2.1 Structure

A structure is a user defined data type in C.

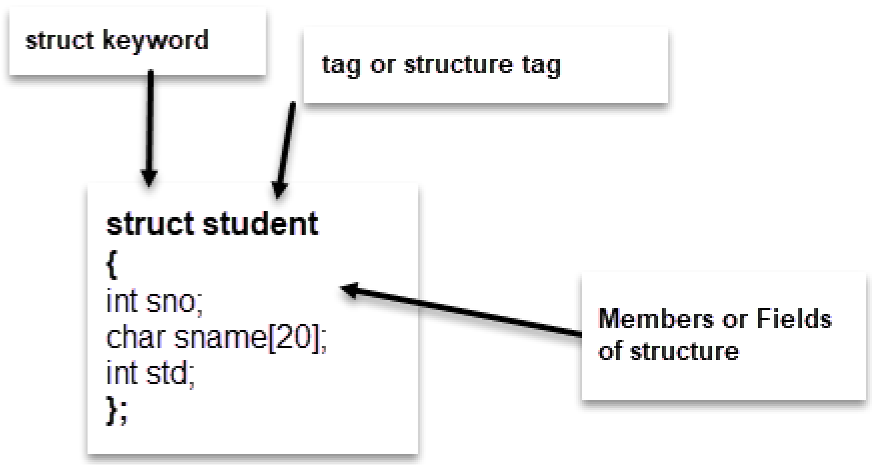
Structure helps to construct a complex data type in more meaningful way.

Arrays allow to define type of variables that can hold several data items of the same kind. For example, int array[10]. Similarly, structure is another user defined data type available in C that allows combining data items of different kinds. For example, struct student{int no,char name[20],int std}. Other way we can say, structure is a method of packing data of different types.

Basically, a structure is used to represent a record. We can define a structure to hold this information.

The structure contain variables are known as: members, fields or structure elements.

The block of the structure is also known as template.



Defining a structure

struct keyword is used to define a structure.

Syntax#

struct struct_name
{
Members;
};
	struct
	{
	Members;
	} struct_name;
		struct struct_name
		{
		Members;
		}[optional]structure variables;
			Struct
			{
			Members;
			} structure variables;

Example#

struct Stu
{
int sno;
	struct
	{
	int sno;
		struct Stu
		{
		int sno;
			struct
			{
			int sno;

char sname[20];
};
	char sname[20];
	} Stu;
		char sname[20];
		}s1,s2,s3;
			char sname[20];
			}s1,s2;

Points to remember structure

- The template of the structure is terminated with " ; "
- Use short and meaningful structure tag names.
- Members of the structure is also separated by " ; "
- After creating template, we need to create object or say structure variables of structure.
- Structure never occupies memory.
- The variable or object of structure occupies the memory.
- We can use sizeof(varname).
- The sizeof structure variable is addition of members of the variable.
- Structure can be nested.

Array vs Structure

Array	Structure
Elements are same as data type	Elements are different data type
Derived data type	programmer-defined one
Behave like a built in data type.	We have to design and declare a data structure before the variables used

Example:
int no[10];
	Example:
	Struct Emp
	{
	int eno;
	char name[20];
	};

2.2 Declaring Structure Variables

A structure variable can either be declared with structure declaration or as a separate declaration like basic types.

Structure variables can be declared in following two ways.

Declaring Structure variables separately

```
struct Bank
{
 char name[20];
 int accno;
 float balance;
} ;

struct Bank b1 , b2;   //declaring variables of Structure Bank
```

RECOMMENDED

Declaring Structure Variables with Structure definition

```
struct Bank
{
 char name[20];
 int accno;
 float balance;
}b1,b2 ;
```

NOT RECOMMENDED

Here b1 and b2 are variables of structure Bank.

2.2.1 Accessing a structure member

After declaring structure variable, to access any member of a structure, we use the member access operator (.).

Syntax#

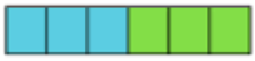
Structure Variable Name.Member name

Example#

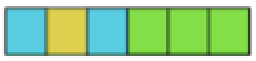
```
struct bank
{
char custname[20];
int accno;
}

main()
{
Bank b1;
b1.accno=101
}
```

2.3 Memory allocation of Structure Variable

Memory is allocated to structure when structure variables are created. Total memory Occupied is equal to the total of each data member size.



