حافظه Stack (پشته):

تعریف: حافظه Stack یک ناحیه از حافظه است که برای ذخیره‌سازی متغیرهای محلی و فراخوانی‌های تابعی استفاده می‌شود. این حافظه به صورت خودکار توسط کامپایلر مدیریت می‌شود.

ویژگی‌ها:

سازمان‌دهی: به صورت LIFO (Last In, First Out) سازمان‌دهی می‌شود، به این معنا که آخرین متغیری که وارد می‌شود، اولین متغیری است که خارج می‌شود.

اندازه: اندازه محدودی دارد و معمولاً از پیش تعیین‌شده است.

زمان دسترسی: دسترسی به داده‌ها در حافظه Stack بسیار سریع است.

عمر متغیر: متغیرها به محض خروج از بلوک کد یا تابعی که در آن‌ها تعریف شده‌اند، از بین می‌روند.

مدیریت حافظه: نیازی به مدیریت دستی ندارد؛ حافظه به طور خودکار تخصیص و آزاد می‌شود.

حافظه Heap (پشته):

تعریف: حافظه Heap ناحیه‌ای از حافظه است که برای ذخیره‌سازی داده‌هایی که طول عمر نامعلومی دارند و به صورت دینامیک تخصیص داده می‌شوند، استفاده می‌شود.

ویژگی‌ها:

سازمان‌دهی: به صورت غیرسازمان‌یافته و پراکنده است؛ مکان‌های حافظه به صورت تصادفی تخصیص داده می‌شوند.

اندازه: اندازه بسیار بزرگتر و انعطاف‌پذیرتر از Stack است.

زمان دسترسی: دسترسی به داده‌ها در حافظه Heap کندتر از Stack است به دلیل نیاز به مدیریت دستی.

عمر متغیر: داده‌ها تا زمانی که به صورت دستی آزاد نشوند، باقی می‌مانند.

مدیریت حافظه: نیاز به مدیریت دستی دارد؛ برنامه‌نویس باید حافظه تخصیص داده شده را به طور دستی آزاد کند (مثلاً با استفاده از free در C یا delete در C++).

تفاوت‌های اصلی بین Stack و Heap

سازمان‌دهی:

Stack: سازمان‌دهی LIFO.

Heap: سازمان‌دهی غیرسازمان‌یافته و پراکنده.

اندازه:

Stack: محدود و از پیش تعیین‌شده.

Heap: بزرگتر و انعطاف‌پذیرتر.

زمان دسترسی:

Stack: بسیار سریع.

Heap: کندتر به دلیل نیاز به مدیریت دستی.

مدیریت حافظه:

Stack: مدیریت خودکار.

Heap: مدیریت دستی.

عمر متغیر:

Stack: محدود به بلوک کد یا تابع.

Heap: تا زمانی که به صورت دستی آزاد نشود.

کاربرد:

Stack: برای متغیرهای محلی و فراخوانی‌های تابعی.

Heap: برای داده‌هایی که نیاز به طول عمر نامشخص و دینامیک دارند.

در کل، انتخاب بین استفاده از حافظه Stack و Heap بستگی به نیازهای برنامه و طول عمر داده‌ها دارد.