

### Memory Allocation in C++

Now that we have an understanding of the available process memory, let us take a look at memory allocation in C++.

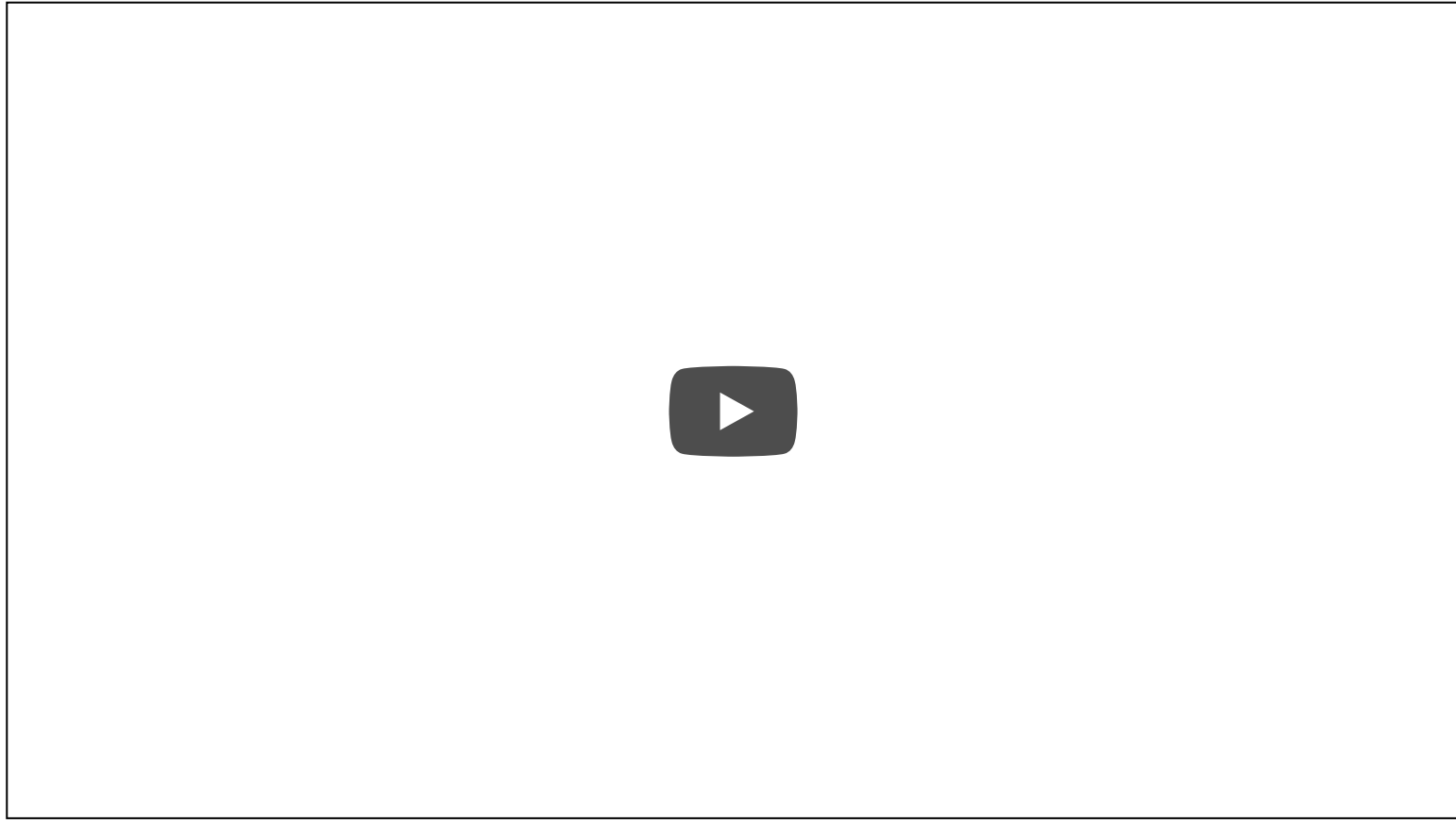
Not every variable in a program has a permanently assigned area of memory. The term **allocate** refers to the process of assigning an area of memory to a variable to store its value. A variable is **deallocated** when the system reclaims the memory from the variable, so it no longer has an area to store its value.

Generally, three basic types of memory allocation are supported:

1. **Static memory allocation** is performed for static and global variables, which are stored in the BSS and Data segment. Memory for these types of variables is allocated once when your program is run and persists throughout the life of your program.
2. **Automatic memory allocation** is performed for function parameters as well as local variables, which are stored on the stack. Memory for these types of variables is allocated when the path of execution enters a scope and freed again once the scope is left.
3. **Dynamic memory allocation** is a possibility for programs to request memory from the operating system at runtime when needed. This is the major difference between automatic and static allocation, where the size of the variable must be known at compile time. Dynamic memory allocation is not performed on the limited stack but on the heap and is thus (almost) only limited by the size of the address space.

From a programmer's perspective, stack and heap are the most important areas of program memory. Hence, in the following lessons, let us look at these two in turn.

#### Outro



<https://youtu.be/JrxMUJWfKhA>