Computer Programming & Problem Solving

CS100

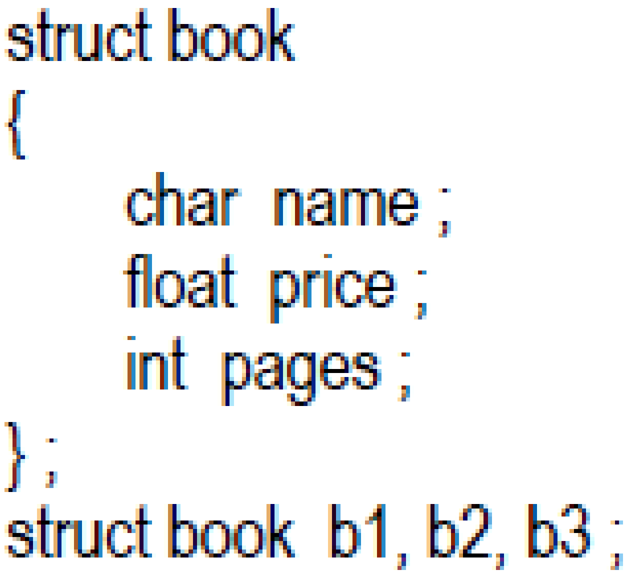
Mrs Sanga G. Chaki
Department of Computer Science and Engineering
National Institute of Technology, Goa

January, 2023

Structures

1. User-defined data type – Defined by user as per need
2. Complex data structure.
3. Convenient tool for handling a group of logically related data items.
4. Helps in organizing complex data in more meaningful way
5. Example:
   a) Student name, roll number, and marks.
   b) Real part and complex part of a complex number.
   c) Name, price, number of pages of a book
6. The individual elements of a structure are called members.

Structure - Declaration

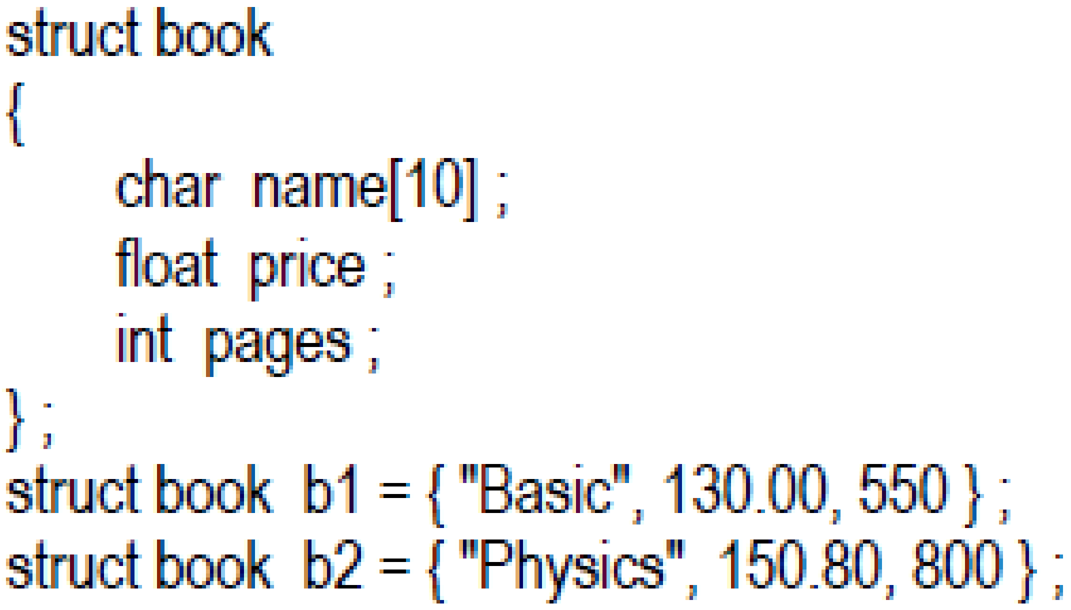


```
struct book
{
      char  name ;
      float  price ;
      int  pages ;
} ;
struct book  b1, b2, b3 ;
```

1. Defines a new data type called struct book
2. Each variable of this data type – b1, b2, b3 - will consist of a character variable (name), a float variable (price) and an integer variable (pages).
3. These are the structure elements

