Structure and Union

Lalit Matoliya

Structure

- A structure gathers together, different atoms of information that comprise a given entity.
- A book is a collection of things such as a title, an author, a call no, a publisher, number of pages, date of publication etc.
- For example, suppose you want to store data about book. You might want to store its name (a string), its price (a float) and number of pages in it(an int) and so on.

Declaring Structure

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

Accessing Structure Elements

b1.pages , b1.price & b1.name

How structure Elements are stored

```
main()
{
    struct book
    {
            char name;
            float price;
            int pages;
    };
    struct book b1={‘B’,130.00,550};
    printf(“Address of name=%d\n”,&b1.name);
    printf(“Address of price=%d\n”,&b1.price);
    printf(“Address of pages=%d\n”,&b1.pages);
}
```

Copy one structure to another

```
main()
{
    struct employee
    {
            char name[10];
            int age;
            float salary;
    };
    struct employee e1={“Sanjay”,30,5500};
    struct employee e2,e3;
    //Copy piece by piece
    strcpy(e2.name,e1.name);
    e2.age=e1.age;
    e2.salary=e1.salary;
```

```
//Copying all elements at one go
e3=e2;

printf(“%s%d%f\n”,e1.name,e1.age,e1.salary);
printf(“%s%d%f\n”,e2.name,e2.age,e2.salary);
printf(“%s%d%f\n”,e3.name,e3.age,e3.salary);

}
```

Array of structure

```
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);
}
```

Passing Structure element to Functions

```
main()
{
    struct book
    {
            char name[25];
            char author[25];
            int callno;
    };
    struct book b1={“Easy Steps to Learn C”, “Lalit Matoliya”, 101};
    display(b1.name, b1.author, b1.callno);
}
display(s,t,n)
char *s,*t;
int n;
{
    printf(“%s\n%s\n%d”,s,t,n);
}
```

Passing Entire Structure to Functions

```
struct book
{
    char name[25];
    char author[25];
    int callno;
};
main()
{
    struct book b1={“Easy Steps to Learn C”,”Lalit Matoliya”,101};
    display(b1);
}
display(b)
struct book b
{
    printf(“%s\n%s\n%d”,b.name,b.author,b.callno);
}
```

Structure Pointers

```
main()
{
    struct book
    {
            char name[25];
            char author[25];
            int callno;
    };
    struct book b1={“Easy Steps to Learn C”,”Lalit Matoliya”,101};
    struct book *ptr;
    ptr=&b1;
    printf(“%s%s%d\n”,b1.name,b1.author,b1.callno);
    printf(“%s%s%d\n”,ptr->name,ptr->author,ptr->callno);
    printf(“%s%s%d\n”,*(ptr).name, *(ptr).author, *(ptr).callno);
}
```

Passing address of structure variable

```
struct book
{
    char name[25];
    char author[25];
    int callno;
};
main()
{
    struct book b1={“Easy Steps to Learn C”,”Lalit Matoliya”,101};
    display(&b1);
}
display(b)
struct book *b;
{
    printf(“%s\n%s\n%d”,b->name,b->author,b->callno);
}
```

Structure within structure

```
main()
{
    struct address
    {
            char phone[15];
            char city[25];
            long int pin;
    };
    struct emp
    {
            char name[25];
            struct address a;
    };
    struct emp e={“Kapil”,”325644”,”Sikar”,332001};
    printf(“Name=%s Phone=%s\n”,e.name, e.a.phone);
    printf(“City=%s Pin=%d”,e.a.city,e.a.pin);
}
```

Union

- Both Structure and Unions are used to group a number of different variables together.
- But while a structure enables us to treat a number of different variables stored at different places in memory, a union enables us to treat the same space in memory as a number of different variables.
- Union offers a way for a section of memory to be treated as a variable of one type on one occasion, and as a different variable, of a different type, on another occasion.
- **Declaration of Union**

```
union a
{
        int i;
        char j;
        float k;
};
```

Union…

- They are useful for applications involving multiple elements, where values need not be assigned to all the elements at any one time.

THANKS