Unit–4

- **Decision Statements**
  - **Unconditional branching: goto statement**
  - **Conditional branching statements:**
  - **If statement**
  - **If-else statement**
  - **Nested If-else statement**
  - **If-else-if Ladder statement**
  - **break, continue and goto statements**
  - **switch statements**

- **Unconditional branching: goto statement OR Explain go to statement with example also explain the difference between forward and backward.**
  - **Use:** the go to statement in the c programming is used to transfer the control unconditionally from one part of the program to other part of the program.
  - 'c' language provide a unconditional branching mechanism called as go to statement.
  - The 'c' is a structural programming language where use of go to statement is a dangerous because the use of the go to statement in the program makes it difficult to understand and debug.
  - The syntax of the go to statement is as follow:
  - **Syntax:**

    go to label name;
  - Here, label is the label to the statement to which go to transfer control.
  - Following shows the two possible use of go to statement.
    - **forward reference**
    - **backward reference**

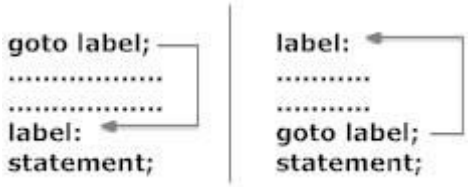


Figure: Working of goto statement

Forward reference	**Backward reference**
go to label; ……. …… Label: Statement;	Label: Statement; …… ……. go to label;
Target statement comes after the go to	Target statement comes before the go to.

  - **Example:**

```
# include <stdio.h>
# include <conio.h>
void main()
{
    int i=1,n;
```

```
        printf ("enter the number n: ");
        scanf ("%d",&n);
        a:
        if (i<=n)
        {
            printf ("%d\n",i);
            i++;
            go to a:
        }
    }
```

- **'C' Language provides a different control Structure for decision making:**
  - if…else statement,
  - else-if ladder
  - switch statement,
  - go to statement and,
  - Break statement.

- **If statement or Explain Simple if statement with flow-chart and example.**
  - General Syntax of simple if statement is as follow:
  - **Syntax:**

```
if (condition)
{
    statement Block;
}
 statement - X;
```

  - **Flowchart.**

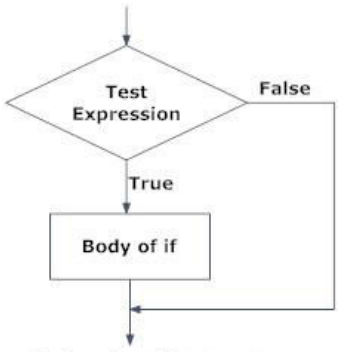


Fig: Operation of if statement

  - In the above syntax the statement block may be a single statement or a group of statements.
  - The simple if statement is executed in the following order.
    - First the condition is checked.
    - If the condition is true then the statement block is executed and then statement-x is executed.
  - If the condition is false then only statement –x is executed.
  - **Example :**

```
#include <stdio.h>
#include<conio.h>
void  main()
```

```
{
    int a;
    printf("Enter value of a");
    scanf("%d",&a);
    if(a>0)
    {
        printf("\n%d",a);
    }
    getch();
}
```

❖ **If-else statement or Explain if...else statement with flow chart and example.**

- The general syntax of if...else statement is as follow:
- **Syntax:**

```
if(condition)
{
    True Block statement(s);
}
else
{
    False block statement(s);
}
```

- **Flowchart:**

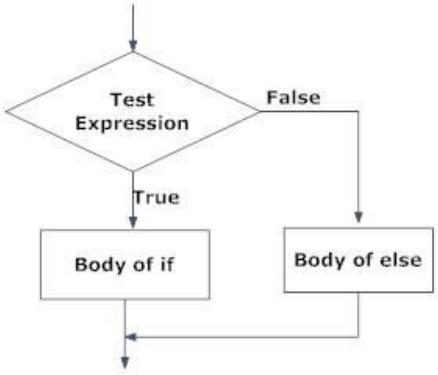


Fig: Operation of if...else statement

- **The if...else statement is executed in the following order:**
  - First the condition is checked.
  - if condition is true the true statement is executes.
  - if condition is false the false statement is executed.
- **Example:**

