CS 225
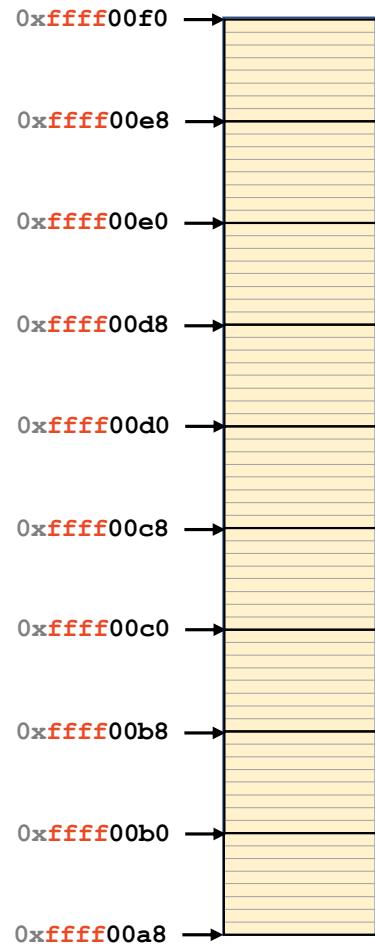
Data Structures

August 30 – Heap Memory
G Carl Evans

Stack Memory

0xffff00f0 →

0xffff00e8 →

0xffff00e0 →

0xffff00d8 →

0xffff00d0 →

0xffff00c8 →

0xffff00c0 →

0xffff00b8 →

0xffff00b0 →

0xffff00a8 →

Location	Value	Type	Name
0xffff00f0 →			
0xffff00e8 →			
0xffff00e0 →			
0xffff00d8 →			
0xffff00d0 →			
0xffff00c8 →			
0xffff00c0 →			
0xffff00b8 →			
0xffff00b0 →			
0xffff00a8 →			

puzzle.cpp

```cpp
1   #include "Cube.h"
2   using cs225::Cube;
3
4   Cube *CreateCube() {
5     Cube c(20);
6     return &c;
7   }
8
9   int main() {
10    Cube *c = CreateCube();
11    SomeOtherFunction();
12    double v = c->getVolume();
13    double a = c->getSurfaceArea();
14    return 0;
15  }
```

Location	Value	Type	Name
0xffff00f0 →			
		main's stack frame	
0xffff00e8 →			
		Cube *	**c**
0xffff00e0 →			
0xffff00d8 →			
0xffff00d0 →			
0xffff00c8 →			
0xffff00c0 →			
0xffff00b8 →			
0xffff00b0 →			
0xffff00a8 →			

puzzle.cpp

```cpp
1  #include "Cube.h"
2  using cs225::Cube;
3
4  Cube *CreateCube() {
5     Cube c(20);
6     return &c;
7  }
8
9  int main() {
10    Cube *c = CreateCube();
11    SomeOtherFunction();
12    double v = c->getVolume();
13    double a = c->getSurfaceArea();
14    return 0;
15 }
```

Location	Value	Type	Name
0xffff00f0			
0xffff00e8		main's stack frame	
		Cube *	c
0xffff00e0			
0xffff00d8			
0xffff00d0		CreateCube frame	
		Cube	c
0xffff00c8			
0xffff00c0			
0xffff00b8			
0xffff00b0			
0xffff00a8			

```cpp
1   #include "Cube.h"
2   using cs225::Cube;
3
4   Cube *CreateCube() {
5       Cube c(20);
6       return &c;
7   }
8
9   int main() {
10      Cube *c = CreateCube();
11      SomeOtherFunction();
12      double v = c->getVolume();
13      double a = c->getSurfaceArea();
14      return 0;
15  }
```

puzzle.cpp

| Location	Value	Type	Name
0xffff00f0			
0xffff00e8			**main's stack frame**
		Cube *	c
0xffff00e0			
0xffff00d8			
0xffff00d0			**CreateCube frame**
		Cube	c
0xffff00c8			
0xffff00c0			
0xffff00b8			
0xffff00b0			
0xffff00a8			

```cpp
1  #include "Cube.h"
2  using cs225::Cube;
3
4  Cube *CreateCube() {
5     Cube c(20);
6     return &c;
7  }
8
9  int main() {
10    Cube *c = CreateCube();
11    SomeOtherFunction();
12    double v = c->getVolume();
13    double a = c->getSurfaceArea();
14    return 0;
15 }
```

