THEORY

DEFINITION

Define of Dynamic Array:

An array is a data structure that stores a collection of elements, all of the same data type, in a contiguous block of memory. Each element in the array is identified by an index, which is a numerical value that represents the position of the element in the array.

Define of Linked list

A linked list is a linear data structure that consists of a series of nodes connected by pointers. Each node contains data and a reference to the next node in the list.

Define of Binary file

A binary file is a file whose content is in a binary format consisting of a series of sequential bytes, each of which is eight bits in length.

(A binary file stores data directly in the binary number system (0s and 1s), the same format used by computer memory for efficient data storage.)

Define of Recursion

Recursion is a programming technique that breaks down a problem into smaller subproblems and calls itself repeatedly until a base case is reached.

The difference between array and linked list:

	Array	Linked List
Definition	An array is a linearly-ordered data structure with the same type of elements in contiguous memory addresses.	A linked list data structure represents a sequence of nodes.
Size	Fixed Size	Dynamic Size
Memory Allocation	Contiguous	Disparate
Memory Usage	Memory efficient due to the contiguous memory allocation and no extra pointers.	Requires extra memory for pointers.
Insertion/Deletion Time	O(n): worst case (when insertion or deletion is done at the beginning or in the middle)	O(1): if insertion is done at the beginning.O(n): If insertion is done at the end without a tail pointer.
Search Time	O(n): Worst Case (linear search)O(logn): With binary search if sorted.	O(n): Worst Case (linear search)
Access Time	O(1): For direct access using indexes	O(n), since elements need to be traversed sequentially.
Delete Time	O(n): If removing from the beginning and the middle.O(1): If removing from the end.	O(1): Removing from the beginning.O(n): removing from the middle and the end.
Use Cases	When the size is known and fixed, it can also be if the insertion and deletion are not required much.	Best when the deletion and insertion require more than simple access to the data.
Example:	Storing pixel data in image, lookup tables.	Implementing data structure lie stack and queue.

Advantages of Linked List:

- **Dynamic data structure**: A linked list is a dynamic arrangement so it can grow and shrink at runtime by allocating and deallocating memory. So there is no need to give the initial size of the linked list.
- **No memory wastage:** In the Linked list, efficient memory utilization can be achieved since the size of the linked list increase or decrease at run time so there is no memory wastage and there is no need to pre-allocate the memory.
- **Implementation**: Linear data structures like stacks and queues are often easily implemented using a linked list.
- **Insertion and Deletion Operations**: Insertion and deletion operations are quite easier in the linked list. There is no need to shift elements after the insertion or deletion of an element only the address present in the next pointer needs to be updated.

- **Flexible**: This is because the elements in Linked List are not stored in contiguous memory locations unlike the array.
- **Efficient for large data**: When working with large datasets linked lists play a crucial role as it can grow and shrink dynamically.
- **Scalability**: Contains the ability to add or remove elements at any position.

Disadvantages of Linked List:

- **Memory usage**: More memory is required in the linked list as compared to an array. Because in a linked list, a pointer is also required to store the address of the next element and it requires extra memory for itself.
- **Traversal**: In a Linked list traversal is more time-consuming as compared to an array. Direct access to an element is not possible in a linked list as in an array by index. For example, for accessing a node at position n, one has to traverse all the nodes before it.
- **Reverse Traversing**: In a singly linked list reverse traversing is not possible, but in the case of a doubly-linked list, it can be possible as it contains a pointer to the previously connected nodes with each node. For performing this extra memory is required for the back pointer hence, there is a wastage of memory.
- **Random Access**: Random access is not possible in a linked list due to its dynamic memory allocation.
- **Lower efficiency at times**: For certain operations, such as searching for an element or iterating through the list, can be slower in a linked list.
- **Complex implementation**: The linked list implementation is more complex when compared to array. It requires a complex programming understanding.
- **Difficult to share data**: This is because it is not possible to directly access the memory address of an element in a linked list.
- **Not suited for small dataset**: Cannot provide any significant benefits on small dataset compare to that of an array.

Advantage of Dynamic Array:

- **Dynamic arrays are expandable**: A dynamic array can increase in size as needed, which makes it especially useful when you don't know how large the array will be. This way, you can avoid the issue of having a static array that is too small to store the data you need to store.
- **Dynamic arrays are efficient**: When compared to linked lists and hash tables, a dynamic array can be more efficient when you need to retrieve elements quickly. Since a dynamic array is indexed, all elements can be accessed and retrieved in constant time (O(1)). In contrast, with a linked list, it can take O(n) time, where n is the number of elements in the list.
- **Dynamic arrays are easily manageable**: A dynamic array is easy to read and manage. To access a specific element, all you need to do is use its index. This makes dynamic arrays easier to maintain than other data structures, such as linked lists and hash tables.