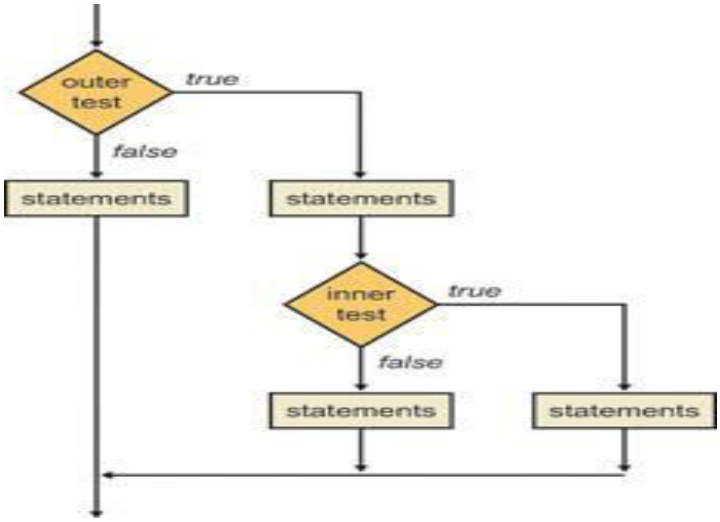
```
#include <stdio.h>
#include <conio.h>
void main()
{
    int a,b;
    printf ("Enter a number a and b.\n");
    scanf ("%d %d",&a,&b);
    if (a>b)
```

```
    {
        printf ("a is maximum");
    }
    else
    {
        printf ("b is maximum");
    }
    getch ();
}
```

- **Nested If-else statement or Explain nesting if-else statement with flowchart and Example.**
  - When the more than one condition is to be checked then we can use nesting of if...else first the condition is checked.
  - The syntax of nesting if...else statement is as follow:
  - **Syntax:**

```
if (condition 1)
{
    if (condition 2)
    {
        statement-1;
    }
    else
    {
        statement-2;
    }
 }
else
{
    statement-3;
}
```

  - **Flowchart:**
  -



  - **The nesting if-else statement is executed in the following manner.**

- First the condition 1 is checked
- If the condition 1 is true then condition 2 is checked. If condition 2 is true then statement-1 is executed.
- But if the condition 2 is false the statement-2 is executed.
- If condition 1 is false the statement-3 is executed.

➢ **Example :**

```
#include <stdio.h>
#include <conio.h>
void  main()
{
    int n1, n2,n3;
    printf("enter three integers to check".\n);
    scanf("%d %d%d",&n1,&n2,&n3);
    if(n1>n2)
    {
        if(n1>n3)
        {
            printf(“n1 is maximum”);
        }
        else
        {
            printf(“n3 is maximum”);
        }
    }
    else
    {
        if(n2>n3)
        {
            printf(“n2 is maximum”);
        }
        else
        {
            printf(“n3 is maximum”);
        }
    }
    getch();
}
```

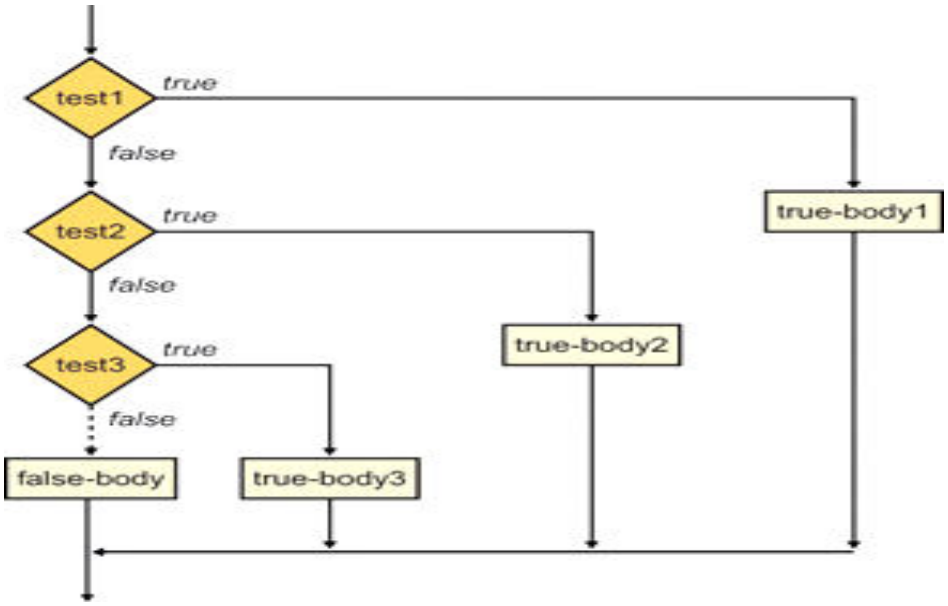
❖ **If-else-if Ladder statement or Explain if-else-if ladder (Multiple if...else) with flow chart and example.**

