Arrays and iteration

CS 211

Winter 2020

Initial code setup

The code in this course is available online. To download a copy of this lecture into your Unix shell account:

```
% cd cs211
% wget -O- $URL211/lec/04array.tgz | tar zxv
…
% cd 04array
```

Review: variables, objects, values

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

Review: variables, objects, values

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

- Variables name objects, which contain values

Review: variables, objects, values

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

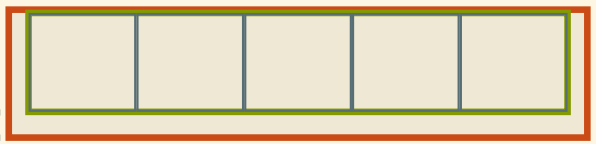
- Variables name objects, which contain values
- Assignment changes the value in an object

Arrays are indexable, aggregate objects

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

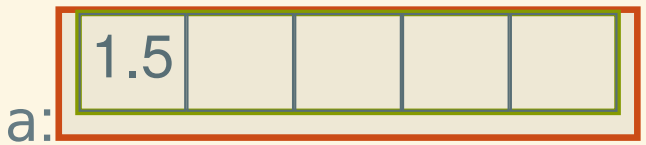
Arrays are indexable, aggregate objects

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



Arrays are indexable, aggregate objects

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



Arrays are indexable, aggregate objects

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

Arrays are indexable, aggregate objects

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

– To the terminal! –

The meaning of `while`

`while (`⟨*cond*⟩`)` ⟨*body*⟩

means

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

The meaning of `while`, using `goto`

`while (`⟨*cond*⟩`)` ⟨*body*⟩

means

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

finish:
```

The meaning of `for`

`for (`⟨*init*⟩`;` ⟨*cond*⟩`;` ⟨*step*⟩`)` ⟨*body*⟩

means

```
{
    ⟨init⟩;
    while (⟨cond⟩) {
        ⟨body⟩
        ⟨step⟩;
    }
}
```

Idiomatic counting using `for`

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

Idiomatic counting using `for`

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

Note:

- We are counting up to `limit - 1`

Idiomatic counting using `for`

```
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`