Bit Field

Bitfield provides exact amount of bits required for storage of values .If a variable value is 1 or 0 we need a single bit to store it. In the same way if the variable is expressed between 0 and 3 , then the two bits are sufficient for storing these values . Similarly if a variable assumes values between 0 and 7 then three bits will be sufficient to hold the variable and so on . The number of bits required for a variable is specified by non-negative integer followed by colon .

To hold the information we use the variables. The variables occupy minimum one byte for **char** And two byte for int . Instead for int using complete integer if bits are used , and space of memory can be saved .

Bit Field Declaration

The declaration of a bit-field has the form inside a structure:

```
struct
{
  type [member_name] : width ;
};
```

Below the description of variable elements of a bit field:

Elements	**Description**
Type	An integer type that determines how the bit-field's value is interpreted. The type may be int, signed int, unsigned int.
member_name	The name of the bit-field.
Width	The number of bits in the bit-field. The width must be less than or equal to the bit width of the specified type.

The variables defined with a predefined width are called **bit fields**. A bit field can hold more than a single bit for example if you need a variable to store a value from 0 to 7 only then you can define a bit field with a width of 3 bits as follows:

```
struct
{
  unsigned int age : 3;
} Age;
```

The above structure definition instructs C compiler that age variable is going to use only 3 bits to store the value, if you will try to use more than 3 bits then it will not allow you to do so. Let us try the following example:

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

struct
{
  unsigned int age : 3;
} Age;

int main( )
{
   Age.age = 4;
   printf( "Sizeof( Age ) : %d\n", sizeof(Age) );
   printf( "Age.age : %d\n", Age.age );

   Age.age = 7;
   printf( "Age.age : %d\n", Age.age );

   Age.age = 8;
   printf( "Age.age : %d\n", Age.age );

   return 0;
}
```

When the above code is compiled it will compile with warning and when executed, it produces the following result:

```
Sizeof( Age ) : 3
Age.age : 4
Age.age : 7
Age.age : 0
```

Example :-

Q1 . WAP to Read and store information about insurance policy holder.The information contains details like gender , whether the holder is minor and major , policy name and duration of the policy . Make use of bit-fields to store this information .

Solution :-

```
#include <stdio.h>
#include <conio.h>
void main()
{
struct policy_holder
{
unsigned gender  : 1; // 0-Male , 1-Female
unsigned status    : 1; // 0-Minor , 1-Major
char name[20];
unsigned  dr  :5;
};
Struct  policy_holder  h;
int  g , s , d ;
char n[20];
printf(“Enter gender(0-Male , 1- Female) of the policy holder:”);
scanf(%d”,&g);

printf(Enter status(0-Minor, 1- Major) of the policy holder:”);
scanf(%d”,&s);

printf(Enter name of the  policy holder:”);
scanf(%s”,n);

printf(Enter duration(1 to 25) of the policy holder:”);
scanf(%d”,&d);

h.gender=g;
h.status=s;
strcpy(h.name ,n);
h.dr=d;
printf(“\n Name : %s”, h.name);
printf(“\n Gender : %s”, h.gender==0?”Male ”:”Female”);
printf(“\n Status : %s”, h.status==0?”Minor”:”Major”);
printf(“\nDuration  : %d”, h.dr);
getch();
}
```

Q 2. WAP , which stores information about a date in a structure containing three members – day ,month and year. Using bit fields the day number should get stored in first 5 bits of day , the month number in 4 bits of month and year in 12 bits of year . Write a program to read date of joining of 10 employees and display them in ascending order of year .

Solution :-

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

```
#include <conio.h>
void main()
{
struct date
{
unsigned day :5;
unsigned month :4;
unsigned year :12;
};
Struct date dt[10] , temp;
int  i , j , d , m , y ;

printf(Enter joining dates (dd-mm-yyyy) of 10 employees \n”);
for(i=0;i<10;i++)
        {
        scanf(“%d %d %d ”, &d , &m , &y);
                if(((d<1)||(d>31)) || ((m<1)||(m>12))|| ((y<1900)||(y>2014)))
                {
                        Printf(“\n invalid date , enter new date \n”);
                        i--;
                        continue ;
                }
        dt[i].day=d;
        dt[i].month=m;
        dt[i].year=y;
}

for(i=0;i<9;i++)
{
        for(j=0;j<9;j++)
       {
                if(dt[j].year < dt[i].year)
                        {
                        temp=dt[i];
                          dt[i]=dt[j];
                          dt[j]=temp;
                           }
        }
}

for(i=0;i<10;i++)
        printf(“\n%d  %d  %d  ”, dt[i].day ,dt[i].month,dt[i].year);

getch();
}
}
```