Introduction to C

C Language

- The C programming language is a general-purpose, high-level language that was originally developed by Dennis M. Ritchie to develop the UNIX operating system at Bell Labs.
- C was originally first implemented on the DEC PDP-11 computer in 1972.
- In 1978, Brian Kernighan and Dennis Ritchie produced the first publicly available description of C, now known as the K&R standard.
- The UNIX operating system, the C compiler, and essentially all UNIX applications programs have been written in C.

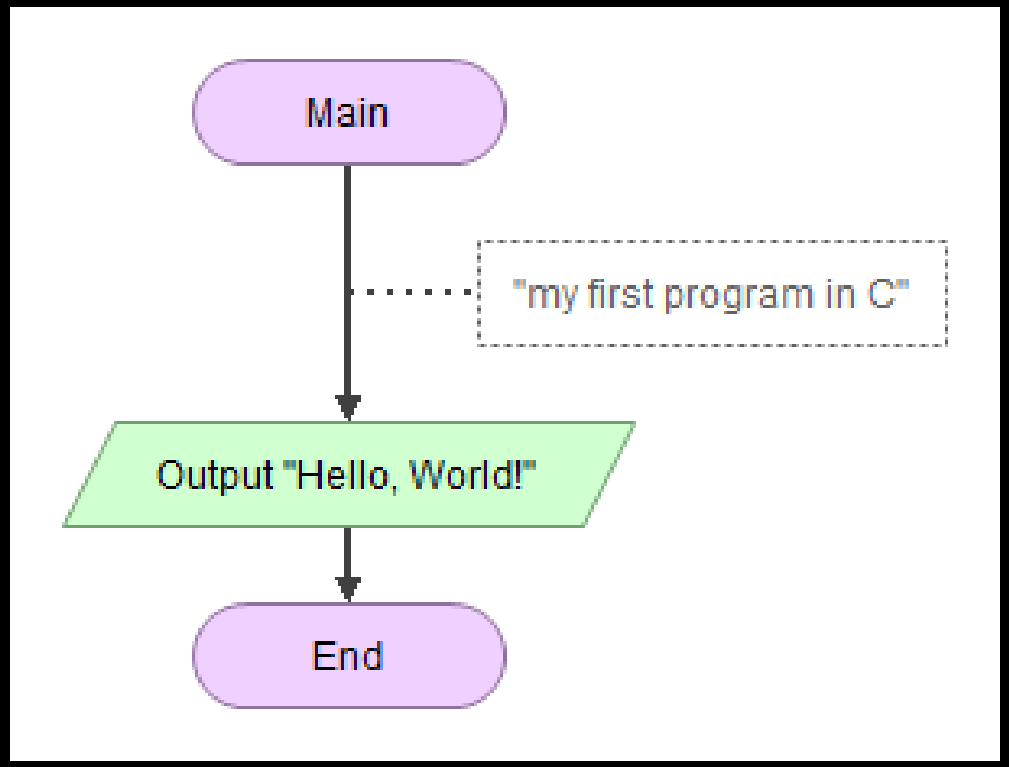
Facts about C

- C was invented to write an operating system called UNIX.
- C is a successor of B language, which was introduced around 1970.
- The language was formalized in 1988 by the American National Standard Institute. (ANSI).
- The UNIX OS was totally written in C by 1973.
- Today, C is the most widely used and popular System Programming Language.
- Most of the state-of-the-art software have been implemented using C.
- Today's most][popular Linux OS and RBDMS MySQL have been written in C.

Minimum C Program Structure

1. Pre-processor Commands
2. Functions
3. Variables
4. Statements & Expressions
5. Comments

Flowgorithm



C Program

```c
#include <stdio.h>

int main()
{
    /* my first program in C */
    printf("Hello, World! \n");

    return 0;
}
```

Various Parts of the above Program

1. The first line of the program #include is a preprocessor command, which tells a C compiler to include stdio.h file before going to actual compilation.
2. The next line int main() is the main function where program execution begins.
3. The next line /*...*/ will be ignored by the compiler and it has been put to add additional comments in the program. So such lines are called comments in the program.
4. The next line printf(...) is another function available in C which causes the message "Hello, World!" to be displayed on the screen.
5. The next line retur 0; terminates main()function and returns the value 0.

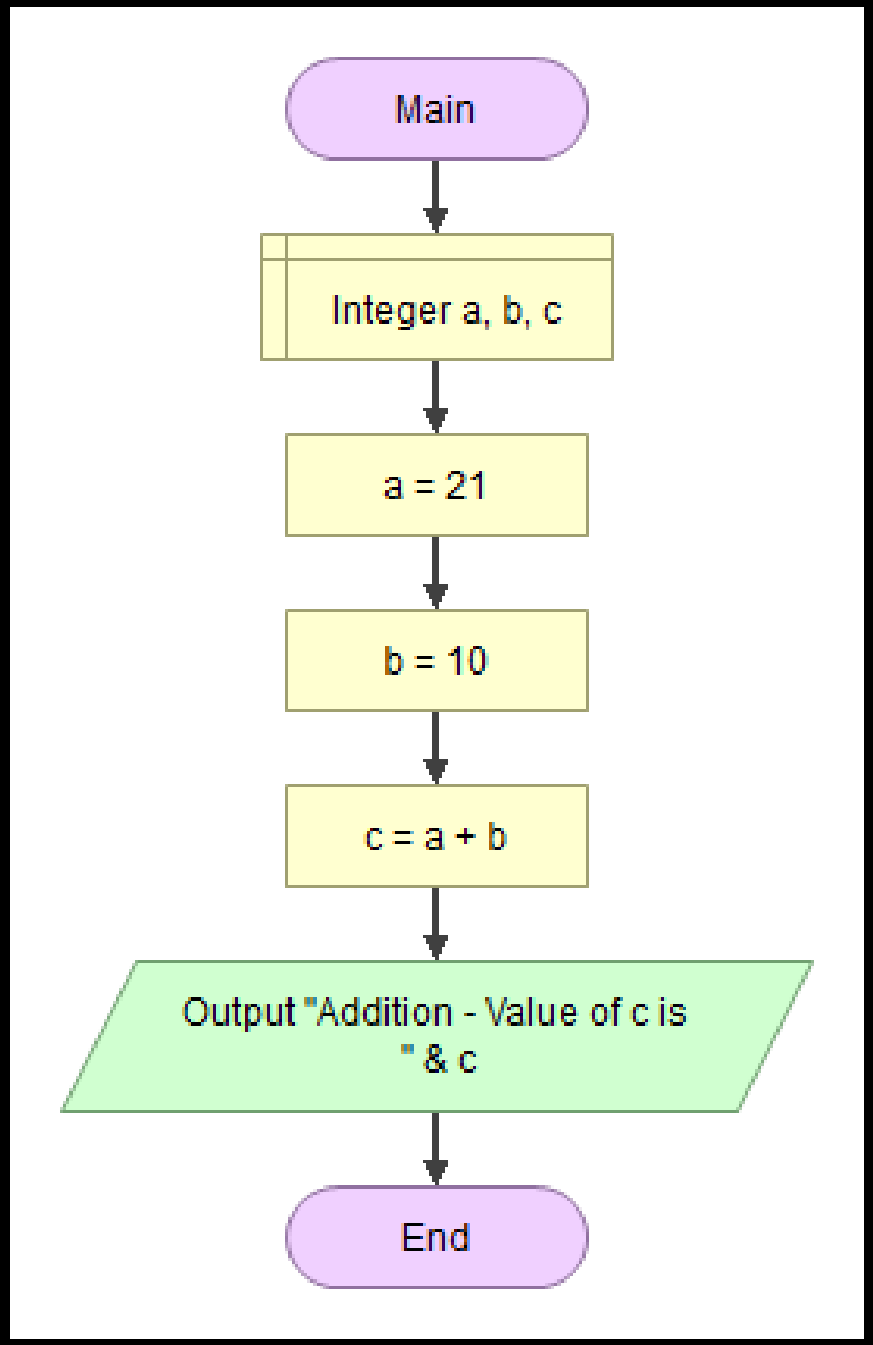
Variables

Type	Description
Char	Typically a single octet(one byte). This is an integer type.
Int	The most natural size of integer for the machine.
Float	A single-precision floating point value.
Double	A double-precision floating point value.
Void	Represents the absence of type.

