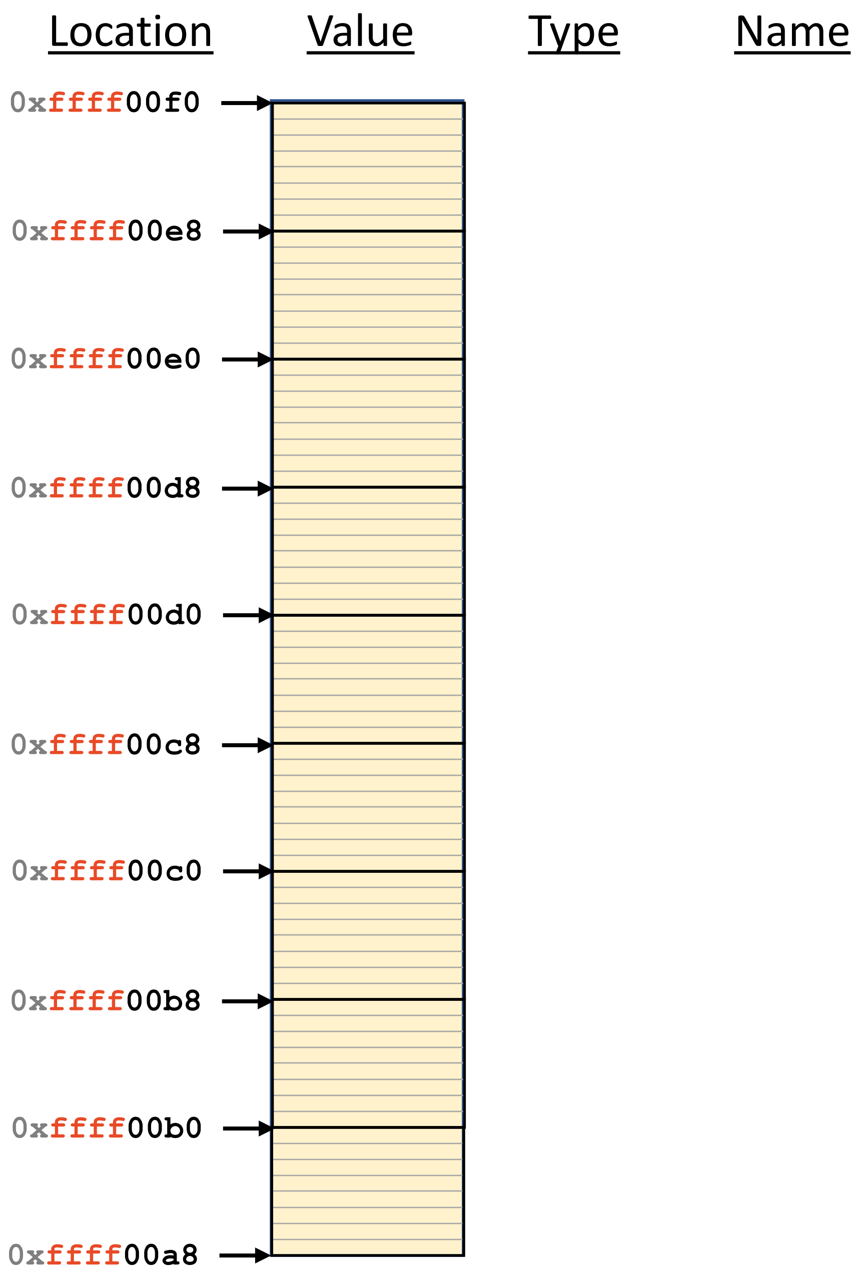
CS 225

Data Structures

Wade Fagen-Ulmschneider

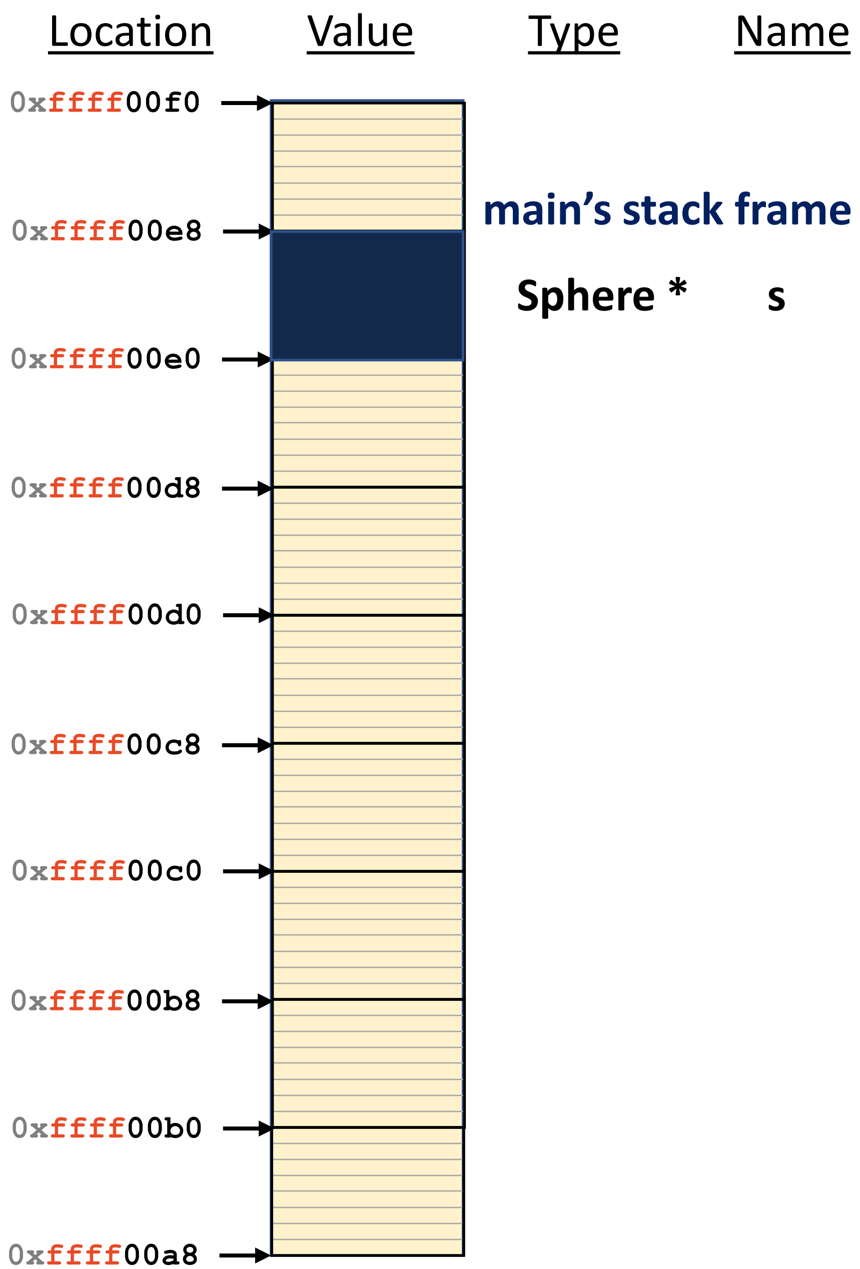


puzzle.cpp

```cpp
#include "sphere.h"
using namespace cs225;

Sphere *CreateUnitSphere() {
  Sphere s(1);
  return &s;
}

int main() {
  Sphere *s = CreateUnitSphere();
  someOtherFunction();
  double r = s->getRadius();
  double v = s->getVolume();
  return 0;
}
```

