

CS 225

Data Structures

Wade Fagen-Ulmschneider

<u>Location</u>	<u>Value</u>	<u>Type</u>	<u>Name</u>
0xffff00f0			
0xffff00e8			
0xffff00e0			
0xffff00d8			
0xffff00d0			
0xffff00c8			
0xffff00c0			
0xffff00b8			
0xffff00b0			
0xffff00a8			

```

1 #include "sphere.h"      puzzle.cpp
2 using namespace cs225;
3
4 Sphere *CreateUnitSphere() {
5     Sphere s(1);
6     return &s;
7 }
8
9 int main() {
10    Sphere *s = CreateUnitSphere();
11    someOtherFunction();
12    double r = s->getRadius();
13    double v = s->getVolume();
14    return 0;
15 }
```

<u>Location</u>	<u>Value</u>	<u>Type</u>	<u>Name</u>
0xffff00f0			
0xffff00e8		Sphere *	s
0xffff00e0			
0xffff00d8			
0xffff00d0			
0xffff00c8			
0xffff00c0			
0xffff00b8			
0xffff00b0			
0xffff00a8			

main's stack frame

```

1 #include "sphere.h"          puzzle.cpp
2 using namespace cs225;
3
4 Sphere *CreateUnitSphere() {
5     Sphere s(1);
6     return &s;
7 }
8
9 int main() {
10    Sphere *s = CreateUnitSphere();
11    someOtherFunction();
12    double r = s->getRadius();
13    double v = s->getVolume();
14    return 0;
15 }
```

<u>Location</u>	<u>Value</u>	<u>Type</u>	<u>Name</u>
0xffff00f0			
0xffff00e8		Sphere *	s
0xffff00e0			
0xffff00d8			
0xffff00d0		Sphere	s
0xffff00c8			
0xffff00c0			
0xffff00b8			
0xffff00b0			
0xffff00a8			

Diagram illustrating the stack frames:

- main's stack frame**: Located at address 0xffff00e8, containing a pointer to a Sphere object.
- CreateUnitSphere frame**: Located at address 0xffff00d0, containing a local Sphere object named s.

```

1 #include "sphere.h"          puzzle.cpp
2 using namespace cs225;
3
4 Sphere *CreateUnitSphere() {
5     Sphere s(1);
6     return &s;
7 }
8
9 int main() {
10    Sphere *s = CreateUnitSphere();
11    someOtherFunction();
12    double r = s->getRadius();
13    double v = s->getVolume();
14    return 0;
15 }
```

<u>Location</u>	<u>Value</u>	<u>Type</u>	<u>Name</u>
0xffff00f0			
0xffff00e8		Sphere *	s
0xffff00e0			
0xffff00d8			
0xffff00d0		Sphere	s
0xffff00c8			
0xffff00c0			
0xffff00b8			
0xffff00b0			
0xffff00a8			

Diagram illustrating the stack frames:

- main's stack frame**: Located at address 0xffff00e8. It contains a pointer to a Sphere object.
- CreateUnitSphere frame**: Located at address 0xffff00d0. It contains a local Sphere object named s.

A blue arrow points from the value at address 0xffff00e8 in the main stack frame to the sphere object in the CreateUnitSphere frame.

```

1 #include "sphere.h"          puzzle.cpp
2 using namespace cs225;
3
4 Sphere *CreateUnitSphere() {
5     Sphere s(1);
6     return &s;
7 }
8
9 int main() {
10    Sphere *s = CreateUnitSphere();
11    someOtherFunction();
12    double r = s->getRadius();
13    double v = s->getVolume();
14    return 0;
15 }
```

<u>Location</u>	<u>Value</u>	<u>Type</u>	<u>Name</u>
0xffff00f0			
0xffff00e8			main's stack frame
0xffff00e0		Sphere *	s
0xffff00d8			
0xffff00d0			
0xffff00c8			
0xffff00c0			
0xffff00b8			
0xffff00b0			
0xffff00a8			

```

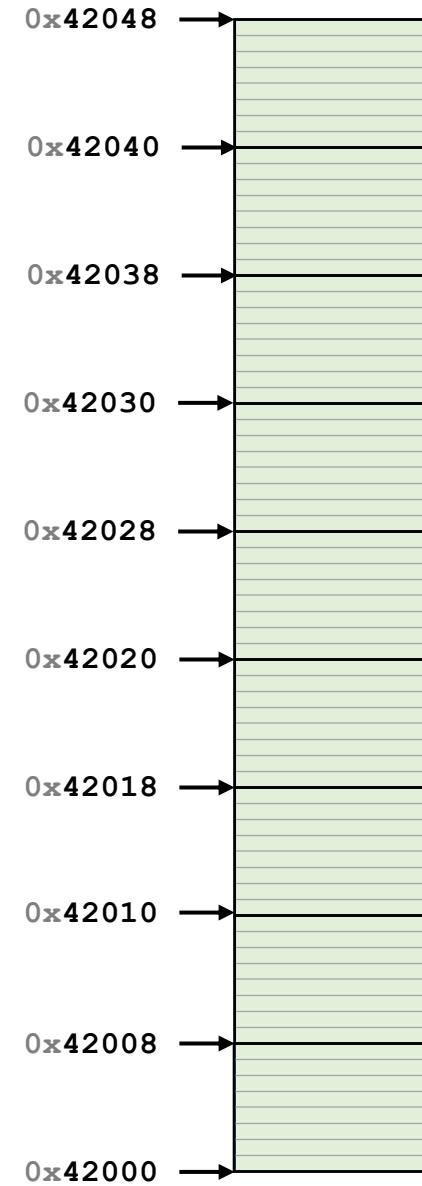
1 #include "sphere.h"          puzzle.cpp
2 using namespace cs225;
3
4 Sphere *CreateUnitSphere() {
5     Sphere s(1);
6     return &s;
7 }
8
9 int main() {
10    Sphere *s = CreateUnitSphere();
11    someOtherFunction();
12    double r = s->getRadius();
13    double v = s->getVolume();
14    return 0;
15 }
```

