PDEU PANDIT DEENDAYAL ENERGY UNIVERSITY

Formerly **Pandit Deendayal Petroleum University**

Computer Programming – I (24CP101T)

Dr. Rajeev Kumar Gupta
Assistant Professor
Pandit Deendayal Energy University
Gandhinagar, Gujarat

Structures

- Data used in real life is complex.
- The primitive data types which are provided by all programming languages are not adequate enough to handle the complexities of real life data.
- **Examples:**
- **Date:** A date is a combination of day of month, month and year.
- **Address:** Address of a person can consist of name, plot number, street, city, pin (zip) code and state.

- A structure is a set of interrelated data.
- A structure in C, is a set of primitive and non primitive data types which are related to business and are grouped together to form a new data type.
- A structure is a mechanism provided by the language to create custom and complex data types
- A structure can be declared using the '**struct**' keyword.
- Each variable inside a structure can be of different data type

Declaring a Structure

- **Syntax:**

```
struct structur_name
{
data-type member-1;
data-type member-2;
……
data-type member-n;
};
```

- A structure declaration ends with a semicolon.

Example:

- Date is a simple data structure, but not available as a built-in data type in C

```
struct date {
int Day;
int Month;
int Year;
};
```

Ex

- Create book structure to store book information.

```
struct book
{
char name[10];
float price;
int pages;
};
```

- Once the new structure data type has been defined one or more variables can be declared to be of that type

```
struct book b1, b2, b3 ;
```

structure variables

```
struct book
{
char name[10];
float price;
int pages;
} ;
struct book b1, b2, b3;
```

```
struct book
{
char name [10];
float price;
int pages;
} b1, b2, b3;
```

```
typeof struct
{
char name[10] ;
float price ;
int pages ;
} book;
```

In this type of declaration, no need to use struct keyword during the declaration of a Variable.

Accessing Structure Elements

- In arrays we can access individual elements of an array using a subscript.
- They use a dot (.) operator.
- So to refer to **pages** of the structure defined in our sample program we have to use.

```
b1.pages
```

- Similarly, to refer to **price** we would use

```
b1.price
```

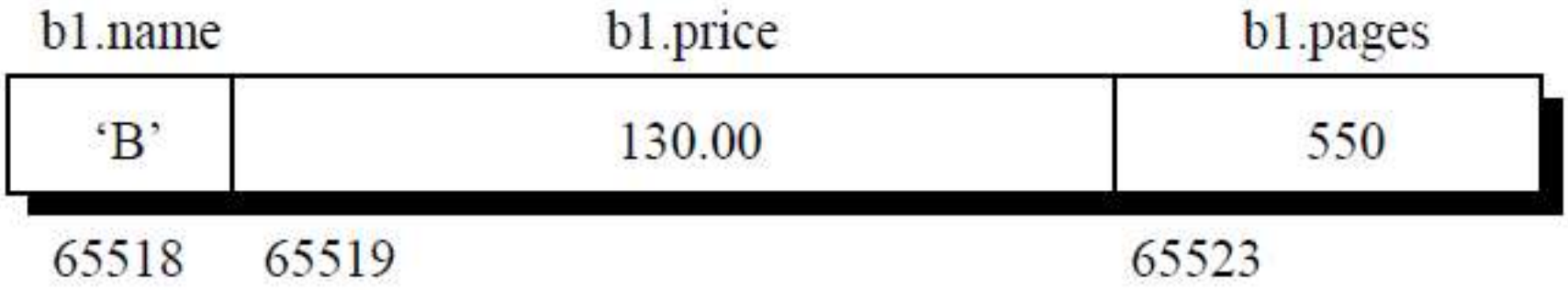
- If you are dealing with the pointer then use -> instead of . operator.

```
B1->price
```

How Structure Elements are Stored

```
main( )
{
struct book
{
char name;
float price;
int pages;
};
struct book b1={'B',130.00,550};
printf("\nAddress of name=%u",&b1.name);
printf("\nAddress of price=%u",&b1.price);
printf("\nAddress of pages=%u",&b1.pages);
}
```

Address of name = 65518
Address of price = 65519
Address of pages = 65523



Structure variables initialization

- Like primary variables and arrays, structure variables can also be initialized where they are declared.

