UNIT II - Managing simple Input and Output operations - Operators and Expressions - Decision Making: Branching statements, looping statements - Function: Declaration, Definition - Passing arguments by value - Recursion - Storage classes.

2.5 RECURSION

Recursion is defined as the function that calls itself repeatedly until condition is reached. But while using **recursion**, programmers need to be careful to define an exit condition from the function; otherwise it will go into an infinite loop.

Syntax:

```
Function1()
{
Function1();
}
```

Example:

Calculating the factorial of a number

Fact (n)= n*fact(n-1);

6! = 6*fact(n);

6! = 6 *5*fact(4)

6! = 6 * 5 * 4 *fact(3)

6! = 6 * 5 * 4 * 3 *fact(2)

6! =6 *5 * 4 * 3 *2 * fact(1)

6! = 6 *5 * 4 * 3 *2 * 1

6!=120

Advantage of recursion

- Recursion makes program elegant and cleaner.
- All algorithms can be defined recursively which makes it easier to visualize and prove.
- Reduce unnecessary calling of function
- Easy to solve complex problems

Direct Recursion:

A function is directly recursive if it calls itself.

```
A( )
{
```

….
A( ); // call to itself
….
}

Indirect Recursion:
Function calls another function, which in turn calls the original function.

```
A( )
{
…
B( );
…
}
B( )
{
…
A( );// function B calls A
…
}
```

Linear Recursion - It makes only one recursive call.

Binary Recursion - It calls itself twice.

N-ary recursion - It calls itself n times.

Program 1 : Find factorial using recursion

```
#include<stdio.h>
#include<conio.h>
int fact(int);
void main()
{
int n, Result;
printf("\n Enter any number:");
scanf("%d", &n);
Result = fact(n);
printf ("Factorial value = %d", Result);
getch();
}
int fact (int x)
{
```

```
if (x == 0)
return 1;
else
return x * fact(x – 1);
}
```

Output:

Enter any number: 4

Factorial value = 24

Program 2 : Generate the Fibonacci Series Using Recursive Function

```
#include<stdio.h>
#include<conio.h>
int fib(int val);
void main ()
{
int i, n;
printf(“Enter the number”);
scanf (“%d”, &n);
printf(“\n Fibonacci sequence:”);
for (i = 0 , i < n, i++)
printf(“%d ”, fib(i));
getch();
}
int fib (int n)
{
if (n== 0)
{
return 0;
}
else if (n == 1)
{
return 1;
}
else
```

return fib(n – 1) + fib (n– 2);

}

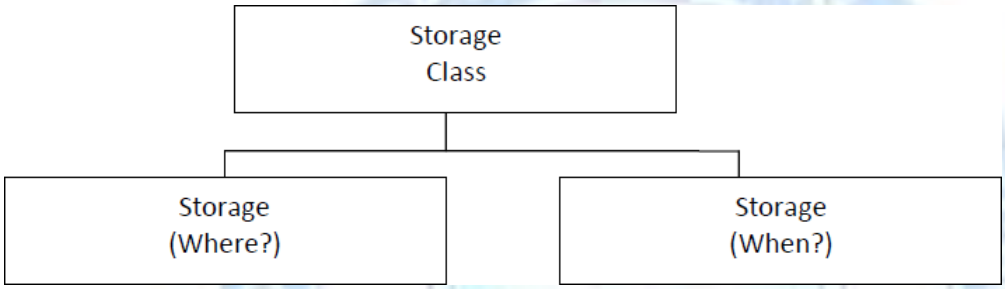
Output

Enter the number: 6

Fibonacci sequence : 0 1 1 2 3 5

2.1 STORAGE CLASSES

A storage class defines the scope and life-time of variables and functions within a C Program. It determines the part of memory.



In C, There are 4 storage classes. They are,

1. Automatic Storage class
2. External Storage class
3. Static Storage class
4. Register Storage class

(i) Automatic Storage class: auto

The auto storage class is the default storage class for all local variables. It is the temporary memory space.

Scope: Variable defined with auto storage class is local to the function in which they are defined.

Default Initial Value: Any random value i.e garbage value.

Lifetime: Till the end of the function/method block where the variable is defined.

Syntax:

auto datatype var1,var2,…..,varn;

Example

#include<stdio.h>

#include<conio.h>

```
void main()
{
int a;
// or
auto int a; //Both are same
….
}
```

(ii) External or Global Storage class: extern

It is the external storage class for all global variables. It is declared out of the main function.

Scope: Global i.e everywhere in the program.

Default initial value: 0(zero).

Lifetime: Till the program doesn't finish its execution, we can access global variables.

Syntax:

```
extern datatype var1,var2,…..,varn;
```

Example

```
#include<stdio.h>
#include<conio.h>
extern int a; // global variable
void main()
{
…..
}
```

(iii) Static Storage class : static

The static storage class is the default storage class for all global variables. It is the permanent memory space. It is declared out of the main function.

Scope: Local to the block in which the variable is defined

Default initial value: 0(Zero).

Lifetime: Till the whole program doesn't finish its execution.

Syntax:

```
static datatype var1,var2,…..,varn;
```

Example

```
#include<stdio.h>
#include<conio.h>
static int a; // both are same
//or
int a;
void main()
{
…..
}
```

(iv) Register Storage class: register

It is the special storage area within the computer‟s central processing unit.

Scope: Local to the function in which it is declared.

Default initial value: Any random value i.e garbage value

Lifetime: Till the end of function/method block, in which the variable is defined.

Syntax:

```
register datatype var1,var2,…..,varn;
```

Example

```
#include<stdio.h>
#include<conio.h>
void main()
{
register int a;
register int b;
…..
}
```