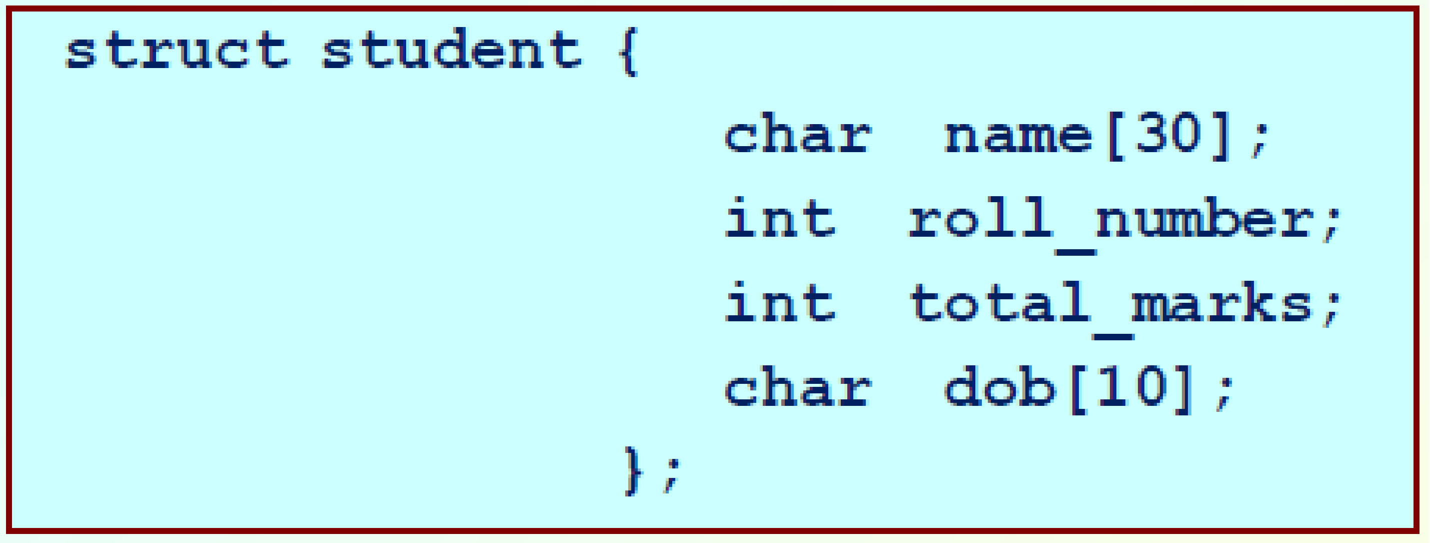
4. The second statement sets aside space in memory in adjacent memory locations

Structure - Initialization

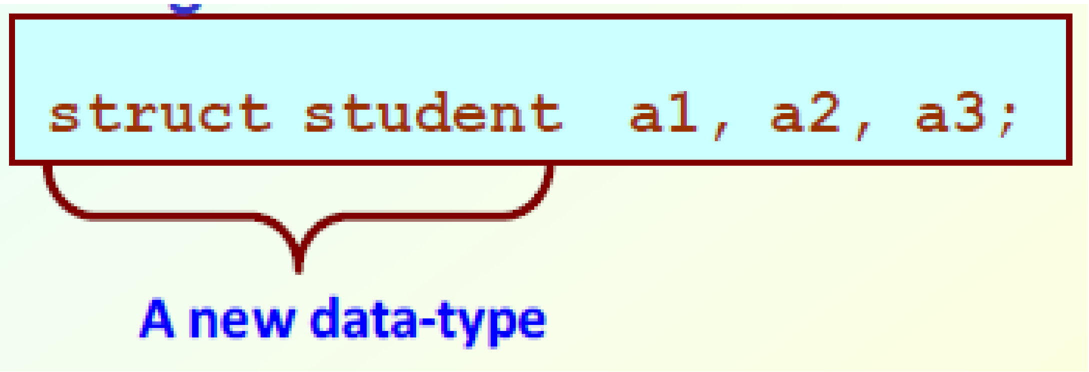


```
struct book
{
        char  name[10] ;
        float  price ;
        int  pages ;
} ;
struct book  b1 = { "Basic", 130.00, 550 } ;
struct book  b2 = { "Physics", 150.80, 800 } ;
```

Structure - Example



```
struct student {
                        char  name[30];
                        int  roll_number;
                        int  total_marks;
                        char  dob[10];
                    };
```



