UNIT 4 : ITERATIVE STATEMENTS

4.1 Use of goto statement for iteration

4.2 while loop

4.3 do..while loop

4.4 for loop

4.5 Nested while, do..while and for loops

4.6 Jumping statement: (break and continue)

Control Statement or Control Structure

- Control Statements or Control Structures are used to control the flow of Program.
- Control structures are the statements that control the flow of the source code.

- There are three categories of flow controls:
  1. Branching or Selection Statement
  2. Looping or Iterative Statement
  3. Jumping

Looping or Iteration or Iterative Statements

- A **Loop** executes the sequence of statements many times until the stated condition becomes false.

OR

- **Looping** is the way to execute a set of instructions any number of times.

4.1 Use of goto statement for iteration

- The **goto statement** is known as jump **statement in C**.
- As the name suggests, **goto** is **used** to transfer the program control to a predefined label.
- The **goto** statment can be **used** to repeat some part of the code for a particular condition.

Syntax:

label:

….

Code to Execute

….

….

goto label;

Example:

```
void main()
{
	int i=1,n;
	clrscr();
	printf("Enter Any Number:");
	scanf("%d",&n);
	a1:
		if(i<=n)
		{
			printf("%d\n",i);
			i++;
			goto a1;
		}
	getch();
}
```