Example# Sizeof() operator with structure



```
struct Stu
{
int no;
char name[20];
}s2={2,"Bhavin"};

main()
{
struct Stu s1={1,"Ram"};
clrscr();

printf("\nSizeof(s1) = %d",sizeof(s1));
printf("\nSizeof(s2) = %d",sizeof(s2));

getch();
}
```

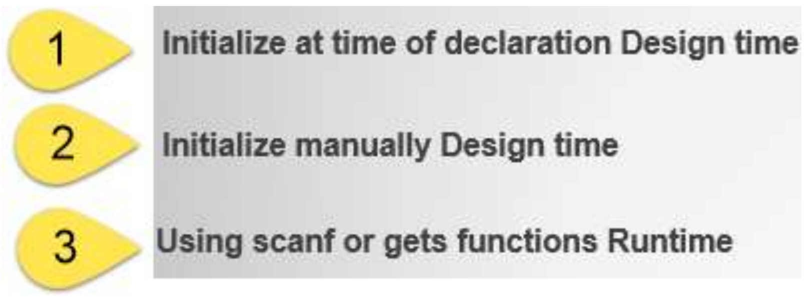
Output#

```
Sizeof(s1) = 22  // Char name[20] + int no =22
Sizeof(s2) = 22
```

2.4 Operations on structure members

2.4.1 Initialization

There are three ways to initialize and accessing members of the structure.



1 Initialize at time of declaration Design time

2 Initialize manually Design time

3 Using scanf or gets functions Runtime

Example# Student structure with three members sno, sname and std. Create three objects and assigned it with different ways.

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

struct stu
{
int sno;
char sname[20];
int std;
};

main()
{
struct stu s1={1,"Ram",11};//declaration time , design time
struct stu s2;
struct stu s3;

clrscr();

//design time
s2.sno=2;
strcpy(s2.sname,"Laxman");
s2.std=11;

//runtime
printf("\nEnter sno =>");
scanf("%d",&s3.sno);

fflush(stdin);

printf("\nEnter sname =>");
gets(s3.sname);

printf("\nEnter std =>");
scanf("%d",&s3.std);

printf("\nS1's Sno = %d Sname = %s Std = %d",s1.sno,s1.sname,s1.std);
printf("\nS2's Sno = %d Sname = %s Std = %d",s2.sno,s2.sname,s2.std);
printf("\nS3's Sno = %d Sname = %s Std = %d",s3.sno,s3.sname,s3.std);

getch();
}
```

Output#

```
Enter sno =>3

Enter sname =>Krishna

Enter std =>11

S1's Sno = 1 Sname = Ram Std = 11
S2's Sno = 2 Sname = Laxman Std = 11
S3's Sno = 3 Sname = Krishna Std = 11
```

2.4.2 Adding structure data members

Example# Create structure student which contains four data members sno,sname , eng and hindi , Scan three objects and print details along with Total of marks

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

struct stu
{
int sno;
char sname[20];
int eng,hindi;
};

main()
{
struct stu s1,s2,s3;

clrscr();

printf("\nEnter sno =>");
scanf("%d",&s1.sno);
fflush(stdin);

printf("\nEnter sname =>");
gets(s1.sname);

printf("\nEnter English and Hindi =>");
scanf("%d %d",&s1.eng,&s1.hindi);

printf("\nEnter sno =>");
scanf("%d",&s2.sno);

fflush(stdin);

printf("\nEnter sname =>");
gets(s2.sname);

printf("\nEnter English and Hindi =>");
scanf("%d %d",&s2.eng,&s2.hindi);

printf("\nEnter sno =>");
scanf("%d",&s3.sno);

fflush(stdin);

printf("\nEnter sname =>");
gets(s3.sname);

printf("\nEnter English and Hindi =>");
scanf("%d %d",&s3.eng,&s3.hindi);

printf("\nNo\tName\tEnglish\tHindi\tTotal");
printf("\n==================================");
printf("\n%d\t%s\t%d\t%d\t%d",s1.sno,s1.sname,s1.eng,s1.hindi,s1.eng+s1.hindi);
printf("\n%d\t%s\t%d\t%d\t%d",s2.sno,s2.sname,s2.eng,s2.hindi,s2.eng+s2.hindi);
printf("\n%d\t%s\t%d\t%d\t%d",s3.sno,s3.sname,s3.eng,s3.hindi,s3.eng+s3.hindi);
printf("\n==================================");

getch();
}
```

