CS F111: Computer Programming

(Second Semester 2020-21)

Lect 23 Structures & Unions

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

BITS Pilani

Hyderabad Campus

Pointers as Structure members

- The use of pointers save a lot of memory.
- If the month in the stamp structure has to be stored as string in place of integer?

```
char jan[ ] ="January";
char feb[ ] ="February";
…
```

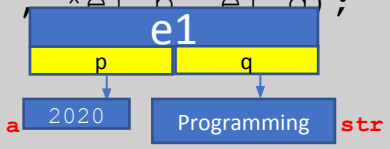
```
stamp.date.month = feb;
```

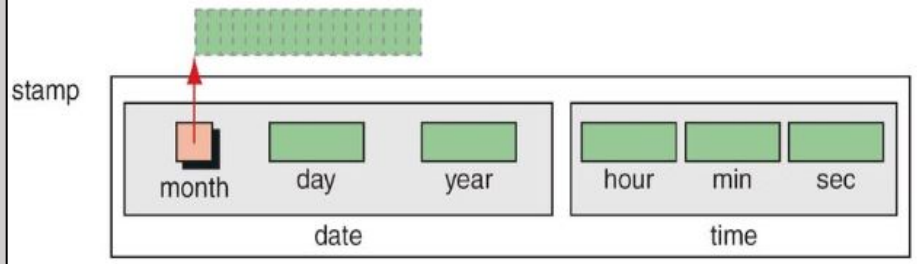
(copying a pointer to a string, not the string itself)

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

```
#include <stdio.h>
int main(){
 typedef struct{int *p;char *q;}ex;
 ex e1;
 int a = 2020;
 char str[] ="Programming";
 e1.p = &a; e1.q = str;
 printf ("\n %d %s", *e1.p, e1.q);
return 0;
}
```

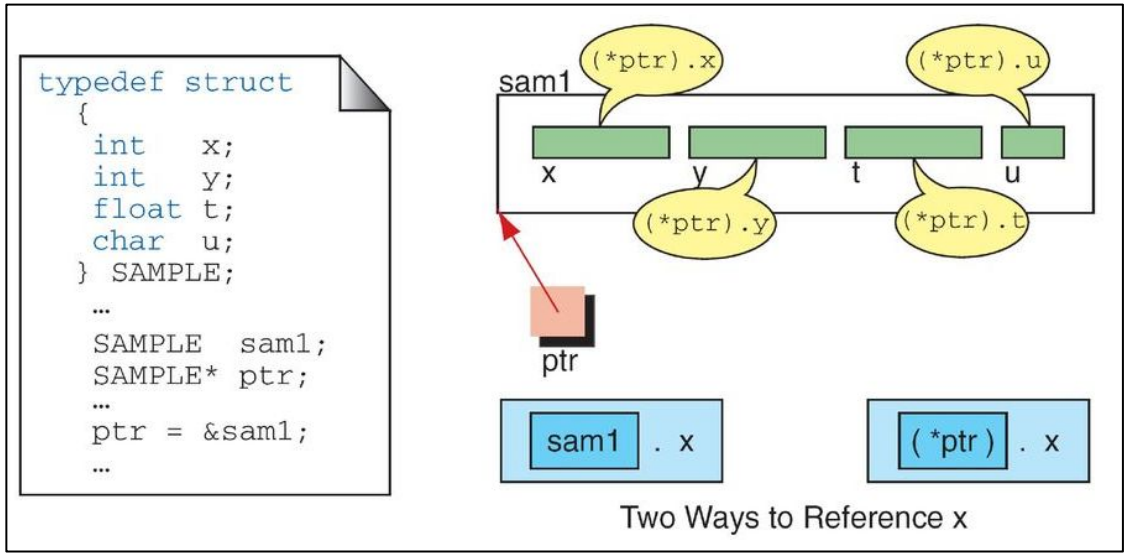
2020 Programming





Pointers to Structures

- Pointers to structures are often used when passing structure as an argument to a function (pass by reference)



- *ptr: refers to whole structure i.e. SAMPLE (* is the indirection operator).