➢ The if...else ladder statement provide two-way decision where we select one of the alternative.
➢ It is used for multiple choices.
➢ The two way decision is done by nested if...else is not sufficient.
➢ Following is the syntax of the if...else ladder.
➢ **Syntax:**

```
if (condition-1)
    statement(s)-1;
```

```
else if(condition-2)
    statement(s)-2;
else if(condition-3)
    statement(s)-3;
        .
        .
else if(condition-N)
    statement(s)-N;
else
    default statement(s)-  N;
```

- if-else-if ladder is executed in the following order:
  - First condition-1 is executed; if the condition-1 is true then the statement-1 is executed.
  - if the condition-1 is false then condion-2 is checked. If condition-2 is true then statement-2 is executed.
  - This procedure repeated until all the condition is checked. if all the condition became false then the default statement is executed.



- **Example:**

```
#include <stdio.h>
#include <conio.h>
void main()
{
    int a;
    printf ("Enter three integers to check".\n);
    scanf ("%d",&a);
    if(a==1)
    {
        printf("Monday");
    }
    else if(a==2)
```

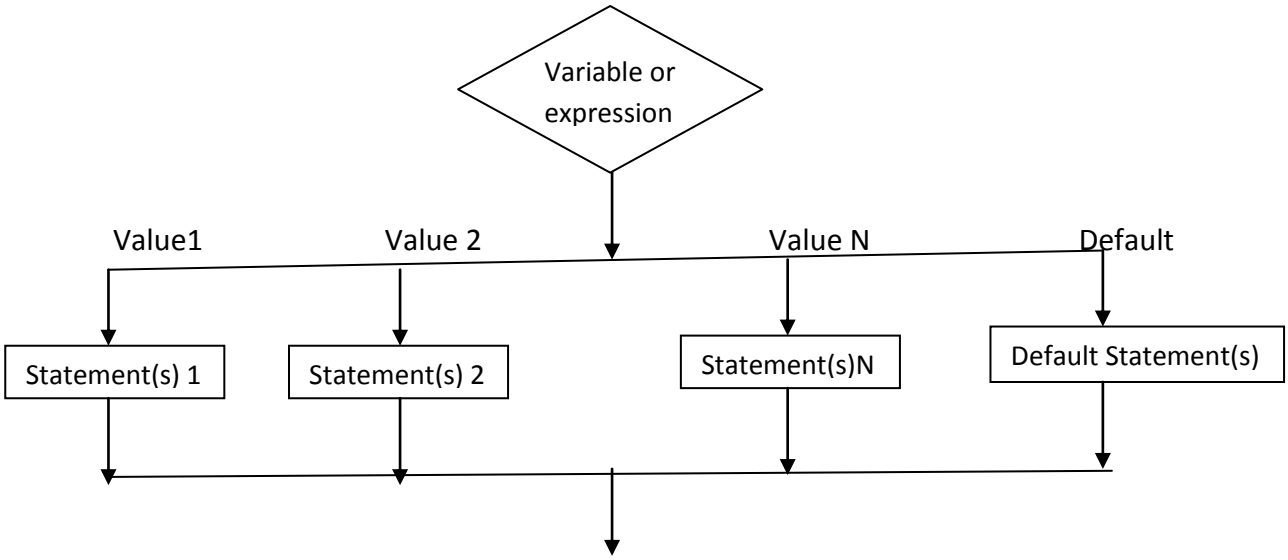
```
        {
            printf("Tuesday");
        }
        else if (a==3)
        {
            printf("Wednesday");
        }
        else if(a==4)
        {
            printf("Thursday");
        }
        else if(a==5)
        {
            printf("Friday");
        }
        else if(a==6)
        {
            printf("Saturday");
        }
        else
        {
            printf("Sunday");
        }
        getch();
    }
```

