CS 225

Data Structures

January 18 – Memory

Wade Fagen-Ulmschneider, Craig Zilles

Pointers and References

A variable containing an instance of an object:

```
Cube s1;
```

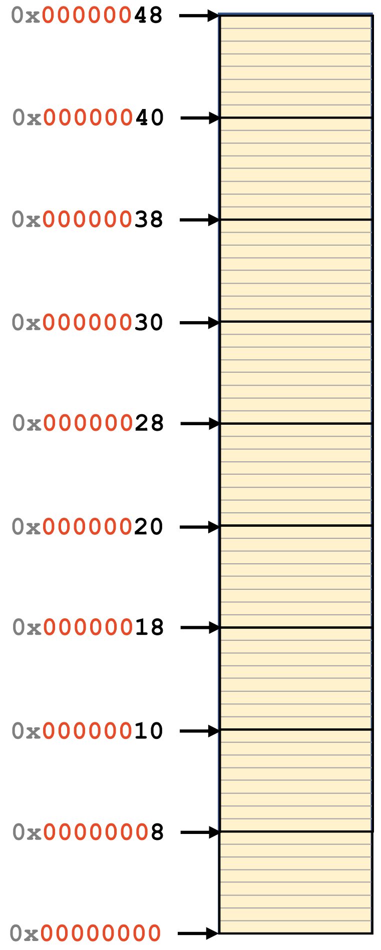
A reference variable of a Cube object:

```
Cube & s1;
```

A variable containing a pointer to a Cube object:

```
Cube * s1;
```

Memory


0x00000048
0x00000040
0x00000038
0x00000030
0x00000028
0x00000020
0x00000018
0x00000010
0x00000008
0x00000000

Pointers

Three key ideas:

1.

2.

3.

main.cpp

```cpp
#include <iostream>
#include "Cube.h"

int main() {
  cs225::Cube c;
  std::cout << "Address storing `c`:" << &c << std::endl;

  cs225::Cube *ptr = &c;
  std::cout << "Addr. storing ptr: "<< &ptr << std::endl;
  std::cout << "Contents of ptr: "<< ptr << std::endl;

  return 0;
}
```

Indirection Operators

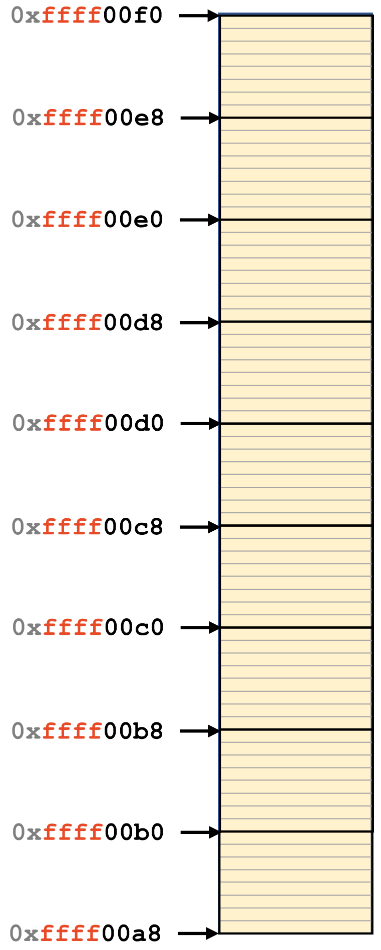
Given any variable v:

&v

***v**

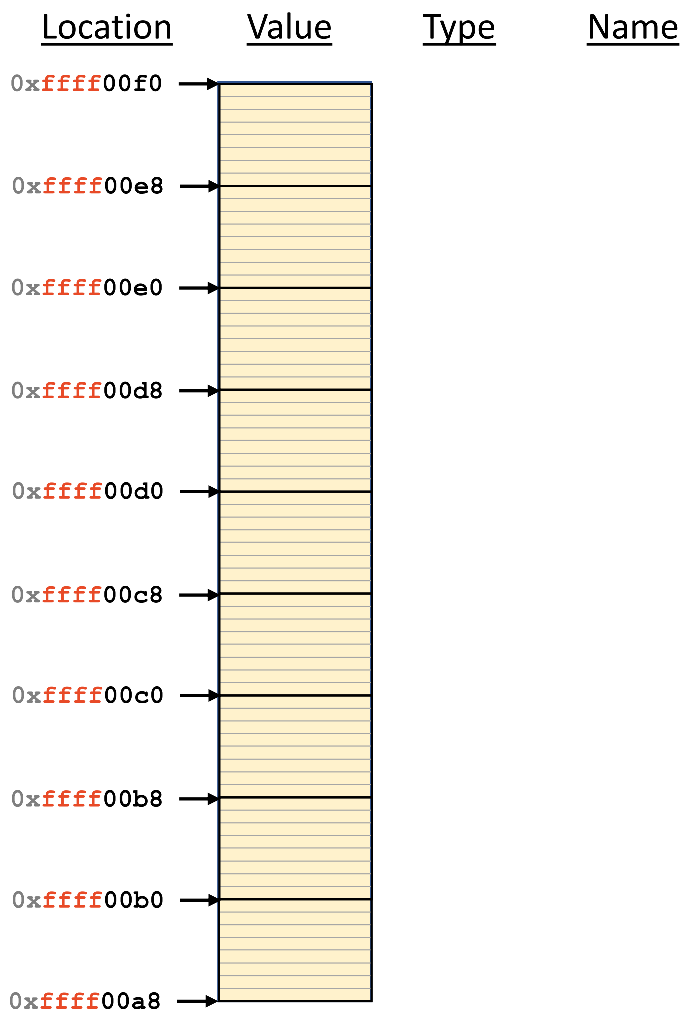
v->

Stack Memory



example1.cpp

```
int main() {
  int a;
  int b = -3;
  int c = 12345;

  int *p = &b;

  return 0;
}
```

