Heaps

Wednesday, 18 January 2023 4:19 AM

Heap Memory

- Heap memory allows us to create memory independent life cycle of a function
- If the memory needs to exist for longer than the life cycle of the function, we use heap memory
- Only way to create heap memory in C++ is with the new operator
- The new operator returns a pointer to the memory storing the data- not an instance of the data itself.

The "new" operator

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

The "new" keyword allocates two chunks of memory:

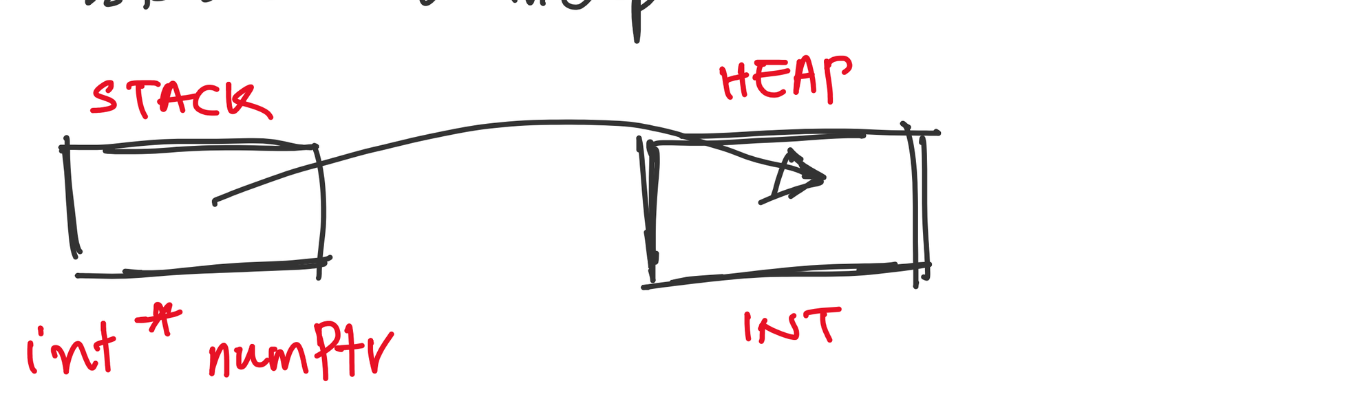
1> memory to store an integer pointer on the stack

