Structures and All That

September 22, 2019

"Here at Brymar College
We can get you prepared for the 31st century
With advanced programming and quad rendering
And Java plus plus plus scripting language
We offer advanced job placement assitance"
from Upgrade by Deltron 3030

Data Descriptions Matter

We have taken a weird approach by fixating on data structured in the form of “or” first.

Data Descriptions Matter

We have taken a weird approach by fixating on data structured in the form of “or” first.

- We would say that a traffic light’s state is red **or** yellow **or** green. I will sometimes refer to data in this form as defining a *sum type*, but for now I will stick to saying itemization.

Data Descriptions Matter

We have taken a weird approach by fixating on data structured in the form of "or" first.

- We would say that a traffic light's state is red **or** yellow **or** green. I will sometimes refer to data in this form as defining a *sum type*, but for now I will stick to saying itemization.
- But tons of data is written in a compound manner. A person has a head **and** a face **and** a body ... I will sometimes refer to data in this form as being a *product type*, but will usually stick to saying *struct*. When I talk about classes, I will be talking about more than compound data.

Data Descriptions Matter

We have taken a weird approach by fixating on data structured in the form of “or” first.

- We would say that a traffic light’s state is red **or** yellow **or** green. I will sometimes refer to data in this form as defining a *sum type*, but for now I will stick to saying itemization.
- But tons of data is written in a compound manner. A person has a head **and** a face **and** a body ... I will sometimes refer to data in this form as being a *product type*, but will usually stick to saying *struct*. When I talk about classes, I will be talking about more than compound data.
- Whereas with “or” we would check which kind of data we would have and then use a computation specific to that data, with products we can directly project out data.

Data Descriptions Matter

We have taken a weird approach by fixating on data structured in the form of “or” first.

- We would say that a traffic light’s state is red **or** yellow **or** green. I will sometimes refer to data in this form as defining a *sum type*, but for now I will stick to saying itemization.
- But tons of data is written in a compound manner. A person has a head **and** a face **and** a body … I will sometimes refer to data in this form as being a *product type*, but will usually stick to saying *struct*. When I talk about classes, I will be talking about more than compound data.
- Whereas with “or” we would check which kind of data we would have and then use a computation specific to that data, with products we can directly project out data.
- Let’s say that in Java that you have some person class with a first and last name represented as strings.

Data Descriptions Matter

We have taken a weird approach by fixating on data structured in the form of “or” first.

- We would say that a traffic light’s state is red **or** yellow **or** green. I will sometimes refer to data in this form as defining a *sum type*, but for now I will stick to saying itemization.
- But tons of data is written in a compound manner. A person has a head **and** a face **and** a body ... I will sometimes refer to data in this form as being a *product type*, but will usually stick to saying *struct*. When I talk about classes, I will be talking about more than compound data.
- Whereas with “or” we would check which kind of data we would have and then use a computation specific to that data, with products we can directly project out data.
- Let’s say that in Java that you have some person class with a first and last name represented as strings.
- It is easy to define a method that returns the person’s full name by concatenating the first and last name.

Who Needs Structs Anyway?

So, why do we need compound data?

Who Needs Structs Anyway?

So, why do we need compound data?

- The obvious answer is that we have programs that have some kind of compound state.

Who Needs Structs Anyway?

So, why do we need compound data?

- The obvious answer is that we have programs that have some kind of compound state.
- Consider the simple program where we wanted to move a dot left and right.

Who Needs Structs Anyway?

So, why do we need compound data?

- The obvious answer is that we have programs that have some kind of compound state.
- Consider the simple program where we wanted to move a dot left and right.
- We were able to represent the state of the world as a single position number.

Who Needs Structs Anyway?

So, why do we need compound data?

- The obvious answer is that we have programs that have some kind of compound state.
- Consider the simple program where we wanted to move a dot left and right.
- We were able to represent the state of the world as a single position number.
- Let's add another dimension of movement where we can now move the dot up and down.

Who Needs Structs Anyway?

So, why do we need compound data?

- The obvious answer is that we have programs that have some kind of compound state.
- Consider the simple program where we wanted to move a dot left and right.
- We were able to represent the state of the world as a single position number.
- Let's add another dimension of movement where we can now move the dot up and down.
- Can we represent the state of the world as a single number?

Who Needs Structs Anyway?

So, why do we need compound data?

