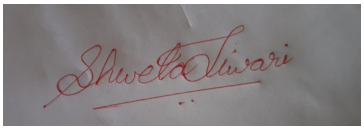
Covered Topics Under UNIT-3 of "PPS-PROGRAMMING FOR PROBLEM SOLVING (BCS101 / BCS201)"

shwetatiwari08@recabn.ac.in
shwetatiwari08aug@gmail.com



Class Notes

Published Date: November, 2022

PPS: UNIT-3

Iteration and Looping

Array

FALL SEMESTER, YEAR (I/II sem, 1st yr)

FALL SESSION (2022-23)
(PPS)
MS. SHWETA TIWARI
Published Date: November, 2022

shwetatiwari08@recabn.ac.in
shwetatiwari08aug@gmail.com

By SHWETA TIWARI
Under On: UNIT-3

PREPARED FOR
Engineering Students
All Engineering College

PREPARED BY
SHWETA TIWARI

Units Classified into Chapters

PPS: UNIT-3:

Chapter-9 STORAGE CLASSES
Chapter-10 STRUCTURE
Chapter-11 UNION

Chapter 9

STORAGE CLASSES

Storage classes tell us about three things.

1. Storage classes tell us about the memory. It means a variable uses which type of memory. It uses RAM or any other memory (CPU Memory).
2. Storage classes tells us about the scope of variable.The scope of a variable is defined as the area of program in which it is accessible.
3. Storage classes tells us about the default value of the variable.

There are four types of storage classes.

1. **Auto**
2. **Static**
3. **External**
4. **Register**

1. Auto: - Auto means automatically. It is the default scope of variables and they are declared with or without auto keyword, belong to auto storage class. Auto variable are active in the block in which they are declared. Block means statements inside the braces ({}).

- ✓ The scope of auto variable is limited. It means they are local variable. They can access within the block where they are declared.
- ✓ The auto storage class variable uses primary memory for storing data. It means RAM.
- ✓ The default value of auto storage class is garbage.

For E.g :-

```
 void main( )
{
auto int a,b;
int c;
clrscr( );
printf("Enter two numbers=");
scanf("%d%d",&a,&b);
c = a + b;
printf("Sum=%d",c);
getch( );
}
```

Note: - As shown above a,b and c all are belong to the auto storage class .

2) **Static variable**: - Static variable are those variable, which start with static keyword. Static variables retain their values through out the execution of the program. Their initial value is zero. Static variable also use primary memory for storing data. Static variables are also active in the block in which they are declared, means the scope of variable is

local. The declaration part of static variable execute only once even if we call function more than once. Static variables are mostly used in functions. As shown in example: -

```
#include <stdio.h>
#include<conio.h>
void show();
void main( )
{
clrscr( );
show( );
show( );
show( );
getch( );
}
void show( )
{
static int a;
printf("\n%d",a);
a=a+1;
}

Output :-    0
             1
             2
```

3) **Extern variable: -** Extern variable are those variables, which are, declared outside of the block/main. This variable can be accessible by each block of the program. They are known as global variables. These variables are declared with extern keyword. Its initial value is zero. They consider as local variable but work as global variables. They also use a primary memory (RAM) for storing a data.E.g :-

```
#include<stdio.h>
#include<conio.h>
void show();
void main( )
{
extern int a;
clrscr( );
printf("\n the value of a=%d",a);
show( );
a = a + 10;
printf("\nNow the value of a=%d",a);
getch( );
}
void show( )
{
extern int a;
```

```
a=a + 5;
printf("\nThe value of a=%d",a);
}

Output :- 0
          5
          15
```

4) **Register variable: -**Register variable are those variable, which start with register keyword. We can use register keyword to increase the speed of program. The Resister variable uses the CPU memory. Their default value is garbage. They are also block level variables, which can access with in the block where they are declared. In case of non-availability of CPU memory, these variables use primary memory (RAM).

