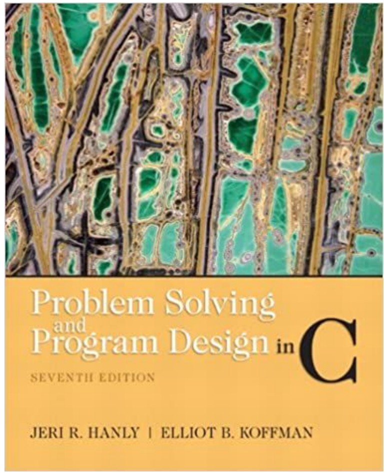
BIRZEIT UNIVERSITY


Problem Solving and Program Design in C
SEVENTH EDITION
JERI R. HANLY | ELLIOT B. KOFFMAN

Faculty of Engineering and Technology Department of Computer Science

Introduction to Computers and Programming (Comp 133)

References :
Book : Problem Solving and Program Design in C (7th Edition) 7th Edition
Slides : Dr. Radi Jarrar , Dr. Abdallah Karakra , Dr. Majdi Mafarja.

Structures

Chapter 10

Structures

- In C a structure is a customised user-defined data type.
- structure definition : the template used to create structure variables.
- **structure elements** : the member variables of the structure type.
- **Syntax Method1**

```
struct tag
    {     type var_1 ;
          type var_2 ;
             ...
          int rec_no ;
    } ;
```

Structure
present the new type

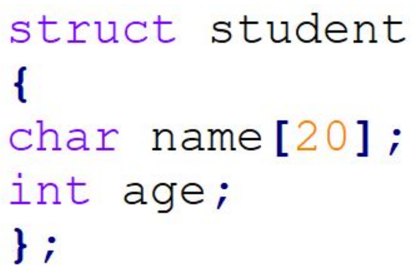
tag is an identifier name given to the customised "**type**"

Chapter 10

- Structure
  - User-Defined structure types

User-Defined structure types

- Example define structure for student with information: **Name , Age**

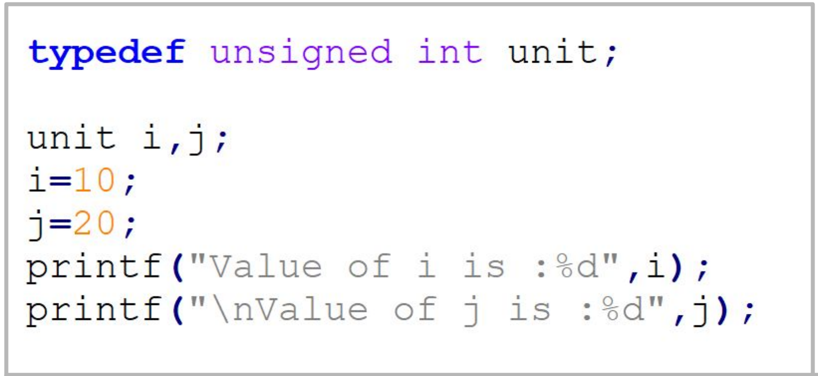


```
struct student
{
char name[20];
int age;
};
```

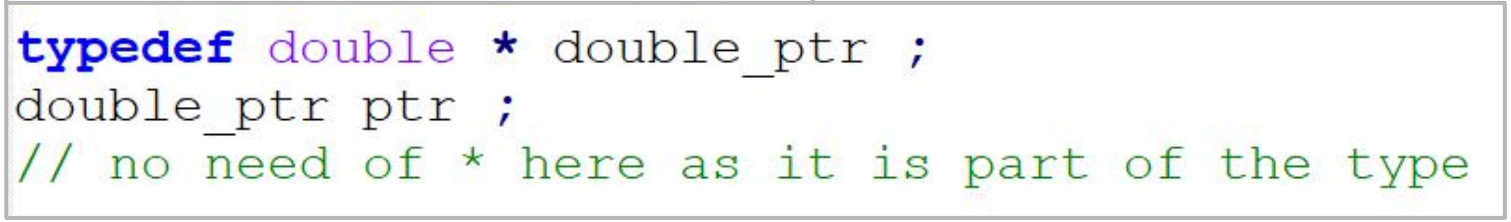
- To created the variable of student type
    - **struct** student s1;
- To access the structure elements
    - strcpy(s1.name, ”Ahmed “);
    - s1.age=25;

User-Defined structure types

- **typedef in C** : a keyword used to provide alternative names to the already existing predefined variable.
- Syntax
  - typedef <existing_name> <alias_name>



