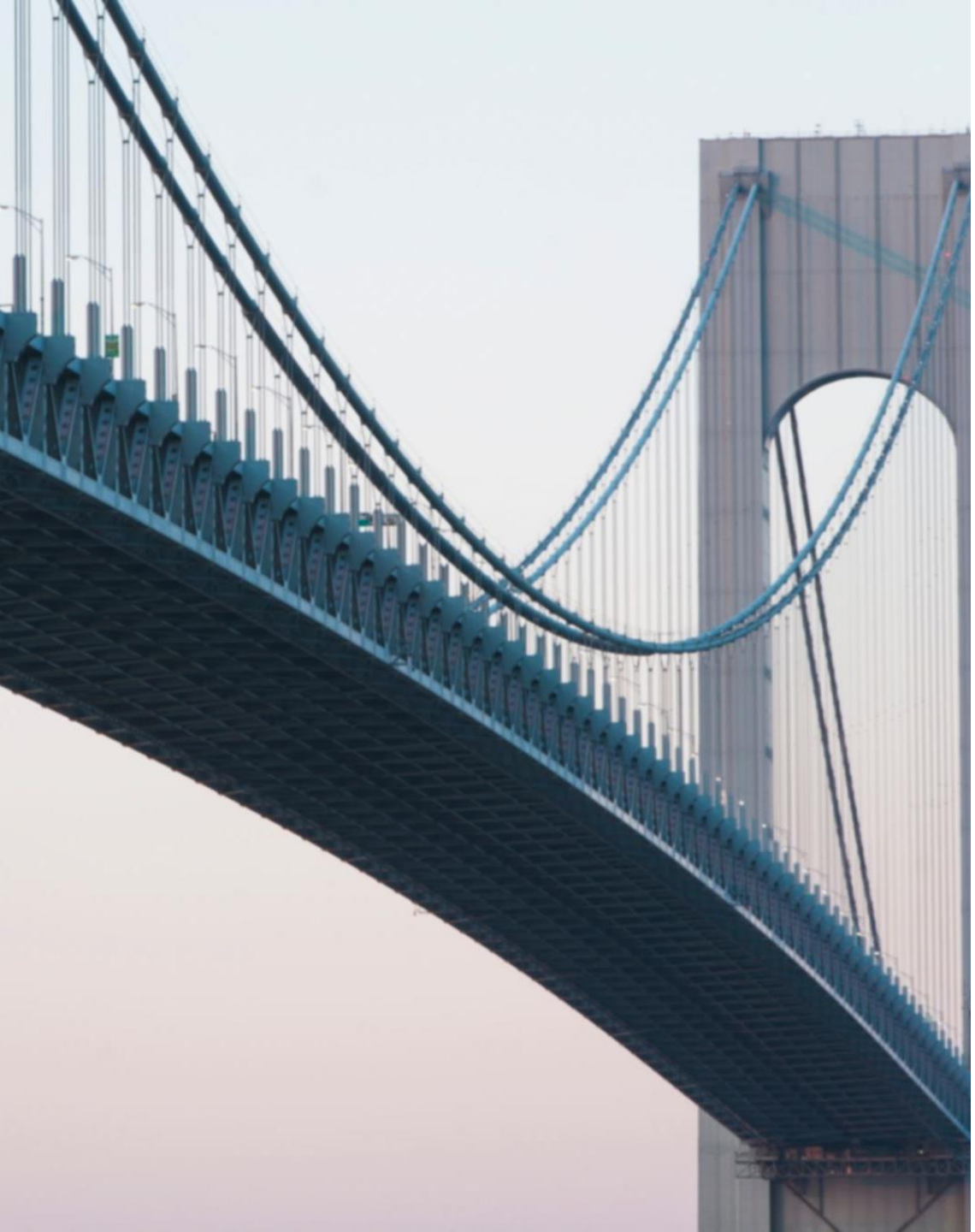


Structure

Why we need structure?

Structure is a user-defined datatype in C language which allows us to combine data of different types together.

Structure helps to construct a complex data type which is more meaningful.