E.g :-

```
void main( )
{
register int a;
clrscr( );
printf("\nThe value of a=%d",a);
getch( );
}
```

Difference between auto variable and a static variable

auto variable	static variable
Auto variables are start with or without auto keyword.	.Static variable start with static keyword.
They cannot retain their value throughout the execution of the program.	They retains their value throughout the execution of the program.
Their default value is garbage.	It's default value is zero.
These variable initialize everytime when we call the function.	These variable initialize a single time and retain their value,
Example: auto int a;	Example: static int a;

Important Long Answer Type Questions

1. Explain the scope of variables in C?.
2. Give the difference between local and global variables in C?.
3. Explain the storage classes in C? Discuss their scope and use.
4. Discuss in brief, the difference between the following declarations.
   a. static int i;
   b. auto int i;

Chapter 10

STRUCTURE

Structure is a group of similar and dissimilar data types. Structure is user defined data type. To create a structure, we use struct keyword. Structure starts with a delimiters ({) and close with a delimiter(}). After closing the delimiters, we place a semicolon. To access the elements of the structures, we create a structure variable/ structure field. With the help of structure variable we use the dot operator and a specific element name, which we want to access. As shown in the syntax below:

Syntax

```
struct<structure name>
{
<data type> <variable>;
<data type><variable>;
};
```

Example

```
struct student
{
char name[20];
int roll, marks ;
};
```

In this example student is structure name and in this structure we declare two different data types , i.e. char & int.

Example of structure

```
#include<stdio.h>
#include<conio.h>
struct student
{
char name[20];
int roll,marks;
};
void main()
{
struct student s; //s is structure variable
clrscr();
printf("enter student name");
scanf("%s",&s.name);
printf("enter student roll no");
scanf("%d",&s.roll);
printf("enter student marks");
scanf("%d",&s.marks);
printf("\n your name=%s",s.name);
printf("\n your roll=%d",s.roll);
printf("\n your marks=%d",s.marks);
```

```
getch();
}
```

Initialization of structure

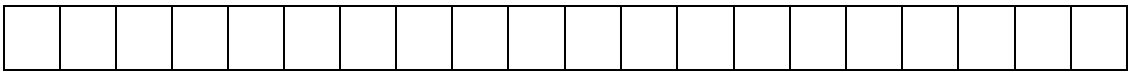
Initialization of structure means to give the initial values to the structure elements in place of giving the values at run time. As shown in the example given below:

```
#include<stdio.h>
#include<conio.h>
struct student
{
char name[20];
int roll,marks;
};
void main()
{
struct student s={"Anju",1,89};          //initialization of structure
variable s
clrscr();
printf("\n your name=%s",s.name);
printf("\n your roll=%d",s.roll);
printf("\n your marks=%d",s.marks);
getch();
}
```

Memory representation in structure: - We can represent the memory allocation of structure as given below. In it we can try to show the memory presentation of structure.

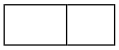
```
struct student
{
char name[20];
int roll, marks;
};
```

name



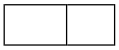
0 1 2 3 4 ---------- 16 17 18 19

roll



0 1

marks



0 1

As shown above it allocates 24 bytes memory for structure variables, 20 bytes memory for name, 2 bytes for roll number and 2 bytes for marks. Total memory is 24 bytes.

Structure within array:- We can also declare an array variable inside the structure. Structure and array both represents the group of elements. As shown in the example

```
struct emp
{
char  name[30];
char  address[80];                    //array
};
```

Dot Operator:- In structure we can use dot operator to access the elements of the structure. We can use structure variable with dot operator and then given the element name, which we want to access. As shown in the example

```
struct student s;
s.name="MONCY"
s.marks=89;
```

Here s is the structure variable and name,marks are the elements of structure which we are accessing using dot operator.

Structure Array

We can also create an array of the structure for storing the details of more than one student.

```
struct student  s[5];
```

Here s is the name of the structure array.The elements of this array are s[0],s[1],s[2],s[3],s[4].