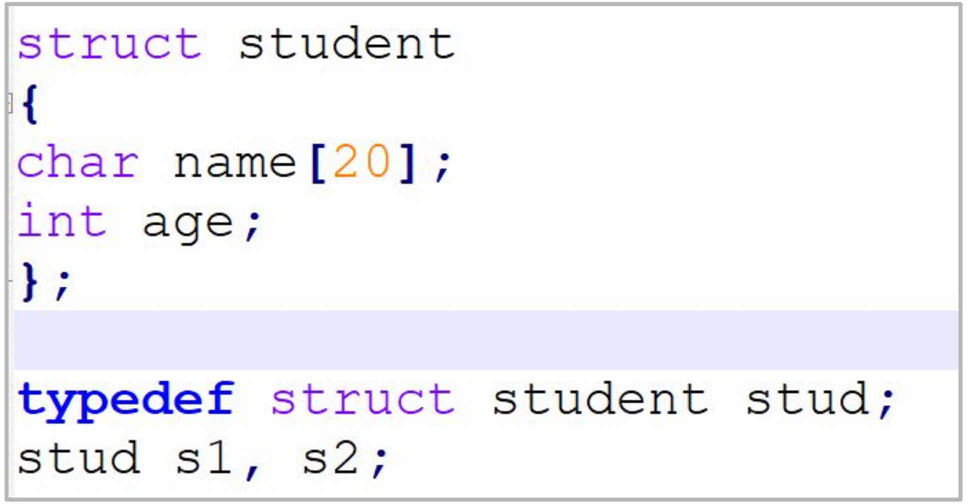
```
typedef unsigned int unit;

unit i,j;
i=10;
j=20;
printf("Value of i is :%d",i);
printf("\nValue of j is :%d",j);
```



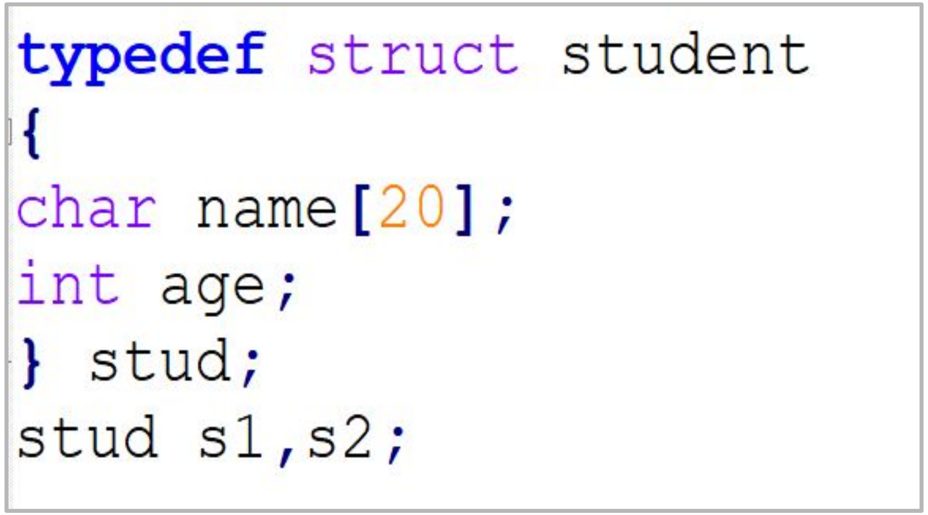
```
typedef double * double_ptr ;
double_ptr ptr ;
// no need of * here as it is part of the type
```

User-Defined structure types

- Using **typedef** with structures



```
struct student
{
char name[20];
int age;
};

typedef struct student stud;
stud s1, s2;
```



```
typedef struct student
{
char name[20];
int age;
} stud;
stud s1,s2;
```

User-Defined structure types

- C provides several ways to define structures, we will explore just one approach defining a **new data type for each category of structured objects**.
- To developing a database of the planets in our solar system. For each **planet**, we need to represent information like :
    - **Name**: Jupiter , **Diameter**: 142,800 km , **Moons**: 16
    - **Orbit time**: 11.9 years , **Rotation time**: 9.925 hours

