GE3 COMPUTER SCIENCE

C AND C ++ LECTURE SERIES *FOR*

B.SC 3RD SEMESTER *BY*

SUBHADIP MUKHERJEE

DEPARTMENT OF COMPUTER SCIENCE

KHARAGPUR COLLEGE

LECTURE 12

STRUCTURES AND UNIONS

A single structure might contain integer elements, floating-point elements and character elements. Pointers, arrays and other structures can also be included as elements within a structure. The individual structure elements are referred to as ***members.***

```
struct  tag  {
        member 1;
        member 2;
        . . . . .
        member m;
};
```

```
storage-class  struct  tag  variable 1,  variable 2,  . . .,  variable n;
```

```
struct account  {
        int acct_no;
        char acct_type;
        char name[80];
        float balance;
};
```

```
storage-class  struct  tag  {
                member 1;
                member 2;
                . . . . .
                member m;
}  variable 1,  variable 2,  . . .,  variable n;
```

STRUCTURES AND UNIONS

```
struct date  {
      int month;
      int day;
      int year;
};
```

```
struct account  {
      int acct_no;
      char acct_type;
      char name[80];
      float balance;
      struct date lastpayment;
};
```

```
static  struct  account  customer = {12345, 'R', "John W. Smith", 586.30, 5, 24, 90};
```

STRUCTURES AND UNIONS

```
#include <stdio.h>

main()    /* determine the size of a structure */
{
     struct date  {
          int month;
          int day;
          int year;
     };

     struct account  {
          int acct_no;
          char acct_type;
          char name[80];
          float balance;
          struct date lastpayment;
     }  customer;

     printf("%d\n", sizeof customer);
     printf("%d", sizeof (struct account));
}
```

```
93
93
```

STRUCTURES AND UNIONS

USER-DEFINED DATA TYPES (typedef)

- The **typedef** feature allows users to define new data-types that are equivalent to existing data types. Once a user-defined data type has been established, then new variables, arrays, structures, etc. can be declared in terms of this new data type.

```
typedef   type   new-type;
```

```
typedef  struct  {
      int acct_no;
      char acct_type;
      char name[80];
      float balance;
}  record;

record oldcustomer, newcustomer;
```

STRUCTURES AND UNIONS

STRUCTURES AND POINTERS

```
typedef struct  {
      int acct_no;
      char acct_type;
      char name[80];
      float balance;
}  account;

account  customer, *pc;
```

```
pc = &customer;
```

```
struct  {
      member 1;
      member 2;
      . . . . .
      member m;
}  variable,  *ptvar;
```

STRUCTURES AND UNIONS

```
#include <stdio.h>

main()

{
    int n = 3333;
    char t = 'C';
    float b = 99.99;

    typedef struct  {
        int month;
        int day;
        int year;
    }  date;

    struct  {
        int *acct_no;
        char *acct_type;
        char *name;
        float *balance;
        date lastpayment;
    }  customer, *pc = &customer;

        customer.acct_no = &n;
        customer.acct_type = &t;
        customer.name = "Smith";
        customer.balance = &b;

        printf("%d %c %s %.2f\n", *customer.acct_no, *customer.acct_type,
                                   customer.name, *customer.balance);
        printf("%d %c %s %.2f", *pc->acct_no, *pc->acct_type,
                                  pc->name, *pc->balance);
}
```

```
3333 C Smith 99.99
3333 C Smith 99.99
```

STRUCTURES AND UNIONS

UNIONS

- Unions, like structures, contain members whose individual data types may differ from one another. However, the members within a union all share the same storage area within the computer's memory, whereas each member within a structure is assigned its own unique storage area.

```
union  tag  {
     member 1;
     member 2;
     . . . . .
     member m;
};
```

```
storage-class  union  tag  variable 1,  variable 2,  . . .,  variable n;
```

```
storage-class  union  tag  {
          member 1;
          member 2;
          . . . . .
          member m;
}  variable 1,  variable 2,  . . .,  variable n;
```

```
union  id  {
     char color[12];
     int size;
}  shirt, blouse;
```

STRUCTURES AND UNIONS

UNIONS

```
#include <stdio.h>

main()

{
    union id  {
        char color;
        int size;
    };

    struct  {
        char manufacturer[20];
        float cost;
        union id description;
    } shirt, blouse;
```

```
    /* assign a value to color */
    shirt.description.color = 'w';
    printf("%c %d\n", shirt.description.color, shirt.description.size);

    /* assign a value to size */
    shirt.description.size = 12;
    printf("%c %d\n", shirt.description.color, shirt.description.size);
}
```

```
2
w -24713
@ 12
```

COMPILE AND RUN A C CODE

Thank You

End of Lecture 12

Subhadip Mukherjee

Department of Computer Science

Kharagpur College

Kharagpur, India