Name	Umar Hayyat
Roll No.	2019-EE-360

Programming Fundamentals Assignment

Pointers

In computer science, a **pointer** is an object in many programming languages that stores a memory address. This can be that of another value located in computer memory, or in some cases, that of memory-mapped computer hardware. ... A **pointer** is a **simple**, more concrete implementation of the more abstract reference data type.

Type of Pointer

Pointer is a user defined data **type** which creates special **types** of variables which can hold the address of primitive data **type** like char, int, float, double or user defined data **type** like function, **pointer**, etc. or derived data **type** like an array, structure, union, enum.

Working of Pointer in C

To use pointers in C, we must understand below two operators.

1. To access address of a variable to a **pointer**, we **use** the unary operator & (ampersand) that returns the address of that variable. For example &x gives us address of variable x. ...
2. One more operator is unary * (Asterisk) which is used for two things.

Use of Pointer in C

C uses **pointers** to create dynamic data structures -- data structures built up from blocks of memory allocated from the heap at run-time. **C** uses **pointers** to handle variable parameters passed to functions. **Pointers** in **C** provide an alternative way to access information stored in arrays.

Drawbacks of Pointer in C

Drawbacks of **pointers** in c:

Uninitialized **pointers** might cause segmentation fault. Dynamically allocated block needs to be freed explicitly. Otherwise, it would lead to memory leak. **Pointers** are slower than normal variables. If **pointers** are updated with incorrect values, it might lead to memory corruption.

Pointers not use in Java

So overall **Java** doesn't have **pointers** (in the C/C++ sense) because it doesn't need them for general purpose OOP programming. Furthermore, adding **pointers** to **Java** would undermine security and robustness and make the language more complex.

Pointers are Dangerous

The "**pointer**" is the step in abstracting the idea of an address. ... **Pointer** arithmetic is the reason that many programmers like **pointers** but it is also the reason why **pointers** are **dangerous**. A mistake in the **pointer** computation can result in it pointing somewhere it shouldn't and the whole system can crash as a result.

Dynamic Memory Allocation

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.

Static and Dynamic Memory Allocation

Static Memory Allocation: **Static Memory** is **allocated** for declared variables by the compiler. ... Functions calloc() and malloc() support **allocating dynamic memory**. In the **Dynamic allocation** of **memory** space is **allocated** by using these functions when the value is returned by functions and assigned to pointer variables.

Dynamic memory allocation is better

Dynamic memory allocation is the process of assigning the **memory** space during the execution time or the run time. Reasons and Advantage of **allocating memory**

dynamically: When we do not know how much amount of **memory** would be needed for the program beforehand. ... When you want to use your **memory** space more efficiently.

Dynamic Memory is Allocated

The remainder of the **dynamic** storage area is commonly **allocated** to the heap, from which application programs may **dynamically allocate memory**, as required. In C, **dynamic memory** is **allocated** from the heap using some standard library functions. The two key **dynamic memory** functions are malloc() and free().

Advantages of Dynamic Memory Allocation

Use of **dynamic memory allocation over static** are :

Memory is **allocated during** the execution of the program. **Memory** Bindings are established and destroyed **during** the Execution. **Allocated** only when program unit is active. Less **Memory** space required.

Dynamic Storage

Dynamic storage, also known as forward pick, is the section of warehouse in which items are consistently picked for order fulfillment. In this area, high selectivity is vital to a successful operation, therefore often integrates multiple types of racking to reduce overall pick time, also known as a pick module.

Function of dynamic storage management

Dynamic memory management involves the use of pointers and four standard **library functions**, namely, malloc, calloc, realloc and free. The first three functions are used to allocate memory, whereas the last function is used to return memory to the system (also called freeing/deallocating memory).