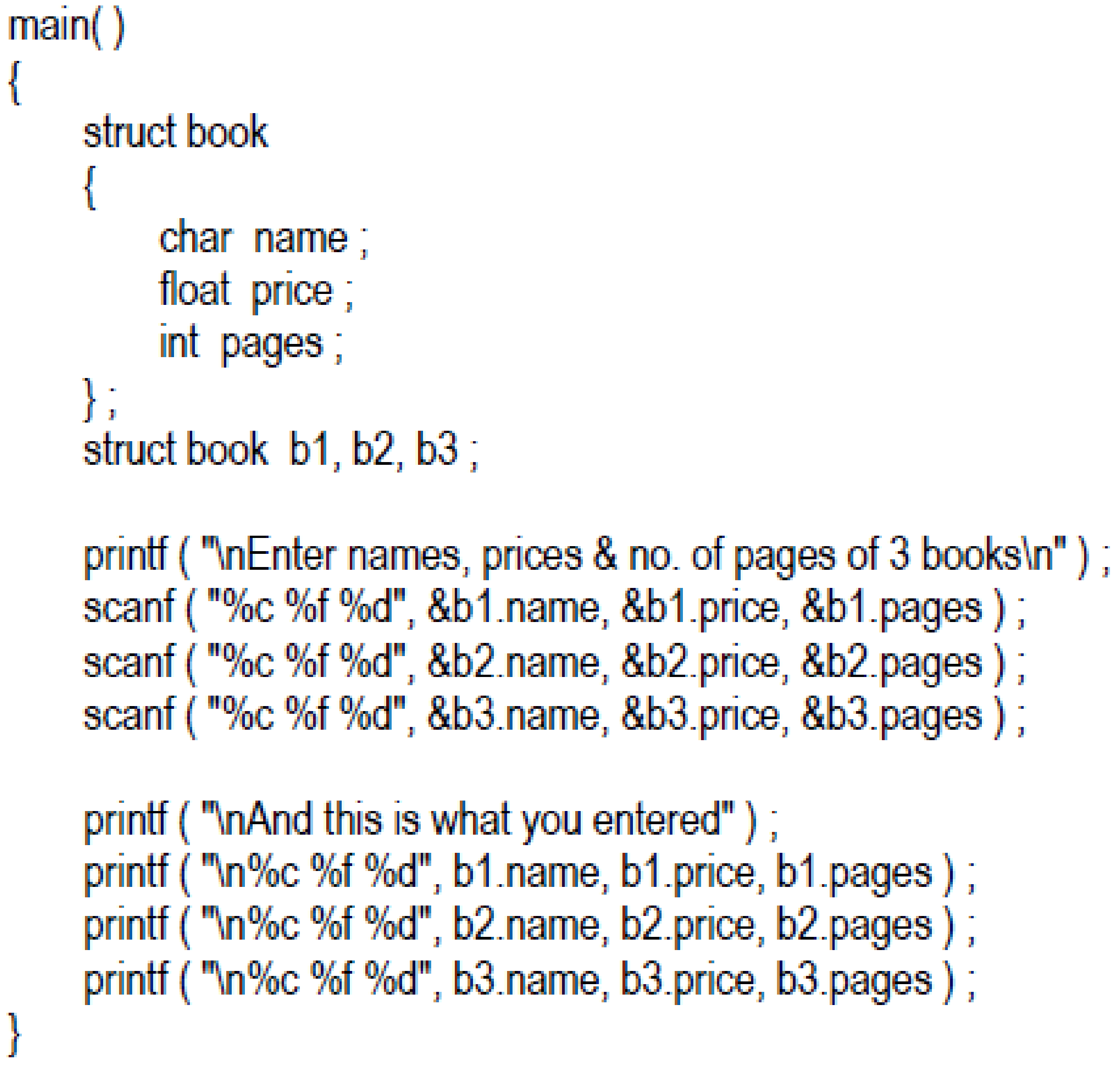
```
struct student  a1, a2, a3;
```

A new data-type

Structure – Accessing Elements

1. How to refer to pages of the structure book?
   a) b1.pages
2. How to refer to price of a book variable?
   a) b1.price
3. For the variable of struct student: a1.name, a2.name, a3.dob, a1.roll_number etc