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

```c
#include <stdio.h>
struct student
{
    int roll;
    char name[50];
    float grade;
};

int main() {
  struct student s1, s2;
  printf("Enter roll no");
  scanf("%d", &s1.roll);
  getchar();
   printf("\nEnter name");
  scanf("%[^\n]", s1.name);
```

```c
   printf("\nEnter grade");
  scanf("%f", &s1.grade);


 printf("\nS1 Roll No is =%d", s1.roll);
  printf("\nS1 Name is =%s", s1.name);
   printf("\nS1 grade =%f", s1.grade);
    return 0;
}
```

Array of Structures

```
void main()
{
struct book
{
char name;
float price;
int pages;
};
struct book b[100];
int i;
                    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);
                      }
                    for(i=0;i<=99;i++)
                        printf("\n%c %f %d",b[i].name,b[i].price,b[i].pages);
                    }
```

Additional Features of Structures

1) The values of a structure variable can be assigned to another structure variable of the same type using the assignment operator.

```
main( )
{
struct employee
{
char name[10];
int age;
float salary;
};
struct employee e1={"Sanjay",30,5500.50};
struct employee e2,e3;
/* piece-meal copying */
strcpy(e2.name,e1.name);
e2.age=e1.age;
e2.salary=e1.salary;
/* copying all elements at one go */
e3 = e2;
printf("\n%s %d %f",e1.name,e1.age,e1.salary);
printf("\n%s %d %f",e2.name,e2.age,e2.salary);
printf("\n%s %d %f",e3.name,e3.age,e3.salary);
}
```

```
Sanjay 30 5500.500000
Sanjay 30 5500.500000
Sanjay 30 5500.500000
```

3) Like an ordinary variable, a structure variable can also be passed to a function.

We may either pass individual structure elements or the entire structure variable at one go

```
/* Passing individual structure elements */
main()
{
struct book
{
char name[25];
char author[25];
int callno;
};
struct book b1 = {"Let us C","YPK",101};
display(b1.name,b1.author,b1.callno);
}
```

```
display(char *s,char *t,int n)
{
printf("\n%s %s %d",s,t,n);
}
```

```
Let us C YPK 101
```

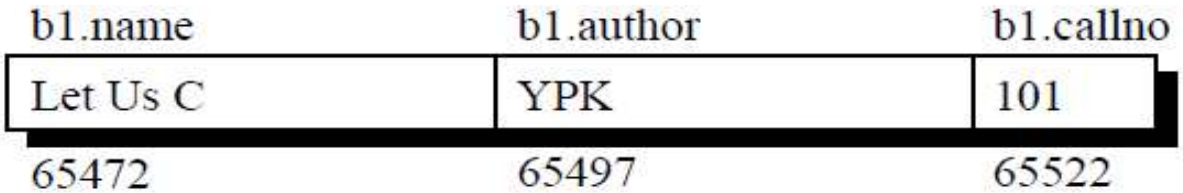
/* Passing structure variable*/

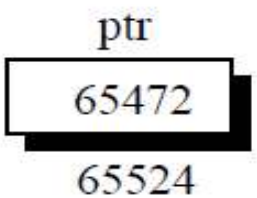
```
struct book
{
char name[25] ;
char author[25] ;
int callno ;
} ;
main( )
{
struct book b1 = { "Let us C", "YPK", 101 } ;
display ( b1 ) ;
}
display ( struct book b )
{
  printf ( "\n%s %s %d", b.name,    b.author, b.callno ) ;
}
```

```
Let us C YPK 101
```

Pointers to structure

- The way we can have a pointer pointing to an **int**, similarly we can have a pointer pointing to a **struct**.






```
main( )
{
struct book
{
char name[25];
char author[25];
int callno;
};
struct book b1={"Let us C","YPK",101};
struct book *ptr;
ptr=&b1;
printf("\n%s %s %d",b1.name,b1.author,b1.callno);
printf("\n%s %s %d",ptr->name,ptr->author,ptr->callno);
}
```

```c
#include <stdio.h>
struct student
{
    int roll;
    char name[50];
    float grade;
};

int main() {
  struct student s1, *s2;
  s2=&s1;
  printf("Enter roll no");
  scanf("%d", &s2->roll);
  getchar();
```

```c
   printf("\nEnter name");
  scanf("%[^\n]", s2->name);

   printf("\nEnter grade");
  scanf("%f", &s2->grade);


 printf("\nS1 Roll No is =%d", s2->roll);
  printf("\nS1 Name is =%s", s2->name);
   printf("\nS1 grade =%f", s2->grade);
    return 0;
}
```

Union

- **Union** can be defined as a user-defined data type which is a collection of different variables of different data types in the same memory location.