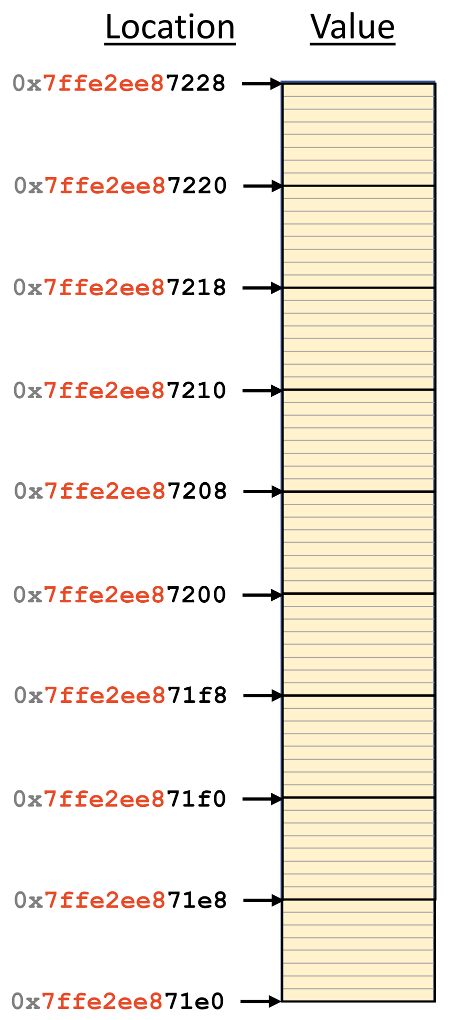


sizeof-int.cpp

```cpp
#include <iostream>

int main() {
  std::cout << sizeof(int) << std::endl;
  return 0;
}
```

sizeof-intptr.cpp

```cpp
#include <iostream>

int main() {
  std::cout << sizeof(int *) << std::endl;
  return 0;
}
```

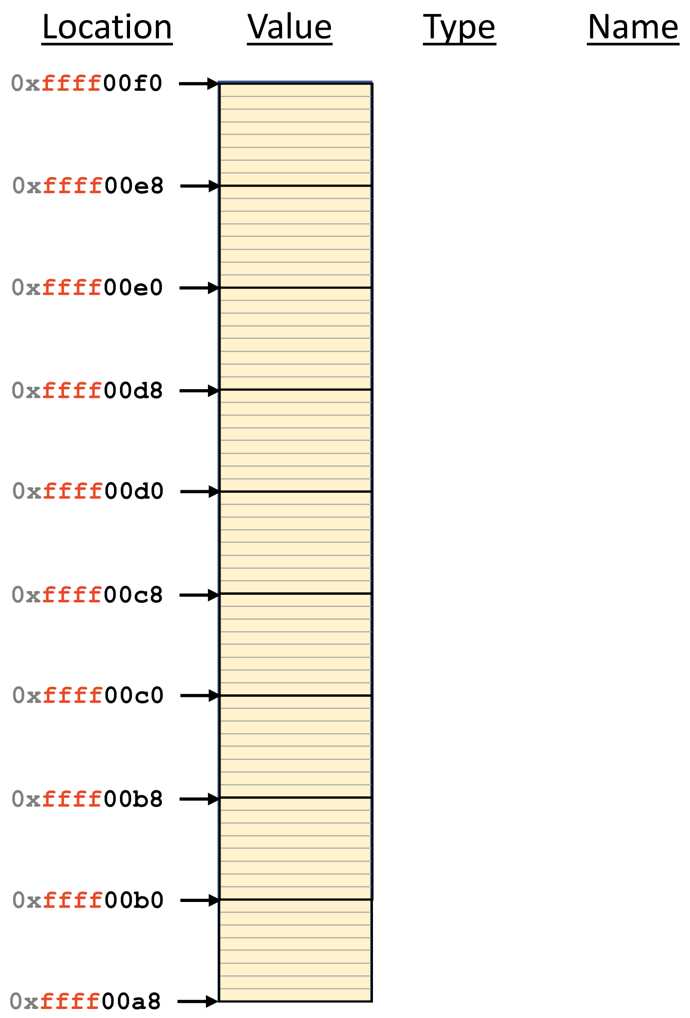


example1.cpp

```cpp
int main() {
  int a;
  int b = -3;
  int c = 12345;

  int *p = &b;

  return 0;
}
```

