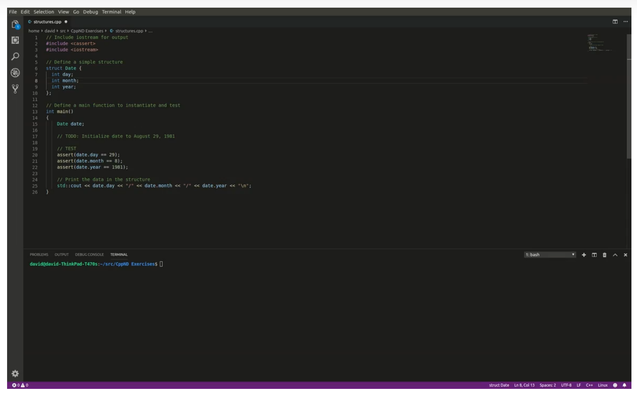

https://youtu.be/iKIEdY_pdzY

Structures

Structures allow developers to create their own types ("user-defined" types) to aggregate data relevant to their needs.

For example, a user might define a `Rectangle` structure to hold data about rectangles used in a program.

```
struct Rectangle {
  float length;
  float width;
};
```

Types

Every C++ variable is defined with a **type**.

```
int value;
Rectangle rectangle;
Sphere earth;
```

In this example, the "type" of `value` is `int`. Furthermore, `rectangle` is "of type" `Rectangle`, and `earth` has type `Sphere`.

Fundamental Types

C++ includes **fundamental types**, such as `int` and `float`. These fundamental types are sometimes called **"primitives"**.

The Standard Library [includes additional types](, such as `std::size_t` and `std::string`.

User-Defined Types

Structures are "user-defined" types. Structures are a way for programmers to create types that aggregate and store data in way that makes sense in the context of a program.

For example, C++ does not have a fundamental type for storing a date. (The Standard Library does include types related to **time**, which can be converted to dates.)

A programmer might desire to create a type to store a date.

Consider the following example:

```
struct Date {
  int day;
  int month;
  int year;
};
```

The code above creates a structure containing three "member variables" of type `int`: `day`, `month` and `year`.

If you then create an "instance" of this structure, you can initialize these member variables:

```
// Create an instance of the Date structure
Date date;
// Initialize the attributes of Date
date.day = 1;
date.month = 10;
date.year = 2019;
```