevOps & Cloud
GATE

DSA Course DSA Tutorial Data Structures Algorithms Array Strings Linked List Stack

Software and Tools

Offline Centers

Noida
Bengaluru
Pune
Hyderabad
Patna

Preparation Corner

Aptitude
Puzzles
GfG 160
DSA 360
System Design

@GeeksforGeeks, Sanchhaya Education Private Limited, All rights reserved

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our [Cookie Policy](#) & [Privacy Policy](#).