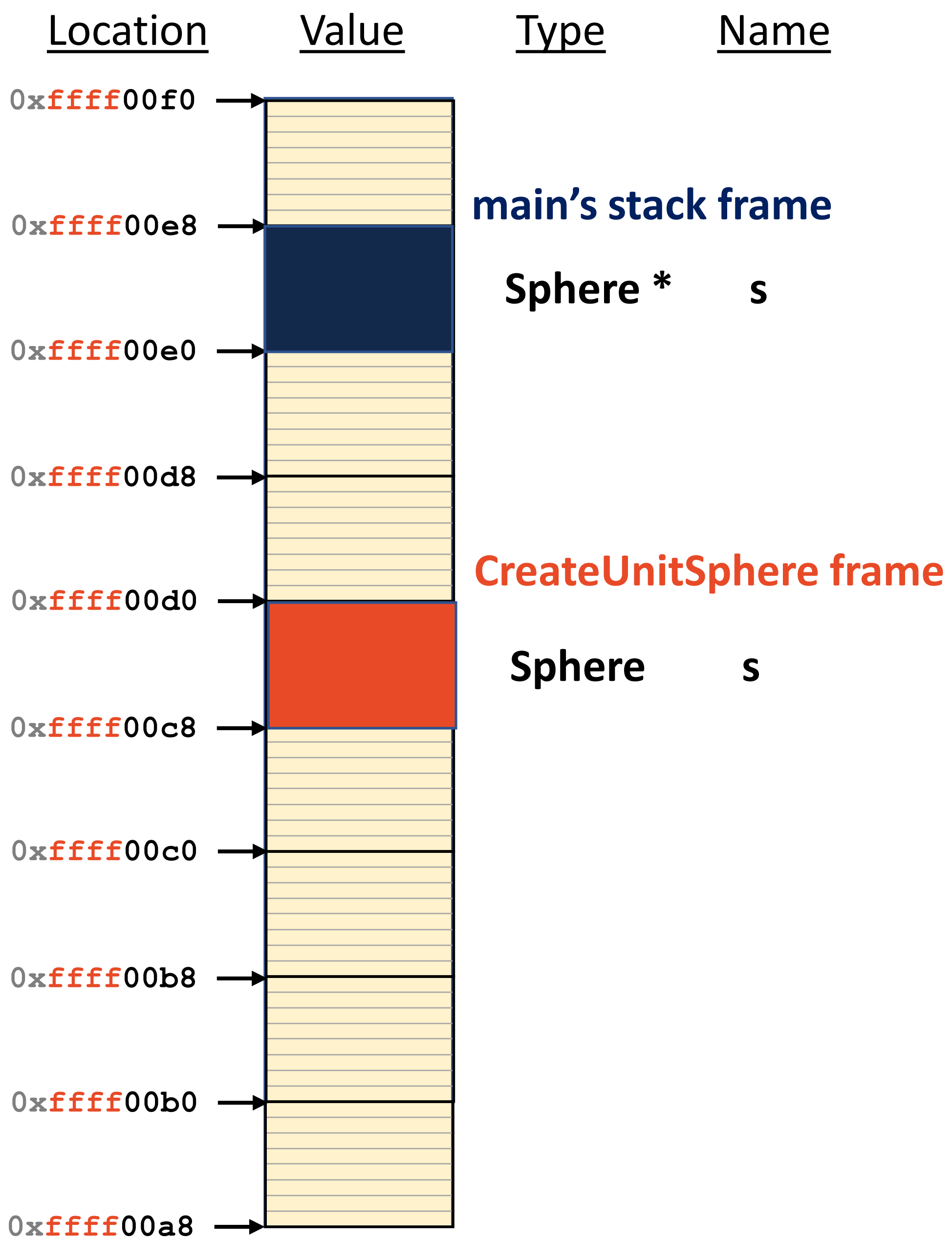


puzzle.cpp

```
#include "sphere.h"
using namespace cs225;

Sphere *CreateUnitSphere() {
  Sphere s(1);
  return &s;
}

int main() {
  Sphere *s = CreateUnitSphere();
  someOtherFunction();
  double r = s->getRadius();
  double v = s->getVolume();
  return 0;
}
```