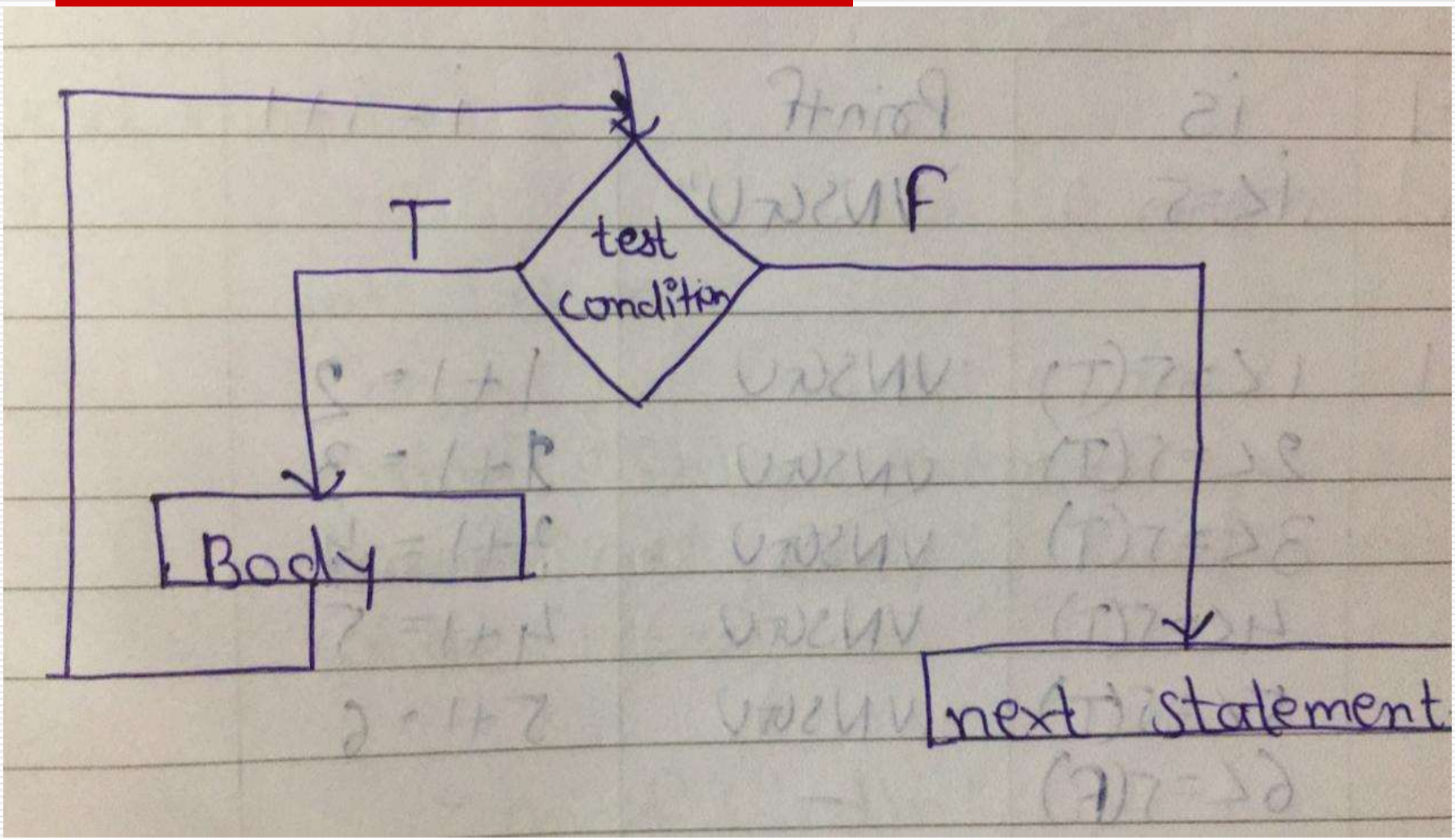
4.2 while loop

- A **while** loop in C programming repeatedly executes a target statement as long as a given condition is true.

Syntax:

```
while(Test Condition)
{
    //Body of Loop
}
Next-statements;
```

Flowchart:



Example:

```
#include<stdio.h>
#include<conio.h>
void main()
{
      int i=1,n;
      clrscr();
      printf("Enter any number :");
      scanf("%d",&n);
      while(i<=n)
      {
            printf("%d",n);
            i=i+1;
      }
      getch();
}
```