Real results when running on **linus.ews.illinois.edu**

```
&a:  0x7ffe2ee87218
&b:  0x7ffe2ee87214
&c:  0x7ffe2ee87210
&p:  0x7ffe2ee87208
```



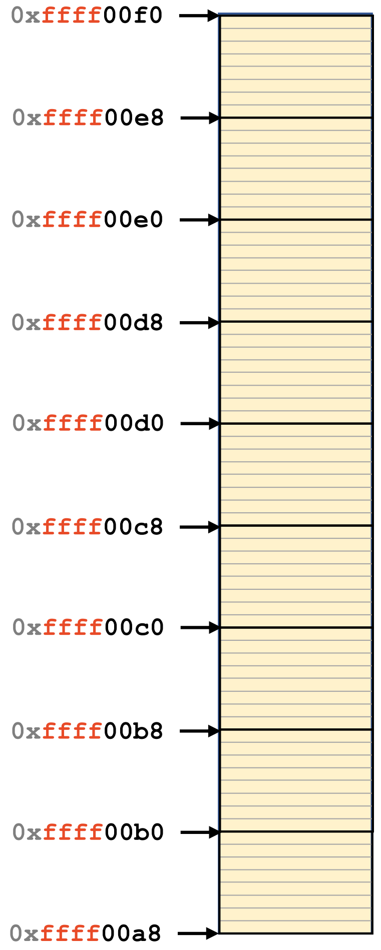
example2.cpp

```
#include "Cube.h"

int main() {
  cs225::Cube c;
  cs225::Cube *p = &c;

  return 0;
}
```

sizeof-cube.cpp

```cpp
#include <iostream>
#include "Cube.h"

int main() {
  std::cout << sizeof(cs225::Cube) << std::endl;
  std::cout << sizeof(cs225::Cube *) << std::endl;
  return 0;
}
```

Stack Frames



stackframe.cpp

```
int hello() {
  int a = 100;
  return a;
}

int main() {
  int a;
  int b = -3;
  int c = hello();
  int d = 42;

  return 0;
}
```

Problems of the Day (POTD)

POTDs are small, daily problems for you to practice programming in an environment similar to the CBTF exam environment.

Each POTD is worth **+1** extra credit point, capped at **+40**.
(Course-wide, all extra credit is capped at +100.)

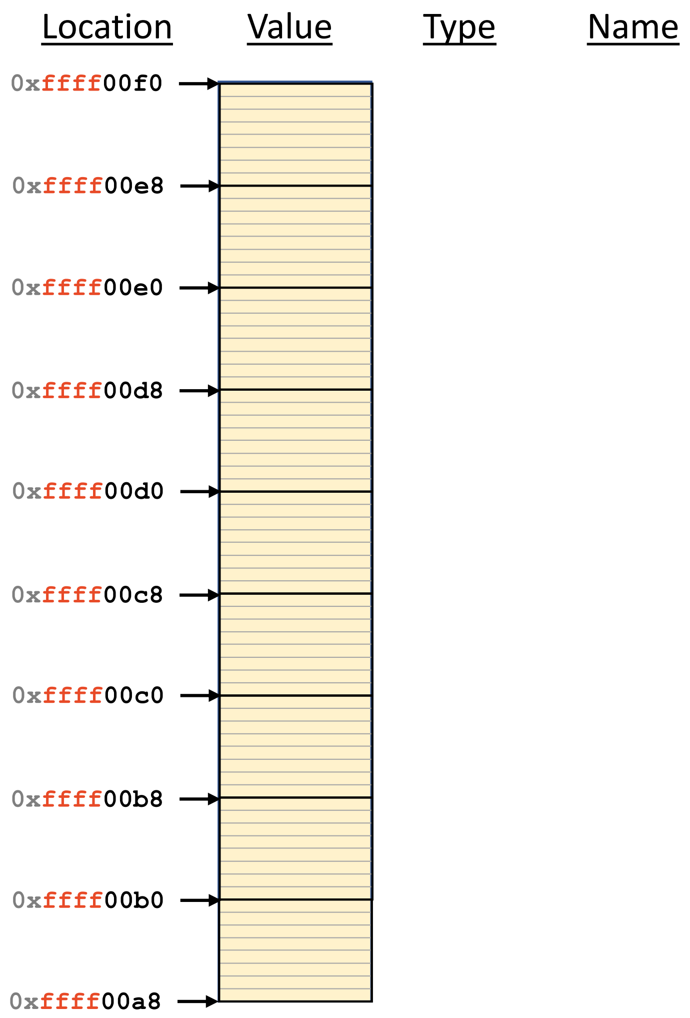
POTD#1 is available on Tuesday, until 8:00am Wednesday morning when POTD#2 becomes available!

Code Reading Questions

Code reading questions are also small problems to practice your programming knowledge (+1 extra credit, capped at 5)

```
int f(int x, int y) {
    if (x > y) {
        return x;
    }

    return y;
}
```

Give a high-level description of the highlighted code



puzzle.cpp

```cpp
#include "Cube.h"
using cs225::Cube;

Cube *CreateCube() {
  Cube c(20);
  return &c;
}

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