Structure – Populating Variable Elements



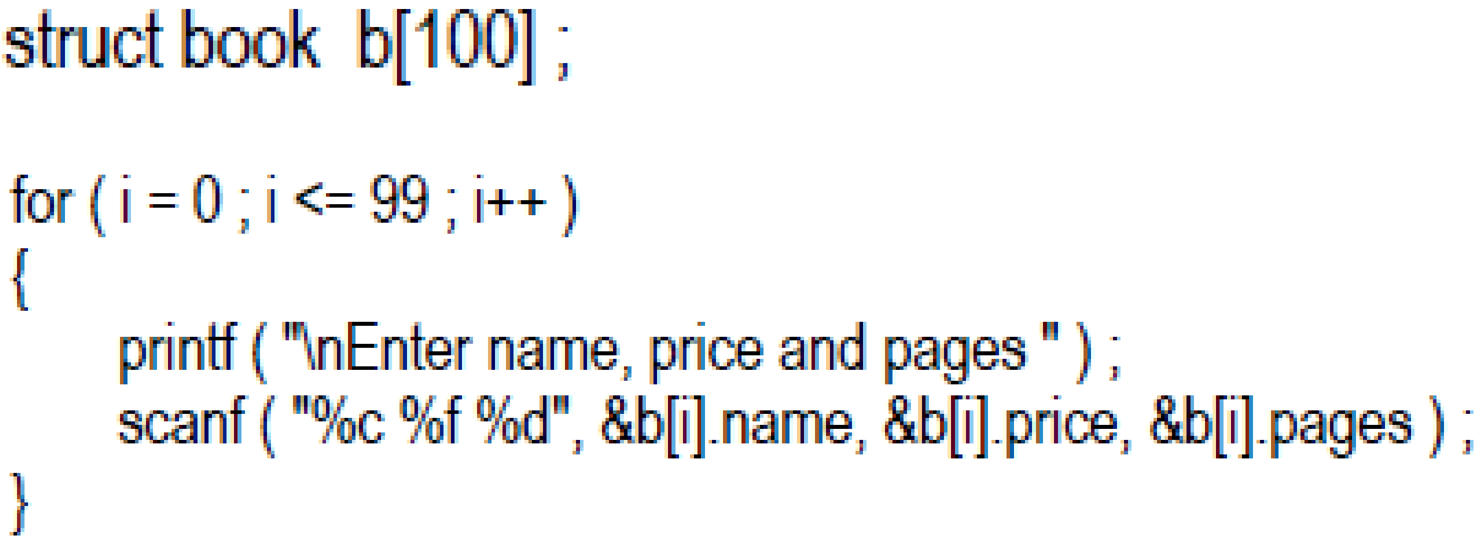
```
main( )
{
    struct book
    {
        char  name ;
        float  price ;
        int  pages ;
    } ;
    struct book  b1, b2, b3 ;

    printf ( "\nEnter names, prices & no. of pages of 3 books\n" ) ;
    scanf ( "%c %f %d", &b1.name, &b1.price, &b1.pages ) ;
    scanf ( "%c %f %d", &b2.name, &b2.price, &b2.pages ) ;
    scanf ( "%c %f %d", &b3.name, &b3.price, &b3.pages ) ;

    printf ( "\nAnd this is what you entered" ) ;
    printf ( "\n%c %f %d", b1.name, b1.price, b1.pages ) ;
    printf ( "\n%c %f %d", b2.name, b2.price, b2.pages ) ;
    printf ( "\n%c %f %d", b3.name, b3.price, b3.pages ) ;
}
```

Array of Structures

1. How to store data of 100 books using structures?
   a) Use an array of structures!



```
struct book  b[100] ;

for ( i = 0 ; i <= 99 ; i++ )
{
      printf ( "\nEnter name, price and pages " ) ;
      scanf ( "%c %f %d", &b[i].name, &b[i].price, &b[i].pages ) ;
}
```

Type Definitions

1. **The typedef construct can be used to define new (derived) data types in C.**
2. **The typedef is a keyword that is used in C programming to provide existing data types with a new name**
3. **Example:**