(*ptr).x ⇔ ptr -> x

(Alternativ ely)

Example digital clock simulation

```
#include <stdio.h>
typedef struct {
  int hr;
  int min;
  int sec;
} clock;
void inc (clock
*clk);
void show (clock
*clk);
```

```
int main(){
  clock clk = {9, 15, 51};
  while (1) {
     inc (&clk);
     show (&clk);
  }
  return 0;
}
```

9: 15:52

...

```
10: 28:58
10: 28:59
10: 29:00
10: 29:01
10: 29:02
10: 29:03
```

...

```
13: 51:25
13: 51:26
13: 51:27
13: 51:28
13: 51:29
```

...

```
void inc (clock *clk)
{
     (clk -> sec)++;
     if (clk -> sec == 60)
     {
          clk -> sec = 0;
          (clk -> min)++;
          if (clk -> min == 60) {
                    clk -> min = 0;
                    (clk -> hr)++;
                    if (clk -> hr == 24)
                    clk -> hr = 0;
          }
     }
     return;
}
```

```
void show(clock *clk) {printf ("%02d: %02d:%02d\n", clk ->
hr, clk -> min, clk -> sec);     return;}
```

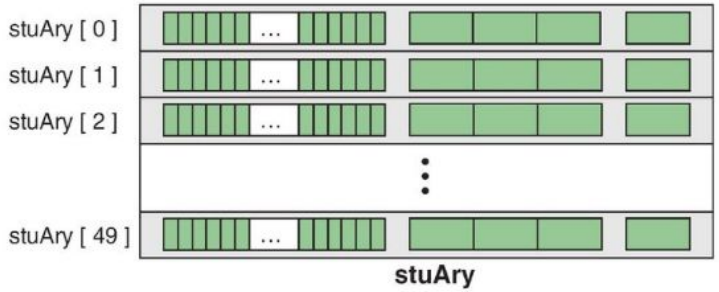
Arrays of Structures

```
struct student
{
   char name[20];
   int midterm[3];
   int final;
};
```

```
struct student s1;
struct student s2;
struct student s3;
struct student s4;
struct student s5;
```

```
struct student stuAry[50];
```

Creates an array of structures that can hold 50 student's details



Awkward to write like this if 50 students are there in a course

We will have to declare a variable every time we add a student.

```c
#include<stdio.h>
#include<string.h>
struct student {
    char name [30];
    char id[15];
    float marks[2];};
int main() {
struct  student  stud[2];
int i, j; float sum = 0;
for(i = 0; i < 2; i++ ) {
printf("\nEnter details of the
          student %d\n\n",i+1);
printf("Enter name: ");
scanf("%s", stud[i].name);
printf("Enter id no: ");
scanf("%s", stud[i].id);
for(j = 0; j < 2; j++)    {
  printf("Enter marks: ");
  scanf("%f", &stud[i].marks[j]);
  }
}
```

```c
printf("\n");
printf("Name\tID no\tAverage\n\n");
for(i = 0; i < 2; i++ ) {
  sum = 0;
  for (j = 0; j < 2; j++) {
    sum += stud[i].marks[j];
  }
  printf("%s\t%s\t%.2f\n", stud[i].name,
                          stud[i].id, sum/2);
 }
return 0;
}
```

Example

```
Name    ID no
Average

Ankit   2019A7PS0008H   87.00

Ruban   2019A7PS0004H   87.50
```

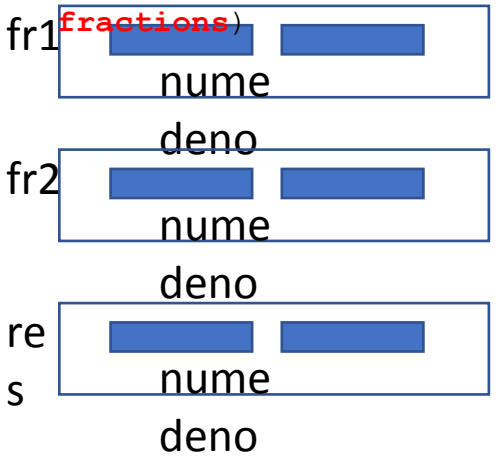
```
Enter details of the student 1

Enter name: Ankit
Enter id no: 2019A7PS0008H
Enter marks: 86
Enter marks: 88

Enter details of the student 2

Enter name: Ruban
Enter id no: 2019A7PS0004H
Enter marks: 89
Enter marks: 86
```

Structures and Functions

For structures to be useful, we must be able to pass them to functions and return them. A function can access members of a structure in below possible ways:

1. Individual members can be passed to it (Pass-by-value)
2. The whole structure can be passed to it (Pass-by-value)
3. The address of a structure or the address of a member can be passed to it (Pass-by-reference)