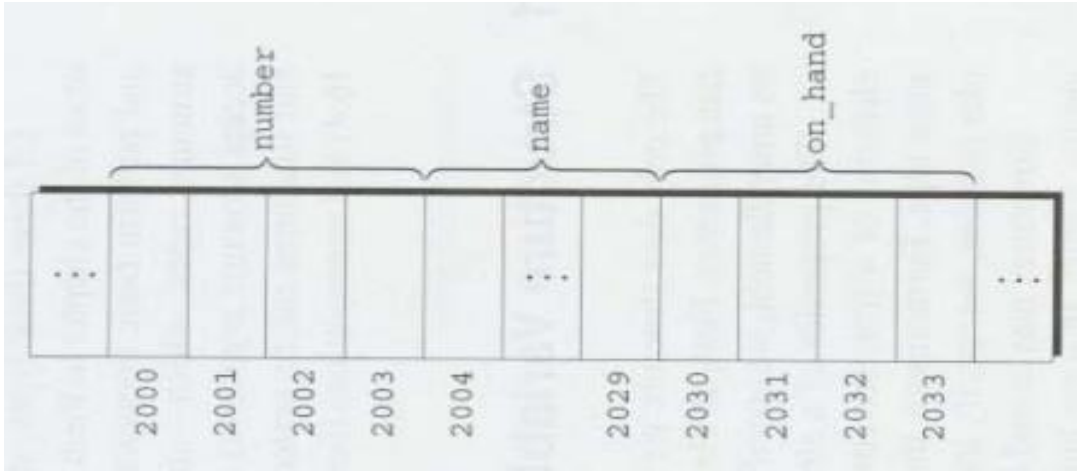
It is somewhat similar to an Array, but an array holds data of similar type only. But structure on the other hand, can store data of any type, which is practical more useful.

Array vs Structure

- For example: If I have to write a program to store Student information, which will have Student's name, age, branch, permanent address, father's name etc, which included string values, integer values etc, how can I use arrays for this problem, I will require something which can hold data of different types together.

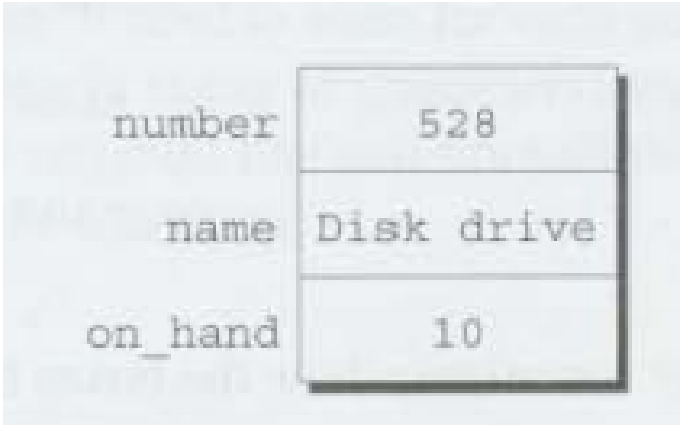
Declaring Structure Variables

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



Initializing Structure Variables

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



Designated Initializer

```
{528, "Disk drive", 10}
```

A designated initializer would look similar, but with each value labeled by the name of the member that it initializes:

```
{.number = 528, .name = "Disk drive", .on_hand = 10}
```

The combination of the period and the member name is called a ***designator.*** (Designators for array elements have a different form.)

```
{.on_hand = 10, .name = "Disk drive", .number = 528}
```

```
{.number = 528, "Disk drive", .on_hand = 10}
```

Declaring a Structure tag

A ***structure tag*** is a name used to identify a particular kind of structure. The following example declares a structure tag named `part`:

```
struct part {
  int number;
  char name[NAME_LEN+1];
  int on_hand;
};
```

Notice the semicolon that follows the right brace—it must be present to terminate the declaration.

Operations on Structure

- All most all kinds of operations can be performed on structure
- Print
- Scan
- Assignment
- Mathematical
- Array of Structure
- Pointer
- Pass as function arguments
- And others

Problem Set

To illustrate how nested arrays and structures are used in practice, we'll now develop a fairly long program that maintains a database of information about parts stored in a warehouse. The program is built around an array of structures, with each structure containing information—part number, name, and quantity—about one part. Our program will support the following operations:

- ***Add a new part number, part name, and initial quantity on hand.*** The program must print an error message if the part is already in the database or if the database is full.
- ***Given a part number, print the name of the part and the current quantity on hand.*** The program must print an error message if the part number isn't in the database.
- ***Given a part number, change the quantity on hand.*** The program must print an error message if the part number isn't in the database.
- ***Print a table showing all information in the database.*** Parts must be displayed in the order in which they were entered.
- ***Terminate program execution.***

We'll use the codes `i` (insert), `s` (search), `u` (update), `p` (print), and `q` (quit) to resent these operations. A session with the program might look like this:

```
Enter operation code: i
Enter part number: 528
Enter part name: Disk drive
Enter quantity on hand: 10

Enter operation code: s
Enter part number: 528
Part name: Disk drive
Quantity on hand: 10

Enter operation code: s
Enter part number: 914
Part not found.

Enter operation code: i
Enter part number: 914
Enter part name: Printer cable
Enter quantity on hand: 5

Enter operation code: u
Enter part number: 528
Enter change in quantity on hand: -2

Enter operation code: s
Enter part number: 528
Part name: Disk drive
Quantity on hand: 8

Enter operation code: p
Part Number   Part Name                 Quantity on Hand
    528       Disk drive                               8
    914       Printer cable                            5

Enter operation code: q
```

The End