   a) **typedef float kilometers_per_hour;**

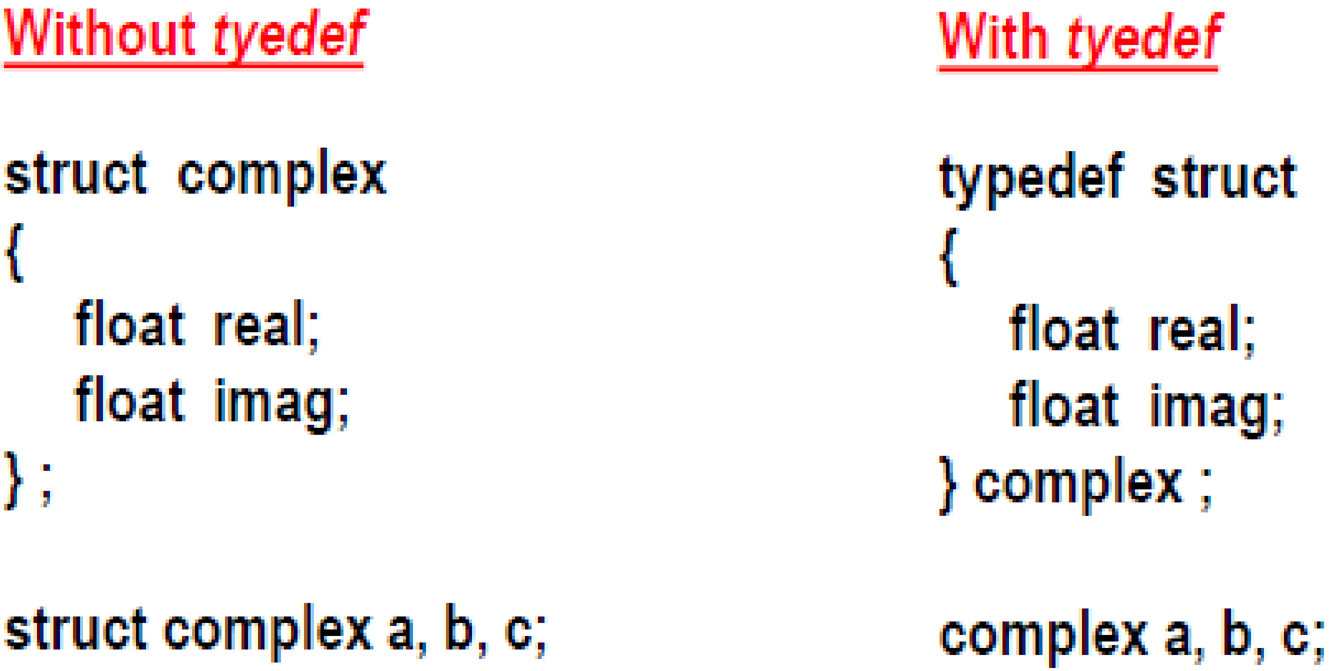
   // kilometers_per_hour is a new data type

   // Note that no variable is allocated space here

   b) kilometers_per_hour speed; // Here speed is a variable

   c) speed = 40;

Using Typedef with Structures



Without tyedef

```
struct  complex
{
    float  real;
    float  imag;
} ;

struct complex a, b, c;
```

With tyedef

```
typedef  struct
{
    float  real;
    float  imag;
} complex ;

complex a, b, c;
```

1. A new data type can be created and used to define the structure variable.

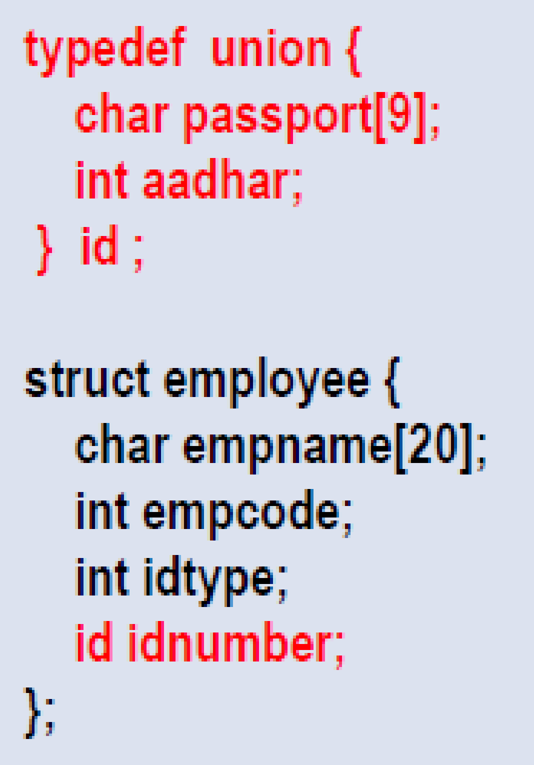
Unions

1. **In a struct, space is allocated as the sum of the space required by its members.**
2. **We use union when we want only one of the members, but don't know which one.**
3. **Members within a union all share the same storage area – save memory**
4. **Whereas each member within a structure is assigned its own unique storage area.**