Output#

```
Enter sno =>1

Enter sname =>ram

Enter English and Hindi =>11 22
Enter sno =>2

Enter sname =>laxman

Enter English and Hindi =>22 33

Enter sno =>3

Enter sname =>krishna

Enter English and Hindi =>33 44

No     Name    English Hindi   Total
===================================
1      ram     11      22      33
2      laxman  22      33      55
3      krishna 33      44      77
===================================
```

2.4.2 Subtraction operation on data member

Example# Create structure bank which contains accno, customerno and balance. Create an object, initialize members and print also perform Withdrawn operation.

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

struct bank
{
int acno;
char custname[20];
float bal;
};

main()
{
float x;
struct bank b1;
clrscr();

printf("\nEnter Account No =>");
scanf("%d",&b1.acno);

fflush(stdin);

printf("\nEnter Customer Name =>");
gets(b1.custname);
printf("\nEnter Balance =>");
scanf("%f",&b1.bal);

printf("\nBefore WithDrawn");
printf("\nAccount No\tCustomer Name\tBalance");
printf("\n======================================");
printf("\n%d\t\t%s\t\t%.2f",b1.acno,b1.custname,b1.bal);
printf("\n======================================");

printf("\n\nEnter amount of withdrawn =>");
scanf("%f",&x);

b1.bal=b1.bal-x;

printf("\n\nAfter WithDrawn");
printf("\nAccount No\tCustomer Name\tBalance");
printf("\n======================================");
printf("\n%d\t\t%s\t\t%.2f",b1.acno,b1.custname,b1.bal);
printf("\n======================================");

getch();
}
```

Output#

```
Enter Account No =>101

Enter Customer Name =>Jayul

Enter Balance =>10000

Before WithDrawn
Account No      Customer Name   Balance
======================================
101             Jayul           10000.00
======================================

Enter amount of withdrawn =>2000


After WithDrawn
Account No      Customer Name   Balance
======================================
101             Jayul           8000.00
======================================
```

2.4.3 Copying and comparing variables

A structure object can also be assigned to another structure object only if both the structures are of same structure type.

Example: Copy

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

struct Stu
{
int no;
char name[20];
}

main()
{
struct Stu s1={1,"Ram"},s2;
clrscr();

s2=s1;

printf("\n\nS1 's No = %d Name = %s",s1.no,s1.name);
printf("\n\nS2 's No = %d Name = %s",s2.no,s2.name);

getch();
}
```

Output

```
S1's No = 1 Name = Ram
S2's No =1 Name = Ram
```

We cannot use s1==s2, s1!=s2 ,s1>s2 ,etc .C has no operator overloading, so == ,!= , etc. operator are defined only for built-in types. But we can compare object's members.

Example# Compare two student's mark and print details of who got highest marks.

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

struct stu
{
int sno;
char sname[20];
int eng,hindi;
};
```

```
main()
{
struct stu s1={11,"Ram",22,12};
struct stu s2={12,"Jayul",33,44};

clrscr();

if(s1.eng+s1.hindi>s2.eng+s2.hindi)
{
 printf("\nstudent %d %s is got highest marks",s1.sno,s1.sname);
}
else
{
 printf("\nstudent %d %s is got highest marks",s1.sno,s2.sname);
}

getch();
}
```

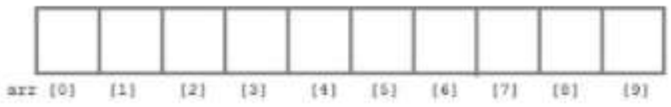
Output#

```
student 11 Jayul is got highest marks
```

2.5 Arrays of structure:

A structure variable of structure represents a single record in memory, if we want more than one record of structure type, we have to create an array of structure or object

Declaring Arrays within Structures



It is possible to define an array of structures, for example: if we are maintaining information of all the employees in the company and if 100 employees are doing job in the company.

We need to use an array with single structure variable. Otherwise we have to take e1,e2,e3,e4….,e100 which is not practically convenient. Instead of that we will use the concept of arrays of structure.

Arrays of structure allow us to store more data items and give flexibility in accessing the data members. Each element of the array represents a structure variable.

Syntax#

struct structurename
{
//members….
}structure variablename[100];
	struct structurename
	{
	members
	};
	main()
	{
	struct structurename variablename[100];
	}

#Example of array in structure :