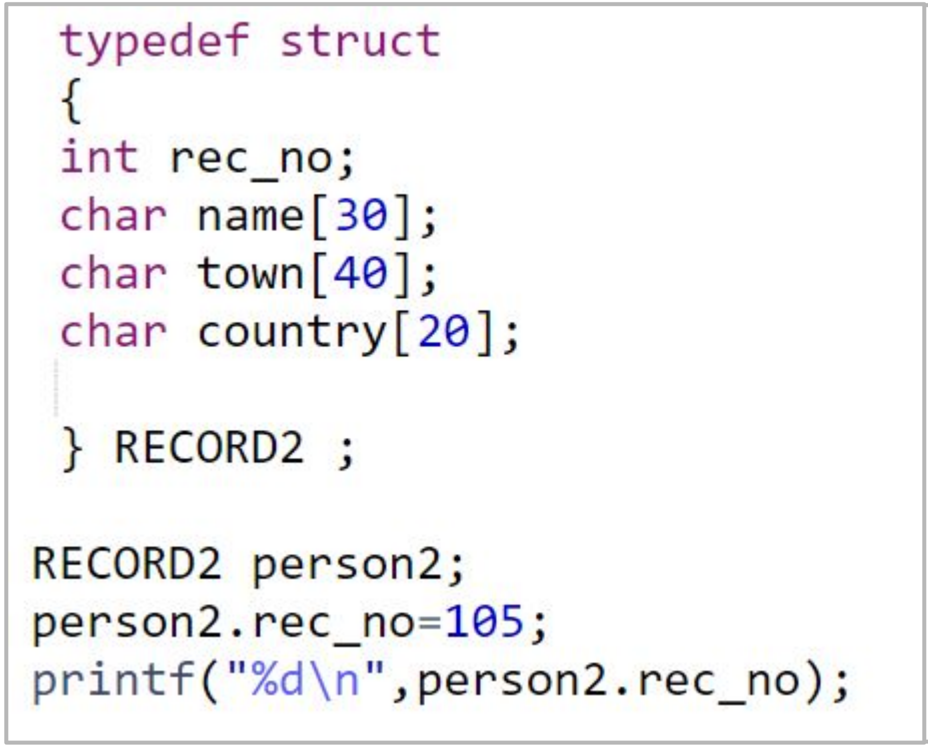
User-Defined structure types

- **Structure type**: data type for a record composed of multiple components.
- We can define a **structure type planet_t** to use in declaring a variable in which to store this information.

```
#define STRSIZ 10

typedef struct {
      char   name[STRSIZ];
      double diameter;              /* equatorial diameter in km   */
      int    moons;                 /* number of moons             */
      double orbit_time,            /* years to orbit sun once     */
             rotation_time;         /* hours to complete one
                                         revolution on axis        */
} planet_t;
```

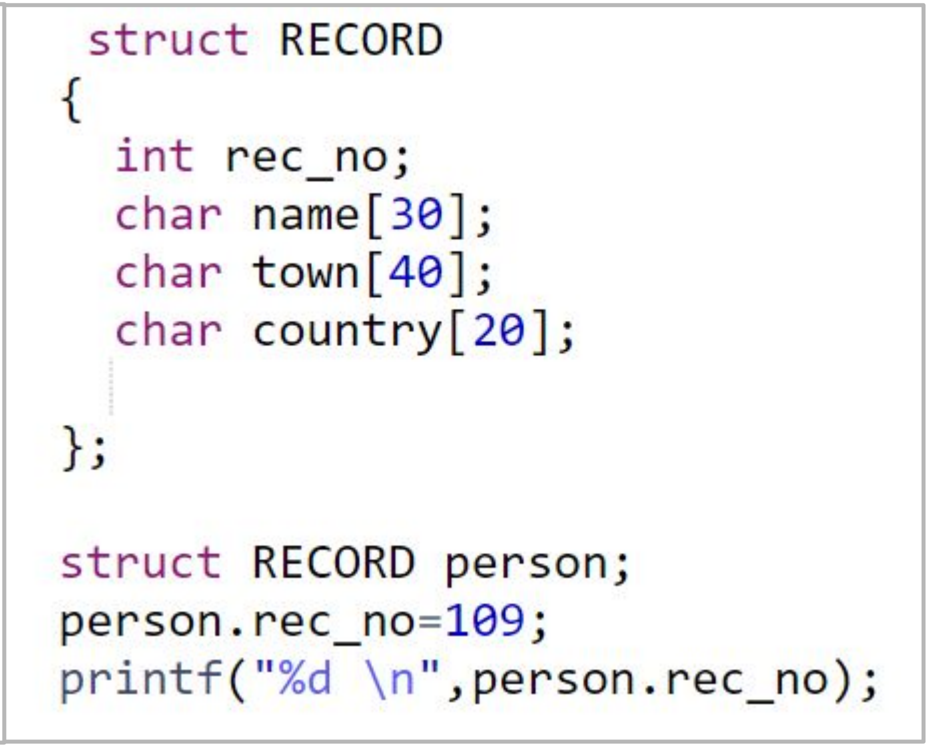
User-Defined structure types

Method 1



```
typedef struct
{
int rec_no;
char name[30];
char town[40];
char country[20];

} RECORD2 ;

RECORD2 person2;
person2.rec_no=105;
printf("%d\n",person2.rec_no);
```