Disadvantage of Dynamic Array:

- **Dynamic arrays can be slow to resize**: When the number of elements in a dynamic array increases and the array needs to be resized, the entire array must be moved to a new location in memory, which can be costly in terms of time and resources.
- **Dynamic arrays can use a lot of memory**: A dynamic array often needs to allocate more memory than is strictly necessary, since it needs to allocate extra space in order to be able to store more elements when needed. This can result in memory wastage.
- **Dynamic arrays cannot store different types of elements**: As with all arrays, a dynamic array can only store homogenous elements, which means that you cannot store different types of elements in the same array. If you need to store different types of elements, then you need to use a different data structure.

The difference between Recursion and Iterration:

Property	Recursion	Iteration
Definition	Function calls itself.	A set of instructions repeatedly executed.
Application	For functions.	For loops.
Termination	Through base case, where there will be no function call.	When the termination condition for the iterator ceases to be satisfied.
Usage	Used when code size needs to be small, and time complexity is not an issue.	Used when time complexity needs to be balanced against an expanded code size.
Code Size	Smaller code size	Larger Code Size.
Time Complexity	Very high(generally exponential) time complexity.	Relatively lower time complexity(generally polynomial-logarithmic).
Space Complexity	The space complexity is higher than iterations.	Space complexity is lower.
Stack	Here the stack is used to store local variables when the function is called.	Stack is not used.
Speed	Execution is slow since it has the overhead of maintaining and updating the stack.	Normally, it is faster than recursion as it doesn't utilize the stack.
Memory	Recursion uses more memory as compared to iteration.	Iteration uses less memory as compared to recursion.
Overhead	Possesses overhead of repeated function calls.	No overhead as there are no function calls in iteration.
Infinite Repetition	If the recursive function does not meet to a termination condition or the base case is not defined or is never reached then it leads to a stack overflow error and there is a chance that the an system may crash in infinite recursion.	If the control condition of the iteration statement never becomes false or the control variable does not reach the termination value, then it will cause infinite loop. On the infinite loop, it uses the CPU cycles again and again.

Advantages of Recursion:

- Recursion is very simple and easy to understand.
- It requires a minimal number of programming statements.
- Recursion will break the problem into smaller pieces of sub problems for example Tower of Hanoi.
- It is used to solve mathematical, trigonometric, or any type of algebraic problems.
- It is more useful in multiprogramming and multitasking environments.
- It is useful in solving data structure problems like linked lists, queues and stacks.
- Recursive function is useful in tree traversal and stacks.
- Complex tasks can be solved easily.

Disadvantages of Recursion:

- It consumes more storage space than other techniques such as Iteration, Dynamic programming etc.
- If base condition is not set properly then it may create a problem such as a system crash, freezing etc.,
- Compared to other techniques recursion is a time-consuming process and less efficient.
- It is difficult to trace the logic of the function.
- Computer memory is exhausted if recursion enters an infinite loop.
- Excessive function calls are being used.
- Each function called will occupy memory in stack. Which will lead to stack overflow.

The difference between Binary file and Text file

S. No.	Text file	Binary File
1.	The text files can easily be transferred from one computer system to another.	Binary files cannot easily be transferred from one computer system to another due to variations in the internal variations in the internal representation which varies from computer to computer.
2.	It stores data using ASCII format i.e. human-readable graphic characters.	It stores data in binary format i.e. with the help of 0 and 1.
3.	These files are easily readable and modifiable because the content written in text files is human readable. Content written in binary files is not human-readable and looks like encrypted content.	These files are not easily readable and modifiable because the content written in binary files is not human-readable and it is encrypted content.
4.	These files create portability problems.	These files are easily portable.
5.	Text files save the data by converting each digit in data into ASCII format which will take up much of the space as compared to the required one. For example, the number 546378 is an integer that should occupy 4 bytes in the disk but it will occupy 6 bytes, 1 byte for each digit in the number.	These save memory because the data of any type will get stored in memory as per its memory size. For example, any integer number irrespective of individual digits in the number will be stored by consuming 4 bytes.
6.	Any file is by default text file.	The ios:: binary mode has to be used with binary files while opening them.
7.	Error in a textual file can be easily recognized and eliminated.	Error in a binary file corrupts the file and is not easily detected.
8.	In a text file, a new line character is first converted to a carriage return-line feed combination and then written to the disk. Vice versa happens when a line is read from the text file.	In binary file, no such conversion from newline to carriage return-line feed combination is done.
9.	In a text file, a special character with ASCII code 26 is inserted at the end of the file. This character signals the EOF to the program when encountered.	There is no such special character in the binary file to signal EOF.
10.	Text files are used to store data more user friendly.	Binary files are used to store data more compactly.
11.	Mostly .txt and .rtf are used as extensions to text files.	Can have any application defined extension.