puzzle.cpp

Location	Value	Type	Name
0xffff00f0			
0xffff00e8			**main's stack frame**
0xffff00e0		Cube *	c
0xffff00d8			
0xffff00d0			
0xffff00c8			
0xffff00c0			
0xffff00b8			
0xffff00b0			
0xffff00a8			

puzzle.cpp

```cpp
1  #include "Cube.h"
2  using cs225::Cube;
3
4  Cube *CreateCube() {
5    Cube c(20);
6    return &c;
7  }
8
9  int main() {
10   Cube *c = CreateCube();
11   SomeOtherFunction();
12   double v = c->getVolume();
13   double a = c->getSurfaceArea();
14   return 0;
15 }
```

Location | Value | Type | Name

| Location		Type	Name
0xffff00f0 →			
		main's stack frame	
0xffff00e8 →		**Cube ***	**c**
0xffff00e0 →			
0xffff00d8 →			
		SomeOtherFunction	
0xffff00d0 →			
0xffff00c8 →			
0xffff00c0 →			
0xffff00b8 →			
0xffff00b0 →			
0xffff00a8 →			

```cpp
 1  #include "Cube.h"
 2  using cs225::Cube;
 3
 4  Cube *CreateCube() {
 5     Cube c(20);
 6     return &c;
 7  }
 8
 9  int main() {
10     Cube *c = CreateCube();
11     SomeOtherFunction();
12     double v = c->getVolume();
13     double a = c->getSurfaceArea();
14     return 0;
15  }
```

puzzle.cpp

Heap Memory
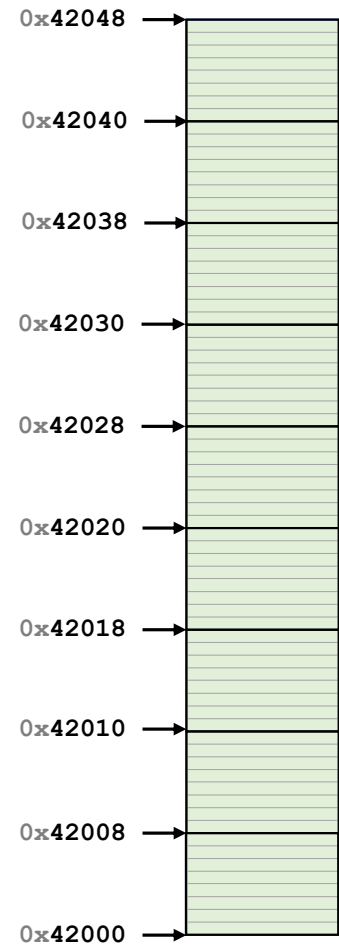
Heap Memory - new

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

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

1.

2.

3.

