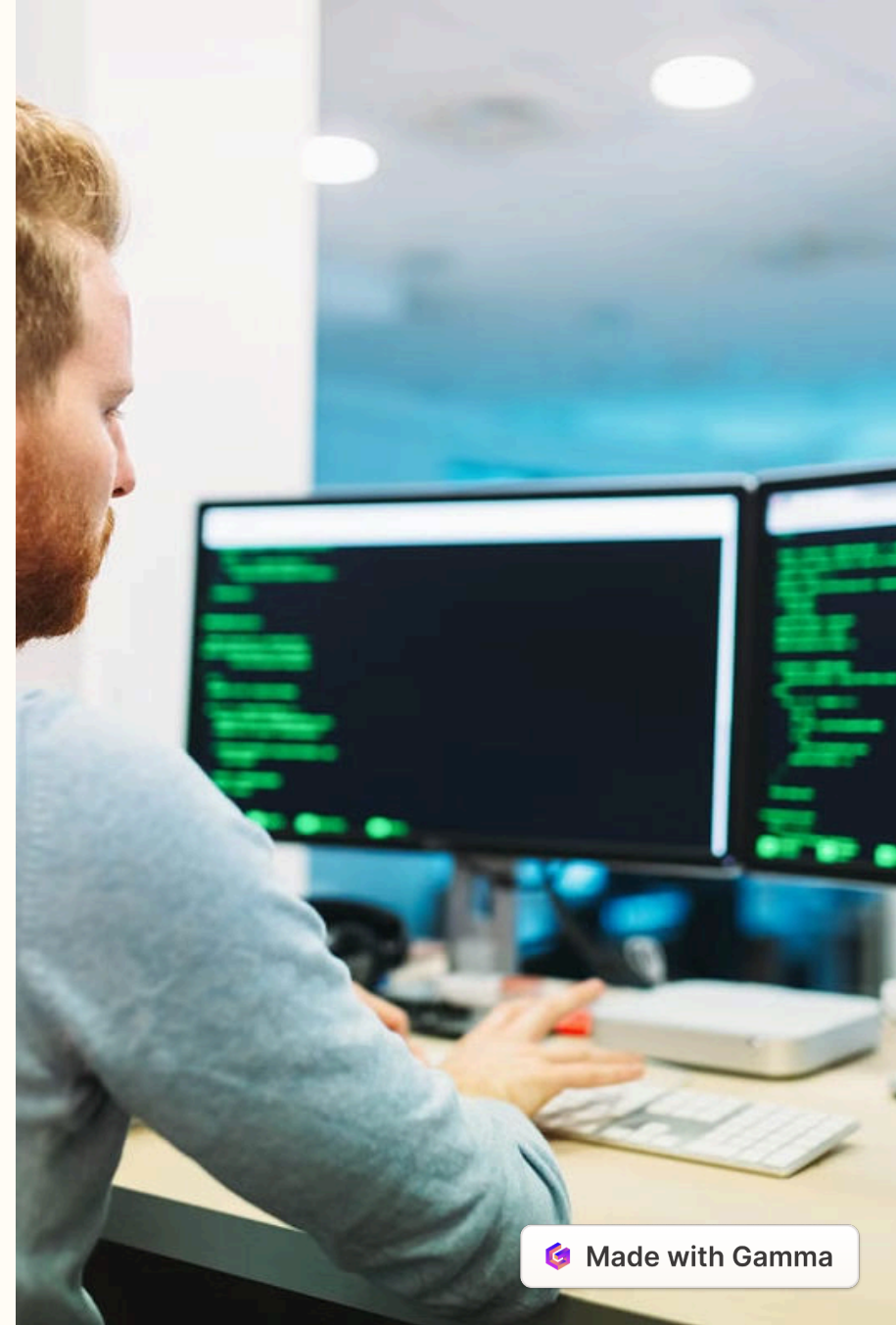


Data Structures in C

Data structures are the fundamental building blocks used to organize and manipulate data in computer programs. In C, developers have access to a wide range of data structures, each with its own strengths and applications. This presentation will explore several key data structures that are essential to C programming.

BA by Bhuvan Goud



Arrays

Fixed Size

Arrays in C have a fixed size, meaning the number of elements they can hold is predetermined and cannot be changed during runtime.

Indexed Access

Elements in an array are accessed using an index, typically starting from 0. This provides constant-time access to any element.

Sequential Storage

Arrays store elements in contiguous memory locations, allowing for efficient memory usage and fast sequential access.

Structures (Structs)

Composite Data Types

Structs allow you to group multiple variables of different data types into a single unit, creating a custom data type.

Representing Complex Entities

Structs are useful for representing real-world objects or concepts, such as a person, a car, or a bank account.

