B.C.A. (Sem – II)

B.C.A. - 201

Advanced Programming Language 'C'

Purushottam Singh

Unit:- 1

unit:-2

Date 9/12/
Page

user defined function

* Function :-

Independent coded programs are called subprograms that are much easier to understand debug and testing.

In c such a subprograms are refer to as function.

Functions can be classified into two categories.

1) library function
2) user defined function

Printf and scanf functions belong to the library functions.

Main() is example of user defined function.

The main difference between library function and user defined function is that library functions are not required to be written by us.
Where ~~were~~ as user defined function has to be developed by the user at the time of writing a program.

Date __/__/__
Page _____

* Need for user defined function or Advantages of user decined function

-> The Program may become to lurge and complex and as a result the tesk for debugging testing and maintaining becomes difficult If program is devided in to functional parts then each part may be independently coded and leter combine in to a single unit.

-> It facilities top-down modulur Programming in this programming style the high level of over all problem is solved first while the details of each lower level function are addressed luter.

length & size -> The length of a source Program can be reduced by using the function at a cppropriate places.

multiple time use function -> It is easy to locate and isolate a function for investigation.

=) elements of user defined function

Three elements of user defined function

1) Function definition
2) Function call
3) Function declaration

1) Function definition :-

Function definition is an independent Program module that is speciall written to implement the requirement of the function

→ A function definition is also known as function implementation that includes following elements.

1) Function name
2) Function Type
3) List of Parameters

} function header

4) local variable declaration
5) function statement
6) A return statement

} function body

Syntax Function definition

```
function_type function Name (Parameter list)
{
    local variable declaration;
    executable statement 1;
    executable statement 2;
    .....
    Return statement;
}
```

Date __/__/__
Page _____

-) Function type Function name (argument list) is known as the function header and the statements within the opening and closing braces. {} is known as function body which is a compound statement.

-) Function header :-
The function header consists of three parts.

-) The function Type (It is also known as return type)
-) Function name and the
-) Formal parameter list

NOTE :- semicolon is not used at the end of function header.

=) Function name & Type :-

Function Type specifies the Types of value like integen, float, double. if the return type is not specified C will be assume the integer.

If the function is not returning anything then we need to specify return type as void.

The function name is any valid C identifier the name should be appropriate to the task performed by the function.

Date __/__/____
Page _____

* Formal parameter list :-

The parameter list declares the variables that will receive the data sent by calling program.

ex.

```
int sum(int a, int b)
Float multiPlication(Float x, Float y)
```

* Function body :-

Function body contains 3 parts

1) local variable declaration that specify the variables needed by the function.
2) function statement that perform the task to the function.
3) Return statement that returns the value evaluated by the function.

ex.

(1)
```
int sum(int a, int b)
{
    int c;
    c= a+b;
    return (c);
}
```

(2)
```
void sum(int a, int b)
{
    PrintF("sum = %d", a+b);
    Return;
}
```

2) void main()
{

```
struct book b1;
Printf("Enter value of books title=");
Scanf("%s", b1.title);
Printf("Enter name of book author=");
Scanf("%s", b1.author);
Printf("Enter the pages of book=");
Scanf("%d", &b1.pages);
Printf("Enter the price of book=");
Scanf("%d", &b1.price);
Printf("Book title = %s", b1.title);
Printf("Authorname=%s", b1.author);
Printf("Book Page=%d", b1.Page);
Printf("Book Price: %f", b1.price);
getch();
```

}

Date __/__/__
Page _____

Return value and their types

A function may or may not send back any value to the calling function.

If it does it is done to the return statement while it is posible to pass to the called function any number of values.

syntax of return

return (expression);

Ex → return (c);

2) function call :-

A function can be called by simply using the function name followed by a list of parameter or arguments.

Ex:-

```
main()
{
    int c;
    c = sum(15, 10);
    printf("%d", c);
}
int sum(int a, int b)
{
    int c;
    c = a+b;
    return (c);
}
```

Date __/__/__
Page _____

-> Here when the compiler encounter function call the control is transfer to the function sum.

-> The function is executed line by line as described and value is return when return statements is executed.

3) function declaration :-

function declaration is also known as function prototype.

It consists 4 parts.

1) Function type (Return type)
2) Function name
3) Parameter list
4) Terminating semicolon

Syntax

Function_type function_name (Parameter list)

Ex

int sum (int a, int b);

Date __/__/__
Page _____

* Definition of calling function :-

Function defined main Program it is known as called calling function. it is user defined function which is defined outside the main Program it is called called function.

=) Actual Argument :-

A variable which are declair in calling function is known as actual argument.

=) Formal Argument :-

A variable which are declaired in called function or user define function is known as formal argument.

* Category of User defined Function

There are 5 categories of udf.

1) Function with no arguments and no return values.
2) Function with Arguments and no return value.
3) Function with Arguments and one return values.

4) Function with no Arguments but Return value.

5) Function that return multiple values.

1) Function with no Arguments no return value :-

when A Function has no Argument it does not receive any data From the calling function.

