

Heap memory allows us to create memory independent of the lifecycle of a function.



Heap Memory

If memory needs to exist for longer than the lifecycle of the function, we must use **heap memory**.

The <u>only</u> way to create heap memory in C++ is with the <u>new</u> operator.

The new operator returns a pointer to the memory storing the data - not an instance of the data itself.



C++'s New Operator

The **new** operator in C++ will always do three things:

- 1. Allocate memory on the heap for the data structure
- 2. Initialize the data structure
- 3. Return a pointer to the start of the data structure

The memory is only ever reclaimed by the system when the pointer is passed to the **delete** operator.



Heap Memory

The code below allocates two chunks of memory:

- Memory to store an integer pointer on the stack
- Memory to store an integer on the heap

```
int * numPtr = new int;
```

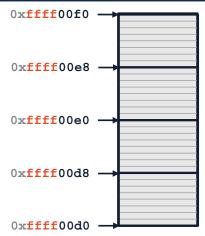


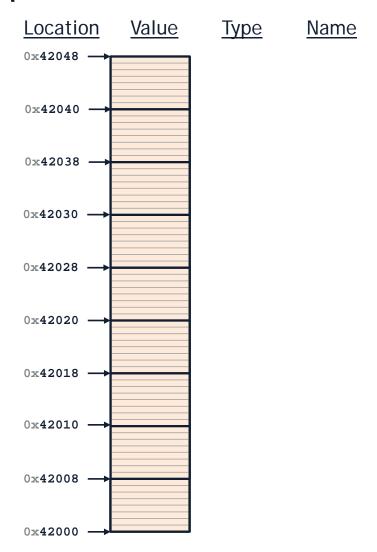
cpp-heapMemory/main.cpp

```
10|int main() {
11
     int *numPtr = new int;
12
     std::cout << "*numPtr: " << *numPtr << std::endl;</pre>
13
14
     std::cout << " numPtr: " << numPtr << std::endl;</pre>
     std::cout << "&numPtr: " << &numPtr << std::endl;</pre>
15
16
17
     *numPtr = 42;
18
     std::cout << "*numPtr assigned 42." << std::endl;</pre>
19
     std::cout << "*numPtr: " << *numPtr << std::endl;</pre>
20
     std::cout << " numPtr: " << numPtr << std::endl;</pre>
21
     std::cout << "&numPtr: " << &numPtr << std::endl;</pre>
22
23
24
     return 0;
25
```

cpp-heapMemory/heap1.cpp

```
11 int main() {
     int *p = new int;
12
13
     Cube *c = new Cube;
14
15
     *p = 42;
     (*c).setLength(4);
16
17
     delete c;
18
     delete p;
19
20
     return 0;
21
```





nullptr

The C++ keyword **nullptr** is a pointer that points to the memory address 0x0.

- nullptr represents a pointer to "nowhere"
- Address 0x0 is reserved and never used by the system.
- Address 0x0 will always generate an "segmentation fault" when accessed.
- Calls to delete 0x0 are ignored.



Arrow Operator (->)

When an object is stored via a pointer, access can be made to member functions using the -> operator:

```
c->getVolume();
    ...identical to...
(*c).getVolume();
```



cpp-heapMemory/heap2.cpp

```
int main() {
11
12
     Cube *c1 = new Cube;
     Cube *c2 = c1;
13
14
     c2->setLength( 10 );
15
16
17
     delete c2;
     delete c1; // !!
18
19
20
     return 0;
21
```