What is a pointer in C?

A pointer is defined as a derived data type that can store the address of other C variables or a memory location. We can access and manipulate the data stored in that memory location using pointers.

Features of Pointer:

- Pointers save memory space.
- Execution time with pointers is faster because data are manipulated with the address, that is, direct access to memory location.
- Memory is accessed efficiently with the pointers. The pointer assigns and releases the memory as well. Hence it can be said the Memory of pointers is dynamically allocated.
- Pointers are used with data structures. They are useful for representing two-dimensional and multi-dimensional arrays.
- An array, of any type, can be accessed with the help of pointers, without considering its subscript range.
- Pointers are used for file handling.
- Pointers are used to allocate memory dynamically.
- In C++, a pointer declared to a base class could access the object of a derived class. However, a pointer to a derived class cannot access the object of a base class.

PASSING AND CALLING

Passing by Pointer:

- Here, the memory location (address) of the variables is passed to the parameters in the function, and then the operations are performed. It is also called the call by pointer method.

Passing by reference:

- It allows a function to modify a variable without having to create a copy of it. We have to declare reference variables. The memory location of the passed variable and parameter is the same and therefore, any change to the parameter reflects in the variable as well.

Passing by value:

- Pass by value means that a copy of the actual parameter's value is made in memory, i.e. the caller and callee have two independent variables with the same value. If the callee modifies the parameter value, the effect is not visible to the caller.

Parameters	Pass by Pointer	Pass by Reference
Passing Arguments	We pass the address of arguments in the function call.	We pass the arguments in the function call.
Accessing Values	The value of the arguments is accessed via the dereferencing operator *	The reference name can be used to implicitly reference a value.
Reassignment	Passed parameters can be moved/reassigned to a different memory location.	Parameters can't be moved/reassigned to another memory address.
Allowed Values	Pointers can contain a NULL value, so a passed argument may point to a NULL or even a garbage value.	References cannot contain a NULL value, so it is guaranteed to have some value.

Difference Between Reference Variable and Pointer Variable:

A reference is the same object, just with a different name and a reference must refer to an object. Since references can't be NULL, they are safer to use.

- A pointer can be re-assigned while a reference cannot, and must be assigned at initialization only.
- The pointer can be assigned NULL directly, whereas the reference cannot.
- Pointers can iterate over an array, we can use increment/decrement operators to go to the next/previous item that a pointer is pointing to.
- A pointer is a variable that holds a memory address. A reference has the same memory address as the item it references.
- A pointer to a class/struct uses '->' (arrow operator) to access its members whereas a reference uses a '.' (dot operator)
- A pointer needs to be dereferenced with * to access the memory location it points to, whereas a reference can be used directly

When to use pass by value?

If we are building multi-threaded application, then we don't have to worry of objects getting modified by other threads. In distributed application pass by value can save the over network overhead to keep the objects in sync.

When to use pass by reference?

In pass by reference, no new copy of the variable is made, so overhead of copying is saved. This makes programs efficient especially when passing objects of large structs or classes.

Difference between the Call by Value and Call by Reference in C

The following table lists the differences between the call-by-value and call-by-reference methods of parameter passing.

Call By Value	Call By Reference
While calling a function, we pass the values of variables to it. Such functions are known as "Call By Values".	While calling a function, instead of passing the values of variables, we pass the address of variables(location of variables) to the function known as "Call By References.
In this method, the value of each variable in the calling function is copied into corresponding dummy variables of the called function.	In this method, the address of actual variables in the calling function is copied into the dummy variables of the called function.
With this method, the changes made to the dummy variables in the called function have no effect on the values of actual variables in the calling function.	With this method, using addresses we would have access to the actual variables and hence we would be able to manipulate them.
In call-by-values, we cannot alter the values of actual variables through function calls.	In call by reference, we can alter the values of variables through function calls.
Values of variables are passed by the Simple technique.	Pointer variables are necessary to define to store the address values of variables.
This method is preferred when we have to pass some small values that should not change.	This method is preferred when we have to pass a large amount of data to the function.
Call by value is considered safer as original data is preserved	Call by reference is risky as it allows direct modification in original data

Credit:

https://hcmus.edu.vn/

https://www.geeksforgeeks.org/

https://www.educative.io/

https://www.w3schools.com/