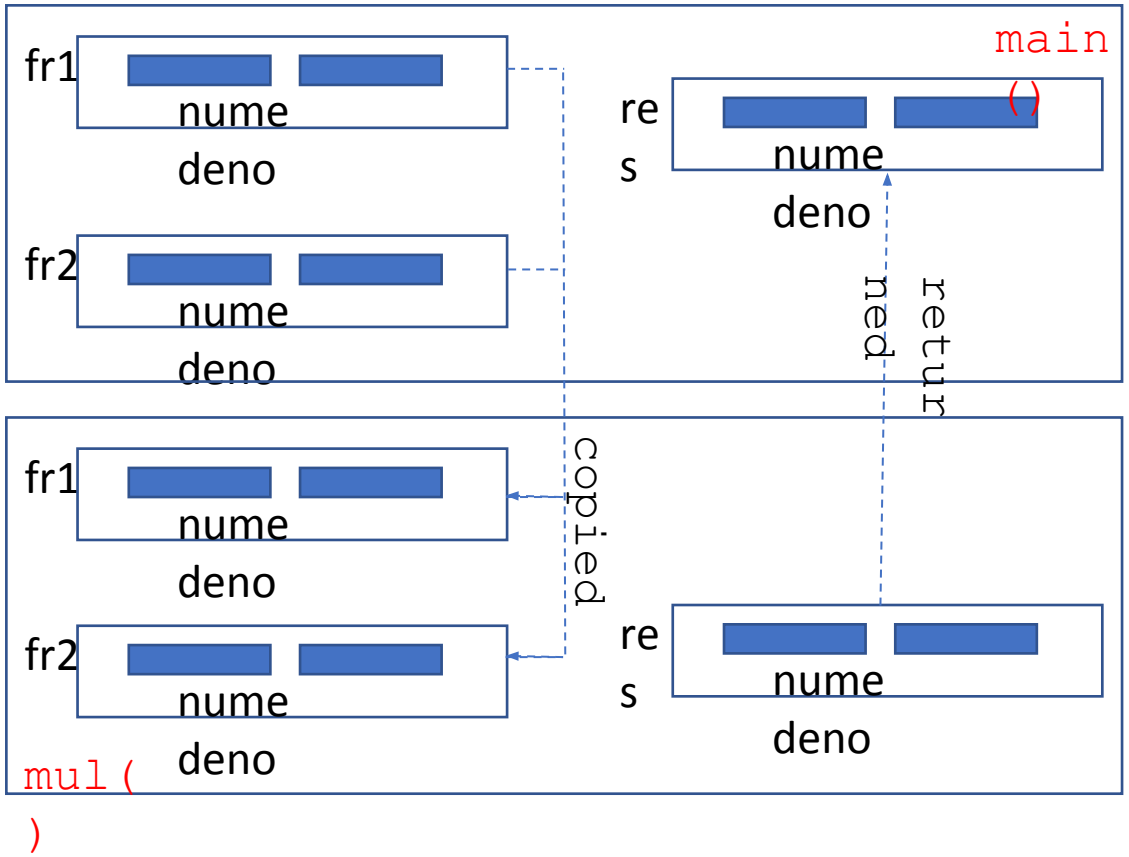
(**Multiplying fractions**)




```
#include <stdio.h>
struct {
   int nume;
   int deno;
}fr1, fr2, res;
int main () {
…
res.nume=mul(fr1.nume,fr2.nume);
res.deno=mul(fr1.deno,fr2.deno);
…
```

```
int mul(int x, int y)
{
  return x*y;
}
```