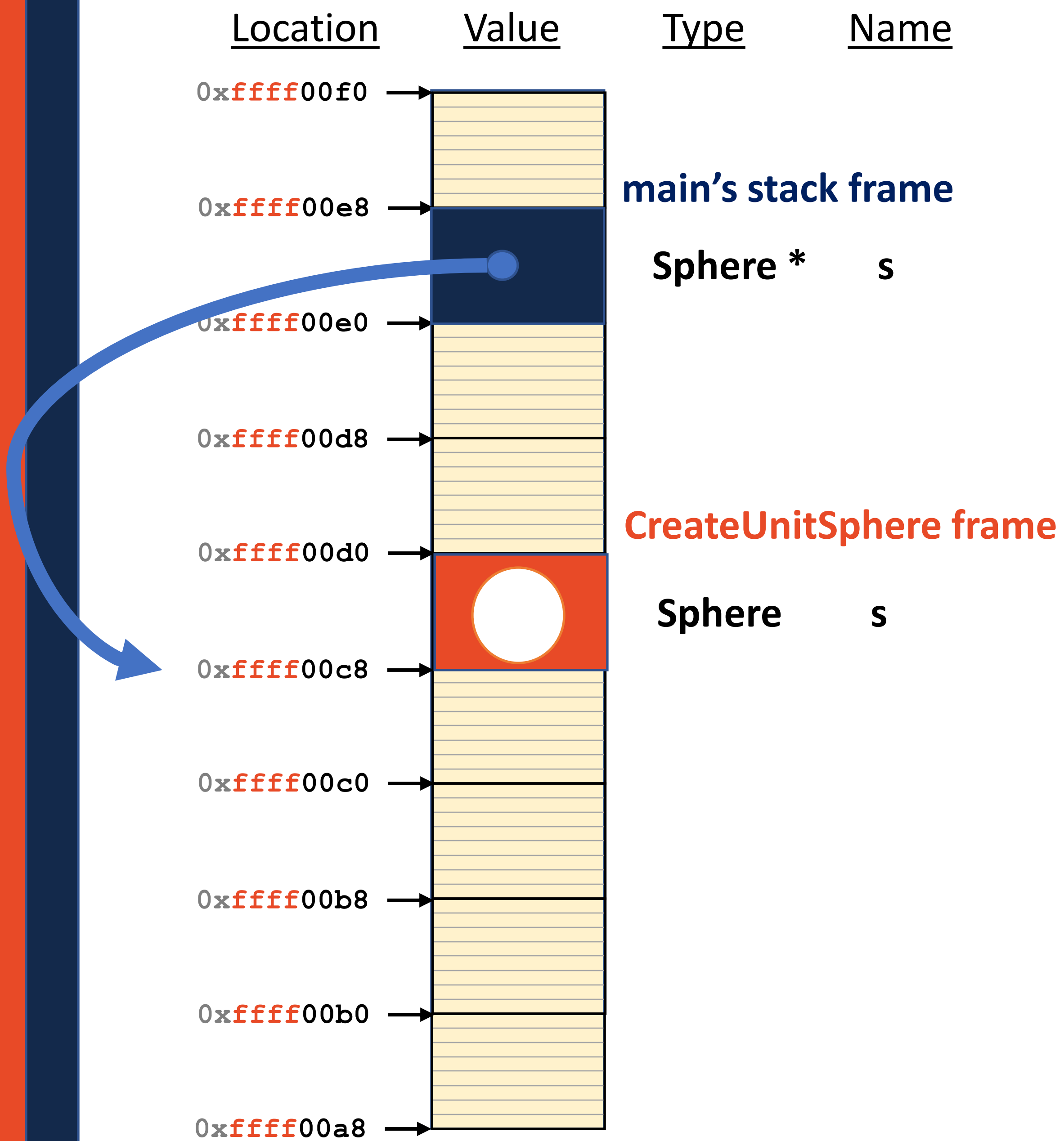


puzzle.cpp

```
#include "sphere.h"
using namespace cs225;

Sphere *CreateUnitSphere() {
  Sphere s(1);
  return &s;
}

int main() {
  Sphere *s = CreateUnitSphere();
  someOtherFunction();
  double r = s->getRadius();
  double v = s->getVolume();
  return 0;
}
```

