COMPUTER PROGRAMMING

Structures and Union

Topics to be covered

- Defining a Structure
- Declaring a structure variables
- Accessing Structure Elements
- Union

Need of Structure

- Problem:
  - How to group together a collection of data items of different types that are logically related to a particular entity??? (~~**Array**~~)

  Solution: ***Structure***

Structure

- A structure is a collection of variables of different data types under a single name.
- The variables are called members of the structure.
- The structure is also called a user-defined data type.

Defining a Structure

```
struct structure_name {
        member 1;
        member 2;
        :
        member m;
      };
```

- struct is the required C keyword
- structure_name is the name of the structure
- member 1, member 2, … are individual member declarations
- ***The members of a structure do not occupy memory until they are associated with a structure_variable.***

Structure variables

- Once a structure has been defined, the individual structure-type variables can be declared as:

  struct structure_name var_1, var_2, …, var_n;

Example

- A structure definition

```
struct item {
        char name[30];
        int  itemcode;
        float  price;
      };
```

- Defining structure variables:

```
struct item  a1, a2, a3;
```

A new data-type

Another way of declaring structure variables

- It is possible to combine the declaration of the structure with that of the structure variables:

```
struct item
        {
           member 1;
           member 2;
           :
           member m;
         } var_1, var_2,…, var_n;
```

- Declares three variables of type struct item

Accessing Structure Elements

- The member variables are accessed using the dot (.) operator or member operator.
- A structure member can be accessed by writing

  variable.member

  where variable refers to the name of a structure-type variable, and member refers to the name of a member within the structure.
- Examples:

  a1.name, a2.name, a1.itemcode, a3.price

Example

```
#include <stdio.h>
struct complex
{
   int real, img;
};
int main()
{
   struct complex a, b, c;
   printf("Enter a and b where a + ib is the first complex number.\n");
   scanf("%d%d", &a.real, &a.img);
   printf("Enter c and d where c + id is the second complex number.\n");
   scanf("%d%d", &b.real, &b.img);
   c.real = a.real + b.real;
   c.img = a.img + b.img;
 printf("Sum of the complex numbers: (%d) + (%di)\n", c.real, c.img);
 return 0;
}
```

Arrays of Structures

- Once a structure has been defined, we can declare an array of structures

```
struct item a[5];
```

type name

  - The individual members can be accessed as:

```
a[i].name
a[2].price;
```

Arrays within Structures

- A structure member can be an array

```
struct  student
{
    char  name[30];
    int  roll_number;
    int  marks[5];
    char  dob[10];
} a1, a2, a3;
```

- The array element within the structure can be accessed as:

  a1.marks[2], a1.dob[3],…

Structure Initialization

- Structure variables may be initialized following similar rules of an array. The values are provided within the second braces separated by commas
- An example:

```
struct complex a={1.0,2.0}, b={-3.0,4.0};
```

```
a.real=1.0;   a.imag=2.0;
b.real=-3.0;  b.imag=4.0;
```

Union

- A **union** is a special data type available in C that allows to store different data types in the same memory location.
- You can define a union with many members, but only one member can contain a value at any given time.

Union (Contd…)

- A union is declared using the keyword *union as:*

```
union student
                    {
                    char name[20];
                    int roll_no;
                    float marks;
                    char section;
                    };
              union student s;
```

- While accessing union members, we should make sure that we are accessing the member whose value is currently residing in the memory. Otherwise we will get erroneous output (which is machine dependent).

Example 1

```
#include <stdio.h>
 #include <string.h>
  union Data
   { int i;
   float f;
   char str[20];
     };
 int main( )
{
union Data data;
 printf( "Memory size occupied by data : %d\n", sizeof(data));
return 0;
 }
```

Output:

Memory size occupied by data : 20

Example 2

```
#include <stdio.h>
#include <string.h>
 union Data
{ int i;
 float f;
char str[20];
};
int main( )

{
 union Data data;
 data.i = 10;
 data.f = 220.5;
 strcpy( data.str, "C Programming");
 printf( "data.i : %d\n", data.i);
 printf( "data.f : %f\n", data.f);
 printf( "data.str : %s\n", data.str);
  return 0;
 }
```

Structure vs Union

- Both **structure and unions are used to group a number of different variables together. Syntactically** both structure and unions are exactly same. The main difference between them is in storage.
- In structures, each member has its own memory location but all members of union use the **same memory location** which is equal to the greatest member's size.

Thank You