Write a program to enter the details of five students: their name, roll no and marks using structure.

```
# include<stdio.h>
# include<conio.h>
struct student
{
char name [20];
int roll,marks;
};
void main( )
{
int i;
struct student s [5];
clrscr( );
for(i=0;i<5;i++)
{
printf("\n enter your name");
scanf("%s",& s[i].name);
printf("please enter roll no");
```

```
scanf(“%d”,& s[i].roll);
printf(“please enter your marks”);
scanf(“%d”,& s[i].marks);
}
for(i=0;i<5;i++)
{
printf(“\n your name=%s”,s[i].name);
printf(“\n your roll no=%d”,s[i].roll);
printf(“\nyour marks=%d”,s[i].marks);
}
getch( );
}
```

Nested structure

Nested structure means a structure inside a structure.It is possible to have one structure as a member of another structure.In nested structure the first structure is called the outer structure and second structure which inside the structure is known as inner structure.E.g :-

```
 struct emp
{
            char name[20];
            struct dep
{
int salary ;
}d;
};
void main( )
{
struct emp e;
clrscr( );
printf(“Enter Employ name=”);
sacnf(“%s”,& e.name);
printf(“Enter your salary=”);
sacnf(“%d”,& e.d.salary);
printf(“\n Your name=%s”,e.name);
printf(“\n Your salary=%d”.e.d.salary);
getch();
}
```

As shown in the above example the emp is our outer structure and dep is inner structure.

Passing structure as argument into function: - We can also pass structure as argument into the function. Same as variable and array. As shown in the example.In this program,we write a function show().It accepts a structure as its argument.

```
struct student
{
char name[30];
int roll, marks;
};
void show(struct student s);
void show(struct student s);
{
print("\n Student name=%s",s.name);
print("\n Student roll=%d",s.roll);
print("\n Student marks=%d",s.marks);
}
void main()
{
struct student s={"Harparteek",1,89};
clrscr();
show(s);
getch();
}
```

In the above example first we declare the structure and then declare the function with structure argument. Then we define th body of the function and in last call the function with main function.

To check the size of the structure

To check the size of the structure, we can use sizeof() operator. It tells us how much memory is occupied by the structure elements.

As shown in the example below: -

```
#include<stdio.h>
#include<conio.h>
struct emp
{
char name[15];
int age;
float salary;
};
void main()
{
struct emp e;
clrscr();
printf("the size of the structure=%d",sizeof(e));
getch();
}
```

Difference between Structure And Array

Structure	Array
Structure is a group of similar and dissimilar elements.	Array is a group of elements of similar types.
Structure starts with struct keyword.	It does not start with any keyword but is uses subscript for creating an array.
We can use dot operator to accesses the elements of the structure.	We can use index of the arrayto access the element of the space array.
Structure has declaration and definition.	It has only declaration.

5. Example
struct student
{
char name[20];
int roll;
};
	5. Example
	int a[10];

Important Short Answer type Questions

1. What are the advantages of structures?
2. What is structure?
3. Write a note on nested structure.
4. Explain self-referential structure.
5. Difference between structure and union.

Important Long Answer Type Questions

1. Explain structure in C. How can you differentiate it from an array?
2. How structures are defined and initialized? Explain with example.
3. Explain by giving example how the members of structure are accessed in C.
4. What do you mean by structure within a structure? Give Example.
5. Explain how structure can be passed to a function as argument with example.
6. Write a program in C to define a structure student with data members like rollno, class and marks of 5 students. Calculate total marks and show the students detail.
7. Write a program to create a structure student having data members as student name, rollno and pass it to a function as an argument.

Chapter 11

UNION

Union is a group of similar and dissimilar data types or elements. It is a user defined data type. Union starts with 'union' keyword and terminate with semicolon same as structure. Union allocates a single memory to its elements. All the elements of the union share the single memory. To access the elements of union we can create a union variable. It is same as structure.

The major difference between the union and structure is that in structure different members use different memory locations whereas in union all members of union share the same memory location.