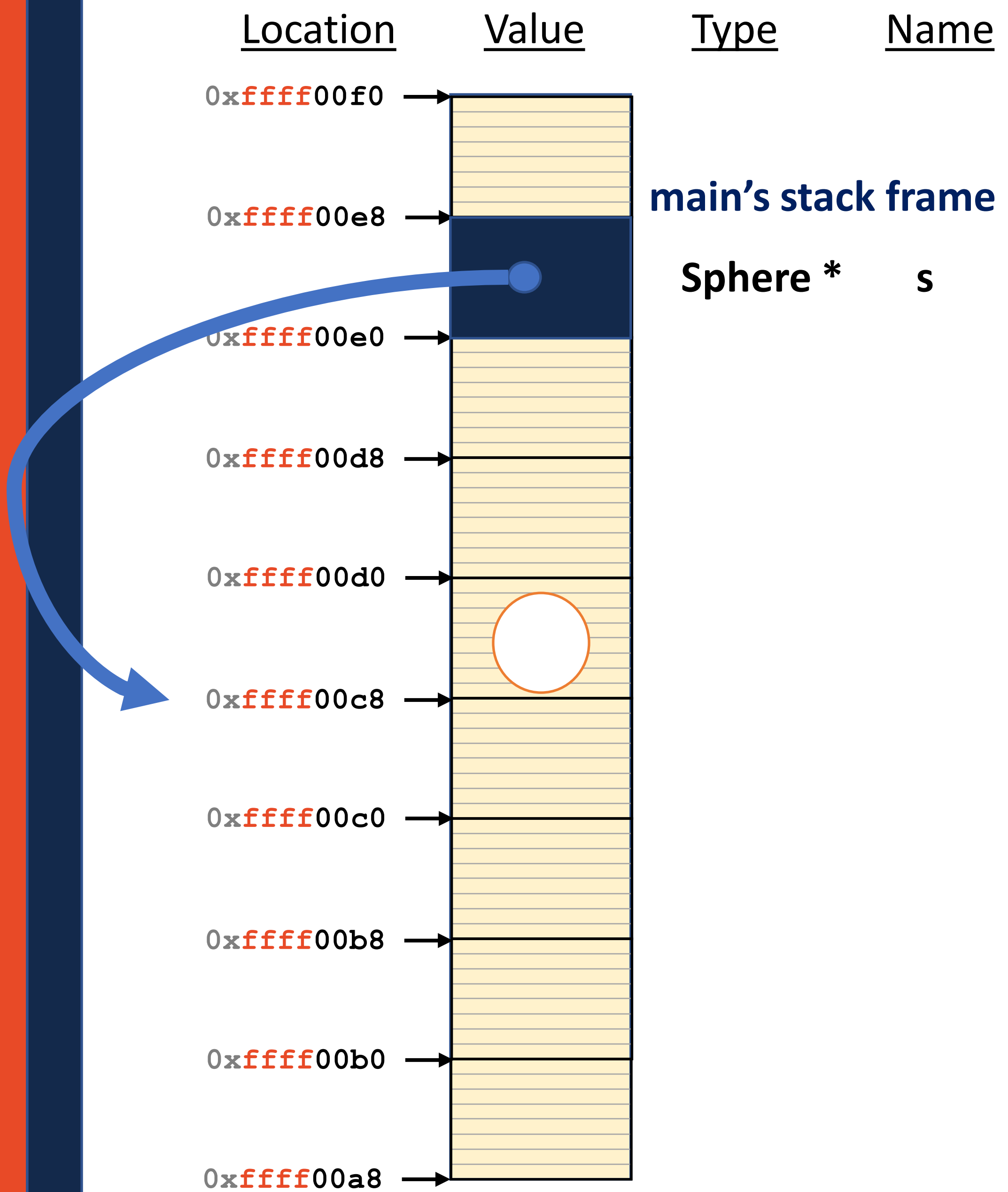


puzzle.cpp

```cpp
#include "sphere.h"
using namespace cs225;

Sphere *CreateUnitSphere() {
  Sphere s(1);
  return &s;
}

int main() {
  Sphere *s = CreateUnitSphere();
  someOtherFunction();
  double r = s->getRadius();
  double v = s->getVolume();
  return 0;
}
```

