**What is a pointer and explain its application?**

A pointer is a variable which contains the address in memory of another variable.

Pointer is used for different purposes. Pointer is low level construct in programming which is used to perform high level task. Some of the pointer applications are listed below

* Pointer are more efficient data array. They are used to manipulate arrays more easily by moving pointers to them instead of moving the arrays themselves.
* Passing Parameter by Reference.
* Dynamic Memory Allocation or (DMA).
* Passing String to a function.
* Reducing size of parameter.
* Pointer are used to create complex data structures such as link list, trees etc.
* Pointer increases the execution speeds.
* Pointer are used to communicate information about memory which returns the location of free memory.

E.g. Using like malloc(), calloc(),realloc(),free().

**Justify that pointer is jewel in C language.**

A Pointer is a variable that contains address of other variables i.e. a variable that pointer to the location of data item. It provides a way of accessing a variable without referring to the variable directly. It uses the address of the variable. The pointer is called jewel of C language because the real power of C lies in the proper use of pointer. This helps in flexible manipulation of the data and memory as well.

**Explain the pointer to structure with example.**

1. Address of Pointer variable can be obtained using ‘&’ operator.
2. Address of such Structure can be assigned to the Pointer variable.
3. Pointer Variable which stores the address of Structure must be declared as Pointer to Structure.

* Basically a pointer to a structure means to user pointer to access and manipulate data/memory of the structure.

Syntax for Pointer to Structure:

struct student\_database

{

char name[10];

int roll;

int marks;

}

stud1;

struct student\_database \*ptr;

ptr = &stud1;

And there is mainly two ways to do this:

1. Referencing pointer to another address to access memory
2. Using dynamic memory allocation

**Explain the pointer arithmetic with example.**

A pointer in c is an address, which is a numeric value. Therefore, you can perform arithmetic operations on a pointer just as you can on a numeric value. There are four arithmetic operators that can be used on pointers: ++, --, +, and -

To understand pointer arithmetic, let us consider that **ptr** is an integer pointer which points to the address 1000. Assuming 32-bit integers, let us perform the following arithmetic operation on the pointer −

ptr++

After the above operation, the **ptr** will point to the location 1004 because each time ptr is incremented, it will point to the next integer location which is 4 bytes next to the current location. This operation will move the pointer to the next memory location without impacting the actual value at the memory location. If **ptr** points to a character whose address is 1000, then the above operation will point to the location 1001 because the next character will be available at 1001.

**What is Dynamic Memory Allocation (DMA)? How can you use it in C?**

**C dynamic memory allocation** refers to performing manual memory management for dynamic memory allocation in the C programming language via a group of functions in the C standard library, namely malloc, realloc, calloc and free.

Although, C language inherently does not have any technique to allocate memory dynamically, there are 4 library functions under "stdlib.h" for dynamic memory allocation.

| Function | Use of Function |
| --- | --- |
| [malloc()](https://www.programiz.com/c-programming/c-dynamic-memory-allocation#malloc) | Allocates requested size of bytes and returns a pointer first byte of allocated space |
| [calloc()](https://www.programiz.com/c-programming/c-dynamic-memory-allocation#calloc) | Allocates space for an array elements, initializes to zero and then returns a pointer to memory |
| [free()](https://www.programiz.com/c-programming/c-dynamic-memory-allocation#free) | deallocate the previously allocated space |
| [realloc()](https://www.programiz.com/c-programming/c-dynamic-memory-allocation#realloc) | Change the size of previously allocated space |

Now, for example:

### Syntax of malloc()

ptr = (cast-type\*) malloc(byte-size)

**What are the advantages of dynamic memory allocation over static memory allocation?**

* When main memory is allocated statically it cannot be altered during the execution of program. When main memory is allocated dynamically it can be altered during the execution of program as per the user wish.
* The length of dynamically allocated memory either can be decreased or increased
* In case of dynamically created lists insertions and deletions can be done very easily just by the manipulation of addresses whereas in case of statically allocated memory insertions and deletions lead to more number of movements and wastage of memory.
* In case of statically allocated memory there is every chance of “overflow”  during insertions in the lists, whereas in case of dynamically allocated memory it does not come into picture unless otherwise unavailability of main memory.

**How is malloc() function different from calloc() function?**

|  |  |
| --- | --- |
| Differences between malloc and calloc | |
| **malloc** | **calloc** |
| The name malloc stands for *memory allocation*. | The name calloc stands for *contiguous allocation*. |
| void \*malloc(size\_t n) returns a pointer to n bytes of uninitialized storage, or NULL if the request cannot be satisfied. If the space assigned by malloc() is overrun, the results are undefined. | void \*calloc(size\_t n, size\_t size)returns a pointer to enough free space for an array of n objects of the specified size, or NULL if the request cannot be satisfied. The storage is initialized to zero. |
| malloc() takes one argument that is, *number of bytes*. | calloc() take two arguments those are: *number of blocks* and *size of each block*. |
| syntax of malloc():  void \*malloc(size\_t n);  Allocates n bytes of memory. If the allocation succeeds, a void pointer to the allocated memory is returned. Otherwise NULL is returned. | syntax of calloc():  void \*calloc(size\_t n, size\_t size);  Allocates a contiguous block of memory large enough to hold n elements of sizebytes each. The allocated region is initialized to zero. |
| malloc is faster than calloc. | calloc takes little longer than mallocbecause of the extra step of initializing the allocated memory by zero. However, in practice the difference in speed is very tiny and not recognizable. |

**Referenced from:**

[**http://cs-fundamentals.com**](http://cs-fundamentals.com)

[**http://www.studytonight.com**](http://www.studytonight.com)

**dotprogramming.blogspot.com**

[**http://stackoverflow.com**](http://stackoverflow.com)

[**www.cs.fsu.edu**](http://www.cs.fsu.edu)

[**http://picturefullfun.blogspot.com**](http://picturefullfun.blogspot.com)

[**www.programiz.com**](http://www.programiz.com)