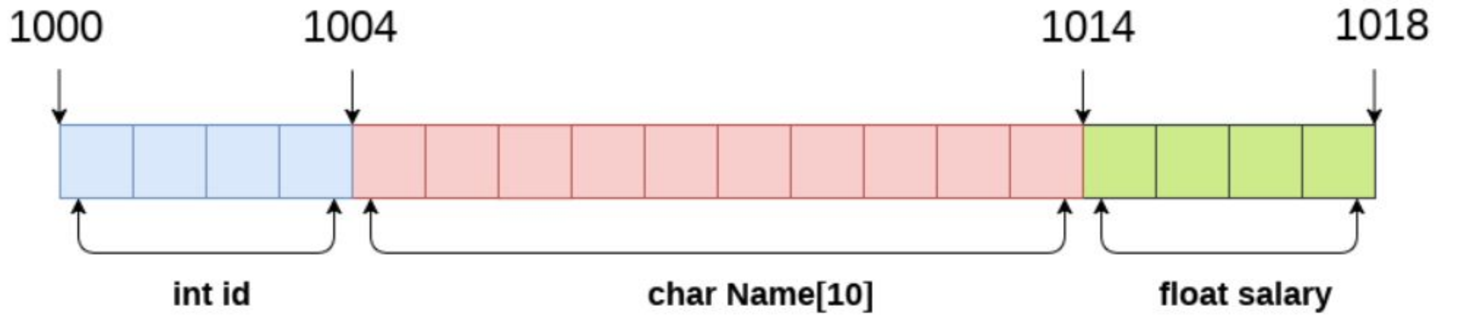
Method 2



```
 struct RECORD
{
  int rec_no;
  char name[30];
  char town[40];
  char country[20];

};

struct RECORD person;
person.rec_no=109;
printf("%d \n",person.rec_no);
```

User-Defined structure types

- Structure memory size allocated .



```
struct Employee
{
    int id;
    char Name[10];
    float salary;
} emp;
```