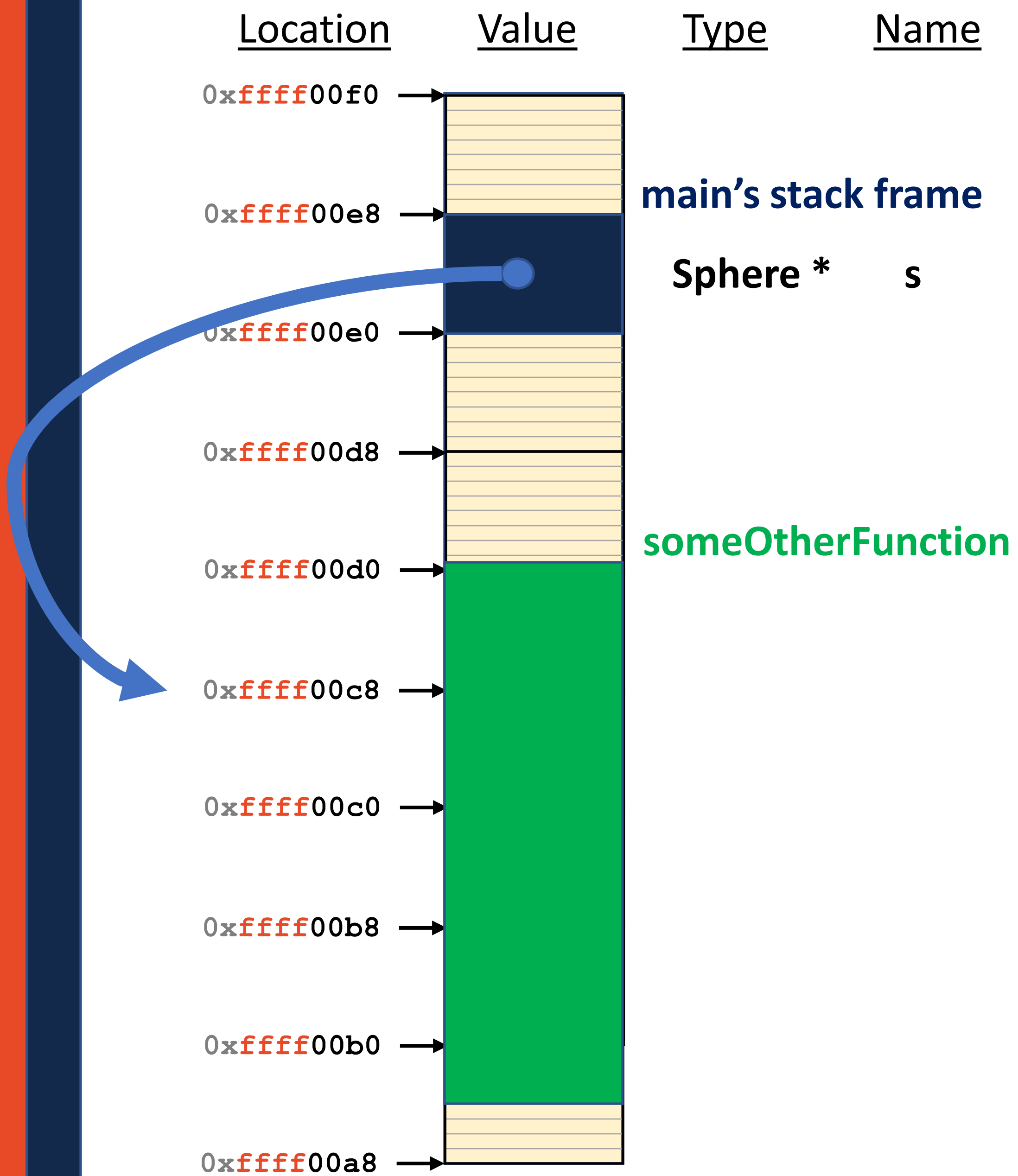


puzzle.cpp

```
#include "sphere.h"
using namespace cs225;

Sphere *CreateUnitSphere() {
  Sphere s(1);
  return &s;
}

int main() {
  Sphere *s = CreateUnitSphere();
  someOtherFunction();
  double r = s->getRadius();
  double v = s->getVolume();
  return 0;
}
```



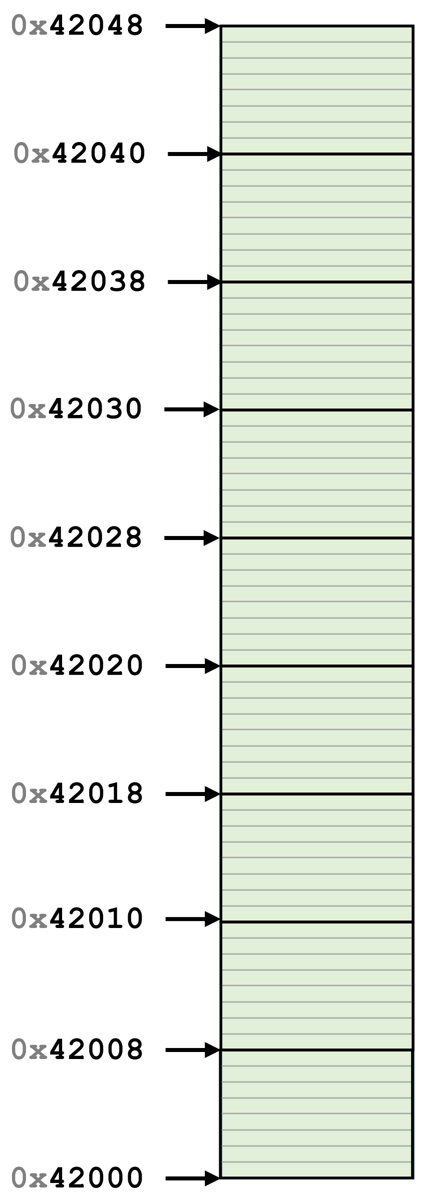
puzzle.cpp

```cpp
#include "sphere.h"
using namespace cs225;

Sphere *CreateUnitSphere() {
  Sphere s(1);
  return &s;
}

int main() {
  Sphere *s = CreateUnitSphere();
  someOtherFunction();
  double r = s->getRadius();
  double v = s->getVolume();
  return 0;
}
```