Ex

```
#include<stelio.h>
#include<conio.h>
void sum(void);   // Function declaration
void main()
{
        sum();    // calling Function
      getch();
}
void sum(void)    // called Function
{
        int a=10, b=5, c=0;
        c=a+b;
        Printf("%d", c);
}
```

2) Function with Arguments and no return value :-

The calling function can send the data to the called function but it can not return any value to the calling function as it has no return statement.

Ex

```
#include<stdio.h>
#include<conio.h>
void sum(int a, int b);
void main()
{
    int a=10, b=20;
    sum(a,b);
    getch();
}
void sum(int a, int b)
{
    int c=0;
    c=a+b;
    printf("%d",c);
}
```

3) Function with Argument and one return value.

```
#include<stdio.h>
#include<conio.h>
int sum(int a, int b);
void main()
```

```
{
        sum(a,b);
        Printf("%d",c);
        getch();
}
int sum(int a, int b)
{
        a=10,b=5,c=;
        c=a+b;
        Return(c);
}
```

4) Function with no Argument with return value.

```
#include<stdio.h>
#include<conio.h>
int sum(void)
void main();
{
        sum();
        Printf("%d",c);
        getch();
}
int sum(void)
{
        int a=10,b=5,c=;
        c=a+b;
        Return(c);
}
```

① function with no arguments and no ~~result~~ return value:-

when the function doesn't have any arguments and doesn't have any return then it will call function with no argument and no return value.

Ex. Void add();
main()
{
add();
}
Void add()
{
int a, b;
Scanf("%d%d", &a, &b);
printf("add = %d", a+b);
}

```
int a=10, b=5, c;
printf("%d", c);
c = a+b;
```

In this kind of user define function the main() does not take value and does not passes the value to the UDF as argument and the call function does not return any value to main() thats called function with no argument and no return value.

Date ___/___/___
Page _____

2) Function with Arguments and no return value :-

The calling function can send the data to the called function but it can not return any value to the calling function as it has no return statement.

Ex

```
#include<stdio.h>
#include<conio.h>
void sum(int a, int b);
void main()
{
    int a=10, b=20;
    sum(a,b);
    getch();
}
void sum(int a, int b)
{
    int c=0;
    c=a+b;
    printf("%d",c);
}
```

③ function with argument and return value

Ex.

```
int add (int, int);
main()
{
    int a, b, c;
    scanf ("%d%d", &a, &b);
    c = add (a, b);
    printf ("%d", c);
}
int add (int a, int b)
{
    int n;
    n = a + b;
    return (n);
}
```

```
add (a, b);
printf ("%d", c);
a=10, b=5, c=0;
c = a+b;
return (c);
```

when the function does have any argument and does have any return value it's called function with argument and return value.

The main() take value and passes the value to the UDF as argument and the called function does return any value to the main().

Date ___/___/___
Page _____

* Nesting of function :-

Nesting of function means main function can call function 1 which calls function 2 which calls function 3 and so on....

There is in principle no limit as to how deeply functions can be nested.

Ex =

```
float ratio(int x, int y, int z);
int difference(int x, int y);
void main()
{
        int a, b, c;
        scanf("%d %d %d", &a, &b, &c);
        printf("%f", ratio(a, b, c));
}
float ratio(int x, int y, int z)
{
        if(difference(y, z))
        {
                return(x/(y-z));
        }
        else
        {
                return(0.0);
        }
}
int difference(int p, int q)
{
        int(p != q)
```

Date ____/____/____
Page _____

```
return(1);
else
    return(0);
}
```

* Passing the array to the function Or

How can we pass array as argument function.

- like the value of simple variable it is also possible to pass the values of an array to a function.

Date __/__/__
Page _____

- Previously we pass the variable in function similary we can also pass the whole array as a actual perameter through the function But the formal perameter should be declair as array of same datatype.

=)
```
#include<stdio.h>
#include<conio.h>
float largest(float a[], int n);
void main()
{
        float value[4] = {2.5, 4.5, 7.5, 8.5};
        printf("%f", largest(value, 4));
}
float largest(float a[], int n)
{
        int i;
        float max;
        for(i=1; i<n; i++)
        {
                if(max < a[i])
                {
                        max = a[i];
                }
                return(max);
        }
}
```

```
    printf("enter value of *b = ");
    scanf("%d", &*b);
    swap(&*a, &*b);
    getch();
}
void swap(int *a, int *b)
{
    int temp;
    temp = *a;
    *a = *b;
    *b = temp;
    printf("value of *a & *b = %d %d", *a, *b);
}
```

* The scope visibility and lifetime of variable.

In c not only do all have datatype but they also have a storage classes.

There are 4 types of storage classes.

1) Automic variable
2) external variable
3) static variable
4) Register variable

1) Automic variable :-

Automic variable is also known as local or internal variable that is called declared inside the funct They are created when the function is created and the functio is exists.

2) External variable :-

external variable is also known as global variable that is declar in inside and outside of the function this variable are both alive and activ through out the entire program.

3) Static variable :-

The value of static variable persist until the end of program. A varic can be declared static using the key.w static.

Ex static int x;

A static variable may be eith an internal or an external type depending on the place of declara

4) Register variable :-

We can tell the compiler that a variable should be kept in on

of the machine Register instead of keeping in memory.

since Register class is much faster than memory access keeping the frequently access variable Register will lead to faster execution.