- **switch statements OR Explain switch case statement with flowchart and example.**
  - The switch statement is also known as multi-choice or multi decision statement.
  - Writing the code using the multiple if...else becomes lengthy and also difficult to manage. Using switch statement it is done by easy.
  - It provides the choice for each value of variable or expression.
  - The switch statement test the value of a given variable against a list of case values and when the match is found, a statement associated with the case is executed.
  - General syntax of switch case statement is as follow:
  - **Syntax**:

```
Switch (variable name or expression)
{
    Case value1:
        statement(s) 1;
        break;
    Case value2:
        statement(s) 2;
        break;
            .
            .
    Case value N:
        statement(s) N;
        break;
```

```
    default:
        default statement(s);
}
```

- **Flow Chart:**



- The expression or variable name is an integer or characters.
- Value-1, value-2 is constant or known as case labels, each case lable value must be unique with a switch statement. each case lable must end with (;).
- The switch case statement is executed in the following order:
  - The value of the expression is compared against the value of case lable.
  - If the case is found whose value match with value of the expression, then the statement associated with the case is executed. There is a single statement or multiple statements. There is break which sends the control to the next statement.
  - If the value of the expression does not match with any case value then the statement associated with the default case is executed.
- **Example:**

```
#include<stdio.h>
#include<conio.h>
void main()
{
    int day;
    clrscr ();
    printf ("Enter Week day no = ");
    scanf ("%d",&day);
    switch (day)
    {
        case 1:
            printf ("Sunday");
            break;
        case 2:
            printf ("Monday");
            break;
        case 3:
            printf ("Tuesday");
            break;
```

```
            case 4:
                printf ("Wednesday");
                break;
            case 5:
                printf ("Thursday");
                break;
            case 6:
                printf ("Friday");
                break;
            case 7:
                printf ("Saturday");
                break;
            default:
                printf ("Wrong Day NO Insert.");
        }
        getch ();
    }
```

- **Explain Break statement with example.**
  - We have use the break statement with the switch statement.
  - The function of break statement is exit from the switch body.
  - If it is not written after each case statement, then control pass to the next statement ,so remaining statement of the next case statement will also execute even if the case value do not match and the program will not function properly.
  - The break statement terminates the loop (for, while and do...while loop) immediately when it is encountered. The break statement is used with decision making statement such as if...else
  - Syntax:- **break;**
  - **Example:-**

```
#include <stdio.h>
void main()
{
    int i;
    clrscr();
    for(i=1;i<10;i++)
    {
        printf("%d\n",i);
        if( i == 5)
        {
            break;
        }
    }
    getch();
}
```

- **Explain continue statement with example.**
  - C **Continue statement** are used to skips the rest of the current iteration in a loop and returns to the top of the loop.
  - The continue statement works like a shortcut to the end of the loop body.
  - **Syntax**
    - jump-statements (loop or switch case);

- **continue;**

➢ **Example:-**

```
#include<stdio.h>
#include<conio.h>
void main()
{
    int i;
    clrscr();
    for(i=1; i<=10; i++)
    {
        if(i==3)
        {
            continue;
        }
        printf("%d \n",i);
    }
    getch();
}
```

Unit–5

- **Loop Control Statements**
  - **for loop**
  - **Nested for loop**
  - **While loop**
  - **Do-while loops**

- **Parts of looping structures in 'c' and how many parts of loop?**
  - **The looping structure provided in the 'c' language are:**
    - **while loop,**
    - **Do...while loop,**
    - **For loop.**
  - **There are two parts of loop.**
    - **Condition**:
      - The control statement tests some condition.
      - If condition is satisfied then the loop is executed otherwise the statement follows the loop is executed**.**
    - **Body:**
      - This statement consists of single or group of statement.

