1. **What is the difference between the malloc () and calloc()?**

* *Malloc* () function creates a single block of memory of a fixed size. *Calloc* () function assigns multiple blocks of memory to a single variable.
* Malloc () function returns only starting address and does not make it zero, on the other hand, the calloc () function returns the starting address and makes it zero.

|  |  |  |
| --- | --- | --- |
| **S.No.** | **malloc()** | **calloc()** |
| 1. | malloc() function creates a single block of memory of a specific size. | calloc() function assigns multiple blocks of memory to a single variable. |
| 2. | The number of arguments in malloc() is 1. | The number of arguments in calloc() is 2. |
| 3. | malloc() is faster. | calloc() is slower. |
| 4. | malloc() has high time efficiency. | calloc() has low time efficiency. |
| 5. | The memory block allocated by malloc() has a garbage value. | The memory block allocated by calloc() is initialized by zero. |
| 6. | malloc() indicates memory allocation. | calloc() indicates contiguous allocation. |

1. **What is the difference between the struct and union in C?**

**Both**[**structures**](https://www.shiksha.com/online-courses/articles/structure-in-c-programming/)**and**[**unions**](https://www.shiksha.com/online-courses/articles/unions-in-c/)**allow you to group different data types together, but**

Structure, there is a specific memory location for every input data member. Thus, it can store multiple values of the various members.

a **Structure** is a user defined datatype. It is basically used to combine different datatypes into a single datatype

*struct [structure name]*

*{*

*member definition;*

*member definition;*

*...*

*member definition;*

*};*

*struct Data {*

*int a;*

*long int b;*

*} data, data1;*

**In the case of a Union**, there is an allocation of only one shared memory for all the input data members. Thus, it stores one value at a time for all of its members.

**All the members of a union share the same memory location.**

You can define a union with many members, but only one member can contain a value at any given time

*union [union name]*

*{*

*member definition;*

*member definition;*

*...*

*member definition;*

*};*

union Data {

   int i;

   float f;

} data, data1;

|  |  |  |
| --- | --- | --- |
| Point of Difference | Structure | Union |
| Memory Allocation | Allocates memory for all its members. | Allocates memory only for the largest member. |
| Total Size | Sum of sizes of all members. | Size of the largest member. |
| Data Storage | Can store values for all members simultaneously. | Can store value for only one member at a time. |
| Use Case | When you want to group different data types and access all of them at once. | When you want to save memory and only need one data type at a time. |
| Example | struct example { int a; float b; } | Unionexample { int a; float b; } |
| Accessing Members | All members can be accessed at any time. | Only the last stored member can be accessed. |
| Modification Impact | Modifying a member doesn’t affect other members. | Modifying one member may overwrite other members. |

**3.What is the cell by reference in functions?**