```
struct Emp
{
int eno;
int salary;
char ename[20];
};

main()
{
struct Emp e[100];
logic.....
}
```

Example# Array of structure variable

```
#include<stdio.h>
#include<conio.h>
#define N 5

struct stu
{
int sno;
char sname[20];
int eng,hindi;
};

main()
{
struct stu s[N];
int i;
clrscr();

for(i=0;i<N;i++)
{
printf("\nEnter sno =>");
scanf("%d",&s[i].sno);

fflush(stdin);

printf("\nEnter sname =>");
gets(s[i].sname);

printf("\nEnter English and Hindi =>");
scanf("%d %d",&s[i].eng,&s[i].hindi);
}

printf("\nNo\tName\tEnglish\tHindi\tTotal");
printf("\n===================================");
for(i=0;i<N;i++)
{
printf("\n%d\t%s\t%d\t%d\t%d",s[i].sno,s[i].sname,s[i].eng,s[i].hindi,s[i].eng+s[i].hin
di);
}
printf("\n===================================");

getch();
}
```

Ouput#

```
Enter sname =>bhavin
Enter English and Hindi =>20 30

Enter sno =>4

Enter sname =>mansi

Enter English and Hindi =>31 36

Enter sno =>5

Enter sname =>neha

Enter English and Hindi =>44 42

No     Name    English Hindi   Total
===================================
1      ram     22      33      55
2      jayul   33      44      77
3      bhavin  20      30      50
4      mansi   31      36      67
5      neha    44      42      86
===================================
```

2.6 Arrays within Structures

C permits the use of arrays in structure members.

As we know, structure is a collection of different data type. Like normal data type, It can also store array as well.

Syntax#	Example#

struct structurename
{
datatype membername;
datatype membername[size];
…
}
	struct stu{
	 int sno;
	 char sname[20];
	 float sub[3];
	}

2.6.1 Accessing Arrays within Structures

```
struct stu{
 int sno;
 char sname[20];
 float sub[3];
}
```

Here, the member sub contains three elements, sub[0], sub[1] and sub[2].
Let's create an object or say structure variable s1 and s2, means we want to store record of two students.

Now we can access, s1.sub[0],s1.sub[1],s1.sub[2] or s2.sub[0],s2.sub[1],s2.sub[2] will refer to the marks obtained by the first s1 and second s2 students.

```
#include<stdio.h>
#include<conio.h>
struct stu
{
int sno;
char sname[20];
int marks[3];
};

main()
{
struct stu s1;
int i,total=0;
clrscr();

printf("\nEnter sno =>");
scanf("%d",&s1.sno);
fflush(stdin);

printf("\nEnter sname =>");
gets(s1.sname);

printf("\nEnter marks of 3 subjects =>");
for(i=0;i<3;i++)
{
scanf("%d",&s1.marks[i]);
}


printf("\n============================");
printf("\nNo = %d\t Name = %s",s1.sno,s1.sname);
for(i=0;i<3;i++)
{
printf("\nMarks of %d Subject => %d",i,s1.marks[i]);
total=total+s1.marks[i];
}
printf("\n==============================");
printf("\nTotal = %d",total);

getch();
}
```

Output#

```
Enter sno =>1

Enter sname =>Ram

Enter marks of 3 subjects =>11 22 33

============================
No = 1   Name = Ram
Marks of 0 Subject => 11
Marks of 1 Subject => 22
Marks of 2 Subject => 33
==============================
Total = 66
```

2.7 Structure within Structure (Nested Structure)

Structure written inside another structure is called nesting of structure.

When one structure is a member element of another structure it is called nesting of structure.

Format of a nested structure#