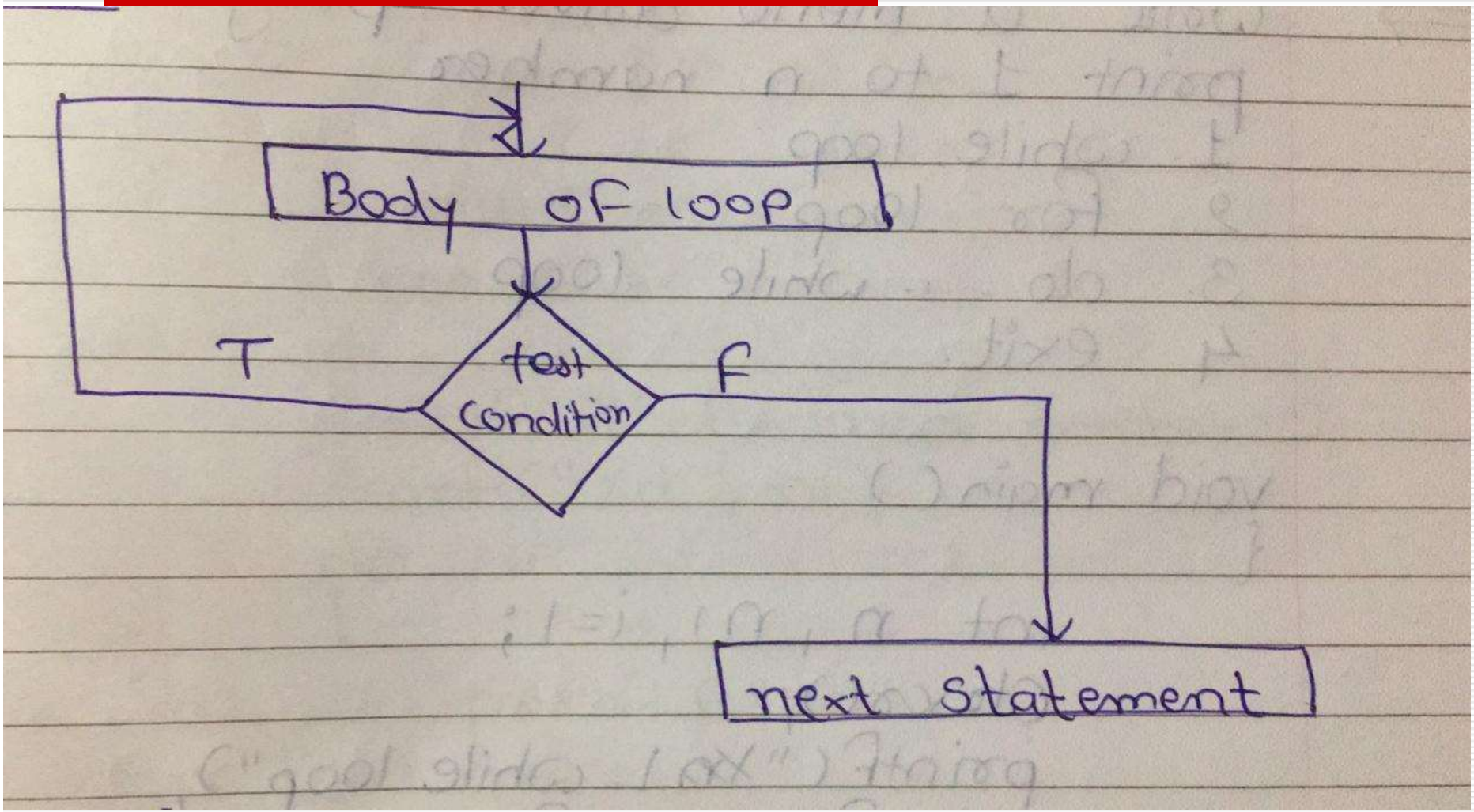
4.3 do..while loop

- The do..while loop is similar to the while loop with one important difference.
- The body of do...while loop is executed **at least once**. Only then, the test expression is evaluated.

Syntax:

```
do
{
    //body of loop
}while(Test-Condition);
```

Flowchart:



Example:

```
#include<stdio.h>
#include<conio.h>
void main()
{
        int i=1,n;
        clrscr();
        printf("Enter any number :");
        scanf("%d",&n);
        do
        {
                printf("%d",n);
                i=i+1;
        }while(i<=n);
        getch();
}
```

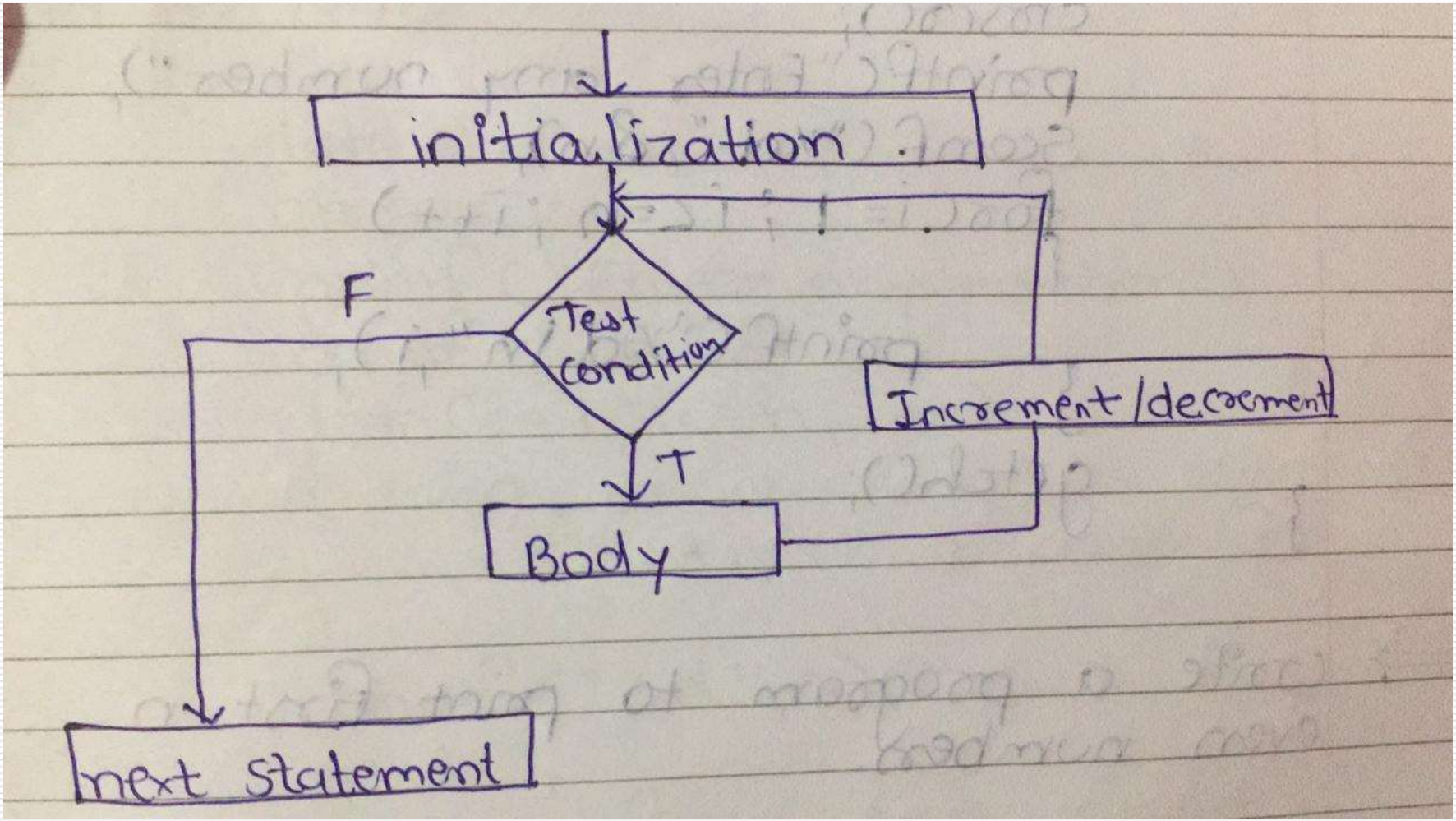
4.4 for loop

- A **for** loop is a repetition control structure that allows you to efficiently write a loop that needs to execute a specific number of times.

