**Fundamentals of Computer programming**

**ASSIGNMENT**

**VI**

**Prepared by: Sushil .S. Bajracharya**

**Shift: Day**

**Roll Number: 24**

**BscCSIT**

1. What is pointer and explain its applications.

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

Pointers are declared as:

type \*var\_name ;

type is the pointer's base type(int, float); it must be a valid C data type and var-name is the name of the pointer variable. The asterisk \*is used to declare a pointer.

Example: int \*p;

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

* Passing Parameter by Reference
* Accessing Array element
* Dynamic Memory Allocation
* Reducing size of parameter
* Passing Strings to function
* Provides effective way of implementing the different data structures such as tree, graph, linked list

1. Justify that pointer is jewel in C language.

A pointer is a programming language object, whose value refers to (or "points to") another value stored elsewhere in the computer memory using its memory address. Pointer is low level construct in programming which is used to perform high level task. It is one of the most distinct and exciting features of C language. It has increased power and flexibility to C language. Pointers are more efficient in handling arrays and tables; it can be used to support dynamic memory management. Pointers reduce length and complexity of programs. As pointer is low level construct it increases the execution speed and hence reduces the program execution time. Due to these reasons it can be justify that pointer is jewel in C language.

1. What is dynamic memory allocation (DMA)? How can you use it in C?

The process of manually allocating and freeing memory at run time is known as dynamic memory allocation. In simple terms, Dynamic memory allocation allows you to manually handle memory space for your program enabling us to use the memory efficiently.

In C, 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 |

Syntax of malloc()

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

Here, ptr is pointer of cast-type. The malloc() function returns a pointer to an area of memory with size of byte size. If the space is insufficient, allocation fails and returns NULL pointer.

ptr = (int\*) malloc(100 \* sizeof(int));

This statement will allocate either 200 or 400 according to size of int 2 or 4 bytes respectively and the pointer points to the address of first byte of memory.

Syntax of calloc()

ptr = (cast-type\*)calloc(n, element-size);

This statement will allocate contiguous space in memory for an array of n elements. For example:

ptr = (float\*) calloc(25, sizeof(float));

This statement allocates contiguous space in memory for an array of 25 elements each of size of float, i.e, 4 bytes.

syntax of free()

free(ptr);

This statement frees the space allocated in the memory pointed by ptr.

Syntax of realloc()

ptr = realloc(ptr, newsize);

Here, ptr is reallocated with size of newsize.

1. What are the advantages of dynamic memory allocation over static memory allocation?

Static memory allocation in general is the allocation of memory at compile time before the associated program is executed, unlike dynamic memory where memory is allocated as required at run time.

In static memory allocation we cannot alter the memory during the time of execution of program. But when the memory is allocated dynamically we get the freedom of altering the memory as we wish.

Dynamic memory allocation saves memory. 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.

There is greater chance of “overflow” during insertion in Static memory than in dynamic memory allocation.

For some liner data structures like STACK and QUEUE, dynamic memory allocation proves very much appropriate because the length of such data structure is not fixed. They may increase or decrease dynamically that is during the execution of program. Similarly non-linear data structures need recursive definitions of “struct” data type, so the need of dynamic memory allocation.

5.How is malloc() function different from calloc() function?

|  |  |
| --- | --- |
| **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. |
| Number of argument is 1  (Number of bytes) | Number of arguments is 2  (size of each block and number) |
| 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 size bytes each. The allocated region is initialized to zero. |
| malloc is faster than calloc. | calloc takes little longer than malloc because of the extra step of initializing the allocated memory by zero. However, in practice the difference in speed is very tiny and not recognizable. |

Sources :

<https://dotprogramming.blogspot.com/2014/02/advantage-of-dynamic-over-static-memory.html>

<https://www.programiz.com/c-programming/c-dynamic-memory-allocation>

<https://en.wikipedia.org/>

<http://cs-fundamentals.com/tech-interview/c/difference-between-malloc-and-calloc.php>