Structs and Arrays

Now that structs have been introduced, you have one more tool to use when representing data in your programs. As always, though, it’s important to know when to use that tool and when to take another approach.

Consider a program that needs to process information about fifty different cars. When designing the program, its author came up with two different ways of holding all the car data needed, shown in examples a) and b), below.

```c
struct car_info {
    string make;
    string model;
    int year;
    float mpg;
};
car_info info[50];
```

```c
string makes[50];
string models[50];
int years[50];
float mpgs[50];
```

Approach a) uses an array of structs, while approach b) uses multiple individual arrays to store the data.

For this part of the exercise, answer the following questions (and explain your answers!):
Are both methods capable of fully representing all the car data needed in the program? Which method do you think would be easier to design a program around? If you had to add an extra car attribute (color, for example) after writing the program, how difficult would it be to add that extra data when using either approach? And finally, which method do you prefer and why?

Arrow Operator

Just like other data types, it’s frequently useful to use pointers to structs. Up until now we’ve always used the dot operator to access a struct’s members— but when we have pointers to structures, you have to dereference the pointer first before using the dot operator to access the member variable.

Alternatively, you can use the arrow operator to access the members variables of pointers to structs directly. Below are two functions that do exactly the same thing, except one uses the dot operator and the other uses the arrow operator.

```c
void print_year(car_info *car) {
    cout << (*car).year << endl;
}
```

```c
void print_year(car_info *car) {
    cout << car->year << endl;
}
```

As practice, you’re going to implement a function called is better. The function takes two pointers to car_info structures as parameters, and returns a bool. Its return value will be true if and only if the first car’s year and mpg members are both greater than the second car’s.

Implement this function twice— once only using the arrow operator to access member variables, and once only using the dot operator.

Course Reflection

For this section of the exercise, just answer the following questions about your experiences in the course. We’ll use your responses to make future CS courses better!

• What was the hardest topic in the class for you? Why do you think you had a difficult time with it?
• If you could pick an assignment to re-do, which would you choose? What would you do differently?
• What was your favorite and least favorite part about the course?

For take-home exercises completed in peer-led groups, each student must participate in the class discussion and write answers to each of the questions on his/her own paper to show for credit.

For take-home exercises completed on your own, turn in your work electronically using the TEACH website.