- The obvious answer is that we have programs that have some kind of compound state.
- Consider the simple program where we wanted to move a dot left and right.
- We were able to represent the state of the world as a single position number.
- Let's add another dimension of movement where we can now move the dot up and down.
- Can we represent the state of the world as a single number?
- If you said no, I get it! But that happens to be incorrect.

Who Needs Structs Anyway?

So, why do we need compound data?

- The obvious answer is that we have programs that have some kind of compound state.
- Consider the simple program where we wanted to move a dot left and right.
- We were able to represent the state of the world as a single position number.
- Let's add another dimension of movement where we can now move the dot up and down.
- Can we represent the state of the world as a single number?
- If you said no, I get it! But that happens to be incorrect.
- We can represent a grid with one number in the same sense that we can simulate a 10x10 2D array with a 100 element array.

Structs Make Things Easier

Personally, I like doing things the easy way.

Unstructured Compound Data?

We can actually represent compound data without needing to provide names for the individual pieces of data.

Unstructured Compound Data?

We can actually represent compound data without needing to provide names for the individual pieces of data.

- Let's consider some individual examples in Python.

Unstructured Compound Data?

We can actually represent compound data without needing to provide names for the individual pieces of data.

- Let's consider some individual examples in Python.
- We can represent a Person with a first name, last name, and age with the following kind of tuple:

Unstructured Compound Data?

We can actually represent compound data without needing to provide names for the individual pieces of data.

- Let's consider some individual examples in Python.
- We can represent a Person with a first name, last name, and age with the following kind of tuple:
- `("Peter", "Campora", 26)`

Unstructured Compound Data?

We can actually represent compound data without needing to provide names for the individual pieces of data.

- Let's consider some individual examples in Python.
- We can represent a Person with a first name, last name, and age with the following kind of tuple:
- `("Peter", "Campora", 26)`
- We could then define functions to act like field accesses.

Unstructured Compound Data?

We can actually represent compound data without needing to provide names for the individual pieces of data.

- Let's consider some individual examples in Python.
- We can represent a Person with a first name, last name, and age with the following kind of tuple:
- `("Peter", "Campora", 26)`
- We could then define functions to act like field accesses.

```
def first_name(tup):
  return tup[0]
```

-

Data (Un)Structures

We can actually define other data structures in terms of things like lists.

Data (Un)Structures

We can actually define other data structures in terms of things like lists.

- Let's consider defining a binary tree in terms of a python list.

Data (Un)Structures

We can actually define other data structures in terms of things like lists.

- Let's consider defining a binary tree in terms of a python list.
- We can define a null node with `[]`

Data (Un)Structures

We can actually define other data structures in terms of things like lists.

- Let's consider defining a binary tree in terms of a python list.
- We can define a null node with `[]`
- A tree with a single root element can be `[[] 1 []]`

Data (Un)Structures

We can actually define other data structures in terms of things like lists.

- Let's consider defining a binary tree in terms of a python list.
- We can define a null node with `[]`
- A tree with a single root element can be `[[] 1 []]`
- Here's a nice balanced tree `[[[] 1 []] 2 [[] 3 []]]`

Data (Un)Structures

We can actually define other data structures in terms of things like lists.

- Let's consider defining a binary tree in terms of a python list.
- We can define a null node with `[]`
- A tree with a single root element can be `[[] 1 []]`
- Here's a nice balanced tree `[[[] 1 []] 2 [[] 3 []]]`
- This representation gets a bit ugly fast, huh?

Data (Un)Structures

We can actually define other data structures in terms of things like lists.

- Let's consider defining a binary tree in terms of a python list.
- We can define a null node with `[]`
- A tree with a single root element can be `[[] 1 []]`
- Here's a nice balanced tree `[[[] 1 []] 2 [[] 3 []]]`
- This representation gets a bit ugly fast, huh?
- So Python gives classes (or named tuples) as a way to more easily define such structured data.

Unstructured Data in Racket

Similarly, we can define data using pairs and lists in Racket.

Unstructured Data in Racket

Similarly, we can define data using pairs and lists in Racket.

- We can write a pair $(1, 2)$ as `(1 . 2)`.

Unstructured Data in Racket

Similarly, we can define data using pairs and lists in Racket.