```
struct structure-name1
{                                                                  outer
        Data members                                               structure
        struct structure-name2
        {                                            inner
                Data members                         structure
        } structure-var1, structure-var2……..;
} structure-var1, structure-var2……..;
```

For example, we have two structures named Date and Student. To make Date nested to Student, we have to define Date structure before and outside Student structure and create an object or say structure variable of Date structure inside Student structure.

Syntax#

1st Way	2nd Way

struct struct1
{
members…..
}
 	
struct struct2
{
members….
struct struct1 members…
}
	struct struct2
	{
	members…..
	  struct struct1
	  {
	  members……
	  }
	}

Example#

1st Way	2nd Way

struct Date
{
int dd,mm,yy;
}
 	
struct Student
	struct Student
	{
	int rno;
	char sname[20];
	struct Date
	{

{
int rno;
char sname[20];
struct Date dob,doj;
};
	int dd,mm,yy;
	} dob,doj;
	};

- The structure Student contains another structure Date as its one of its members.
- Structure members are accessed using dot operator.
- 'Date' structure is nested within Student Structure.
- Members of the 'date' can be accessed using 'Student'

Example#

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

struct student
{
int sno;
char sname[20];
        struct date
        {
        int dd,mm,yy;
        }dob,doj;
};

main()
{
struct student s1;
clrscr();
        printf("\n\nEnter s1's no->");
        scanf("%d",&s1.sno);
        fflush(stdin);
        printf("\n\nEnter s1's Name->");
        scanf("%s",s1.sname);
        fflush(stdin);
        printf("\n\nEnter s1's Date of Birth dd,mm,yy ->");
        scanf("%d %d %d",&s1.dob.dd,&s1.dob.mm,&s1.dob.yy);
        printf("\n\nEnter s1's Date of Join dd,mm,yy ->");
        scanf("%d %d %d",&s1.doj.dd,&s1.doj.mm,&s1.doj.yy);

printf("\n\nS1's No=%d Name=%sDOB=%d/%d/%d DOJ =%d/%d/%d
",s1.no,s1.year,s1.dob.dd,s1.dob.mm,s1.dob.yy,s1.doj.dd,s1.doj.mm,s1.doj.yy);
getch();
}
```

Output#

```
Enter s1's no->1
Enter s1's name->Karna
Enter s1's dd,mm,yy ->21 12 1982
Enter s1's dd,mm,yy ->11 20 2013

S1's No=1 Name=Karna Dob=21/12/1982 DOJ=11/10/2013
```

2.8 Structure and Function

Like normal arguments, we can pass structures as arguments of functions.

A structure variable may be passed into a function as a separate variable or as an array.

Example# Structure variable as a Function argument

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

struct bank
{
int acno;
char custname[20];
float bal;
};


void printData(struct bank b1)
{
printf("\nAccount No\tCustomer Name\tBalance");
printf("\n=====================================");
printf("\n%d\t\t%s\t\t%.2f",b1.acno,b1.custname,b1.bal);
printf("\n=====================================");
}

main()
{
struct bank b1;
clrscr();

printf("\nEnter Account No =>");
scanf("%d",&b1.acno);

fflush(stdin);

printf("\nEnter Customer Name =>");
gets(b1.custname);

printf("\nEnter Balance =>");
scanf("%f",&b1.bal);

printData(b1);

getch();
}
```

Output#

```
Enter Account No =>101

Enter Customer Name =>Bhavin

Enter Balance =>12000

Account No      Customer Name   Balance
========================================
101           Bhavin         12000.00
========================================
```

Example# Array of structure variable as a function arguments

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

struct stu
{
int sno;
char sname[20];
int eng,hindi;
};

void passfail(struct stu s[])
{
int i;
printf("\nNo\tName\tEng\tHindi\tTotal\tResult");
printf("\n=====================================================");
for(i=0;i<3;i++)
{
printf("\n%d\t%s\t%d\t%d\t%d",s[i].sno,s[i].sname,s[i].eng,s[i].hindi,s[i].eng+s[i].hin
di);
        if(s[i].eng+s[i].hindi>50)
        {
         printf("\tPass");
        }
        else
        {
         printf("\tFail");
        }
}
printf("\n=====================================================\n");
}

main()
{
struct stu s[3];
clrscr();
s[0].sno=11;
strcpy(s[0].sname,"Jayul");
s[0].eng=32;
s[0].hindi=43;

s[1].sno=12;
strcpy(s[1].sname,"Arth");
s[1].eng=12;
s[1].hindi=13;

s[2].sno=11;
strcpy(s[2].sname,"Pooja");
s[2].eng=22;
s[2].hindi=33;

passfail(s);

getch();
}
```