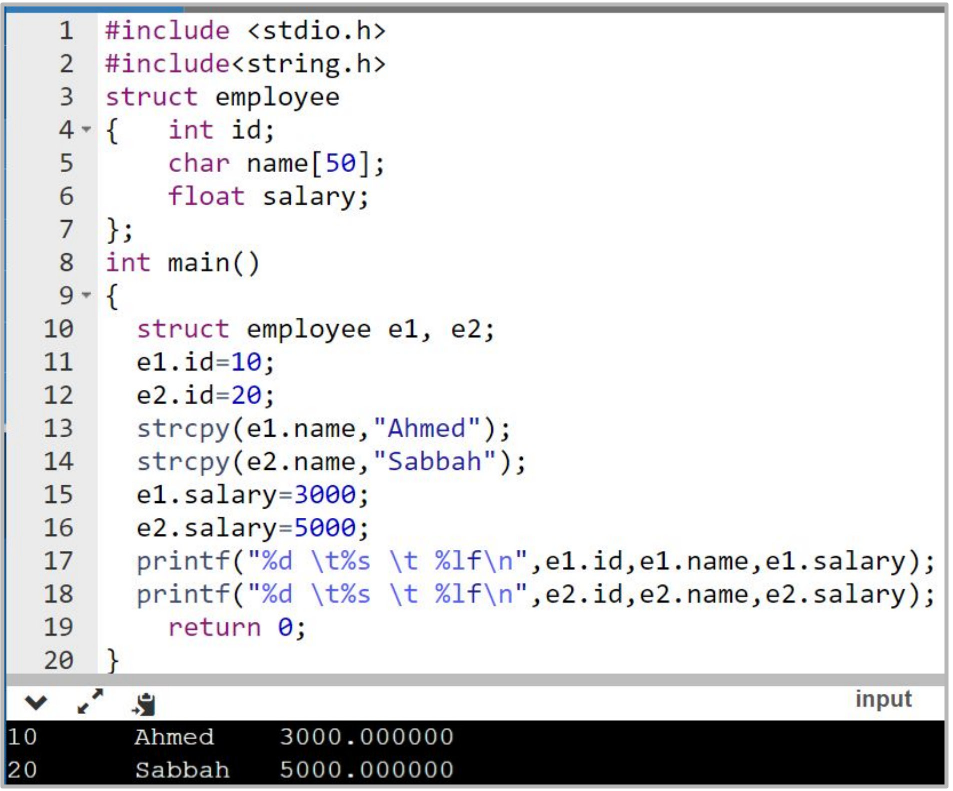
sizeof (emp) = 4 + 10 + 4 = 18 bytes

where;
sizeof (int) = 4 byte
sizeof (char) = 1 byte
sizeof (float) = 4 byte

https://www.javatpoint.com/structure-in-c

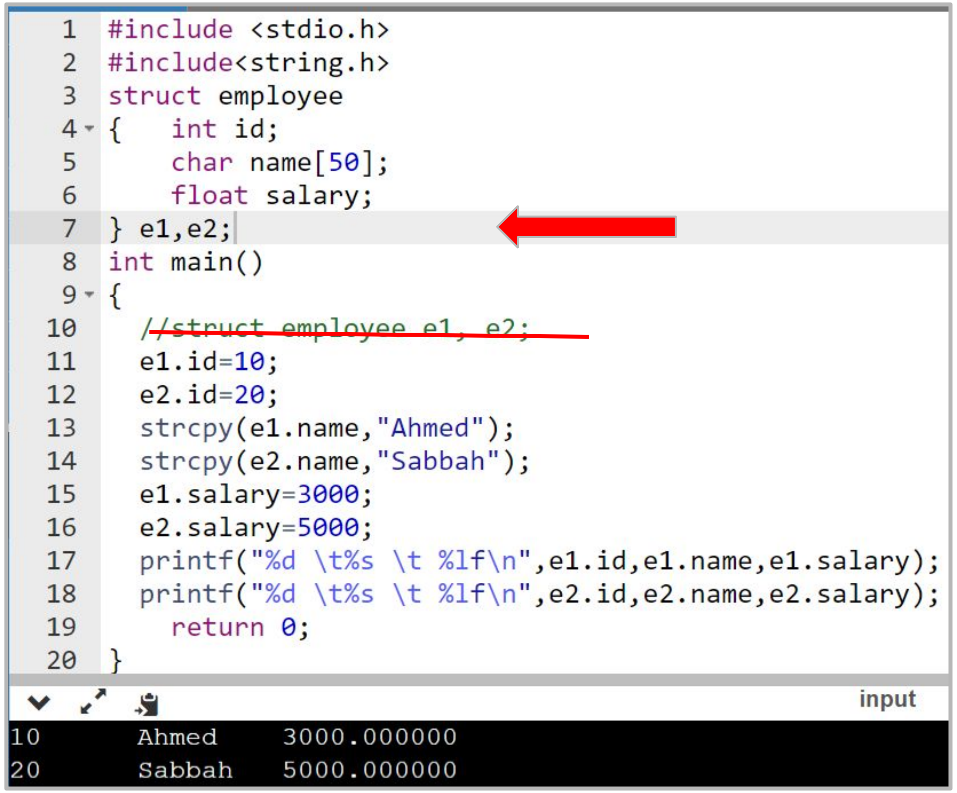
User-Defined structure types

- Define structure for employee with information : **Name, Id , Salary**



```c
#include <stdio.h>
#include<string.h>
struct employee
{   int id;
    char name[50];
    float salary;
};
int main()
{
  struct employee e1, e2;
  e1.id=10;
  e2.id=20;
  strcpy(e1.name,"Ahmed");
  strcpy(e2.name,"Sabbah");
  e1.salary=3000;
  e2.salary=5000;
  printf("%d \t%s \t %lf\n",e1.id,e1.name,e1.salary);
  printf("%d \t%s \t %lf\n",e2.id,e2.name,e2.salary);
    return 0;
}
```

```
10      Ahmed   3000.000000
20      Sabbah  5000.000000
```

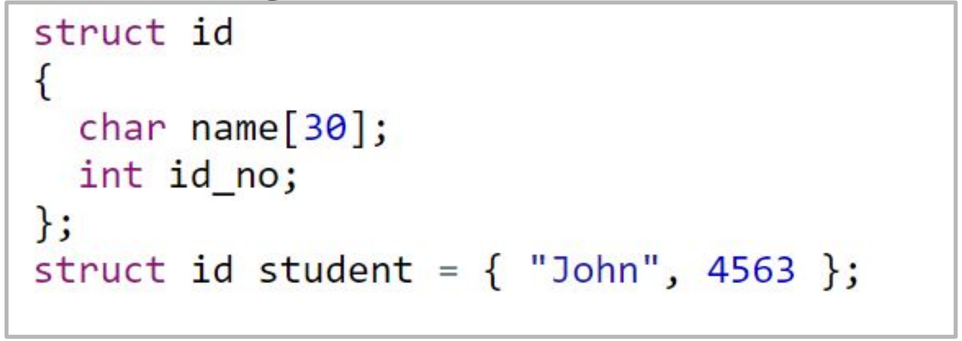


```c
#include <stdio.h>
#include<string.h>
struct employee
{   int id;
    char name[50];
    float salary;
} e1,e2;
int main()
{
  ~~//struct employee e1, e2;~~
  e1.id=10;
  e2.id=20;
  strcpy(e1.name,"Ahmed");
  strcpy(e2.name,"Sabbah");
  e1.salary=3000;
  e2.salary=5000;
  printf("%d \t%s \t %lf\n",e1.id,e1.name,e1.salary);
  printf("%d \t%s \t %lf\n",e2.id,e2.name,e2.salary);
    return 0;
}
```

```
10      Ahmed   3000.000000
20      Sabbah  5000.000000
```