Sequence Structure

Sequence

Step 1: Start
Step 2: Declare a , b and c
Step 3: Assign 21 to a
Step 4: Assign 10 to b
Step 5: Add a and b, then assign it to c
Step 6: Display c
Step 7: Stop

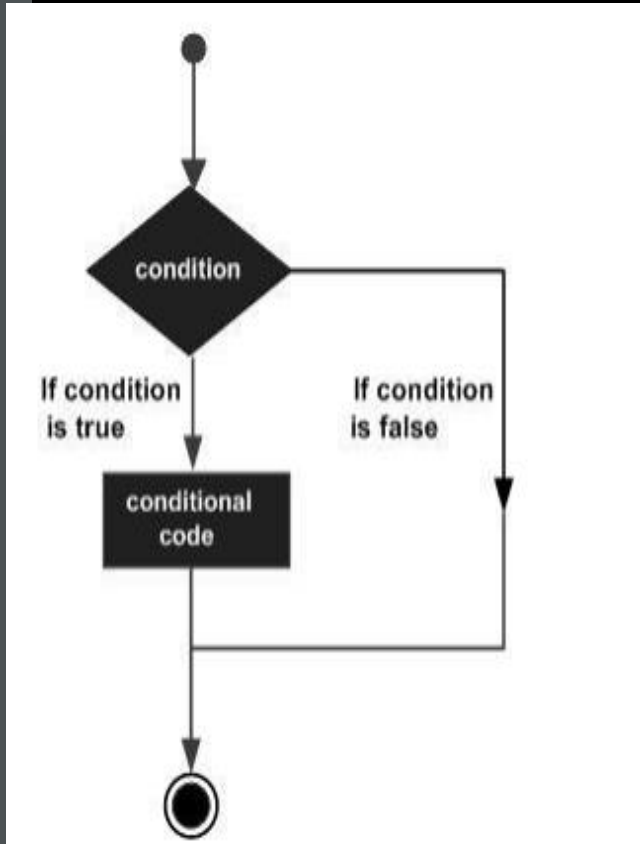



```
#include<stdio.h>
main()
{
        int a = 21;
        int b = 10;
        int c ;
        c = a + b;
        printf("Addition - Value of c is %d\n", c );
        return 0;
}
```

```
#include<stdio.h>
main()
{
        int a = 21, b = 10, c ;
        c = a + b;
        printf("Addition - Value of c is %d\n", c );
        return 0;
}
```

