Arrays and iteration

EECS 211

Winter 2019
Initial code setup

$ cd eecs211
$ curl $URL211/lec/04array.tgz | tar zx
...
$ cd 04pointer
Review: variables, objects, values

```c
int main()
{
    int a = 5, b = 10;
    a = 12;
}
```

Variables name objects, which contain values. Assignment changes the value in an object.
int main()
{
    int a = 5, b = 10;
    a = 12;
}

- Variables name objects, which contain values
Review: variables, objects, values

```c
int main()
{
    int a = 5, b = 10;
    a = 12;
}
```

- Variables name objects, which contain values
- Assignment changes the value in an object
Arrays are indexable, aggregate objects

```c
int main()
{
    double a[5];
    a[0] = 1.5;
    a[2] = 3 * a[0];
    --a[0];
}
```
Arrays are indexable, aggregate objects

```c
int main()
{
    double a[5];
    a[0] = 1.5;
    a[2] = 3 * a[0];
    --a[0];
}
```
Arrays are indexable, aggregate objects

```c
int main()
{
    double a[5];
    a[0] = 1.5;
    a[2] = 3 * a[0];
    --a[0];
}
```
Arrays are indexable, aggregate objects

```c
int main()
{
    double a[5];
    a[0] = 1.5;
    a[2] = 3 * a[0];
    a[0]--;  // a[0] is now 4.5
}
```

```
a
  1.5  4.5  
```
Arrays are indexable, aggregate objects

```c
int main()
{
    double a[5];
    a[0] = 1.5;
    a[2] = 3 * a[0];
    --a[0];
}
```

```
0.5 4.5
```
— To the terminal! —
The meaning of while

while (⟨cond⟩) ⟨body⟩

means

if (⟨cond⟩) {
   ⟨body⟩
   if (⟨cond⟩) {
      ⟨body⟩
      if (⟨cond⟩) {
         ⟨body⟩
         ...
      }
   }
}

6
The meaning of `while`, using `goto`:

```
while ⟨cond⟩ ⟨body⟩
```

means

```
start:
    if (!⟨cond⟩) goto finish;
    ⟨body⟩
goto start;
```

```
finish:
```
The meaning of for

\[ \text{for } (\langle \text{init} \rangle; \langle \text{cond} \rangle; \langle \text{step} \rangle) \langle \text{body} \rangle \]

means

\[
\{
\langle \text{init} \rangle; \\
\text{while } (\langle \text{cond} \rangle) \{ \\
\langle \text{body} \rangle \\
\langle \text{step} \rangle;
\}
\}
\]
Idiomatic counting using for

```cpp
for (size_t i = 0; i < limit; ++i) {
    ... i ... 
}
```
Idiomatic counting using for

```c
for (size_t i = 0; i < limit; ++i) {
    ... i ...
}
```

Note:

- We are counting up to limit - 1
Idiomatic counting using `for`

```cpp
for (size_t i = 0; i < limit; ++i) {
    ... i ... 
}
```

Note:

- We are counting up to `limit - 1`
- This is useful because the last element of an array of size `n` is at index `n - 1`