Syntax: -

```
union < union name >
{
<data type> <variable name> ;
<data type> <variable name> ;
};
```

E.g: -

```
union student
{
char name [20];
int rollno, marks;
};
```

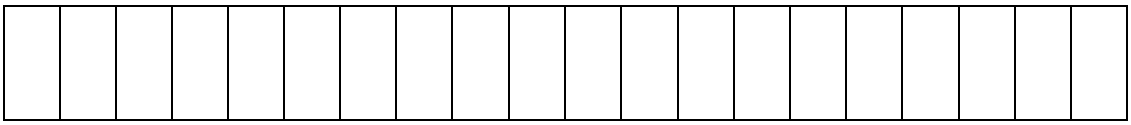
Union memory representation:- As we know that union allocate largest memory of its element and it allocate a single memory to all it elements.

As shown in the figure and Example

```
union student
{
char name[20];
int roll, marks;
};
```

Memory representation

name



0 1 2 3 4 5 6 ---------- 17 18 19

roll

marks

As shown in the example and figure the union allocate 20 bytes memory for name, roll and marks, which is the largest or higher memory in the union.

WAP to create union, which can store student detail like students name, rollno, and marks.

```
union student
{
 char name[20];
int rollno,marks ;
};
void main( )
{
union student s;
clrscr( );
printf(“Enter Employ name=”);
sacnf(“%s”,& s.name);
printf(“\n Your name=%s”,s.name);
printf(“Enter your Rollno.=”);
sacnf(“%d”,& s.rollno);
printf(“\n Your Rollno.is=%d”,s.rollno);
printf(“Enter your Marks=”);
sacnf(“%d”,& s.marks);
printf(“\n Your Marks is=%d”,s.marks);
getch();
}
```

Initialization of union

We cannot initialize the union same as structure because it uses the single memory location and all the elements use the same memory.

As shown in the example below:

```
#include<stdio.h>
#include<conio.h>
union student
{
char name;
int roll,marks;
};
void main()
{
union student s;
clrscr();
s.name='h';                    //initialization of union variable s
printf("\n your name=%c",s.name);
s.roll=1;
printf("\n your roll=%d",s.roll);
s.marks=99;
printf("\n your marks=%d",s.marks);
getch();
 }
```

Union Array

We can also create the array of the union same as structure for storing the details of more than one student.

Write a program to enter the detail of five students: their name, rollno and marks using union.

```
# include<stdio.h>
# include<conio.h>
union student
{
char name [20];
int roll,marks;
};
void main( )
{
int i;
union student s[5];
clrscr( );
for(i=0;i<5;i++)
printf("\n enter your name");
scanf("%s",& s[i].name);
printf("\n your name=%s",s[i].name);
printf("please enter roll no");
scanf("%d",& s[i].roll);
printf("\n your roll no=%d",s[i].roll);
printf("please enter your marks");
scanf("%d",& s[i].marks);
printf("\n   your marks=%d",s[i].marks);
}
getch( );
}
```

To check the size of the union

To check the size of the union we can use sizeof() operator. It tells us how much memory is occupied by the union elements.

As shown in the example below:-

```
#include<stdio.h>
#include<conio.h>
union emp
{
char name[15];
int age;
float salary;
};
void main()
{
```

```
union emp e;
clrscr();
printf("the size of the union=%d",sizeof(e));
getch();
}
```

Difference between structure and union

Structure	Union
Structure start with **'struct'** keyword	Union start with **'union'** keyword
All the elements of the structure use different memory locations for storing the data.	All the elements use the single memory location for storing the data.
It allocates different memory to each element.	It allocates single memory. All the elements share the same memory.
It is commonly used in most of the applications.	It is not commonly used.

Syntax
struct <struct name>
{
<data type> <variable>;
};
	Syntax
	union <union name>
	{
	<data type> <variable>;
	};

Important Short Answer Type Questions

1. State limitations of union.
2. Write advantages of union.
3. How we can check the size of union?

Important Long Answer Type Questions

1. What is union? How can you differentiate it from a structure?
2. How union is defined and initialized? Explain with example.
3. Explain by giving example how the members of union are accessed in C.
4. Write a program to define a union student with data members, rollno as integer, stu_name as string and average marks as float. Display the information of student and calculate the size of union.
5. What are the advantages of structure type over union type?