From last week: Scenarios

Students A and B are paired. They sit together in front of one computer. A starts working on the CS 70 assignment. B pulls out a paper copy of a history paper and starts penciling in edits, while occasionally glancing up and making comments on A's code.

From last week: Scenarios

Students A and B are paired. Because they work on different campuses, they work on separate computers in their own dorm rooms using “screen sharing” and on-line chat to discuss and edit the same file at the same time.

From last week: Scenarios

Students A and B are paired. Before they get very far, B falls ill. Several days later, just before the assignment is due, the professors are asked for an extension (because B was too sick all week to work).

Object lifetimes

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

- **Allocation**: acquire memory for the object buying the land
- **Initialization**: create the object building the building
- **Use**: access the object live/work/play
- **Destruction**: destroy the object demolish the building
- **Deallocation**: relinquish the object's memory sell the land

Object Lifetimes for Local Variables: When?

- Allocation: opening { of the function
- Initialization: line w/ the declaration
- 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 of the parameters
→ all variables declared in the body of the func'n.

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 space for local vars
initialize parameters
initialize vars that are declared
destroy vars declared in block - use
destroy params?, vars declared in f'n
deallocate!

Exercises

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;
```

- index out of bounds?
- move down the stack 10000 ints + return whatever is there!

Looking Forward

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