# **Structures**

#### **CS10001: Programming & Data Structures**



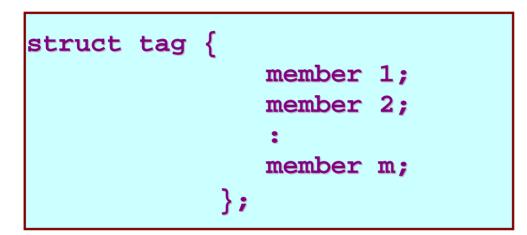
Pallab Dasgupta Professor, Dept. of Computer Sc. & Engg., Indian Institute of Technology Kharagpur

# What is a Structure?

- It is a convenient tool for handling a group of logically related data items.
  - Examples:
    - Student name, roll number, and marks.
    - Real part and complex part of a complex number.
- This is our first look at a non-trivial data structure.
   Helps in organizing complex data in a more meaningful way.
- The individual structure elements are called *members*.

# **Defining a Structure**

• The composition of a structure may be defined as:



- struct is the required keyword.
- tag is the name of the structure.
- member 1, member 2, ... are individual member declarations.

### Contd.

- The individual members can be ordinary variables, pointers, arrays, or other structures.
  - The member names within a particular structure must be distinct from one another.
  - A member name can be the same as the name of a variable defined outside of the structure.
- Once a structure has been defined, the individual structure-type variables can be declared as:

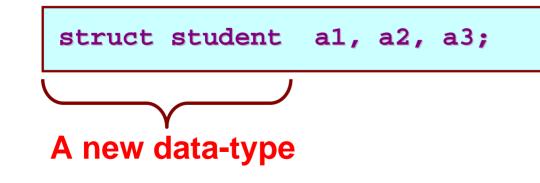
struct tag var\_1, var\_2, ..., var\_n;

#### Example

#### • A structure definition:

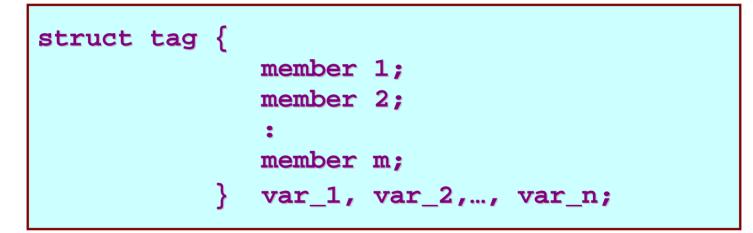
struct	student	{	int	<pre>name[30]; roll_number; total marks;</pre>
				dob[10];
		};		

#### • Defining structure variables:



# **A Compact Form**

• It is possible to combine the declaration of the structure with that of the structure variables:



• In this form, "tag" is optional.

#### **Equivalent Declarations**

struct student	{	
		<pre>char name[30];</pre>
		<pre>int roll_number;</pre>
		<pre>int total_marks;</pre>
		<pre>char dob[10];</pre>
	}	a1, a2, a3;

struct	{
	<pre>char name[30];</pre>
	int roll_number;
	<pre>int total_marks;</pre>
	char dob[10];
	} a1, a2, a3;

#### **Processing a Structure**

- The members of a structure are processed individually, as separate entities.
- A structure member can be accessed by writing variable.member
  - where *variable* refers to the name of a structure-type variable, and *member* refers to the name of a member within the structure.
- Examples:

```
al.name, a2.name, al.roll_number, a3.dob
```

#### **Example: Complex number addition**

```
#include <stdio.h>
main()
{
      struct complex
             float real;
             float cmplex;
       } a, b, c;
       scanf ("%f %f", &a.real, &a.cmplex);
       scanf ("%f %f", &b.real, &b.cmplex);
      c.real = a.real + b.real;
      c.cmplex = a.cmplex + b.cmplex;
      printf ("\n %f + %f j", c.real, c.cmplex);
```

# **Comparison of Structure Variables**

- Unlike arrays, group operations can be performed with structure variables.
  - A structure variable can be directly assigned to another structure variable of the same type.

a1 = a2;

- All the individual members get assigned.
- Two structure variables can be compared for equality or inequality.

if (a1 == a2).....

• Compare all members and return 1 if they are equal; 0 otherwise.

### **Arrays of Structures**

Once a structure has been defined, we can declare an array of structures.

struct student class[50];

- The individual members can be accessed as: class[i].name class[5].roll\_number

# **Arrays within Structures**

• A structure member can be an array:

• The array element within the structure can be accessed as:

al.marks[2]

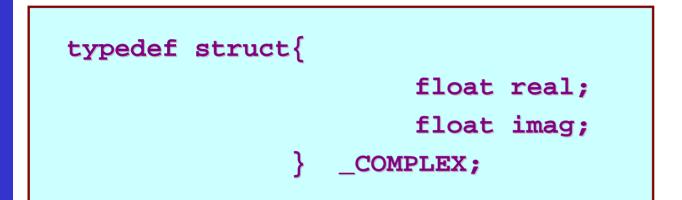
# Defining data type: using typedef

- One may define a structure data-type with a single name.
- General syntax:

typedef struct {	
	<pre>member-variable1;</pre>
	member-variable2;
	•
	member-variableN;
}	tag;

• tag is the name of the new data-type.

# typedef : An example



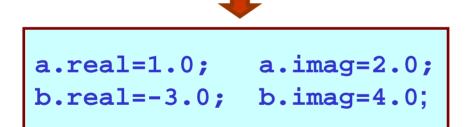
\_COMPLEX a, b, c;

Dept. of CSE, IIT KGP

### **Structure Initialization**

- Structure variables may be initialized following similar rules of an array. The values are provided within the second braces separated by commas.
- An example:

\_COMPLEX  $a = \{1.0, 2.0\}, b = \{-3.0, 4.0\};$ 



### **Parameter Passing in a Function**

 Structure variables can be passed as parameters like any other variables. Only the values will be copied during function invocation.

```
void swap (_COMPLEX a, _COMPLEX b)
{
    _COMPLEX tmp;
    tmp=a;
    a=b;
    b=tmp;
}
```

### **An Example**

```
#include <stdio.h>
typedef struct{
                     float real;
                     float imag;
         } _COMPLEX;
void swap (_COMPLEX a, _COMPLEX b)
    _COMPLEX tmp;
    tmp = a;
    a = b;
    b = tmp;
```

### **Example:: contd.**

```
void print (_COMPLEX a)
            printf("(%f, %f) \n",a.real,a.imag);
main()
            _COMPLEX x = \{4.0, 5.0\}, y = \{10.0, 15.0\};
            print(x); print(y);
            swap(x,y);
            print(x); print(y);
```

#### <u>Output</u>:

(4.000000, 5.000000) (10.000000, 15.000000) (4.000000, 5.000000) (10.000000, 15.000000)

# **Returning structures**

• It is also possible to return structure values from a function. The return data type of the function should be as same as the data type of the structure itself.

```
_COMPLEX add(_COMPLEX a, _COMPLEX b)
{
    _COMPLEX tmp;
    tmp.real = a.real + b.real;
    tmp.imag = a.imag + b.imag;
    return(tmp);
}
```

Direct arithmetic operations are not possible with structure variables.



#### **Exercise Problems**

- 1. Extend the complex number program to include functions for addition, subtraction, multiplication, and division.
- 2. Define a structure for representing a point in two-dimensional Cartesian co-ordinate system.
  - Write a function to compute the distance between two given points.
  - Write a function to compute the middle point of the line segment joining two given points.
  - Write a function to compute the area of a triangle, given the coordinates of its three verwices.