#4: Heap Memory

September 4, 2019 · *G Carl Evans*

Puzzle from last Friday:

puzzle.cpp
```
Cube *CreateCube() {
  Cube c(20);
  return &c;
}

int main() {
  Cube *c = CreateCube();
  SomeOtherFunction();
  double v = c->getVolume();
  double a = c->getSurfaceArea();
  return 0;
}
```

Takeaway:

Heap Memory:

As programmers, we can use heap memory in cases where the *lifecycle of the variable exceeds the lifecycle of the function.*

1. The only way to create heap memory is with the use of the **new** keyword. Using **new** will:
   -
   -
   -

2. The only way to free heap memory is with the use of the **delete** keyword. Using **delete** will:
   -
   -

3. Memory is never automatically reclaimed, even if it goes out of scope. Any memory lost, but not freed, is considered to be "leaked memory".

---

heap1.cpp
```
int main() {
  int *p = new int;
  cs225::Cube *c = new cs225::Cube(10);

  return 0;
}
```

Stack	Value
0xffff00f0 →	
0xffff00e8 →	
0xffff00e0 →	
0xffff00d8 →	
0xffff00d0 →	

Heap	Value
0x42020 →	
0x42018 →	
0x42010 →	
0x42008 →	
0x42000 →	

heap2.cpp
```
int main() {
  Cube *c1 = new Cube();
  Cube *c2 = c1;
  c2->setLength( 10 );
  delete c2;
  delete c1;
  return 0;
}
```

Stack	Value
0xffff00f0 →	
0xffff00e8 →	
0xffff00e0 →	
0xffff00d8 →	
0xffff00d0 →	

Heap	Value
0x42020 →	
0x42018 →	
0x42010 →	
0x42008 →	
0x42000 →	

Copying Memory – Deep Copy vs. Shallow Copy

copy.cpp

```
 6  int  i =  2,  j =  4,  k =  8;
 7  int *p = &i, *q = &j, *r = &k;
 8
 9  k = i;
10  cout << i << j << k << *p << *q << *r << endl;
11
12  p = q;
13  cout << i << j << k << *p << *q << *r << endl;
14
15  *q = *r;
16  cout << i << j << k << *p << *q << *r << endl;
```

Consider how each assignment operator changes the data:

	Type of LHS	**Type of RHS**	**Data Changed?**
Line 8-9			
	i =	**j =**	**k =**
	p =	**q =**	**r =**
Line 11-12			
	i =	**j =**	**k =**
	p =	**q =**	**r =**
Line 14-15			
	i =	**j =**	**k =**
	p =	**q =**	**r =**

Reference Variable

A reference variable is an alias to an existing variable. Modifying the reference variable modifies the variable being aliased. Internally, a reference variable maps to the same memory as the variable being aliased. Three key ideas:

1.

2.

3.

reference.cpp

```
int main() {
  int i = 7;
  int & j = i;   // j is an alias of i

  j = 4;                           // j and i are both 4.
  std::cout << i << " " << j << std::endl;

  i = 2;                           // j and i are both 2.
  std::cout << i << " " << j << std::endl;
  return 0;
}
```

heap-puzzle1.cpp

```
int *x = new int;
int &y = *x;

y = 4;

cout << &x << endl;
cout << x << endl;
cout << *x << endl;

cout << &y << endl;
cout << y << endl;
cout << *y << endl;
```

heap-puzzle2.cpp

```
int *p, *q;
p = new int;
q = p;
*q = 8;
cout << *p << endl;

q = new int;
*q = 9;
cout << *p << endl;
cout << *q << endl;
```

CS 225 – Things To Be Doing:

1. Exam 0 starts on Thursday, know your time slot!
2. Finish up MP1 – Due Monday, Sept. 9 at 11:59pm
3. Complete lab_debug this week in lab sections (due Sunday)
4. POTDs are released daily, worth +1 extra credit point! ☺