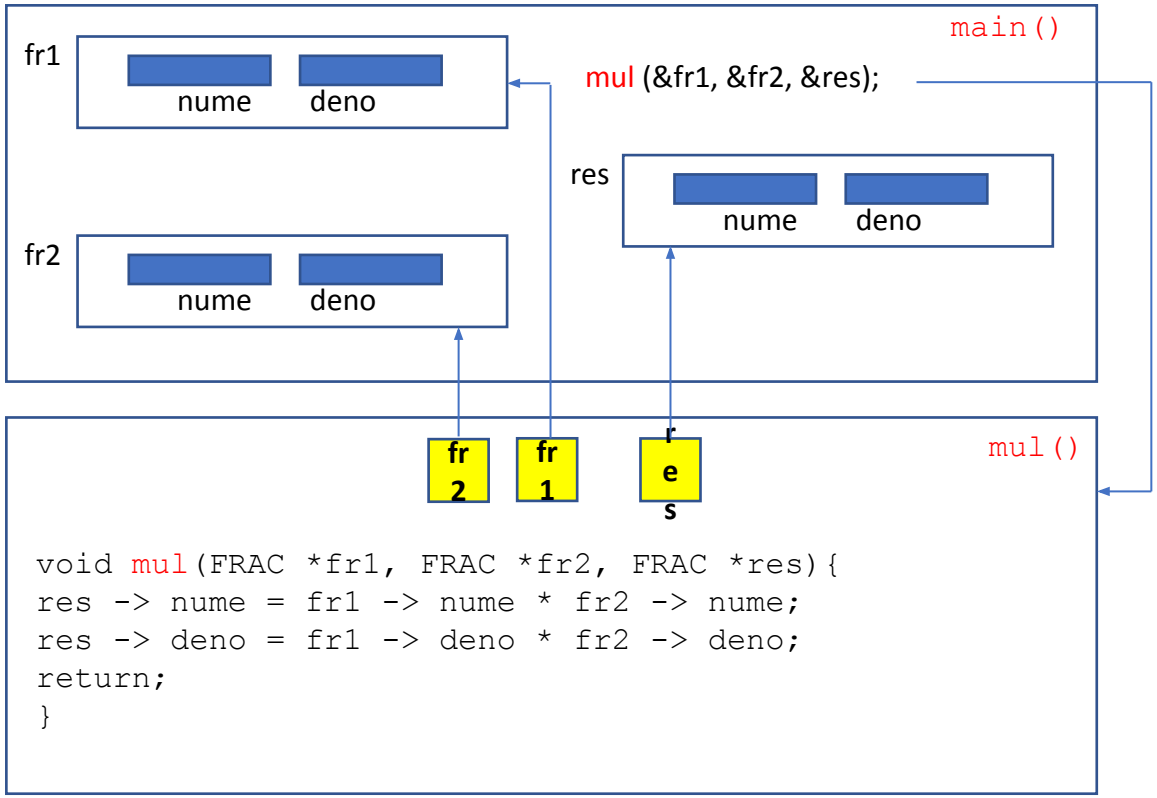
Passing a whole Structure




```
#include <stdio.h>
typedef struct {
  int nume;
  int deno;
}FRAC;
FRAC fr1 = {4, 10};
FRAC fr2 = {25, 2};
FRAC res;
FRAC mul(FRAC fr1, FRAC fr2){
  res.nume=fr1.nume *
fr2.nume;
  res.deno=fr1.deno *
fr2.deno;
  return res;
}
int main() {
res = mul(fr1, fr2);
printf("%d",res.nume/res.deno)
```

```
5

...Program finished with exit code 0
```

Passing Structures through Pointers: Right way




```c
#include <stdio.h>
typedef struct {
  int nume; int deno;
}FRAC;
FRAC fr1 = {4, 10};
FRAC fr2 = {25, 2};
FRAC res;
void mul(FRAC *fr1, FRAC *fr2, FRAC *res){
res -> nume = fr1 -> nume * fr2 -> nume;
res -> deno = fr1 -> deno * fr2 -> deno;
return;
}
int main() {
mul(&fr1, &fr2, &res);
printf("%d",res.nume/res.deno);
return 0;
}
```

Output:

```
5


...Program finished with exit code 0
```

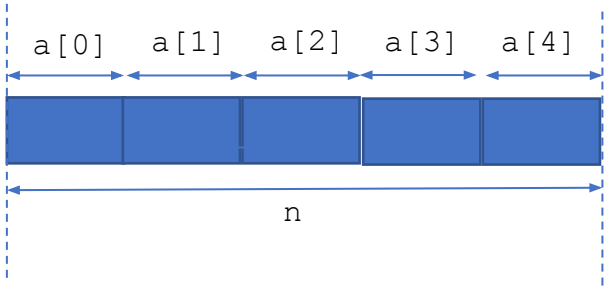
```
struct node
{
   int i;
   float j;
};
struct node *s[10] ;
```

An array, each element of which is a pointer to a structure of type node

Unions

- Similar to structures in terms of defining, declaring and accessing the members.
- The only difference is, only one member of an Union can be used at a time. The reason being all members of an Union occupy the same memory area.
- Applications: That require multiple interpretations of a given piece of memory, e.g, searching a database using a persons' PAN no. and also, by name.

```
union data{
  char a[5];
  float n;
}d1;
```



Examples

```c
#include <stdio.h>
#include <string.h>
int main() {
    union data {
        char a[3];
        float n;
    }d1;
    strcpy (d1.a, "AA");

    d1.n = 345;
    printf("%f", d1.n);

    printf ("%s", d1.a);
    return 0;
}
```

```
345.000000
```

```c
#include <stdio.h>
#include <string.h>
int main(){
    union data {
        char a[3];
        float n;
    }d1;

    d1.n = 345;
    strcpy (d1.a, "AA");
    printf ("%s", d1.a);
    printf ("%f", d1.n);
    return 0;
}
```

```
AA128.254898
```

```c
#include <stdio.h>
#include <string.h>
int main()
{
    union data {
        char a[5];
        float n;
    } d1;

    strcpy (d1.a, "AAAAAAAAAAAAAAA");
    d1.n = 345;
    printf ("%s", d1.a);
    printf ("%f", d1.n);
    return 0;
}
```

Output

```
*** stack smashing detected ***: ./a.out terminated
345.000000Aborted
```

Compiler does not output an error when you access a member that was not the last one to be assigned.

THANK YOU