Selection Structure

if



Step 1: Start

Step 2: Declare a

Step 3: Assign 10 to a

Step 4: if a < 20 then

Step 4.1: Display a is less than 20

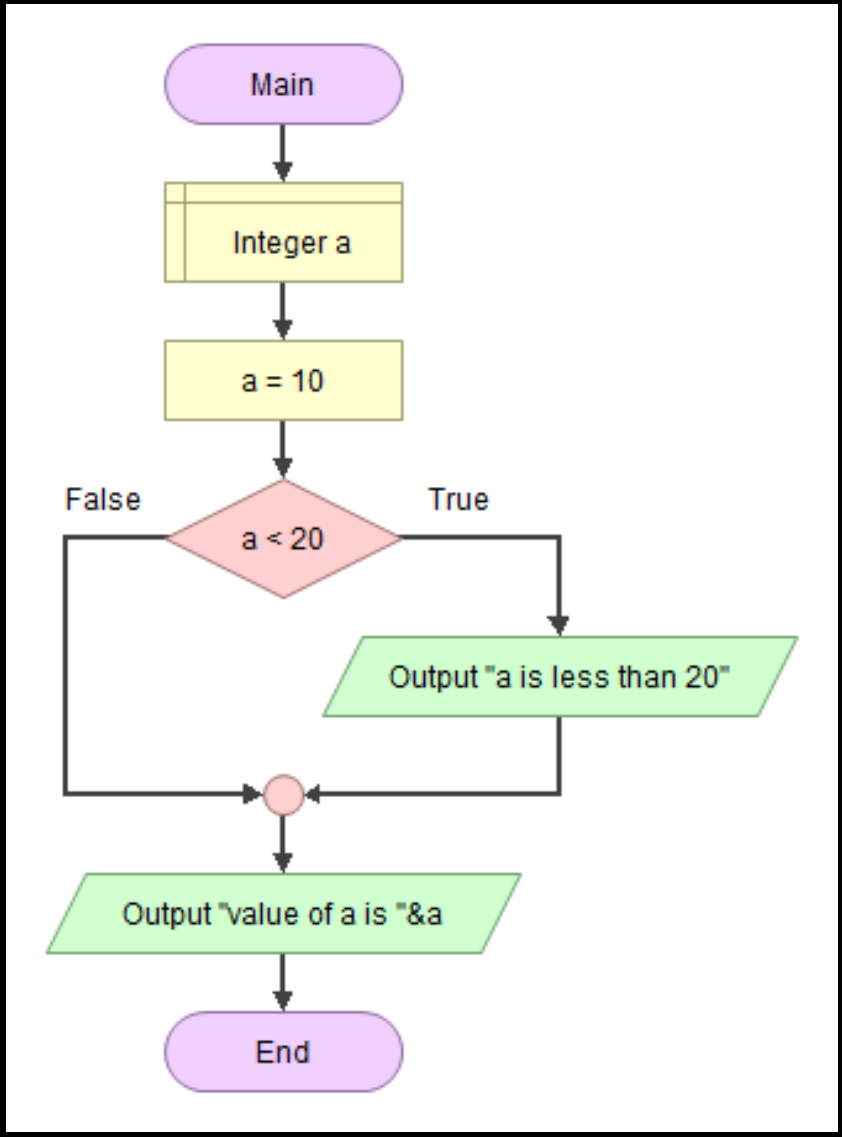
Step 5: End If

Step 6: Display value of a is 10

Step 7: Stop

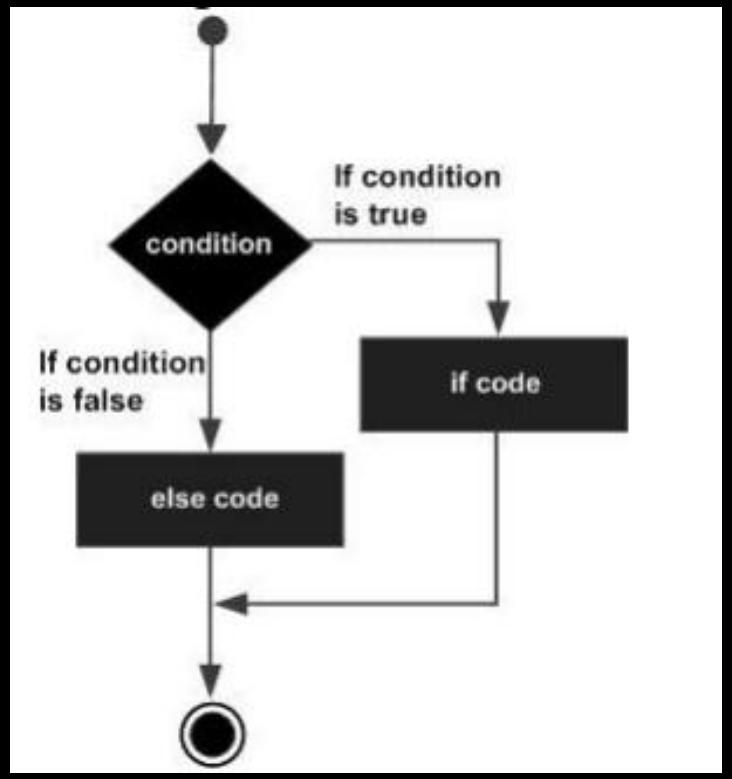
```
if(boolean_expression)
{
   /* statement(s) will execute if the boolean expression is true */
}
```

if




```
#include <stdio.h>
int main ()
{
        int a = 10;
        if( a < 20 )
        {
                printf("a is less than 20\n" );
        }
        printf("value of a is : %d\n", a);
        return 0;
}
```

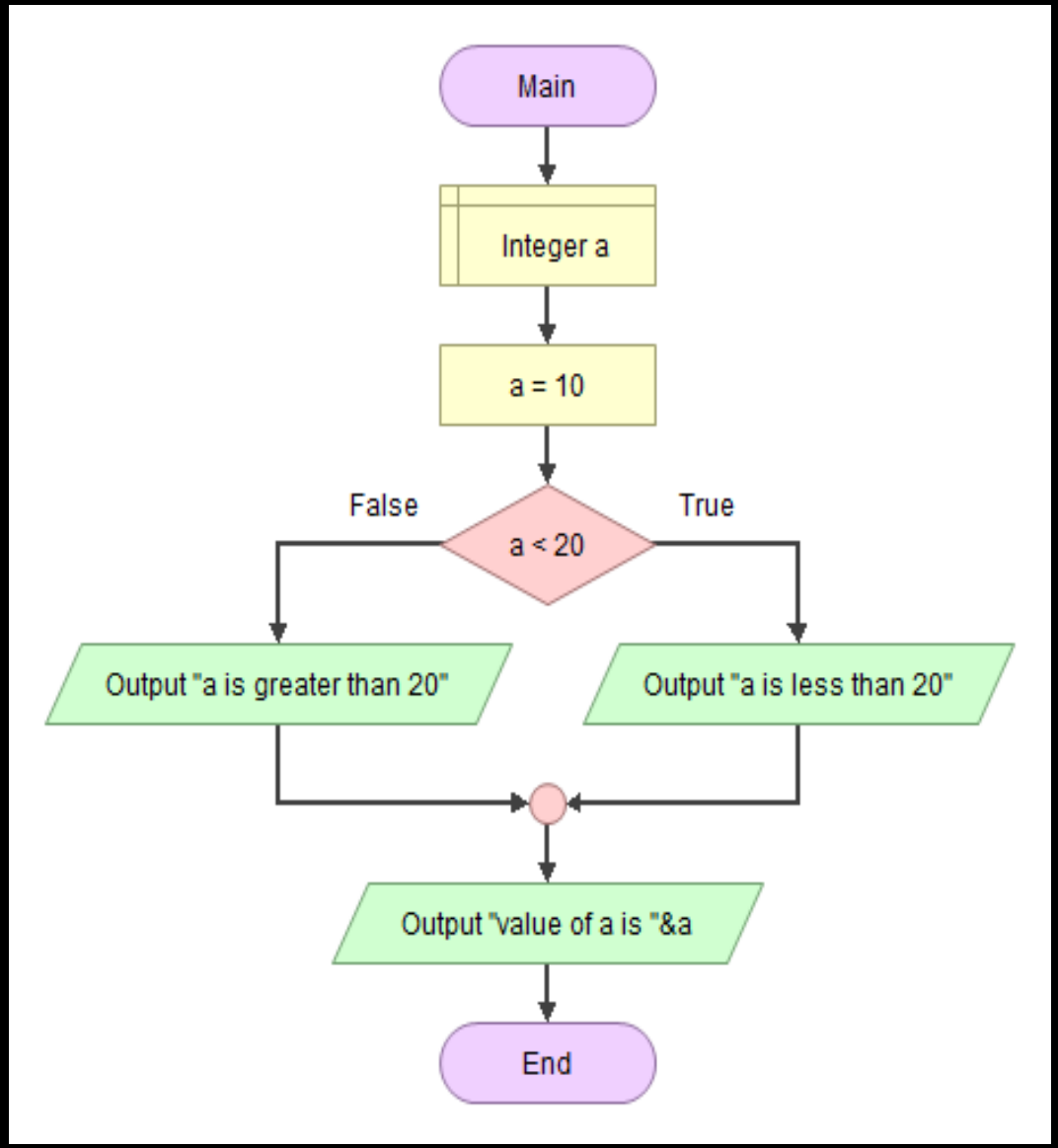
if -else



Step 1: Start
Step 2: Declare a
Step 3: Assign 10 to a
Step 4: if a < 20 then
    Step 4.1: Display a is less than 20
Step 5: Else
    Step 5.1: Display a is greater than 20
Step 6: End If
Step 7: Display value of a is 10
Step 8: Stop

```
if(boolean_expression)
{
   /* statement(s) will execute if the boolean expression is true */
}
else
{
  /* statement(s) will execute if the boolean expression is false */
}
```

if -else




```
#include <stdio.h>
int main ()
{
        int a = 10;
        if( a < 20 )
        {
                printf("a is less than 20\n" );
        }
        else
        {
                printf("a is greater than 20\n" );
        }
        printf("value of a is : %d\n", a);
        return 0;
}
```

