Vector in C++

What is a Vector?

A vector in C++ is a dynamic array provided by the Standard Template Library (STL). Unlike regular arrays, vectors can grow or shrink in size dynamically during runtime.

Where Does It Belong?

Vectors are part of the `<vector>` header in the C++ Standard Template Library (STL).

Syntax of Vector

To declare a vector:

```
std::vector<DataType> vectorName;
```

Example:

```
std::vector<int> vec;
```

Why and Where to Use Vectors

- **Why**: Vectors provide dynamic sizing, rich functionality, and automatic memory management, making them versatile and easy to use compared to arrays.
- **Where**: Use vectors when the size of the collection is not known at compile time or may change dynamically during the program execution.

Common Member Functions of Vectors

1. **push_back()**: Adds an element to the end of the vector.
2. **pop_back()**: Removes the last element of the vector.
3. **size()**: Returns the current number of elements in the vector.
4. **capacity()**: Returns the current allocated capacity of the vector.
5. **resize()**: Changes the size of the vector.
6. **clear()**: Removes all elements, making the vector empty.
7. **at(index)**: Returns the element at the specified position with bounds checking.
8. **empty()**: Checks if the vector is empty.
9. **front()**: Returns the first element of the vector.
10. **back()**: Returns the last element of the vector.

Difference Between Array and Vector

1. **Size**: Arrays have a fixed size; vectors are dynamic.
2. **Memory Management**: Arrays require manual management for dynamic allocation; vectors handle memory automatically.
3. **Flexibility**: Arrays cannot grow or shrink; vectors can resize dynamically.
4. **Initialization**: Arrays have limited initialization methods; vectors support versatile initialization.
5. **Bounds Checking**: Arrays lack bounds checking; vectors provide bounds checking with `at()`.

6. **Functions**: Arrays do not have member functions; vectors have a rich set of member functions.
7. **Performance**: Arrays are faster for small, fixed-size collections; vectors are slightly slower due to dynamic resizing.
8. **Ease of Use**: Vectors are easier to use with dynamic sizing and built-in functionality.

---

in memory mangment vector get memoery on heap

directly and that memory internally deallocated

in normal array defaulty memory created o n stack if we want we can store array on heap also an we need to explicitlyt free the resources

add it on o=your phreses