Heap Memory - delete

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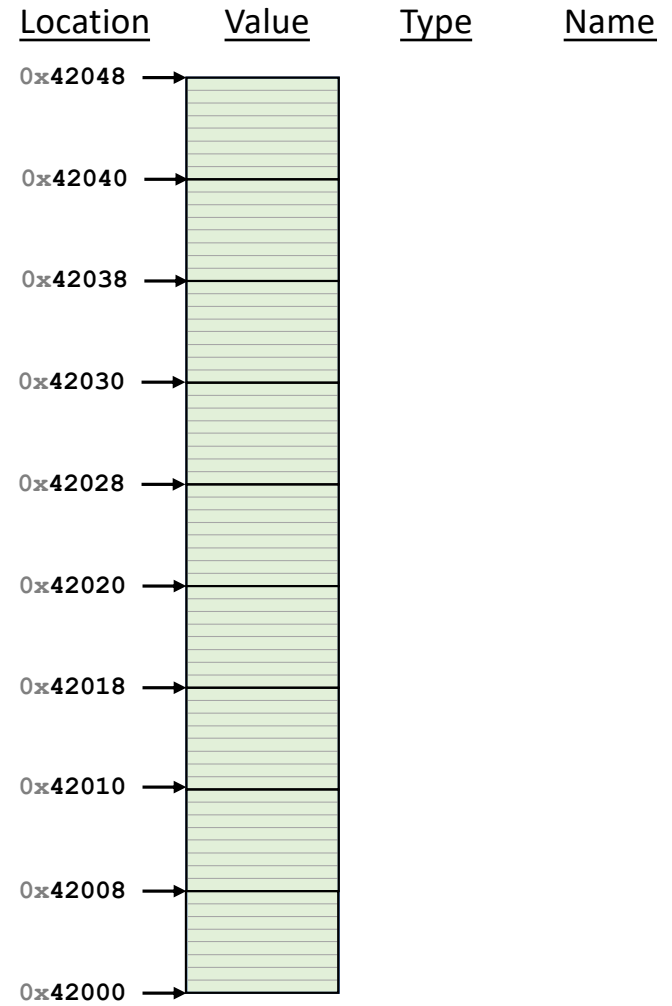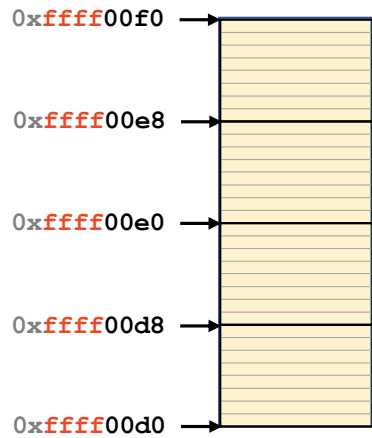
*

*

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".

Heap Memory vs. Stack Memory Lifecycle

```cpp
int main() {                heap2.cpp
  Cube *c1 = new Cube();
  Cube *c2 = c1;
  c2->setRadius( 10 );
  delete c2;
  delete c1;
  return 0;
}
```
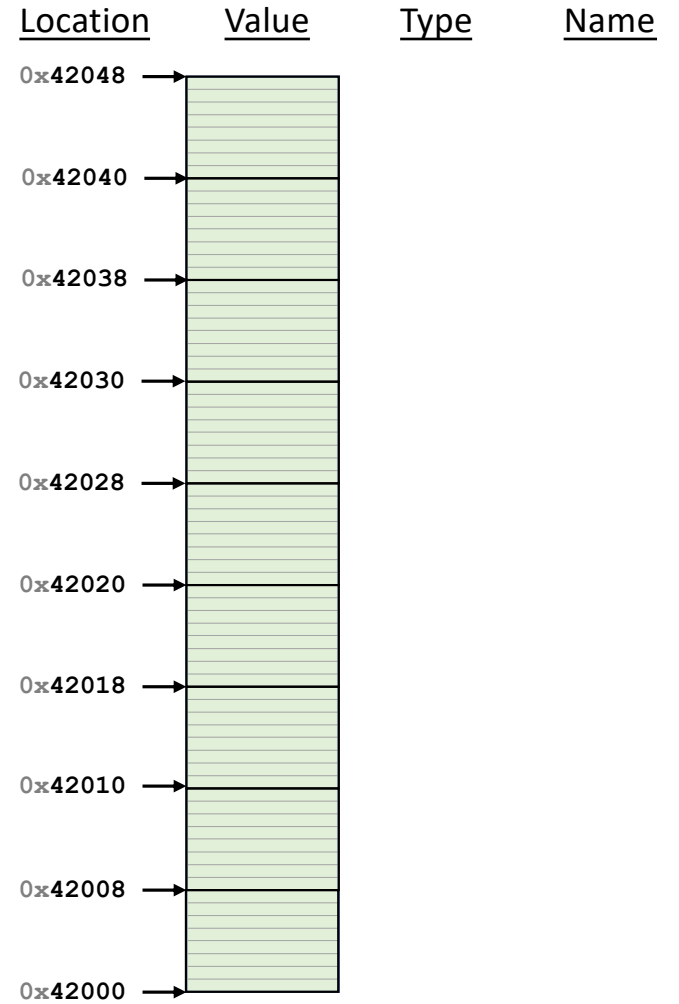
Location	Value	Type	Name
0x42048			
0x42040			
0x42038			
0x42030			
0x42028			
0x42020			
0x42018			
0x42010			
0x42008			
0x42000			