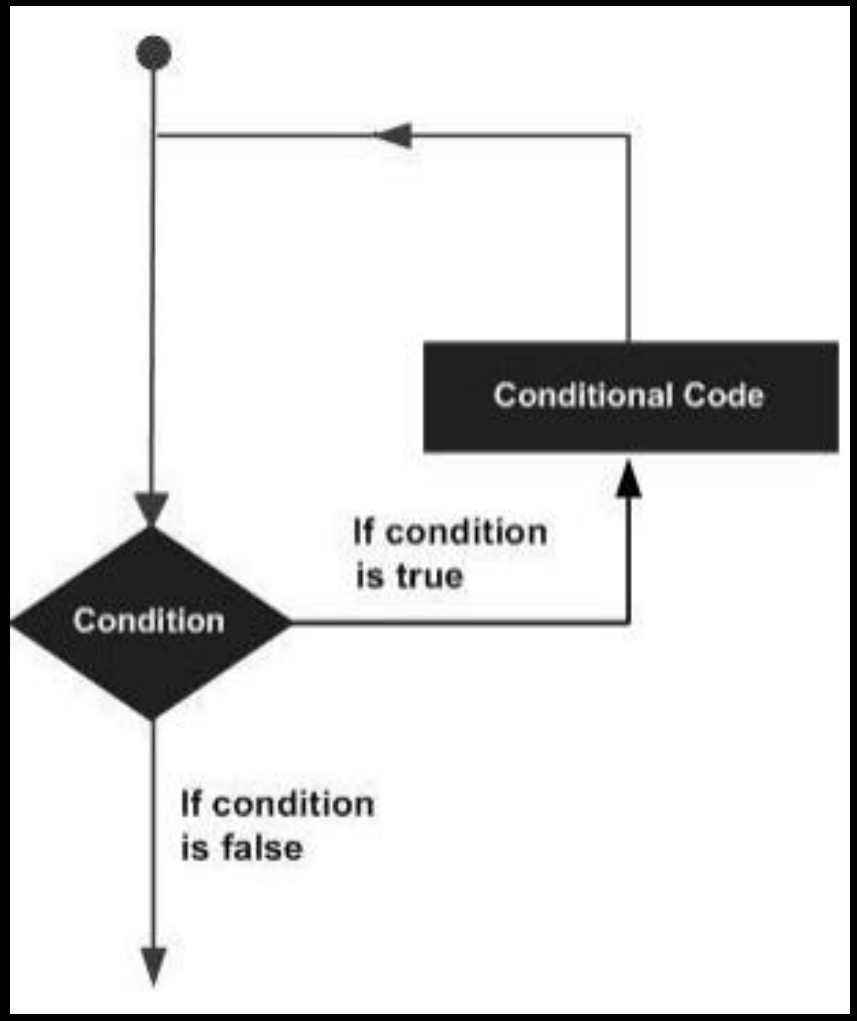
if…else if…else Statement

```
if(boolean_expression 1)
{
   /* Executes when the boolean expression 1 is true */
}
else if( boolean_expression 2)
{
   /* Executes when the boolean expression 2 is true */
}
else if( boolean_expression 3)
{
   /* Executes when the boolean expression 3 is true */
}
else
{
   /* executes when the none of the above condition is true */
}
```

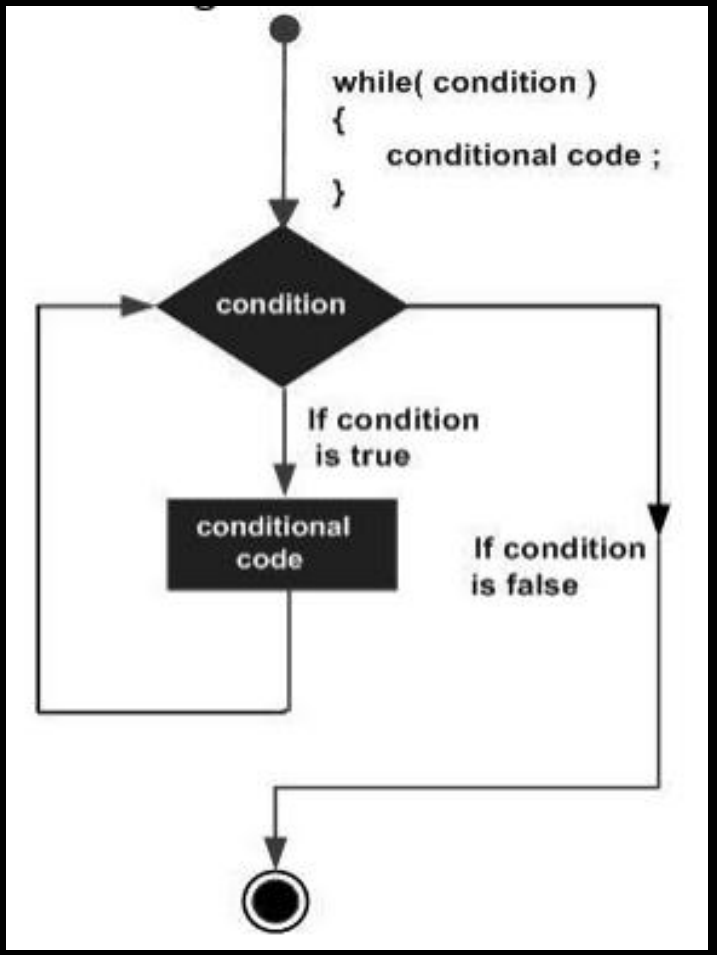
Nested if statements

```
if( boolean_expression 1)
{
   /* Executes when the boolean expression 1 is true */
   if(boolean_expression 2)
   {
      /* Executes when the boolean expression 2 is true */
   }
}
```

Looping Structure



while




```
while(condition)
{
    statement(s);
}
```

Step 1: Start

Step 2: Declare a

Step 3: Assign 10 to a

Step 4: Repeat

Step 4.1: Display value of a

Step 4.2: a= a+1

Step 5: Until (a >= 20)

Step 6: Stop

Step 1: Start

Step 2: Declare a

Step 3: Assign 10 to a

Step 4: while (a <20)