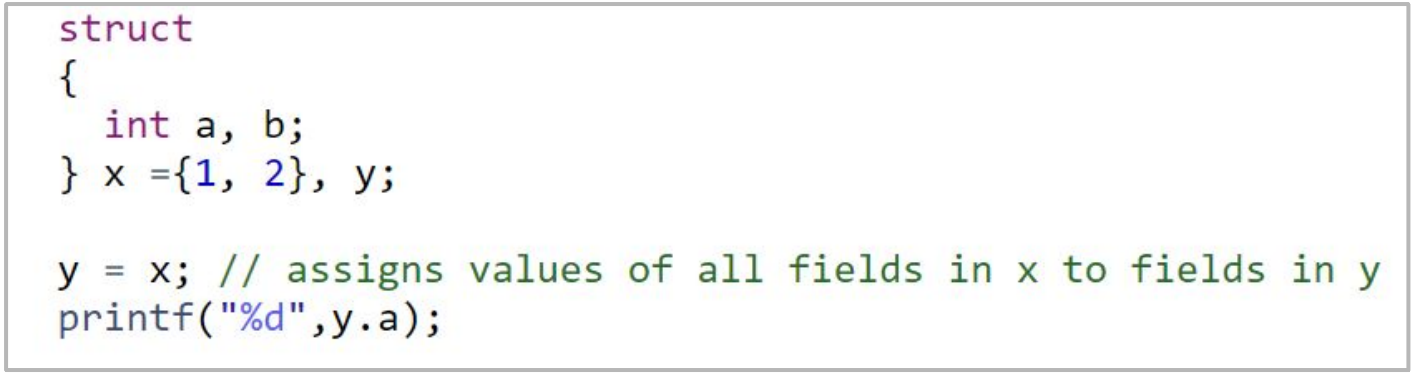
User-Defined structure types

- Initialising Structures



```c
struct id
{
  char name[30];
  int id_no;
};
struct id student = { "John", 4563 };
```

- Structure Assignment



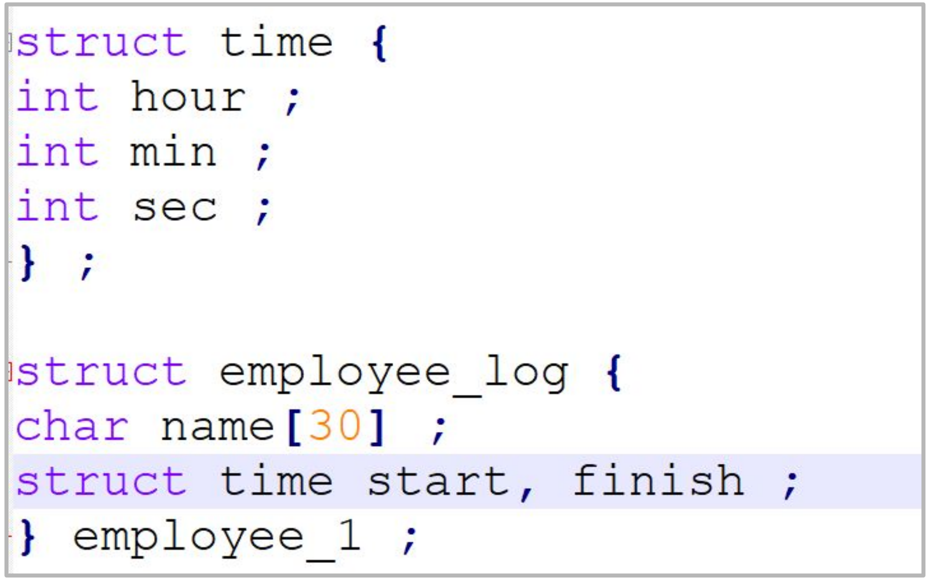
```c
struct
{
  int a, b;
} x ={1, 2}, y;

y = x; // assigns values of all fields in x to fields in y
printf("%d",y.a);
```

Chapter 10

- Structure
  - Structure type data as input and output parameters
  - Functions result values are structures

Structure

- Structures like any other type in C.
- It can create arrays of structures, nest structures, pass structures as arguments to functions.



```
struct time {
int hour ;
int min ;
int sec ;
} ;

struct employee_log {
char name[30] ;
struct time start, finish ;
} employee_1 ;
```

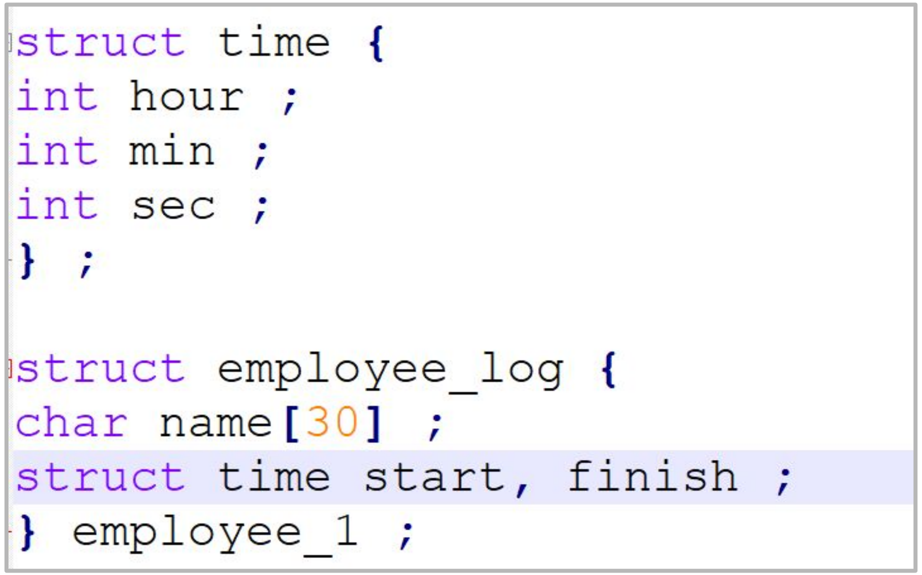
To access the hour field of time in the variable **employee_1**.

employee_1.start.hour = 9 ;

employee_1.finish.hour = 4 ;

Array of Structure

- If need to keep track 100 employees so that an array of **employee_log** would be useful.



