Object lifetimes

Every object goes through these stages, over the course of its life.

- **Allocation**: acquire memory for the object
- **Initialization**: create the object
- **Use**: access the object
- **Destruction**: destroy the object
- **Deallocation**: relinquish the object's memory

buy the land

build the building

enjoy - live/work/play

tear down the building

sell back the land

Object Lifetimes for Local Variables: When?

- Allocation: opening { of the function
- Initialization: declaring line
- Use: scope
- Destruction: closing } of the declaring block
- Deallocation: closing } of the function

Functions and Local variables

Functions manage the lifetimes of their *local variables*. In CS70, a function's local variables are:

→ all parameters
→ all variables declared in the body of the func'n.

```
int hello(int x) {   ← allocate 3 ints;
  int y = 5;         ← initialize y
  int z = x+y;       ← initialize z
  return z;          ← initialize parameters
}
```

Function's Perspective

- At the opening {, …
- During a function, for each line of code, …
- At the end of a block, …
- At the end of the function, …

allocate for local variables / initialize parameters

use variables

destroy vars declared in that block

initialize any declared variables

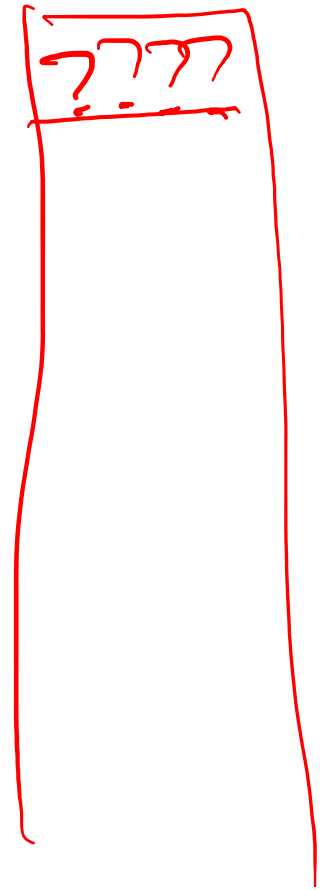
deallocate space

Exercises

int x;

legal but bad idea!!!

cout << x



What is an array?

Why do we have arrays?

(After Homework 4, we'll have `vector`)

C++ primitive arrays

```
// read this declaration "inside out"
int values[42]
```

values is an array of 42 ints

~~int values[];~~

```
// lifecycle rules apply
int weeklyPayments[DAYS_IN_WEEK];
const int weeklyPayments[DAYS_IN_WEEK];
// initialization
int weeklyPayments[DAYS_IN_WEEK] =
{10,5,5,5,5,5,10};
```

Array Idiom

It's okay to default initialize the elements of an array, if we then *immediately* initialize *all* the elements.

```
int weeklyPayments[DAYS_IN_WEEK];

for (size_t day = 0; day < DAYS_IN_WEEK; ++day) {
   cin >> weeklyPayments[day];
}
```

C++ primitive arrays: indexing

```
int weeklyPayments[DAYS_IN_WEEK] =
{10,5,5,5,5,5,10};
cout << weeklyPayments[0] << endl;
cout << weeklyPayments[1] << endl;
```

What happens if we write:

```
int values[3] = {1, 2, 3};
cout << values[10000] << endl;
```

Looking Forward

- Grutoring is up and running!
- Homework 1 is due Wednesday night
- Homework 2 (data visualization with embroidery) is available Thursday