CS 162: Objects & Structures

Moving beyond the basic types

Objects

- The world is filled with them
- Objects are often composed of sub-objects
- Examples:

    Book → title, pages, authors

    Protein Bar → nutritional info, ingredients, manufacturer

    Car → engine, wheels, frame, seats, etc

    Engine → spark plugs, pistons, rods, crankshaft, gaskets

How does this relate to C++?

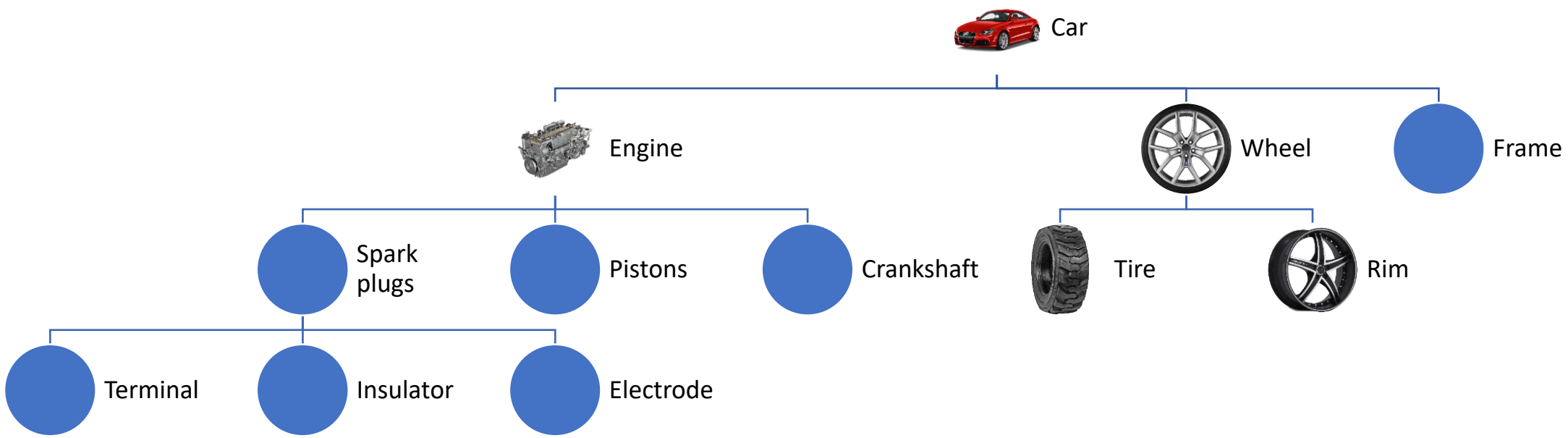
- While programing, we often want to “group” variables
  - Arrays give us a primitive technique (items must be the same type)

    ```
    int grades[150];
    ```
  - We want more flexibility

- What if:
  - We weren’t limited to the primitive variable type?
  - We had the ability to store multiple types of variables in the same container?

Consider the hierarchy of a car

Simplified… obviously.



Images from pngimg.com

C++ allows this behavior

- We can define custom objects called structures
  - Our custom structure can mix and match data types
  - Let's call this a "struct"

- Traditionally, structures contain only data (no member functions)
  - Classes are used when you want member functions
  - Like functions, structures must be defined before usage

How to define a struct

```
struct book {
     int pages;
     unsigned int pub_date;
     string title; // a string inside the struct
     int num_authors;
     string* authors; // a pointer to a string
};

// declare a book struct
book text_book;
```

Working with structs

- Can use the same way as any other type
- The . operator allows us to access the member variables

```
book bookshelf[10];
for (int i = 0; i < 10; i++) {
     bookshelf[i].num_pages = 100;
     bookshelf[i].title = “Place holder”;
     bookshelf[i].authors = new string[2];
}
```

Using pointers with structs

```
book bk1; // statically allocated
book* ptr1 = &bk1;

// dereference the pointer and access the data member
(*ptr1).title = “Old Yeller”;

// a shortcut to dereference the struct
// the -> operator
ptr1->title = “Old Yeller (abridged)”;
ptr1->num_pages = 196;

// this works for objects on the heap as well
book* ptr2 = new book; // dynamically allocated
ptr2->title = “Charlotte’s Web”;
```