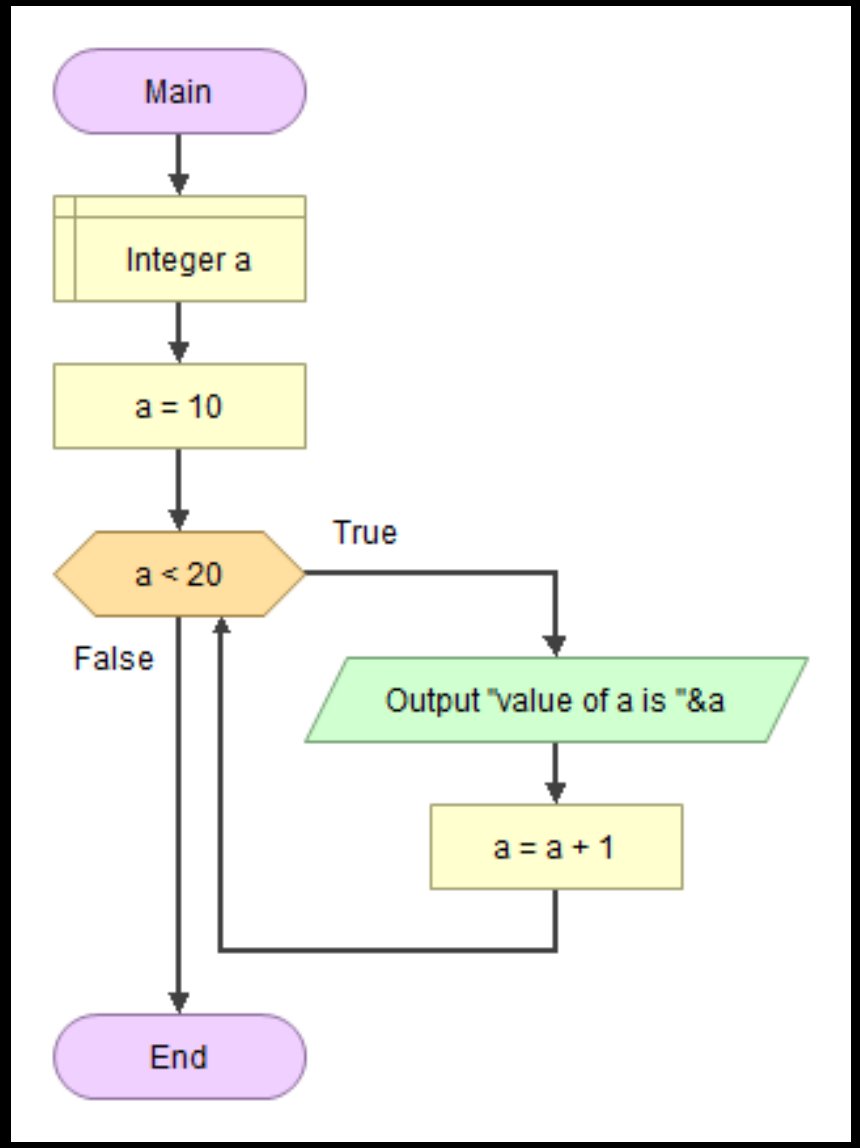
Step 4.1: Display value of a

Step 4.2: a= a+1

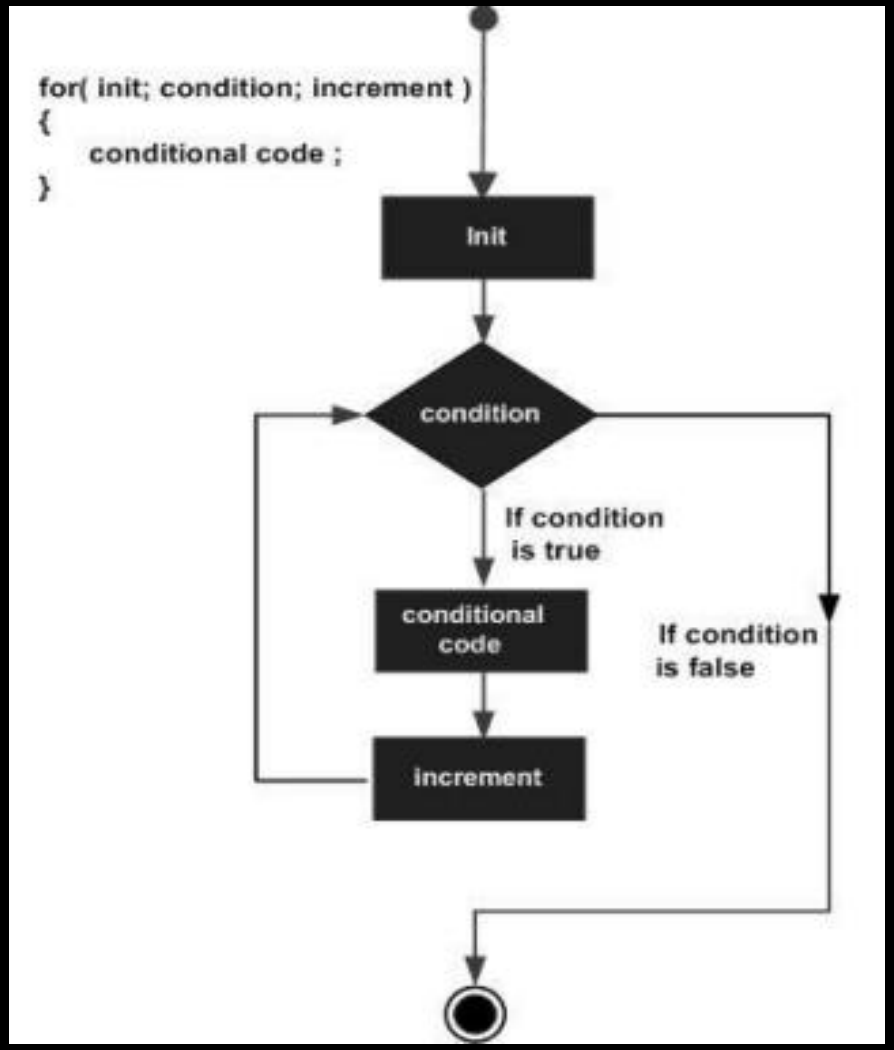
Step 5: End While

Step 6: Stop

while




```
#include <stdio.h>
int main ()
{
        int a = 10;
        while(a<20)
        {
                printf("value of a: %d\n", a);
        a++;
        }
         return 0;
}]
```



for

Step 1: Start

Step 2: Declare a

Step 3: for (a=10; a<20; a=a+1)

