C Programming

Day4

By Manjiri Deshpande

Why Structures?

- Array is a data structure containing elements of same type.
- In real world applications,one needs to store data of different types.
- Diffculty
- More than one array is required.
- Handling arrays becomes difficult with increase in the amount of data.
- E.g:- to store information about a customer,customer_id,name,account balance etc…

What is a Structure?

- User defined data type.
- Convenient way of grouping of members of different types.
- May contain any number of members of different types.
- Can form the basis for more complex constructions ,such as linked lists.
- It may contain any number of members of different types.

Syntax of Structure:-

```
struct structure_name
{
    data_type member1;
    data_type member2;
    .
    .
    data_type memeberN;
};
```

Example:-

```
struct customer
{
 char custName[20];
 int custId;
Float amt;
  char address[25];
 char phone[10];
}
```

Structure Variables

- The members of structure can be processed individuall

```
struct savAcc
{
 int custId;
 char custName[20];
float balAmt;
}
```

Structure variables can be initialized when declared

```
struct savAcc ac1,ac2;
struct savAcc ac3={1005,"smita",6000};
```

Structure variable can also be defined

```
struct savAcc{
Char custName[20],int custId;
float balAmt;}ac1;
```

typedef

- The **typedef** is a keyword used in C programming to provide some meaningful names to the already existing variable in the C program
- . It behaves similarly as we define the alias for the commands. In short, we can say that this keyword is used to redefine the name of an already existing variable.
- Syntax of typedef

typedef <existing_name> <alias_name>

```
typedef int tool;
  tool x,y;
```

Using typedef with structures

- **Consider the below structure declaration:**

```
struct student
{
char name[20];
int age;
};
```

- In the above structure declaration, we have created the variable of **student** type by writing the following statement:

```
struct student s1;
```

- The above statement shows the creation of a variable, i.e., s1, but the statement is quite big. To avoid such a big statement, we use the **typedef** keyword to create the variable of type **student**.

Example of typedef

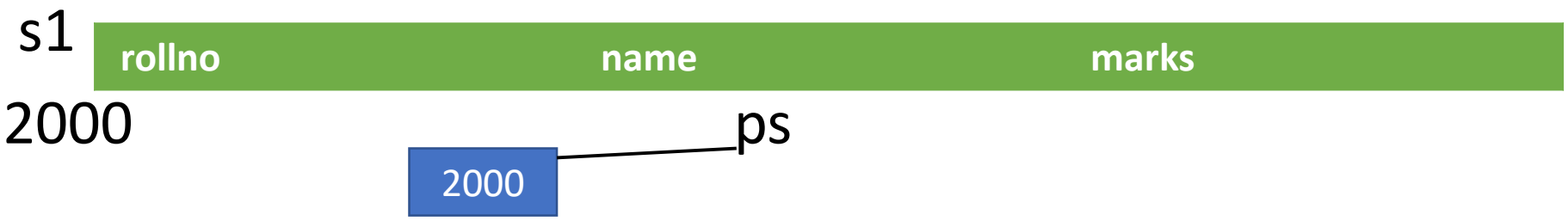
```
struct student
{
char name[20];
int age;
};
typedef struct student stud
stud s1, s2;
```

```
typedef struct student
{
char name[20];
int age;
} stud;
stud s1,s2;
```

Pointer to a Structure

```
Student s1;
Student *ps;
 ps=&s1;

Puts(ps->name);//accessing number using pointer
```



Pointer to a Structure

- The pointer points to the entire structure.
- This is useful when the structure is passed to a function.
- Only the base address of the structure is passed to the function instead of entire structure by value.

Passing Structure to a function

- A structure can be passed as a parameter to a function as
- Individual structure members.
- Entire structure by value
- Entire structure by address.

Array of Structure

- To hold a number of record of similar type ,array of structures is used.
- Student s1[5];
- All records of same type at contiguous memory locations
- The array can be initialized at the time of declaration
- Student s1[2]={(1,”abc”,80.7},{2,”xyz”,70.3}};

S1[0].rollno	S1[0].name	S1[0].marks
S1[1].rollno	S1[1].name	S1[1].marks
S1[2].rollno	S1[2].name`	S1[1].marks

Union

- A user defined type that stores different data types in the same memory space (but not simultaneously).
- Declared in the same way as structures.
- Size of the union is the size of its largest data type.

```
 union employee
{
Int empid;
  double salary
};
```

Memory Representation of union

Memory allocated to structure

Memory allocated to union

Point to Remember

- Union elements are accessed using the same method used for accessing structure elements, using dot or period(.)operator or arrow(->)operator.
- The name of a structure member represents the offset of that member from the start of the structure. In union all members start at the same location in memory.
- The operations performed on union are same as on structures- accessing union members, taking address, assignment of entire union variable to other union variable

Enum

- A user defined data type which consists of set of named integer constants.(symbolic names represent integer constants.

```
enum actype{SAVING=100,FIXED,RECURRING}
```

```
enum actype at;
At=SAVING;
printf(%d”,at);//100
```

- Each member starts from 0 by default and is incremented by 1 for each next member.