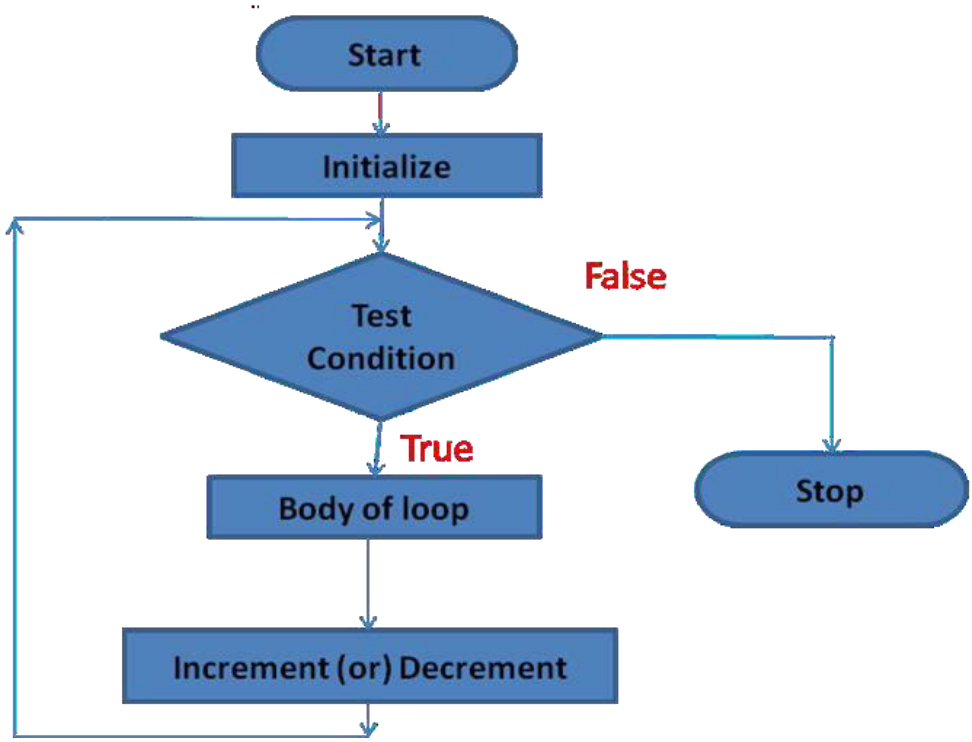
- **How many types of loop? Explain in details.**
  - Depending on where the condition is checked, we can have two types of loop structure:
    - **Entry control loop.**
    - **Exit control loop.**
  - **Entry control loop**:
    - The condition is written first and then the body of statement.
    - If the condition is tested before the body of loop is called Entry control loop.
    - If condition is true then the body of loop is executed otherwise the loop is not executed.
  - **Exit control loop:**
    - The body of statement is written first then the condition is written.
    - If the condition is tested after the body of the loop then it is known as exit control loop.
    - So first body of the loop is executed and then the condition is checked.

- **Explain while loop with flow chart and example.**
  - While loop is the entry control loop. Because in while loop first the condition is checked.
  - Following is the syntax of the while loop.
  - **Syntax:**

```
While (condition)
{
    Body of the loop
}
```

  - {} is known as body of the loop. If body contain one statement then it is not necessary to enclose body of the loop in the pair of brace {}.
  - The while loop is executed in the following format.
  - Here the condition is evaluated first and if it is true then the statement in the body of the loop is executed.

- After executing body, the condition is evaluated again and if it is true body is executed again. This process is repeated as long as the condition is true. The control move out once the condition is false
- **Flow Chart:**



- **Example:**

```
#include <stdio.h>
void main()
{
    int i=1;
    while( i<=5)
    {
        printf("%d\n",i);
        i++;
    }
}
```

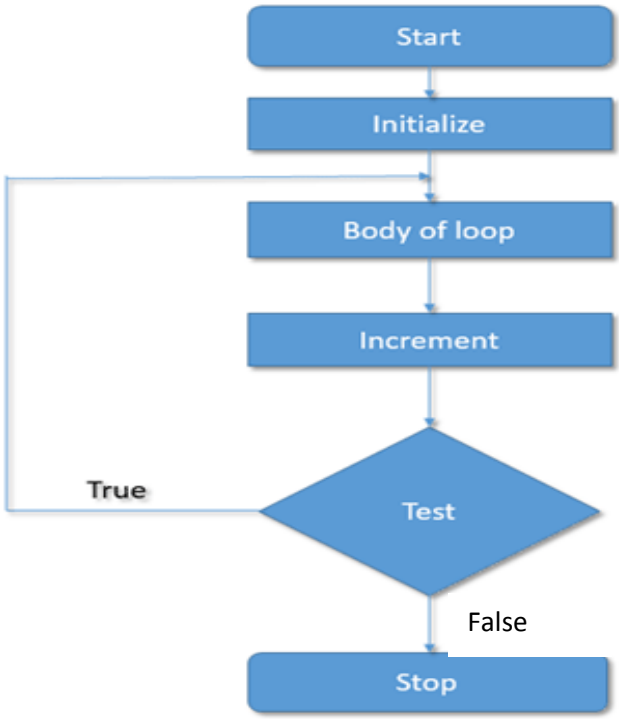
❖ **Explain do..While loop with flow chart and example.**

