Structures

Array of Structures

- Same as declaring an array of any other type
- Example

```
s_type students[50];
```

  - Declares an array of 50 s_type variables
  - Each element can be accessed using students[<index>]
  - A field of the structure variable element can be accessed using students[<index>].fieldName
    - `students[0].age` or `students[0].fname`

Passing Structures to Functions

- Structures can also be passed-by-value or passed-by-reference
- The syntax for doing so is similar to that of normal variables
- Recall: Operations on Structures
  - Assignment statements between structure variables of the ***same structure data type***
  - Taking the address of a structure variable

Passing Structures by Value

- Pass-by-value can be done because it is possible to assign the value of one structure of the same type to another
- Example:

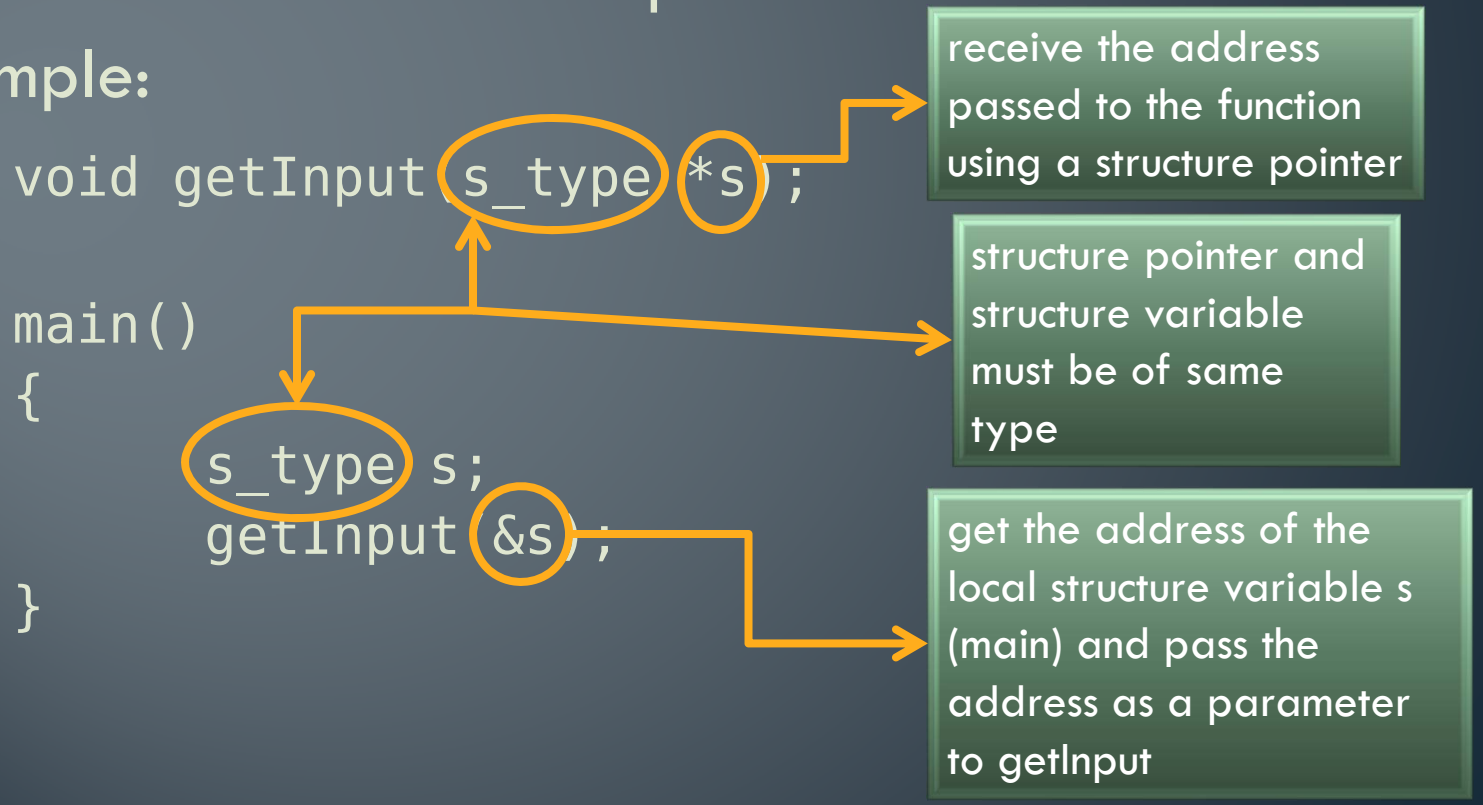
```
int printVariable(s_type s);

int main()
{
        s_type s;
        … //get variable data
        printVariable(s);
}
```

think of it as assigning the local structure variable s from main to the local structure variable s in printVariable ☺

Passing Structures by Reference

- Since the address of a structure variable can be computed using the address operator, it can be passed to a function as a parameter
- Example:



Remarks (1)

- A structure can't have an instance of itself as a field
- It ***can*** have a ***pointer to an instance of itself*** as a field
- Example. Which of the following is valid?

```
typedef struct s_tag
{
      char fname[50];
      int age;
      struct s_tag *p;
} s_type;
```

```
typedef struct s_tag
{
      char fname[50];
      int age;
      struct s_tag s;
} s_type;
```

Remarks (2)

- Recall:
  - A function can not be called unless it is declared first
  - A variable can not be used unless it is declared first
- A structure type can not be used unless it has been defined first.
- Example: Which of the following is valid?

```
typedef struct
{
      char fname[50];
      int age;
      struct s_tag *p;
} s_type;
```

```
typedef struct s_tag
{
      char fname[50];
      int age;
      struct s_tag *p;
} s_type;
```

Remarks(3)

- Therefore, in order to have a pointer to an instance of itself as a field, a structure ***must always have a structure tag***