<u>Location</u>	<u>Value</u>	<u>Type</u>	<u>Name</u>
0xffff00f0			
0xffff00e8			main's stack frame
0xffff00e0		Sphere *	s
0xffff00d8			
0xffff00d0			someOtherFunction
0xffff00c8			
0xffff00c0			
0xffff00b8			
0xffff00b0			
0xffff00a8			

```

1 #include "sphere.h"          puzzle.cpp
2 using namespace cs225;
3
4 Sphere *CreateUnitSphere() {
5     Sphere s(1);
6     return &s;
7 }
8
9 int main() {
10    Sphere *s = CreateUnitSphere();
11    someOtherFunction();
12    double r = s->getRadius();
13    double v = s->getVolume();
14    return 0;
15 }
```

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

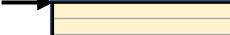
2. The only way to free heap memory is with the use of the **delete** keyword. Using **delete** will:

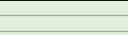
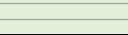
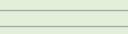
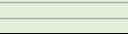
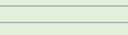
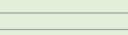
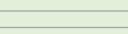
-
-

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”.

Heap Memory vs. Stack Memory Lifecycle

```
1 #include "sphere.h"      heap1.cpp
2 using namespace cs225;
3
4 int main() {
5     int *p = new int;
6     int *s = new Sphere(10);
7
8
9
10    return 0;
11 }
```

<u>Location</u>	<u>Value</u>	<u>Type</u>	<u>Name</u>
0xfffff00f0			
0xfffff00e8			
0xfffff00e0			
0xfffff00d8			
0xfffff00d0			

<u>Location</u>	<u>Value</u>	<u>Type</u>	<u>Name</u>
0x42048			
0x42040			
0x42038			
0x42030			
0x42028			
0x42020			
0x42018			
0x42010			
0x42008			
0x42000			

```
1 #include "sphere.h"      heap2.cpp
2 using namespace cs225;
3
4 int main() {
5     Sphere *s1 = new Sphere();
6     Sphere *s2 = s1;
7     s2->setRadius( 10 );
8
9
10    return 0;
11 }
```



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

Exam 1 Topics

MP1

copy.cpp

```
1 #include <iostream>
2 using namespace std;
3
4 int main() {
5     int i = 2, j = 4, k = 8;
6     int *p = &i, *q = &j, *r = &k;
7
8     k = i;
9     cout << i << j << k << *p << *q << *r << endl;
10
11    p = q;
12    cout << i << j << k << *p << *q << *r << endl;
13
14    *q = *r;
15    cout << i << j << k << *p << *q << *r << endl;
16 }
```

heap-puzzle1.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

```
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

```
1 #include <iostream>
2 using namespace std;
3
4 int main() {
5     int *x;
6     int size = 3;
7
8     x = new int[size];
9
10    for (int i = 0; i < size; i++) {
11        x[i] = i + 3;
12    }
13
14    delete[] x;
15 }
16
17
```

joinSpheres.cpp

```
11  /*
12   * Creates a new sphere that contains the exact volume
13   * of the two input spheres.
14   */
15  Sphere joinSpheres(Sphere    s1, Sphere    s2) {
16      double totalVolume = s1.getVolume() + s2.getVolume();
17
18      double newRadius = std::pow(
19          (3.0 * totalVolume) / (4.0 * 3.141592654),
20          1.0/3.0
21      );
22
23      Sphere result(newRadius);
24
25      return result;
26 }
```

CS 225 – Things To Be Doing

Register for Exam 1 (CBTF)

Details on the course website!

Every day, work on the POTDs

Available on PrairieLearn, every weekday!

Finish MP1

Due: Monday, Sept. 11th (11:59pm)

Attend lab and complete lab_debug

Due: Sunday, Sept. 10th (11:59pm)