- We can write a pair $(1, 2)$ as `(1 . 2)`.
- We can write a linked list 1-¿2-¿3-¿4-¿empty as `'(1 2 3 4)`

Unstructured Data in Racket

Similarly, we can define data using pairs and lists in Racket.

- We can write a pair $(1, 2)$ as `(1 . 2)`.
- We can write a linked list 1-¿2-¿3-¿4-¿empty as `'(1 2 3 4)`
- To get the first element in the linked list, you can write: `(first '(1 2 3 4))` $\hookrightarrow$ `1`

Unstructured Data in Racket

Similarly, we can define data using pairs and lists in Racket.

- We can write a pair $(1, 2)$ as `(1 . 2)`.
- We can write a linked list 1-¿2-¿3-¿4-¿empty as `'(1 2 3 4)`
- To get the first element in the linked list, you can write: `(first '(1 2 3 4))`$\hookrightarrow$ `1`
- To get the rest of the linked list you can write `(rest '(1 2 3 4))`$\hookrightarrow$ `'(2 3 4)`

Unstructured Data in Racket

Similarly, we can define data using pairs and lists in Racket.

- We can write a pair $(1, 2)$ as `(1 . 2)`.
- We can write a linked list 1-¿2-¿3-¿4-¿empty as `'(1 2 3 4)`
- To get the first element in the linked list, you can write: `(first '(1 2 3 4))`$\hookrightarrow$ `1`
- To get the rest of the linked list you can write `(rest '(1 2 3 4))`$\hookrightarrow$ `'(2 3 4)`
- The empty list is represented with `'()` and you can check for the empty list with `(empty? '())` $\hookrightarrow$ `\#t`

Unstructured Data in Racket

Similarly, we can define data using pairs and lists in Racket.

- We can write a pair $(1, 2)$ as `(1 . 2)`.
- We can write a linked list 1-¿2-¿3-¿4-¿empty as `'(1 2 3 4)`
- To get the first element in the linked list, you can write: `(first '(1 2 3 4))` $\hookrightarrow$ `1`
- To get the rest of the linked list you can write `(rest '(1 2 3 4))` $\hookrightarrow$ `'(2 3 4)`
- The empty list is represented with `'()` and you can check for the empty list with `(empty? '())` $\hookrightarrow$ `\#t`
- We will return to discussing lists in more detail later, since they are *extremely* important.

Unstructured Data in Racket

Similarly, we can define data using pairs and lists in Racket.

- We can write a pair $(1, 2)$ as `(1 . 2)`.
- We can write a linked list 1-¿2-¿3-¿4-¿empty as `'(1 2 3 4)`
- To get the first element in the linked list, you can write: `(first '(1 2 3 4))` $\hookrightarrow$ `1`
- To get the rest of the linked list you can write `(rest '(1 2 3 4))` $\hookrightarrow$ `'(2 3 4)`
- The empty list is represented with `'()` and you can check for the empty list with `(empty? '())` $\hookrightarrow$ `\#t`
- We will return to discussing lists in more detail later, since they are *extremely* important.
- But for now, remember that we wanted to avoid the inconveniences given by using other existing data types to represent some piece of compound data!

The Talk

We said that we didn't want to represent all of our compound data with existing structures like lists are tuples, so let's *finally* talk about structs.

The Talk

We said that we didn't want to represent all of our compound data with existing structures like lists are tuples, so let's *finally* talk about structs.

- Let's reconsider our 2 dimensional movement program.

The Talk

We said that we didn't want to represent all of our compound data with existing structures like lists are tuples, so let's *finally* talk about structs.

- Let's reconsider our 2 dimensional movement program.
- We need a natural representation for cartesian coordiantes for the state of our world.

The Talk

We said that we didn't want to represent all of our compound data with existing structures like lists are tuples, so let's *finally* talk about structs.

- Let's reconsider our 2 dimensional movement program.
- We need a natural representation for cartesian coordiantes for the state of our world.
- We could obviously have the state of our our world be a pair `'(x . y)` or a list `'(x y)`

The Talk

We said that we didn't want to represent all of our compound data with existing structures like lists are tuples, so let's *finally* talk about structs.

- Let's reconsider our 2 dimensional movement program.
- We need a natural representation for cartesian coordiantes for the state of our world.
- We could obviously have the state of our our world be a pair `'(x . y)` or a list `'(x y)`
- But it would be better if we had piece of compound data with two fields, one field named x to represent the x coordinate and similarly a y...