CS F111: Computer Programming

(Second Semester 2020-21)

Lect 22: Structures contd..

Dr. Nikumani Choudhury
Asst. Prof., Dept. of Computer Sc. & Information Systems
nikumani@hyderabad.bits-pilani.ac.in

BITS Pilani

Hyderabad Campus

Structure Definitions

General format

```
struct [tag_name]
{
   data_type   member1;
   data_type   member2;
   ...
   data_type   membern;
} [var1, var2, …];
```

Example1

```
struct f_year
{
   char name[30];
   char id[15];
   float marks;
};
```

```
struct f_year stud1;
```

Example2

```
typedef struct
{
   char name[30];
   char id[15];
   float marks;
} f_year;
```

```
f_year stud1;
```

variable declarations

name	id	marks	

f_year

Multiple Variable Declarations

```
struct f_year
{
   char name[30];
   char id[15];
   float marks;
};
```

```
struct f_year
stud1, stud2,
stud3;
```

```
struct
{
   char name[30];
   char id[15];
   float marks;
} stud1, stud2, stud3;
```

(Definition and declaration are combined)

Note: Members do not occupy any memory until they are associated with structure variables.

Accessing Structure Members

- The variables stud1, stud2, etc. are real-life object of type struct f_year.
- The individual members (name, id, marks) are connected to their respective variables with a dot, the member operator (direct selection operator).
- The members of stud1 are accessed as: stud1.name, stud1.id, and stud1.marks
- Ex: stud1.marks = 76.5

```
stud1.id = "2019A7PS0007H"
stud1.name = "Amit Agarwal"
```

id, and name are character arrays that signify constant pointers…

Structure Initialization

```
#include <stdio.h>
#include <string.h>
struct f_year {
 char name[30];
 char id[15];
 float marks;
}stud1={"Amit", "2019A7PS0010H"};
```

Partial Initialization

Rules:

1. Cannot initialize individual members within the template
2. The values must be in the order of definitions
3. Uninitialized members should be at the end of the list
4. Uninitialized members will have default values: int/float (0/0.0) and char ('\0')

```
int main(){
 struct f_year stud2 = {"Raj", "2019A7PS0006H", 98.0};
 printf ("student 1 marks: %f", stud1.marks);
 printf ("\nstudent 1 id:%s", stud1.id);
 printf ("\nstudent 2 marks: %f", stud2.marks);
return 0;
}
```

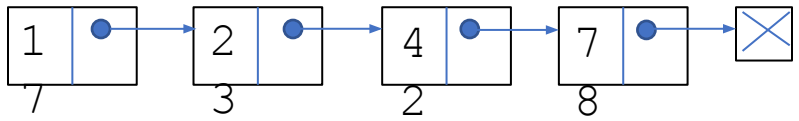
```
student 1 marks: 0.000000
student 1 id:2019A7PS0010H
student 2 marks: 98.000000
```