Step 3.1: Display value of a

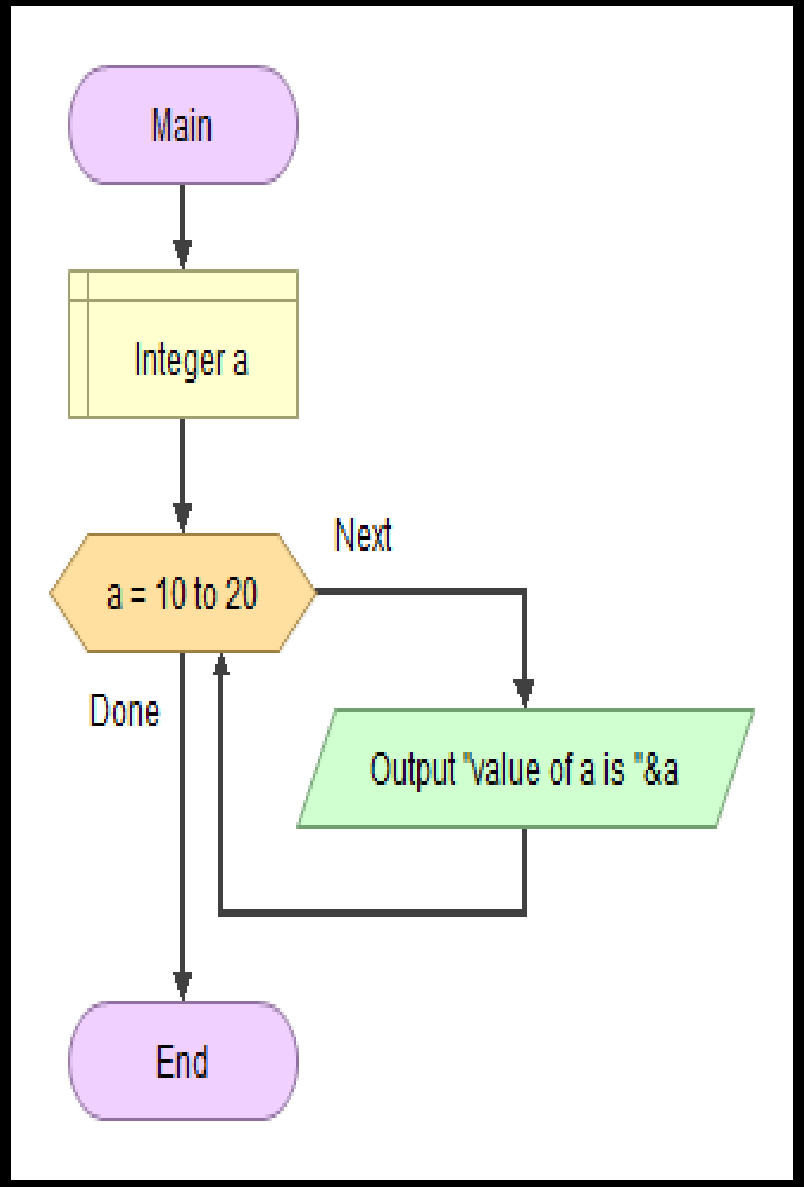
Step 3.2: a= a+1

Step 4: End for

Step 5: Stop

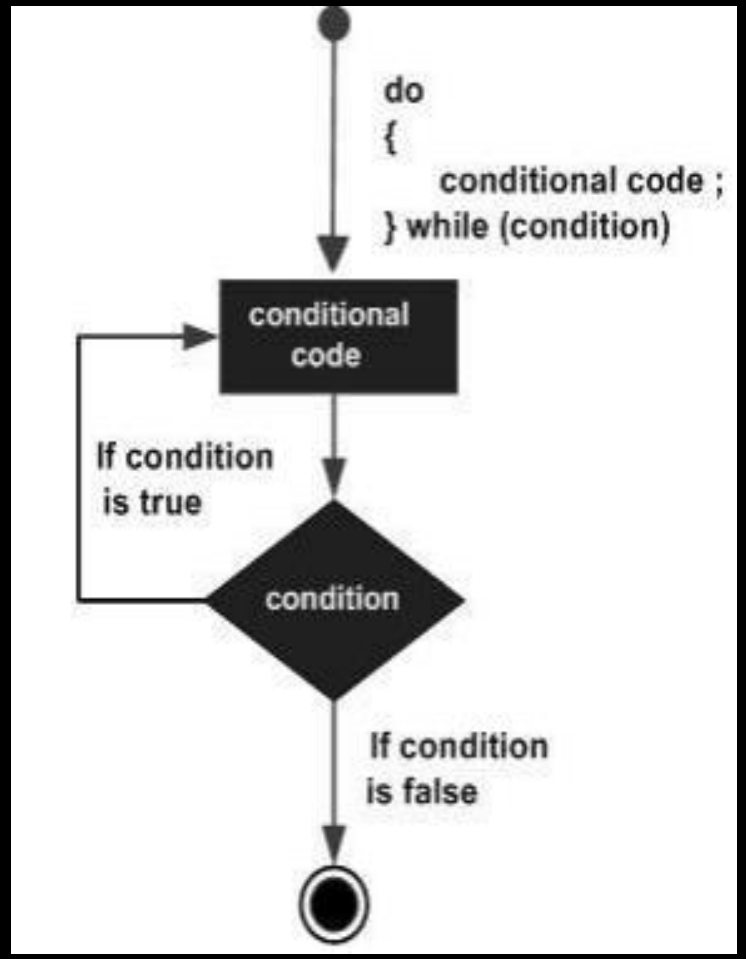
```
for ( init; condition; increment )
{
        statement(s);
}
```

for




```
#include <stdio.h>
int main ()
{
        for(int a=10; a<20; a=a+1)
        {
                printf("value of a: %d\n", a);

        }
         return 0;
}]
```



do-while

Step 1: Start

Step 2: Declare a

Step 3: Assign 10 to a

Step 4: Repeat

Step 4.1: Display value of a

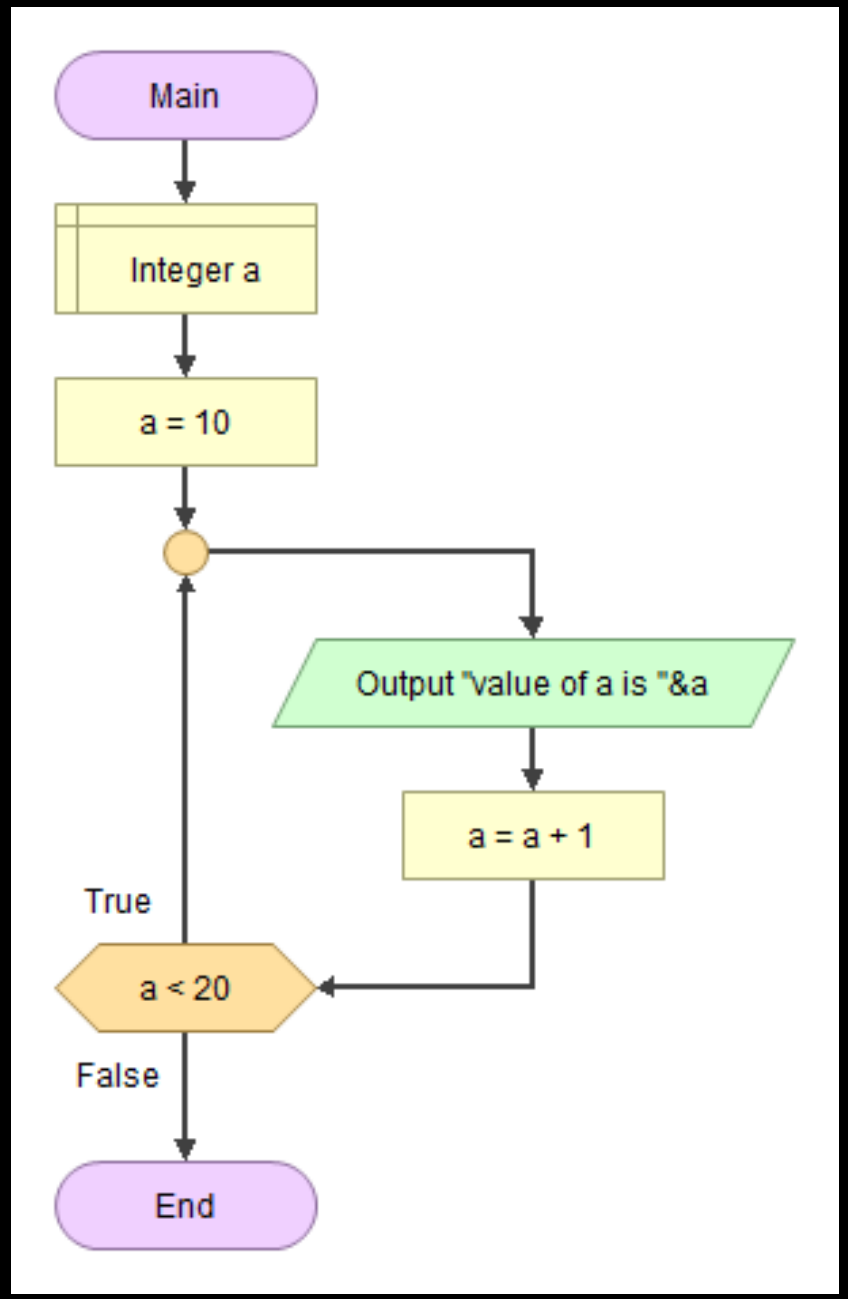
Step 4.2: a= a+1

Step 5: Until (a >= 20)

Step 6: Stop

```
do
 {
        statement(s);
}while( condition );
```

do-while




```
#include <stdio.h>
int main ()
{
        int a = 10;
        do
        {
                printf("value of a: %d\n", a);
                a++;
        } while(a<20);
         return 0;
}
```

Nested Loop

```
for ( init; condition; increment )
{
    for ( init; condition; increment )
    {
        statement(s);
    }
    statement(s);
}
```

```
do
{
    statement(s);
    do
    {
        statement(s);
    }while( condition );

}while( condition );
```