Heap Memory


0x42048
0x42040
0x42038
0x42030
0x42028
0x42020
0x42018
0x42010
0x42008
0x42000

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

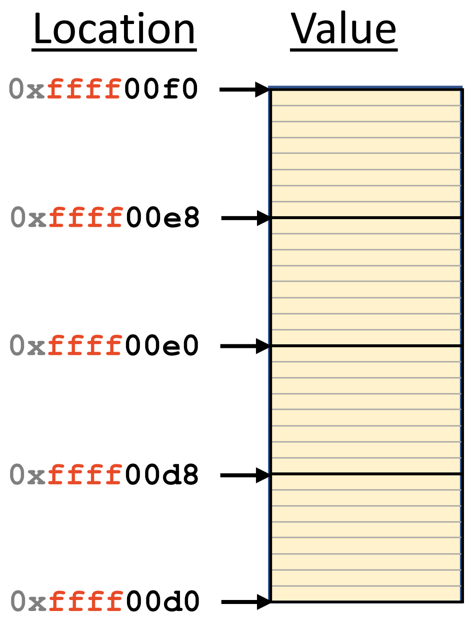
heap1.cpp

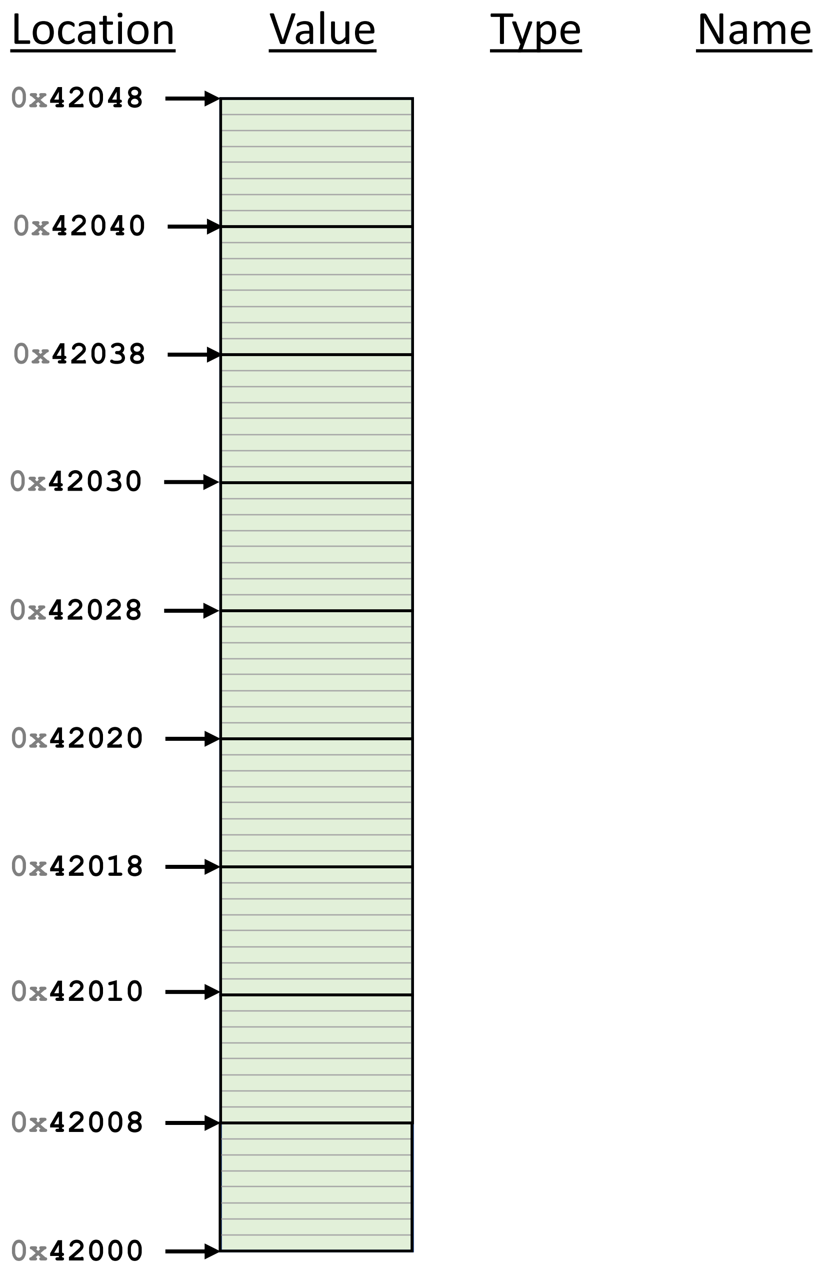
```cpp
#include "sphere.h"
using namespace cs225;

int main() {
  int *p = new int;
  int *s = new Sphere(10);



  return 0;
}
```