Unions - Example

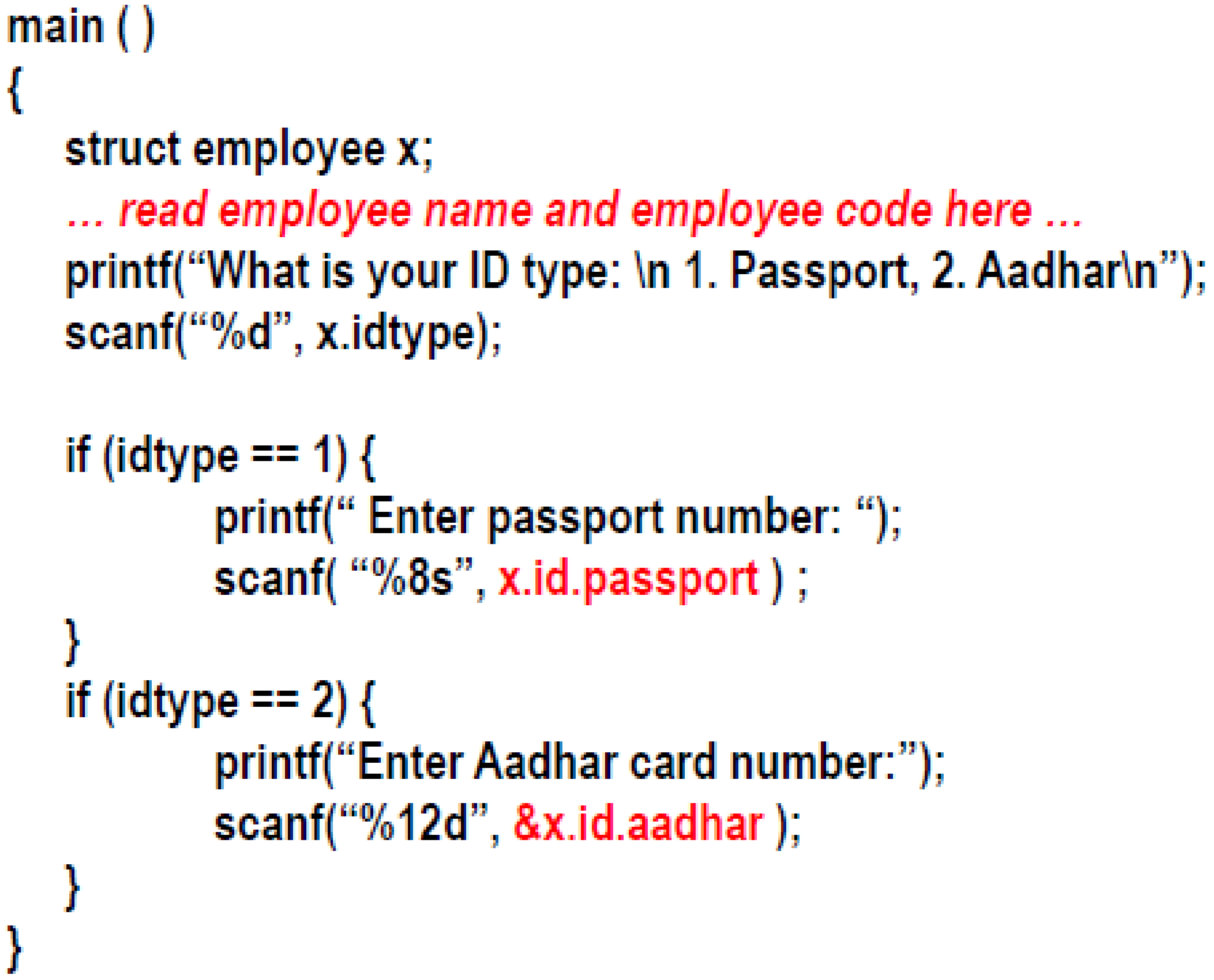
1. Suppose we wish to store an ID for each employee.
2. Some employees may provide passport ID (8 characters)
3. Other employees may provide Aadhar Card Number (12 digit integer)
4. If we use a structure with both these fields, we will waste space
5. So we use Unions

Unions - Example



```
typedef  union {
    char passport[9];
    int aadhar;
 }  id ;

struct employee {
    char empname[20];
    int empcode;
    int idtype;
    id idnumber;
};
```



```
main ( )
{
    struct employee x;
    ... read employee name and employee code here ...
    printf("What is your ID type: \n 1. Passport, 2. Aadhar\n");
    scanf("%d", x.idtype);

    if (idtype == 1) {
            printf(" Enter passport number: ");
            scanf( "%8s", x.id.passport ) ;
    }
    if (idtype == 2) {
            printf("Enter Aadhar card number:");
            scanf("%12d", &x.id.aadhar );
    }
}
```