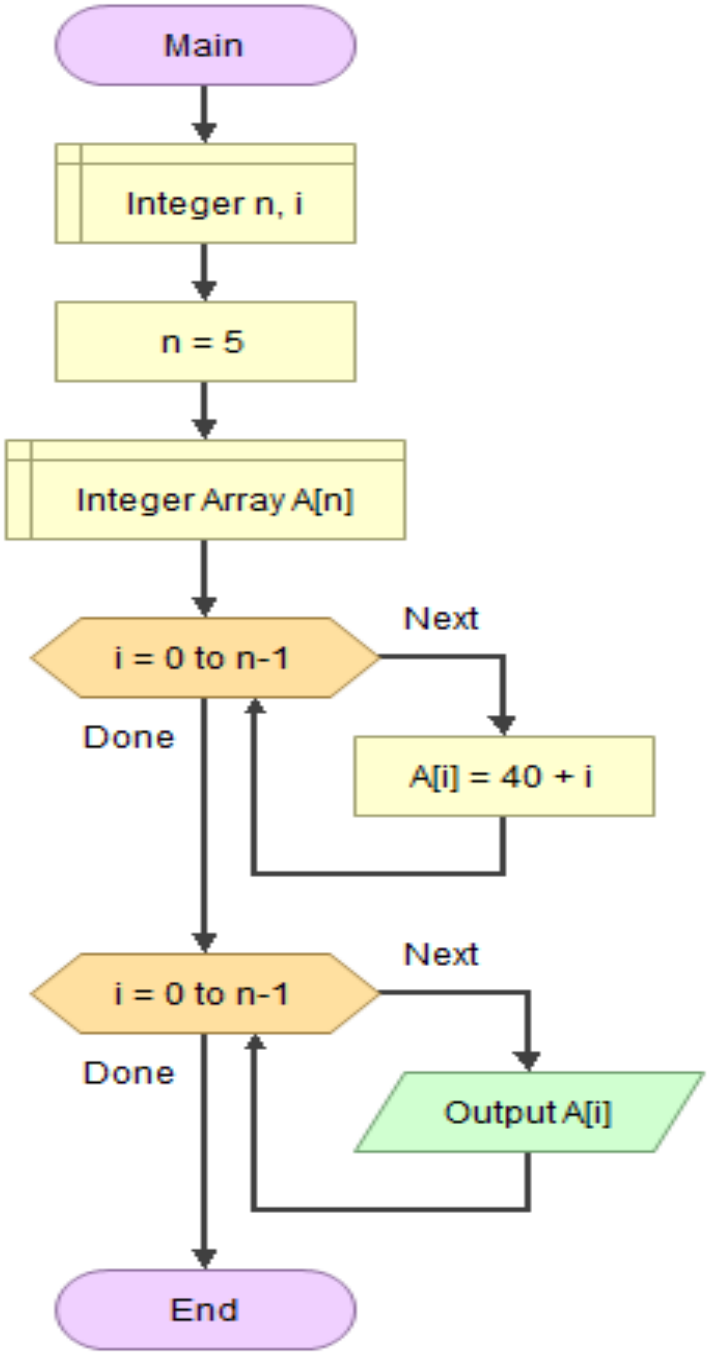
```
while(condition)
{
    while(condition)
    {
        statement(s);
    }
    statement(s);
}
```

Array

Declaring Arrays

```
type arrayName [ arraySize ];
```

- Step 1: Start
- Step 2: Declare size of an array as n and i
- Step 3: Assign 5 to n
- Step 4: Declare an array A[n]
- Step 5: for (i= 0 ; i<5; i=i+1)
  - Step 5.1: Assign A[i]=40 + i
- Step 6: End for
- Step 7: for (i= 0 ; i<5; i=i+1)
  - Step7.1: Display A[i]
- Step 8: Stop



Array

```
#include <stdio.h>
int main ()
{
        int n=5;
        int A[n];
        for ( i = 0; i < 10; i++ )
        {
                A[ i ] = 40 + i;
        }
        for ( i = 0; i < 10; i++ )
        {
                 printf("A[%d] = %d\n", i, A[i]
        }
        return 0;
}
```

Modularization

Factorial of a number- modularization

Step 1: Start

Step 2: Declare N, Fact

Step 3: Assign N=5

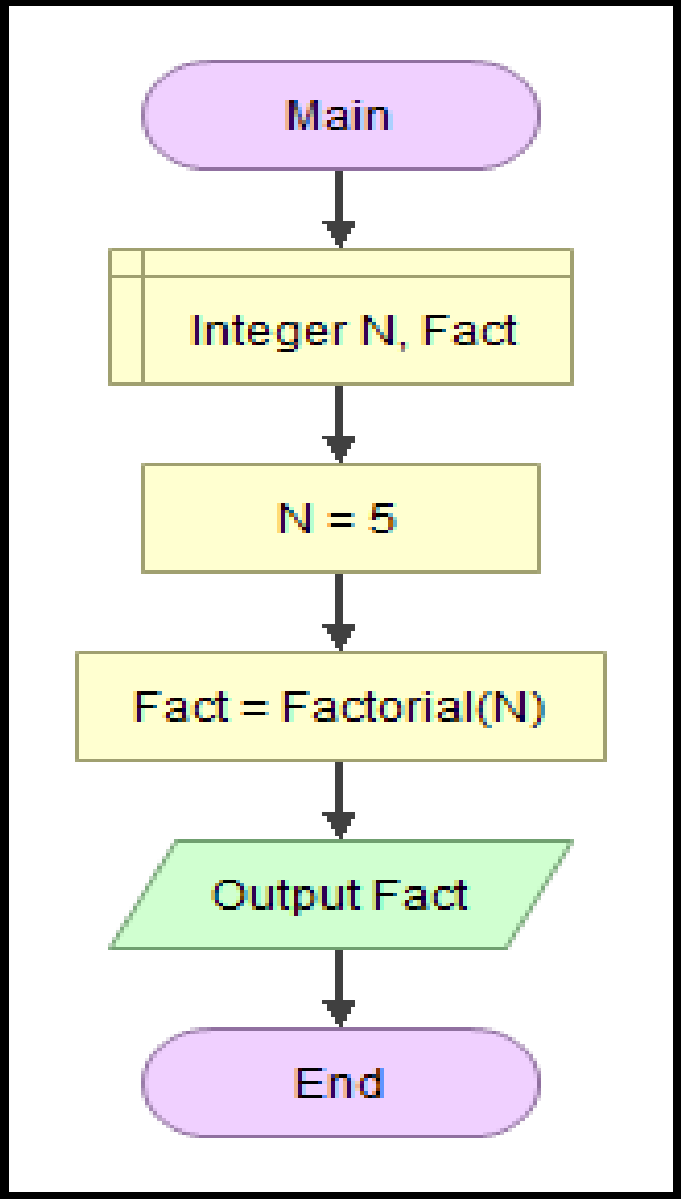
Step 4: Fact=Factorial(N)