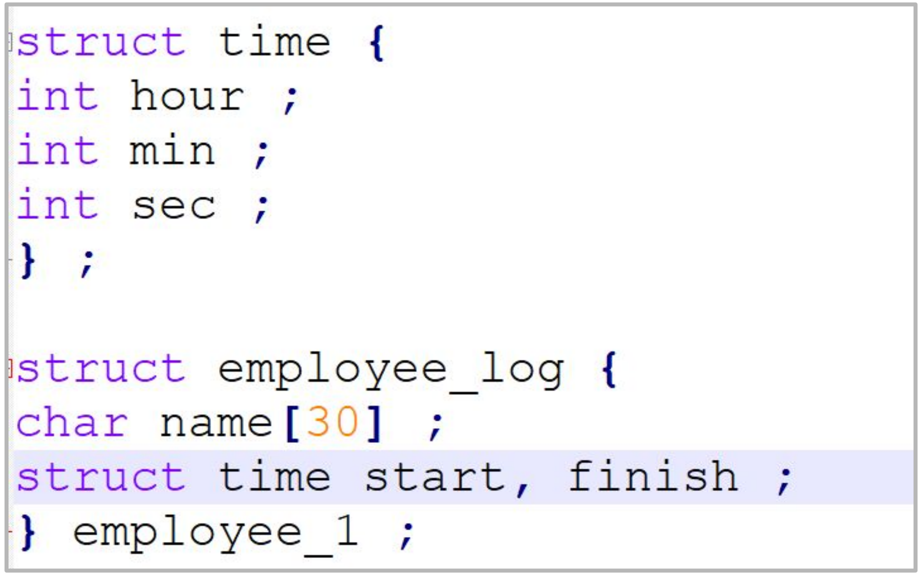
```
struct time {
int hour ;
int min ;
int sec ;
} ;

struct employee_log {
char name[30] ;
struct time start, finish ;
} employee_1 ;
```

```
struct employee_log workers[100] ;
```

To access specific employees

```
workers[10].finish.hour = 10 ;
```

Structure type data as input and output parameter



```
struct time {
int hour ;
int min ;
int sec ;
} ;

struct employee_log {
char name[30] ;
struct time start, finish ;
} employee_1 ;
```

To pass structure variable to functions as parameters.

function1(employee_1) ;

To implements a call to **function1**

void function1(struct employee_log emp)
{ …..}

To Passing an array of structures to a function

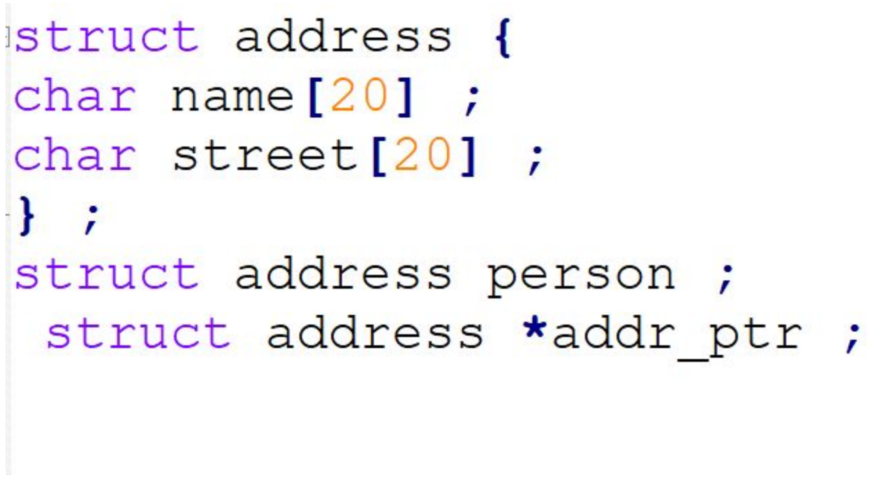
struct employee_log workers[100] ;
function2(workers) ;

To implements a call to **function2**

void function2(struct employee_log staff[ ])
{ …..}

Structure type data as input and output parameter

- Pass by value is less effective than pass by reference with structure.
- Full local copy of the structure passed is made in memory.
- **Structure Pointers**



```
struct address {
char name[20] ;
char street[20] ;
} ;
struct address person ;
 struct address *addr_ptr ;
```

