CS Studies - malloc()

What is `malloc()` ?

`malloc()` is used to dynamically allocate a block of memory on the heap.

```
void* malloc(size_t size);
```

The parameter size is the number of bytes to be allocated. The function returns a void* pointer to the first byte of the allocated memory block.

Since the malloc() function returns a void pointer, it is typically cast to the appropriate pointer type to be used. Following example result is cast to an int* pointer so that it can be used as an array of integers.

```
int* myArray = (int*) malloc(sizeof(int) * arraySize);
```

The preferred way of allocating dynamic memory is using `new`, since it also calls the constructor for any objects created in that memory while allocating memory on the heap. It is also worth noting that delete operator is recommended to release dynamically allocated memory, since it not only deallocates memory but also calls the destructor for any objects created in the memory.

Difference between `malloc()` and `new`

While both `malloc()` and `new` can be used for dynamic memory allocation, there are following differences between the two:

- **Type-safety**
    - `new` is **type-safe**, which means it automatically determines the correct size and alignment of the memory block to be allocated based on the typ[e of object being created.
    - `malloc()` returns a void* which must be **cast to the appropriate pointer type.**
- **Constructor Call**
    - `new` calls the constructor for any objects created in the allocated memory, and objects are **automatically initialized**

    - `malloc()` does not call the constructor of objects created, and instead just returns a block of **uninitialized memory**.
- **Return Type**
    - `new` returns a pointer to the **newly allocated object.**
    - `malloc()` returns a `void*` pointer to the b**eginning of the allocated block.**
- **Memory Overhead**
    - `new` may have some memory overhead[1] due to the allocation of metadata[2] (such as vtables for virtual functions).[3]
    - `malloc()` typically has **no such overhead.**
- **Error Handling**
    - `new` throws a `std::bad_alloc` exception if it fails to allocate memory, while `malloc()` returns a `NULL` pointer in case of failure

The `new` operator in C++ is generally preferred over `malloc()` because of its type-safety, constructor calls, and exception handling. However, `malloc()` can be useful where low-level memory management is required or when working with legacy code that relies on `malloc()`

[1] memory overhead: refers to the additional memory that is required by the system to manage the allocated memory.
[2] metadata: refers to the additional information that is stored by the system to manage the allocated memory.
[3] When using `new` to allocated memory for an object, the C++ runtime may need to add extra information to the allocated memory block to keep track of metadata such as the size of the allocated block, the type of the object being created, and pointers to virtual function tables, if any. This metadata is used by the C++ runtime to manage the memory and support features like dynamic polymorphism. Such metadata adds additional memory overhead to the allocated block, which can be significant for small objects. In general, memory overhead is a trade-off between memory usage and runtime performance- more metadata means more memory usage, but it can also improve runtime performance by reducing the amount of time spent managing memory.