Lecture – 19

Structures: A structure is a collection of one or more variables, possibly different types, grouped together under a single name for convenient handling. Structures help to organize complicated data particularly in large programs because they permit a group of related variables to be treated as a unit instead of as separate entities.

General format of structure:

```
struct tag_name{
        type element 1;
        type element 2;
                .
                .
                .
        type element n;
} variables;
```

The keyword struct introduces a structure declaration, which is a list of declaration enclosed in the braces. An optional name called a structure tag name may follow the word struct. The variables named in the structure are called members. A structure member or tag and an ordinary variable can have the same name without conflict, since they can be always distinguished by context. A structure declaration defines a type. The right brace that terminates the list of members may be followed by a list of variables, just as for any basic types. That is

```
struct {   .   .   .   }a, b, c, … , n;
```

Example 1:

```
void main()
{
        struct book
        {
                char name[20];
                float price;
                int pages;
        }b = {“Teach yourself C”, 45.35, 640};
        printf(“%s\t%f\t%d”,b .name, b .price, b .pages);
}
```

One structure can be nested with another structure.

Example 2:

```
void main()
{
        struct address
        {
                char phone[15];
                char city[25];
                int pin;
        };
        struct emp
        {
                char name[25];
                struct address a;
        };
        struct emp e = {“Karim”, “0171530476”, “Dhaka”,10};
        printf(“ name = %s\n phone = %s”, e .name, e .a .phone);
        printf(“\n city = %s \n pin = %d”, e .a .city, e .a .pin);
}
```

typedef: Syntex of using typedef is given below:

```
typedef   data_type   variable;
```

Example 3:

```
void main()
{
        typedef int x;
        x y;
        y = 5;
        printf(“%d”,y);
}
```

Example 4:

```
void main()
{
        struct student
        {
                char name[100];
                int roll;
                char address[100];
                float gpa;
        }
        typedef struct student std;
        std std1,std2 = {“Rahim”,25, “South Kamalapur, Dhaka”, 3.78};
        std1 = std2;
        printf(“%s\t%d\t%s\t%f ”, std1 .name, std1 .roll, std1 .address, std1 .gpa);
}
```

Example 5:

```
void main()
{
    int i;
    struct point
    {
        int x;
        int y;
    }p[5];
    for(i = 0; i < 5; i ++)
        scanf(“%d%d”, & p[i] . x, & p[i] . y);
    for(i = 0; i < 5; i ++)
        printf(“%d\t%d\n”, p[i] . x, p[i] . y);
}
```

Example 6:

```
void main()
{
    struct point * p;
    struct point
    {
        int x;
        int y;
    };
    p = (struct point *) malloc (sizeof(struct point));
    p → x = 5 ;
    p → y = 7;
    printf(“%d\t%d\n”, p → x, p → y);
}
```