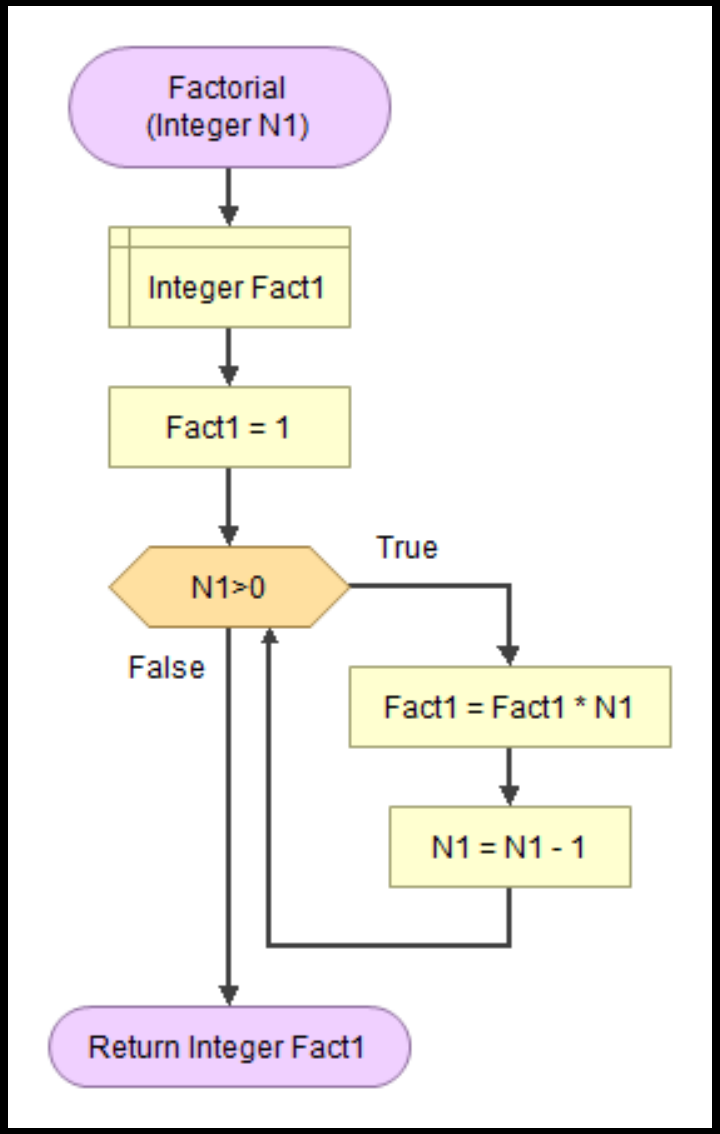
Step 5: Print Fact

Step 6: Stop

```
Factorial(Integer N1)
        Declare Fact1
        Assign Fact1=1
        While N1<0
                Fact1 = Fact1* N1
                N1 = N1 – 1
         End While
        Return Fact1
End Function
```

Modularizing the calculation



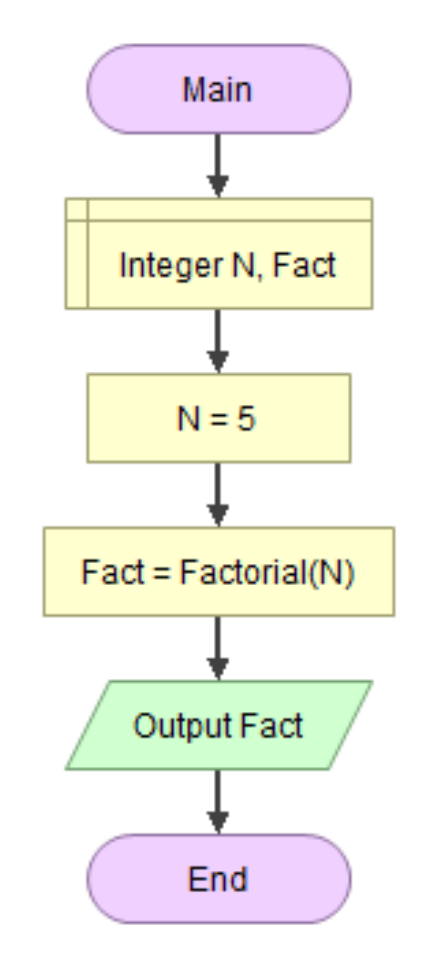


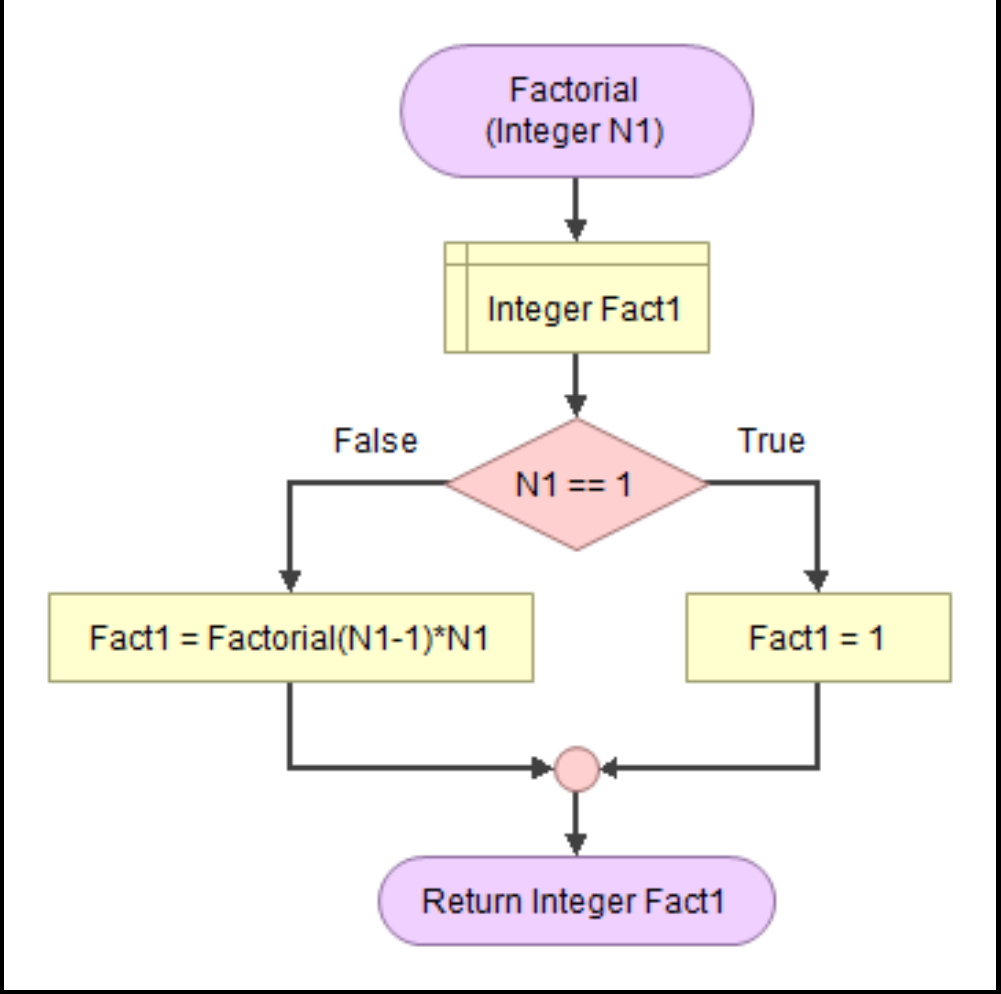
```
#include<stdio.h>
int factorial(int);
main()
{
int N=5, Fact;
Fact=Factorial(N);
printf("%d! = %ld\n", N, Fact);
return 0;
}
```

```
int Factorial(int N1)
{
int Fact1 , i;
for( i = 1 ; i <= N1 ; i++ )
Fact1 = Fact1*1;
return ( Fact1 );
}
```

Recursion

Modified Module- Recursion





```
#include<stdio.h>
int factorial(int);
main()
{
int N=5, Fact;
Fact=Factorial(N);
printf("%d! = %ld\n", N,Fact);
return 0;
```

```
int Factorial(int N1)
{
int Fact1 , i;
if ( N1==1)
        return 1;
else
        return N1 * Factorial( N1-1);

}
```

Thank You