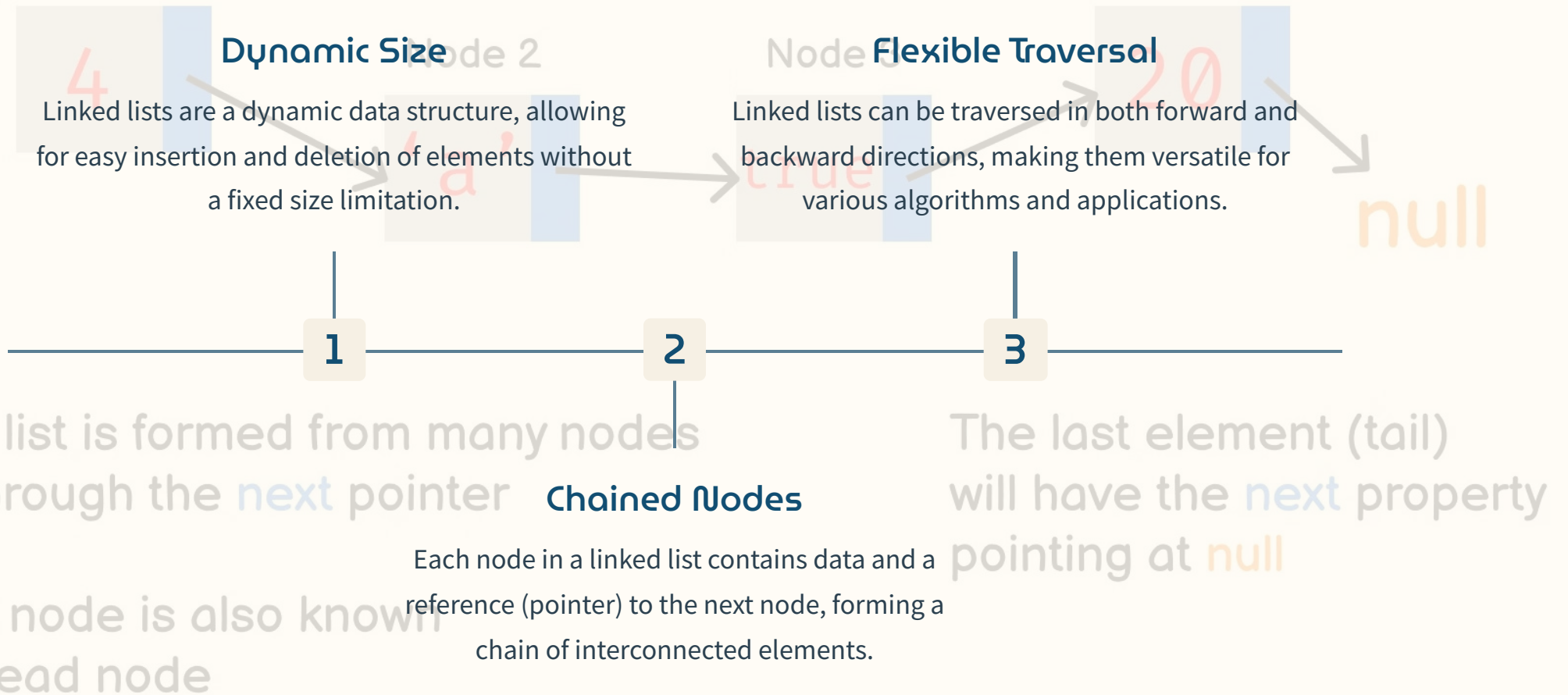
Memory Efficiency

Structs optimize memory usage by storing all their members contiguously in memory, making them efficient for large data sets.

Encapsulation

Structs enable encapsulation, allowing you to hide implementation details and provide a clean interface for interacting with the data.

Linked Lists



Stacks

1 Last-In-First-Out (LIFO)

Stacks follow the LIFO principle, where the most recently added element is the first one to be removed.

3 Memory Efficiency

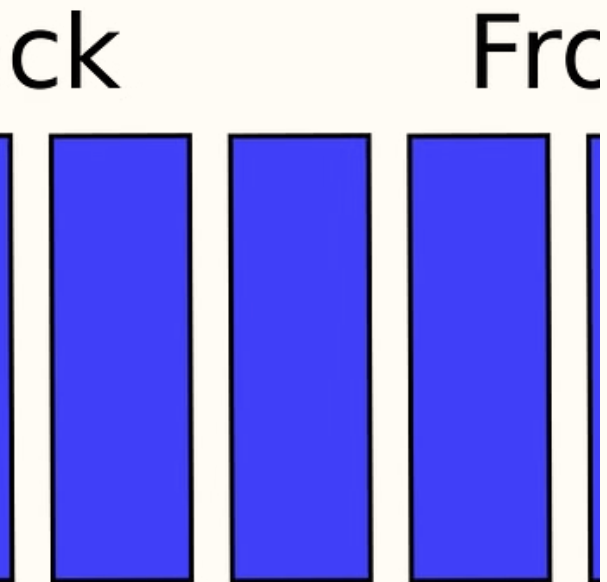
Stacks are memory-efficient, as they only require the storage of the top element and a pointer to the next element.

2 Push and Pop Operations

Elements are added to the top of the stack using the push operation and removed using the pop operation.

4 Common Applications

Stacks are widely used in function calls, expression evaluation, and backtracking algorithms.



Queues

1

First-In-First-Out (FIFO)

Queues follow the FIFO principle, where the first element added to the queue is the first one to be removed.

2

Enqueue and Dequeue

Elements are added to the rear of the queue using the enqueue operation and removed from the front using the dequeue operation.

3

Efficient Memory Usage

Queues can be implemented using an array or a linked list, both of which provide efficient memory utilization.

Trees



Root Node

The topmost node in a tree, which serves as the starting point for the data structure.



Leaf Nodes

The nodes at the bottom of the tree, which have no child nodes.




Branch Nodes

Nodes that have one or more child nodes branching out from them.



Traversal

The process of visiting all the nodes in a tree in a specific order, such as depth-first or breadth-first.

	reader of your stance / opinion.	Inter-changeable
2	Summary of main ideas: summarise the key topics discussed in main body.	
3	Recommendation (suggestion) / Prediction / Solution: the	

Conclusion

Data structures are the foundation of efficient and organized programming in C. By understanding the key characteristics and use cases of arrays, structs, linked lists, stacks, queues, and trees, developers can choose the right data structure for their specific programming needs, leading to more robust and performant applications.