Output#

```
No     Name   Eng    Hindi  Total  Result
==================================================
11     Jayul  32     43     75     Pass
12     Arth   12     13     25     Fail
11     Pooja  22     33     55     Pass
=================================================
```

Example: Billing program , Create a structure which contains bno,item,price,qty print the bills of N customers.

```
#include<conio.h>
#include<stdio.h>
#define N 100

struct bill
{
int bno;
int item[N],price[N],qty[N];
int cnt;
};

void print(struct bill b[],int total)
{
int i,j,pq=0,amt=0;
for(i=0;i<total;i++)
        {
```

```
        amt=0;
        printf("\n\n\n\nBillno -> %d",b[i].bno);
        printf("\n--------------------------");
        printf("\nItem\tQty\tPrice\tQ*P");
        printf("\n--------------------------");
                for(j=0;j<b[i].cnt;j++)
                {
        pq=b[i].qty[j]*b[i].price[j];
printf("\n%d\t%d\t%d\t%d",b[i].item[j],b[i].qty[j],b[i].price[j],pq);
        amt=amt+pq;

                }
                        printf("\n----------------------");
        printf("\nTotal bill is -> %d",amt);
        printf("\n----------------------");

        }
}

main()
{
struct bill b[N];
int total,i,j;
clrscr();

        printf("\n\nEnter total customer ->");
        scanf("%d",&total);

        for(i=0;i<total;i++)
        {
        printf("\n\nEnter bno ->");
        scanf("%d",&b[i].bno);

        printf("\nEnter cnt ->");
        scanf("%d",&b[i].cnt);

                for(j=0;j<b[i].cnt;j++)
                {
                printf("\nEnter Item,Qty,Price ->");
                scanf("%d %d %d",&b[i].item[j],&b[i].qty[j],&b[i].price[j]);
                }
        }

        print(b,total);

getch();
}
```

Output#

```
Enter total customer ->2

Enter bno ->101

Enter cnt ->2

Enter Item,Qty,Price ->1 2 20

Enter Item,Qty,Price ->2 3 15

Enter bno ->102

Enter cnt ->1

Enter Item,Qty,Price ->22 4 10

Billno -> 101
--------------------------
Item    Qty     Price   Q*P
--------------------------
1       2       20      40
2       3       15      45
----------------------
Total bill is -> 85
----------------------

Billno -> 102
--------------------------
Item    Qty     Price   Q*P
--------------------------
22      4       10      40
----------------------
Total bill is -> 40
----------------------
```

Union

Like Structures, union is a user defined data type.

Unions like structure, contain members whose individual data types may differ from one another.

Union allows us to store different data types in the same memory location.

The only differences are in terms of storage.

The members that compose 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.

Difference between structure and union

Structure	Union
Structure allocates the memory equal to the total memory required by the members.	Union allocates the memory equal to the maximum memory required by the member of the union
Each member has their own memory space	One block is used by all the member of the union
Structure cannot have implemented in shared memory.	Union is best in the environment where memory is less, as it shares the memory allocated.
Ambiguity is less in structure.	Ambiguity are more in union
Access more than one data member.	At a time only one data member can be access.

Example#
 	
struct Emp
{
int no;
int age;
};
main()
{
struct Emp e1;
clrscr();
 	
printf("\n\nSizeof(e1) = %d",sizeof(e1));
 	
getch();
}
 	
Output#
 	
Sizeof(e1) = 4 //Different for each variable
	Example#
	
	union Stu
	{
	int no;
	int age;
	};
	main()
	{
	union Stu s1;
	clrscr();
	
	printf("\n\nSizeof(s1) = %d %d",sizeof(s1));
	
	getch();
	}
	
	Output#
	
	Sizeof(s1) = 2 // same memory

Syntax#	Example#

union unionName
{
members….
}
	union student
	{
	int sno;
	char sname[20];
	int eng,hindi;
	}

Example# Union

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

union Stu
{
int no;
int age;
};
main()
{
union Stu s1;
clrscr();

        s1.no=20;
        printf("\n\nNo = %d Age = %d",s1.no,s1.age);

        s1.age=15;
        printf("\n\nNo = %d Age = %d",s1.no,s1.age); // at time only one

getch();
}
```

Output#

```
No = 20 Age = 20 //check the output

No = 15 Age = 15
```