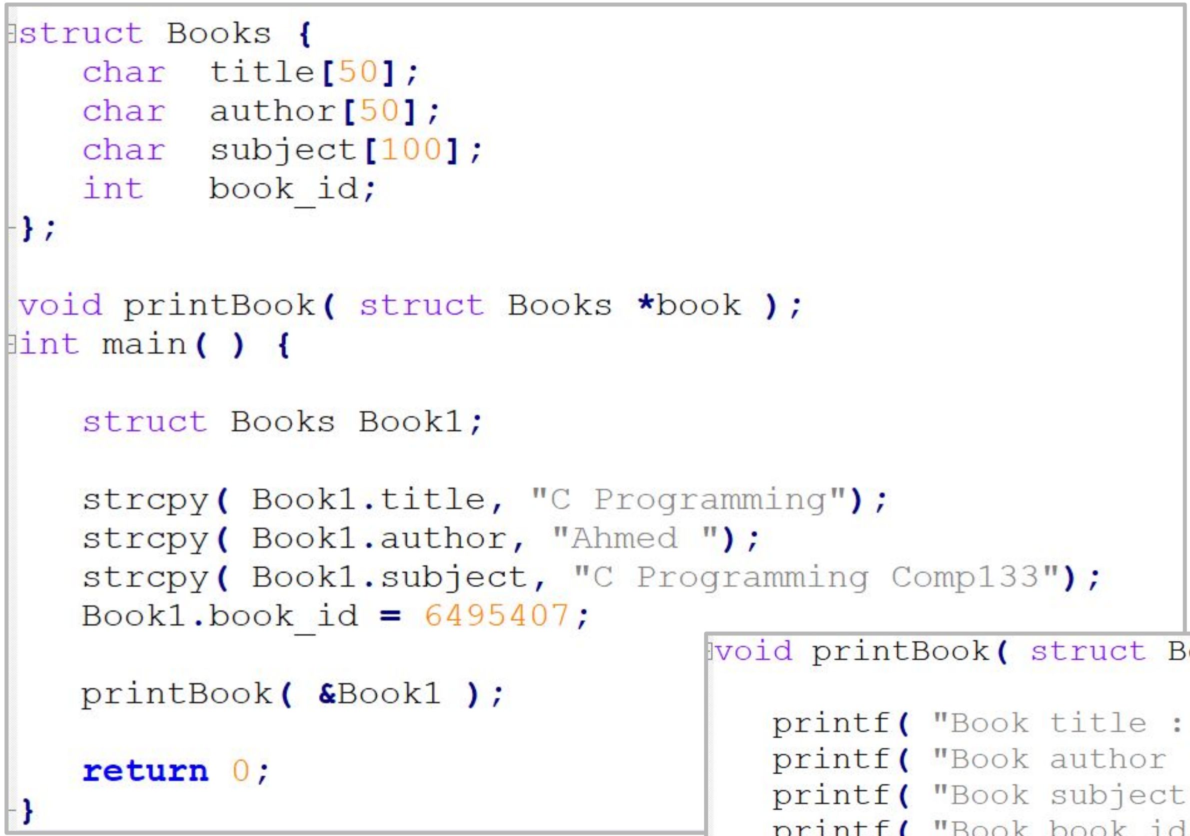
addr_ptr = &person ;

To access the elements using a pointer

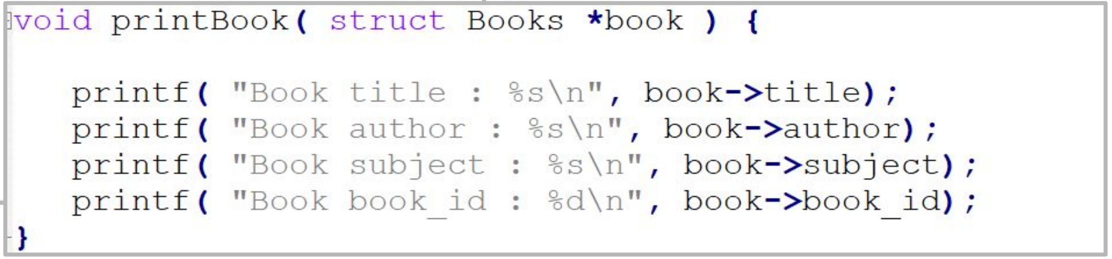
Use **->** operator (**only with pointer)**

addr_ptr -> name

Structure Pointers



```c
struct Books {
    char  title[50];
    char  author[50];
    char  subject[100];
    int   book_id;
};

void printBook( struct Books *book );
int main( ) {

    struct Books Book1;

    strcpy( Book1.title, "C Programming");
    strcpy( Book1.author, "Ahmed ");
    strcpy( Book1.subject, "C Programming Comp133");
    Book1.book_id = 6495407;

    printBook( &Book1 );

    return 0;
}
```



```c
void printBook( struct Books *book ) {

   printf( "Book title : %s\n", book->title);
   printf( "Book author : %s\n", book->author);
   printf( "Book subject : %s\n", book->subject);
   printf( "Book book_id : %d\n", book->book_id);
}
```

Output

```
Book title : C Programming
Book author :Ahmed
Book subject :C Programming Comp133
Book book_id : 6495407
```

Thank You.

جامعة بيرزيت
BIRZEIT UNIVERSITY