Static and Dynamic Allocation

Static and dynamic allocation are two methods of memory allocation in C++.

Static allocation refers to the process of reserving memory for a variable at compile-time. The memory for the variable is allocated from the stack, and its size is determined before the program starts running. Static allocation is used for variables with a fixed size that need to retain their values throughout the lifetime of the program.

Dynamic allocation, on the other hand, refers to the process of allocating memory for a variable at runtime. This is done using special keywords such as `new` or `malloc`. The memory for the variable is allocated from the heap, and its size can be determined and changed during program execution. Dynamic allocation is useful when the size of the variable is not known in advance or when the variable needs to be created and destroyed dynamically.

Both static and dynamic allocation have their advantages and disadvantages, and the choice between them depends on the specific requirements of the program.

Static Allocation:

An example of static allocation in C++ is:

```
int main() {
    int staticVariable = 5;
    // Code goes here...
    return 0;
}
```

In this example, the variable `staticVariable` is allocated statically. Its memory is reserved at compile-time, and it retains its value throughout the lifetime of the program.

Dynamic Allocation:

An example of dynamic allocation in C++ is:

```
int main() {
    int* dynamicVariable = new int;
    *dynamicVariable = 10;
    // Code goes here...
    delete dynamicVariable;
    return 0;
}
```

In this example, the variable `dynamicVariable` is allocated dynamically using the `new` keyword. The memory for the variable is allocated from the heap at runtime. The value 10 is assigned to the variable using the dereference operator `*`. After using the dynamically allocated variable, it should be explicitly deallocated using the `delete` keyword to free the memory.

Difference between Static and Dynamic Allocation

Aspect	Static Allocation	Dynamic Allocation
Memory Allocation	Memory is allocated at compile time.	Memory is allocated at runtime.
Flexibility	Not flexible; fixed memory allocation.	Highly flexible; can allocate memory as needed.
Memory Management	Requires manual memory management.	Typically involves automatic memory management (e.g., garbage collection).
Size Determination	Size is determined at compile time.	Size can be determined at runtime.

Memory Wastage	May lead to memory wastage if the allocated memory is not fully utilized.	Usually minimizes memory wastage as memory is allocated based on actual needs.
Efficiency	Generally more efficient in terms of memory access and performance.	May introduce some overhead due to dynamic memory allocation and deallocation.
Lifetime of Objects	Objects have a fixed lifetime determined at compile time.	Objects can have variable lifetimes determined at runtime.
Examples	Arrays with a fixed size, constants.	Linked lists, dynamic arrays, objects in object-oriented programming.