Allocating memory at runtime in C is essential for several reasons:

**1. Dynamic Data Structures**

* **Flexibility**: You can create data structures like linked lists, trees, and graphs where the size is not known at compile time. This allows for dynamic growth and shrinkage as elements are added or removed.
* **Example**: If you don't know how many items you will need to store (e.g., user input), you can allocate memory as needed.

**2. Efficient Memory Use**

* **Resource Management**: You can allocate memory only when you need it and free it when you're done, leading to more efficient use of memory resources.
* **Example**: If a program processes large amounts of data intermittently, it can allocate memory for those processes and then release it when no longer needed.

**3. Large Data Sizes**

* **Handling Large Data**: Static memory allocation has limits (like stack size). Using dynamic allocation lets you allocate large amounts of memory in the heap that can exceed these limits.
* **Example**: Large arrays or buffers needed for image processing or data analysis can be allocated dynamically.

**4. Avoiding Stack Overflow**

* **Stack vs. Heap**: The stack has limited space, and allocating large arrays or deep recursive calls can lead to stack overflow. Using the heap allows you to avoid this issue.
* **Example**: Recursive algorithms that require substantial memory can benefit from dynamic memory allocation.

**5. Adaptability**

* **Changing Requirements**: Programs often need to adapt to changing conditions. Runtime allocation allows you to adjust memory usage based on current needs without recompiling.
* **Example**: A program that reads data from a file can adjust its memory allocation based on the file size.