2> memory to store an integer on the heap

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

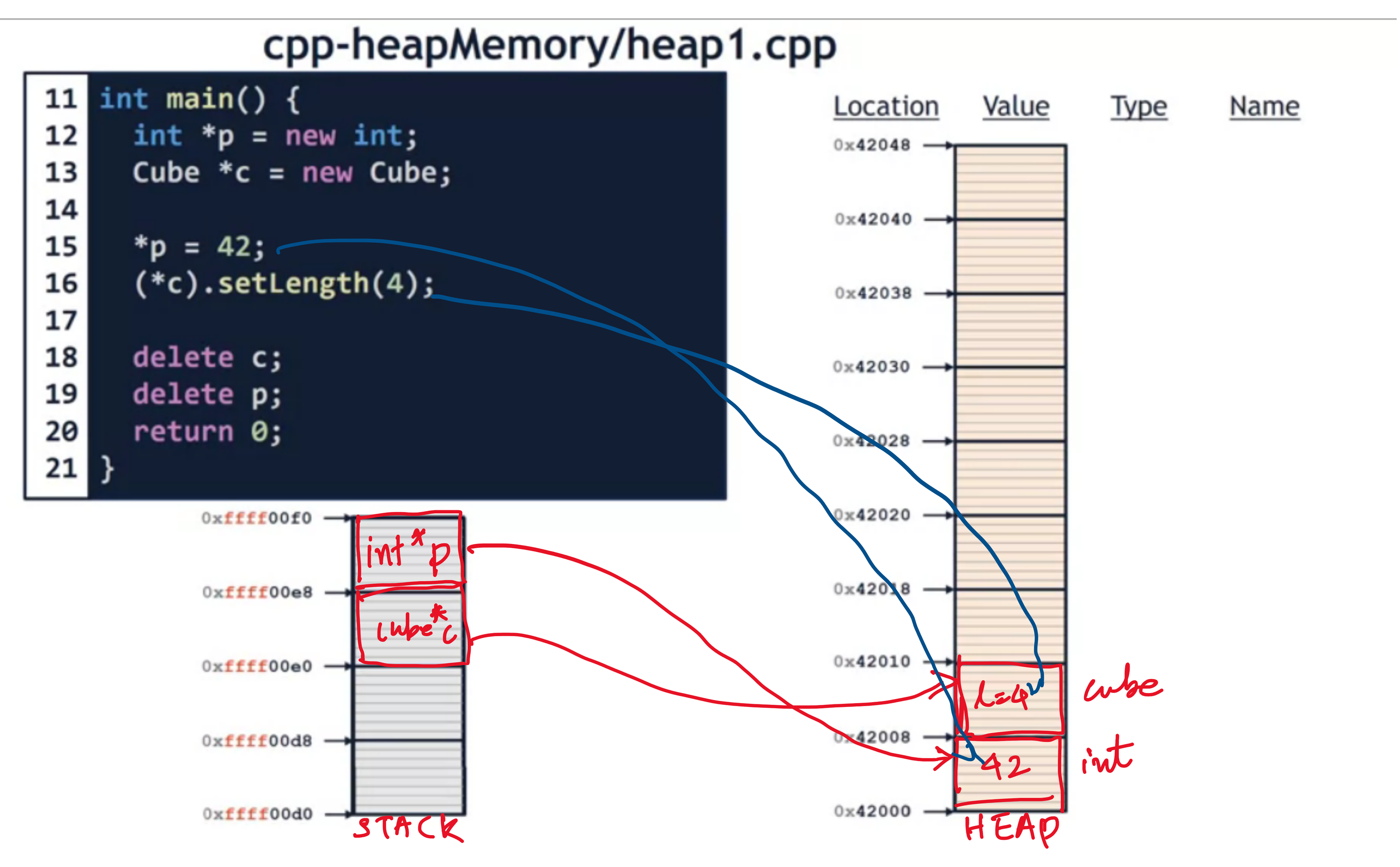
- is a pointer to an integer.
- stored in stack
- it points to an integer stored in heap

- stored in heap
- this will be stored in heap unless we call delete



- The content of numPtr is address of heap memory initiated as int.

- stack memory begins with high address and go down
- heap memory starts at low addresses and grows up.



delete c; → the heap memory c points to is freed

delete p; → the heap memory p points to freed

Even after the line "delete c;" the pointer c still stores the address of the deleted variable, which is no longer valid to dereference and is therefore dangerous. The pointer won't be automatically set to nullptr. Then, you should *manually* set c to nullptr if you want to better protect against coding mistakes:

delete c;

c = nullptr;

Also, this is a good practice in general whenever you use "delete" on a pointer, but in this particular example, the function that is about to return is "main". When main returns, the program exits and the operating system will usually release all the allocated memory back to the system. This isn't an excuse to be sloppy, though ; as you will soon see, many class objects have their "destructor" functions called silently when the program exits, and mistakes can trigger crashes even during that process.

nullptr (in C, NULL)

↳ keyword nullptr is a pointer that points to the memory address 0x0

- nullpointer represents a pointer to nowhere
- address 0x0 is reserved and never used by the system
- address 0x0 will always generate a "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:

```
cube *c

c->getVolume(); ≡ (*c).getVolume();
```

Puzzle

```
int main() {
    Cube *c1 = new Cube;
    Cube *c2 = c1;
    c2->setLength(10);
    delete c2;
    delete c1;
    return 0
}
```

- Cube *c1 — in stack; c1 → contains the address of cube in heap
- Cube *c2 — in stack; c2 contains the address of (points to) same cube c1 contains (points to) (in heap)
- delete c2; — when c2 is deleted, the cube it points to in heap is deleted
- delete c1; — when we try to delete c1, the memory (cube) it points to in the heap, program returns an error because there is nothing at that address, as we already deleted c2