0xffff00f0

0xffff00e8

0xffff00e0

0xffff00d8

0xffff00d0

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

  return 0;
}
```
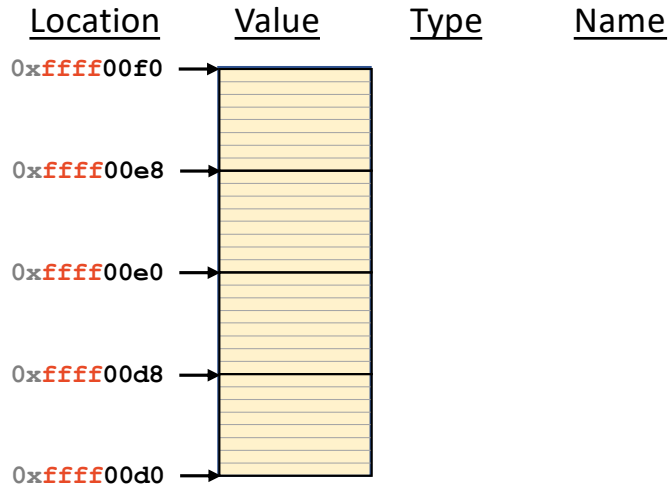
heap1.cpp

Location	Value	Type	Name
0x42048			
0x42040			
0x42038			
0x42030			
0x42028			
0x42020			
0x42018			
0x42010			
0x42008			
0x42000			

Location	Value	Type	Name
0xffff00f0			
0xffff00e8			
0xffff00e0			
0xffff00d8			
0xffff00d0			

copy.cpp

```cpp
#include <iostream>
using std::cout;
using std::endl;

int main() {
  int  i =  2,  j =  4,  k =  8;
  int *p = &i, *q = &j, *r = &k;

  k = i;
  cout << i << j << k << *p << *q << *r << endl;

  p = q;
  cout << i << j << k << *p << *q << *r << endl;

  *q = *r;
  cout << i << j << k << *p << *q << *r << endl;
}
```

Pointers and References

A variable containing an instance of an object:

```
1  Cube s1;
```

A reference variable of a Cube object:

```
2  Cube & r1 = s1;
```

A variable containing a pointer to a Cube object:

```
3  Cube * p1;
```

Reference Variable

A reference variable is an <u>alias</u> to an existing variable.

Key Idea: Modifying the reference variable modifies the variable being aliased.

Reference Variable

Three facts about reference variables:

1.

2.

3.

Reference Variable

A reference variable is an <u>alias</u> to an existing variable.

```cpp
#include <iostream>

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

   j = 4;
   std::cout << i << " " << j << std::endl;

   i = 2;
   std::cout << i << " " << j << std::endl;
   return 0;
}
```

heap-puzzle1.cpp

```cpp
1  #include <iostream>
2  using namespace std;
3
4  int main() {
5     int *x = new int;
6     int &y = *x;
7
8     y = 4;
9
10    cout << &x << endl;
11    cout << x << endl;
12    cout << *x << endl;
13
14    cout << &y << endl;
15    cout << y << endl;
16    cout << *y << endl;
17 }
```

heap-puzzle2.cpp

```cpp
1  #include <iostream>
2  using namespace std;
3
4  int main() {
5     int *p, *q;
6     p = new int;
7     q = p;
8     *q = 8;
9     cout << *p << endl;
10
11    q = new int;
12    *q = 9;
13    cout << *p << endl;
14    cout << *q << endl;
15
16    return 0;
17 }
```

heap-puzzle3.cpp

```cpp
#include <iostream>
using namespace std;

int main() {
  int *x;
  int size = 3;

  x = new int[size];

  for (int i = 0; i < size; i++) {
    x[i] = i + 3;
  }

  delete[] x;
}
```