- Do...While loop is a exit control loop.
- Following is the syntax of the exit control loop. Because after the executing the body of the loop the condition is checked.
- **Syntax:**

```
Do
{
    Body of the loop
```

```
} while (condition);
```

- **Flow Chart:**



- The do while loop is executed in the following format.
- In do...While loop the body is executed and then the condition is checked.
- If the condition is true the body is executed again and again, as long as the condition is true.
- The control moves out of loop once, the condition become false.
- The do...while loop is exit control loop so first body is executed the the condition is checked. This ensures that the body of the loop is executed at lease once even if the condition is false first time.
- E**xample :**

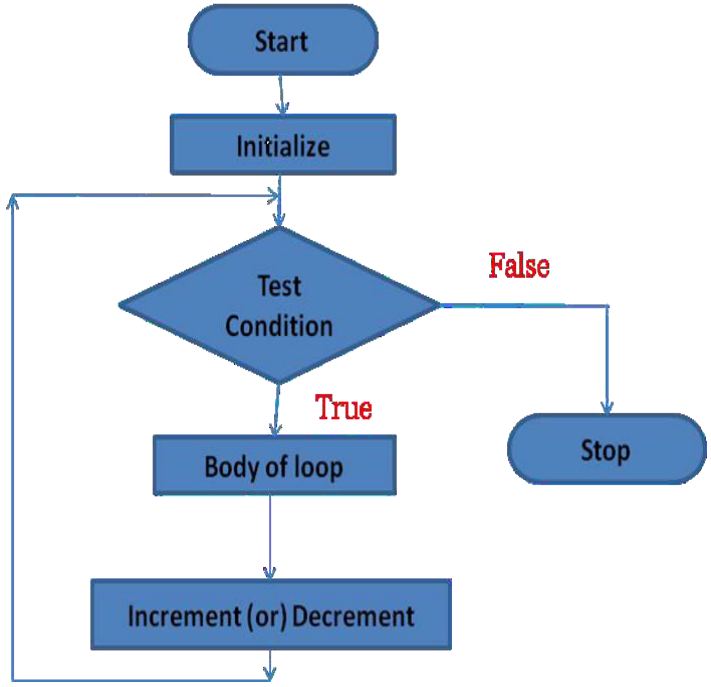
```
#include <stdio.h>
void main()
{
    int i=1;
    do
    {
        printf("%d\n",i);
        i++;
    }while(i<=5);
}
```

❖ **Explain for loop with flow chart and example**

- Following is the syntax of the for loop.
- **Syntax:**

```
for(initialization ; condition ; increment or decrement)
{
    Body of the loop
}
```

- **Initialization:**
  - We require one variable which is used to control the loop, which is called as control variable.
  - The control variable is initialized in the initialization expression.
- **Condition:**
  - The condition statement is checked the value of the control variable.
  - If the condition statement is true, the body of the loop is executed.
- **Increment/Decrement:**
  - The increment/decrement of the control variable is done in this part.
  - After the incrementing/decrementing the value of control variable is tested using condition if condition is true than again the body of loop is executed and this process is repeated until the condition become false.
- **Flow Chart:**



- **Example :**

```
#include <stdio.h>
void main()
{
    int i;
    for(i=1;i<=5;i++)
    {
        printf("%d\n",i);
    }
}
```

❖ **Explain nesting loops with example.**

- When the loop inside another loop is called nesting of loops.
- The nesting can be for any numbers of levels, for certain problem the nesting of loop are needed.

- The outer loop should not end between the starting of inner loop and ending of inner loop.
- **Syntax:**

```
for (i=1;i<=3;i++)
{
    for (j=0;j<=3;j++)
    {
        ..
    }
}
```

- In the above example for(i=0;i<=3;i++)is outer for loop. 'i' is the control variable for outer loop.and 'j' is the control variable for inner loop.
- The above example is executed in the following manner.
- **Example:**

```
#include<stdio.h>
#include<conio.h>
void main()
{
int i,j;
for( i=1;i<=5;i++)
    {
        for( j=1;j<=i;j++)
        {
        printf("%d",j);
        }
    printf("\n");
    }
getch();
}
```