Important attributes of Structures

- As a structure can include int, char, float, struct, union, pointers, etc… as its' member, operations are restricted.
- There are no name conflicts like in:
  - ```
    struct x {
     int x;
     char y[10];
     }x, y;
    ```

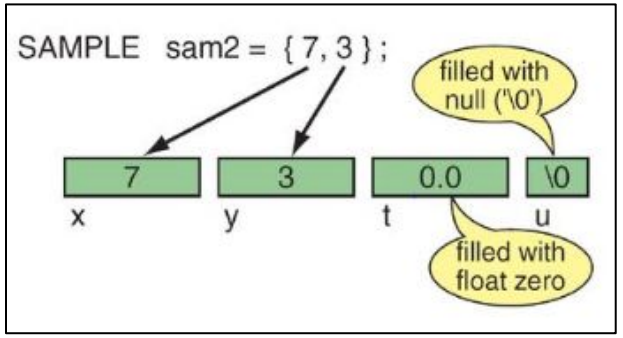
  But, a programmer (you) may not like the idea.

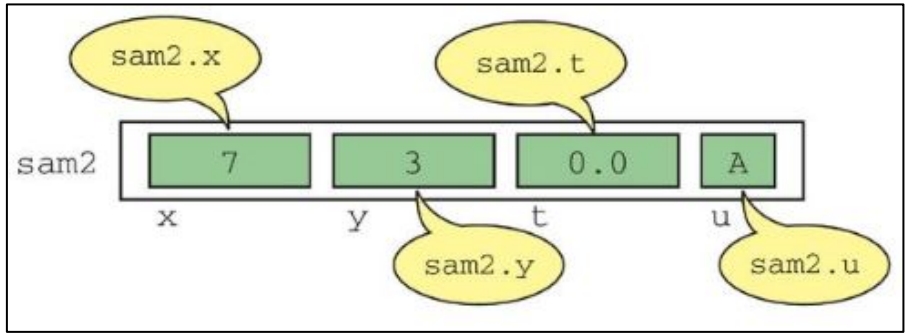
- Dot (.), arrow (->), =, & are the only operators used on structures
- A structure can be copied to another structure if both are based on same template.
  - struct stud1 = {"Amit", "2019A7PS0010H", 86.5};
  - struct stud3 = stud1; (Only operation allowed on a structure)
- A struct is passed-by-value to a function & entire struct is copied.
- A member may refer to itself (self-referential): struct node {int data; struct node *next;}


    ```

Referencing individual fields

```
typedef struct {
 int x;
 int y;
 float t;
 char u;
}SAMPLE;
```






```
scanf("%c", &sam2.u);
 …
if (sam2.u == 'A')
  sam2.x += sam2.y * 18;
```

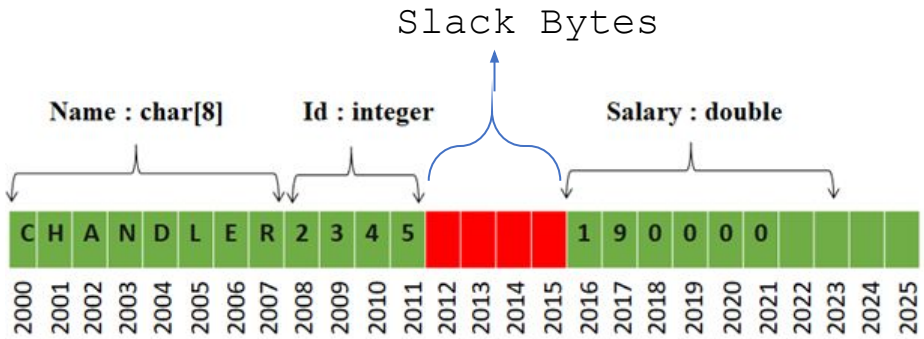
Precedence:

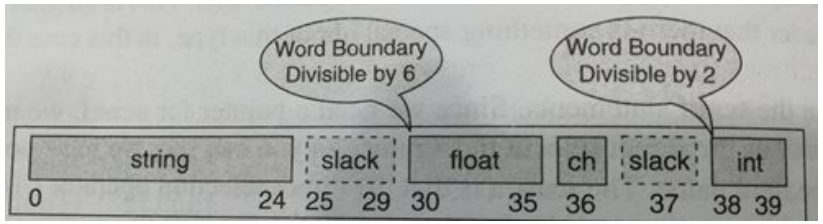
```
++sam2.x;  ++(sam2.x);
fun(&sam2.y);  &(sam2.y) passed
```

Comparing two Structures

- We cannot compare two structures directly because of slack bytes being inserted by the compiler to maintain word boundaries.
- Values of these slack bytes are beyond the control of the program. Hence, compare them member by member.