- But unlike structures, all the members in the C union are stored in the same memory location. Due to this, only one member can store data at the given instance.

```
union abc
{
  int a;
char b;
}var;
int main()
{
  var.a = 66;
  printf("\n a = %d", var.a);
  printf("\n b = %d", var.b);
}
```

a = 66
b=66

```
union abc
{
  int a;
char b;
}var;
int main()
{
  var.a = 66;
  printf("\n a = %d", var.a);
  printf("\n b = %c", var.b);
}
```

a = 66
b=B

```
union abc {
int a;
char b[10];
}var;

int main()
{ var.a = 66;
strcpy(var.b, "rajeev");
printf("\n a = %d", var.a); \
printf("\n b = %s", var.b); }
```

```
#include <stdio.h>
#include <string.h>

union abc {
   int a;
   char b[10];
} var;

int main() {
  var.a = 66;
  printf("\n a = %d", var.a);  // Print `a` before setting
`b`

  strcpy(var.b, "raj");  // Now set `b`
  printf("\n b = %s", var.b);  // Print `b` after setting it

  return 0;
}
```

a = garbage
b=raj

a = 66
b=raj

- Memory Optimization: If you know that at any given time, only one of several variables will be used, a union can save memory by using the same space for all members.

Differences between Structure and Union

	STRUCTURE	UNION
Keyword	The keyword **struct** is used to define a structure	The keyword **union** is used to define a union.
Size	When a variable is associated with a structure, the compiler allocates the memory for each member. The size of structure is **greater than or equal to the sum of sizes of its members.**	when a variable is associated with a union, the compiler allocates the memory by considering the size of the largest memory. So, size of **union is equal to the size of largest member.**
Memory	Each member within a structure is assigned unique storage area of location.	Memory allocated is shared by individual members of union.
Value Altering	Altering the value of a member will not affect other members of the structure.	Altering the value of any of the member will alter other member values.
Accessing members	Individual member can be accessed at a time.	Only one member can be accessed at a time.
Initialization of Members	Several members of a structure can initialize at once.	Only the first member of a union can be initialized.

```c
#include <stdio.h>

typedef struct {
    int roll;
    char name[20];
    double phy, chem, maths;
}Student;

void inputStudent(Student *);
char getGrade(Student);
void printStudent(Student);
double getAverageMarks(Student);

void inputStudent(Student *sp){
    printf("Roll: ");
    scanf("%d", &sp->roll);
    printf("Name: ");
    scanf(" %[^\n]", sp->name);
    printf("Marks in Physics: ");
    scanf("%lf", &sp->phy);
    printf("Marks in Chemistry: ");
    scanf("%lf", &sp->chem);
    printf("Marks in Physics: ");
    scanf("%lf", &sp->maths);
}

double getAverageMarks(Student s){
    double sum = 0.0;
    sum = s.phy + s.chem + s.maths;
    return sum/3.0;

}
```

```c
char getGrade(Student s){
    char grade;

    double avg = getAverageMarks(s);

    if (avg>=90)
        grade = 'A';
    else if(avg>=80 && avg<90)
        grade = 'B';
    else if(avg>=70 && avg<80)
        grade = 'C';
    else if(avg>=60 && avg<70)
        grade = 'D';
    else if(avg>=50 && avg<60)
        grade = 'E';
    else
        grade = 'F';
    return grade;
}

void printStudent(Student s){
    printf("-------------- Student Details ----------------\n");
    printf("-----------------------------------------------\n");
    printf("Roll:      %-4d\n", s.roll);
    printf("Name:      %-20s\n", s.name);
    printf("Physics:   %-10.2lf\n", s.phy);
    printf("Chemistry: %-10.2lf\n", s.chem);
    printf("Math:      %-10.2lf\n", s.maths);
    printf("Grade:     %-4c\n", getGrade(s));
    printf("-----------------------------------------------\n");
}

int main() {
    Student s;
    inputStudent(&s);
    printStudent(s);
    return 0;
}
```

Exercise

1) Create a structure to specify data on students given below:

Roll number, Name, Department, Course, Year of joining

Assume that there are not more than 450 students in the collage.

(a) Write a function to print names of all students who joined in a particular year.

(b) Write a function to print the data of a student whose roll number is given.