

CSC 209 Review 7 Solution

August 26, 2020

1 Exercises

1. First, I need to justify if the following declarations are legal on an individual basis:

```
struct {int x, y;} x;  
struct {int x, y;} y;
```

The first `struct {int x, y;} x;` is legal. `struct {int x, y;} x;` is equivalent to

```
1  struct {  
2      int x;  
3      int y;  
4  } x;
```

and 'x' beside struct represents variable of that type. It is used to declare struct and access members of the struct (e.g. `x.x`, `x.y`).

The same is true for `struct {int x, y;} y;`.

Second, I need to answer if both declarations of struct can appear in a program.

The answer is yes. Each structure has a separate name space for its members.

Notes

- **Declaring Structure Variables**
 - Struct can have many variables that represent the same struct

```
struct part {  
    int number;  
    char name[NAME_LEN+1];  
    int on hand;  
} part1, part2;
```

members of struct

variables that represent
this struct

- **Initializing Structure Variables**

- Struct can be initialized with preset values (like python class under `__init__`)

```
struct {  
    int number;  
    char name[NAME_LEN+1];  
    int on hand;  
} part1 = {528, "Disk drive", 10},  
   part2 = {914, "Printer cable", 5};
```

values
that initialize
structure variables

2. a) I need to declare structure variables named `c1`, `c2` and `c3`, each having members `real` and `imaginary` of type `double`.

The solution to this problem is:

```
1  struct {  
2      double real, imaginary;  
3  } c1, c2, c3;
```

- b) I need to modify the declaration in part a) so that

- `c1`'s members initially have the values 0.0 and 1.0
- `c2`'s members initially have the values 1.0 and 0.0
- `c3` is not initialized

The solution to this problem is:

```

1  struct {
2      double real, imaginary;
3  } c1 = {0.0, 1.0},
4      c2 = {1.0, 0.0},
5      c3;

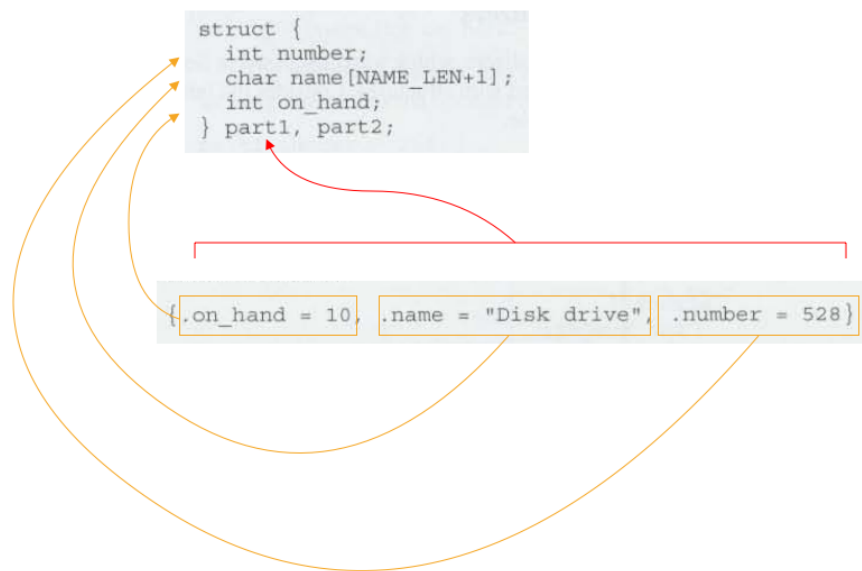
```

Notes

- **Designated Initializer**

- Allows specific member variable to be initialized
- Allows member variables to be initialized in any order

Example



- c) I need to write statements that add the corresponding members of `c1` and `c2` and store the result in `c3`.

The solution to this problem is:

```

1  struct {
2      double real, imaginary;
3  } c1 = {0.0, 1.0},
4      c2 = {1.0, 0.0},
5      c3;
6
7  ...
8
9  c3 = c1 + c2;

```

Notes

- member variables of struct contains two operators & and . (e.g &part1.number and part1.number)
- & accesses memory address of the member variable, where as . accesses value
- part1 = part2 copies contents in part2 to corresponding member variable in part1

```
struct {  
    int number;  
    char name[NAME_LEN+1];  
    int on_hand;  
} part1, part2;
```