```
struct EMPLOYEE{ char name[8]; int id; double salary;};
```

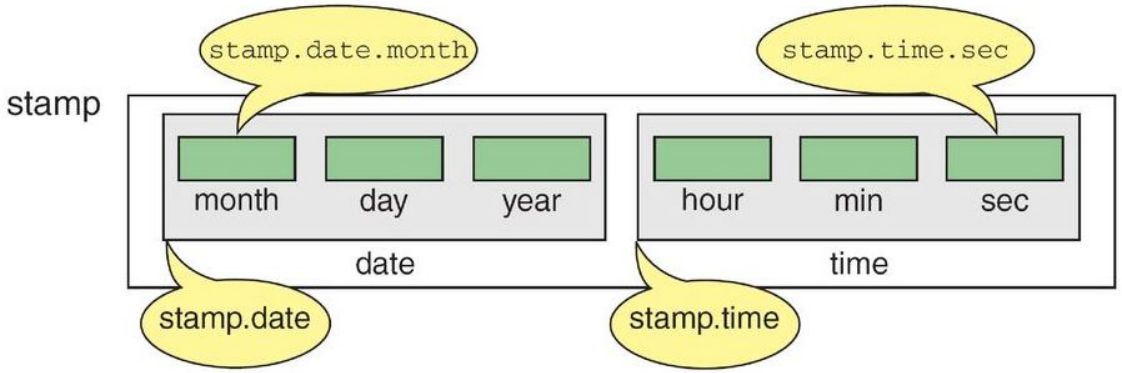




Another Example

Nested Structures

- Structures as members of another structure



- Each structure must be initialized completely before proceeding to next member

```
STAMP stamp = {{09,04,2020},{09,25,10}};
```

```
typedef struct {
  int month;
  int day;
  int year;
} DATE;

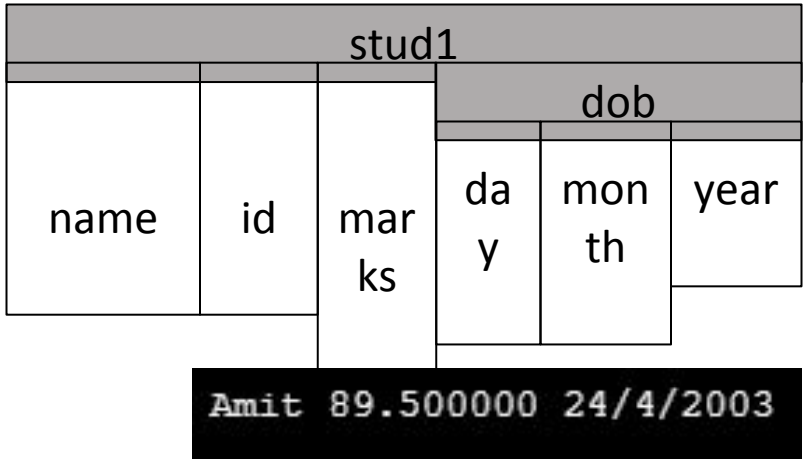
typedef struct {
  int hour;
  int min;
  int sec;
} TIME;

Typedef struct {
  DATE date;
  TIME time;
} STAMP;

STAMP stamp;
```

Another Example

```
#include <stdio.h>
int main() {
    typedef struct  {
        int day;
        int month;
        int year;
    } date;
    typedef struct  {
        char name [30];
        char id [15];
        float marks;
        date dob;
   } student;
   student stud1 = {"Amit", "2019A7PS0007H", 89.5, {24, 4, 2003}};
   printf("\n %s %f %d/%d/%d", stud1.name,
         stud1.marks,stud1.dob.day,stud1.dob.month,
stud1.dob.year);
return 0;
}
```



Find the Output

```c
#include<stdio.h>
struct Ournode{
    char x,y,z;
};
int main(){
    struct Ournode p = {'1', '0', 'a'+2};
    struct Ournode *q = &p;
    printf ("%c", *((char*)q+2) );
    return 0;
}
```

```c
#include <stdio.h>

int main()
{

    struct A {

        int x;

        double z;

        short int y;

    };

    printf("Size of struct: %ld", sizeof(struct A));

    return 0;}
```