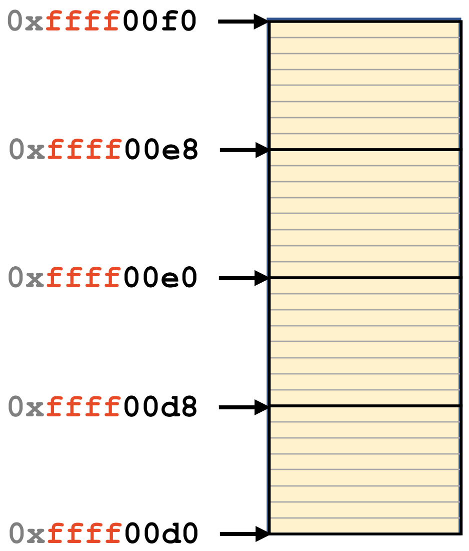


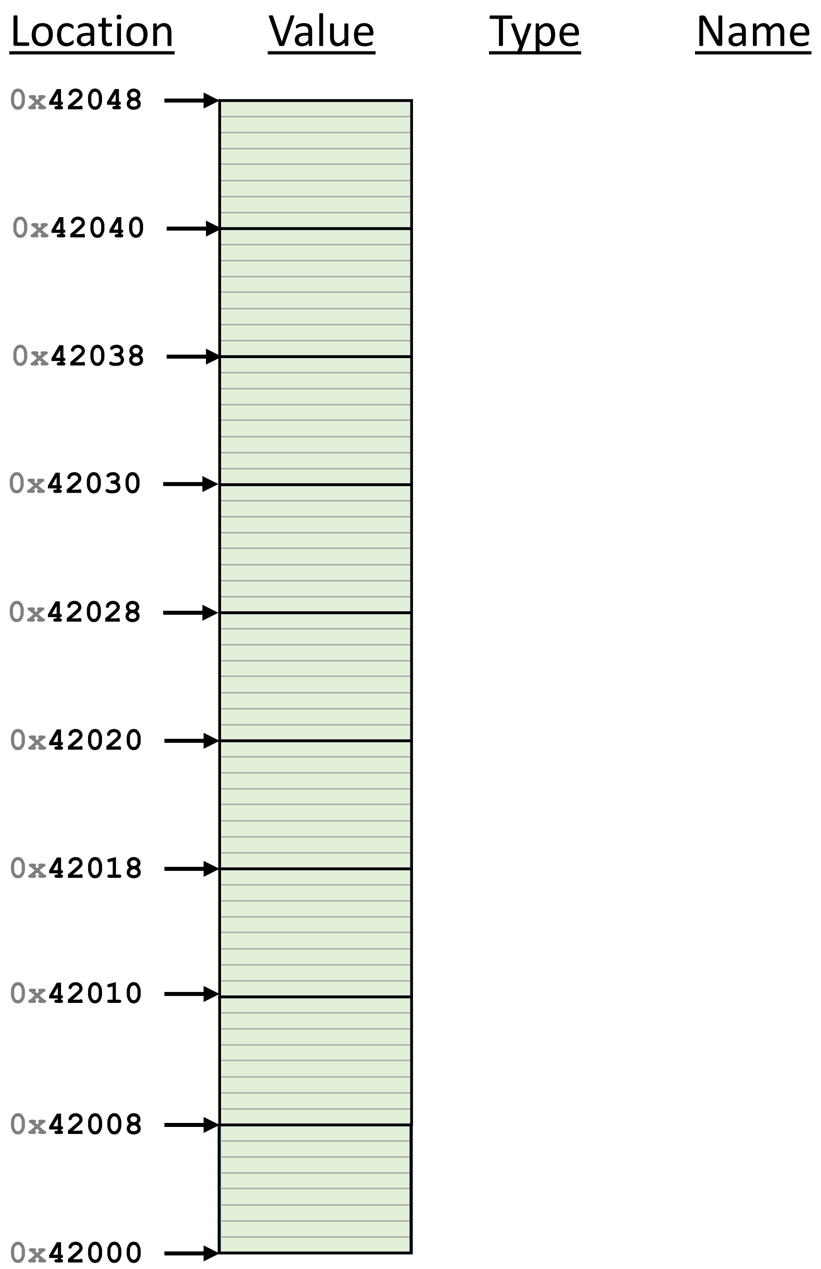
heap2.cpp

```cpp
#include "sphere.h"
using namespace cs225;

int main() {
  Sphere *s1 = new Sphere();
  Sphere *s2 = s1;
  s2->setRadius( 10 );


  return 0;
}
```





Exam 1 Topics

MP1

copy.cpp

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

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;
}
```

heap-puzzle1.cpp

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

int main() {
  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

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

int main() {
  int *p, *q;
  p = new int;
  q = p;
  *q = 8;
  cout << *p << endl;

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

  return 0;
}
```

heap-puzzle3.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;
}


```

joinSpheres.cpp

```cpp
/*
 * Creates a new sphere that contains the exact volume
 * of the two input spheres.
 */
Sphere joinSpheres(Sphere   s1, Sphere   s2) {
  double totalVolume = s1.getVolume() + s2.getVolume();

  double newRadius = std::pow(
    (3.0 * totalVolume) / (4.0 * 3.141592654),
    1.0/3.0
  );

  Sphere result(newRadius);

  return  result;
}
```

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)