Syntax:

```
for ( initialization; Test- Condition; Increment/Decrement )
{
   Body of Loop;
}
```

Flowchart:


initialization
Test condition
F
T
Body
Increment/decrement
next statement

How for loop works?

- The initialization statement is executed only once.
- Then, the test expression is evaluated. If the test expression is evaluated to false, the for loop is terminated.
- However, if the test expression is evaluated to true, statements inside the body of for loop are executed, and the update expression is updated.
- Again the test expression is evaluated.

Example:

```
#include<stdio.h>
#include<conio.h>
void main()
{
      int i=1,n;
      clrscr();
      printf("Enter any number :");
      scanf("%d",&n);
      for(i=1;i<=n;i++)
      {
            printf("%d",n);
      }
      getch();
}
```

- Entry Control Loop
  - while loop
  - for loop
- Exit Control Loop
  - do…while loop

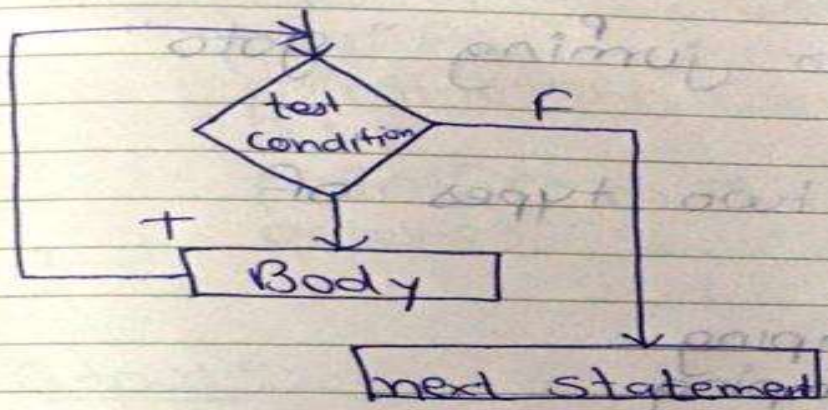
Difference between Entry control loop and exit control loop:

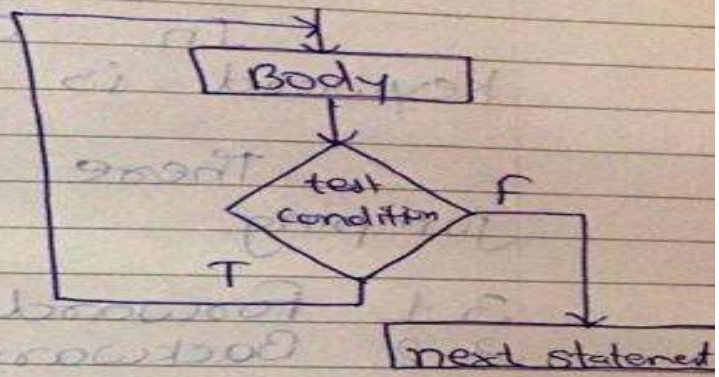
	Entry control loop	Exit control loop
1	The test condition will be checked, if the answer is true then only the body will be executed	first the body will be checked, then the test condition will be checked
2.	if the test condition is true then only the body will be executed	if the test condition is false, then also one time the body will be executed.
3.	while loop and for loop are entry control loop	do while loop is exit control loop

4.	Syntax:
	while (test condition)
	{
	Body
	}
	OR
	For (initialization; test-condition; increment / decrement)
		Syntax:
		do
		{
		body
		} while (Test condition);

```
{
        body
}
```

5.





6. Example:

```
void main()
{
        int  i=1;
        clrscr();
        while (i<=5)
        {
                printf ("\n%d",i);
                i++;
        }
        getch();
}
```

Example:

```
void main()
{
        int i=1;
        clrscr();
        do
        {
                printf ("\n %d",i
                i++;
        } while (i<=5);
        getche();
}
```

4.5 Nested while, do..while and for loops

- Loop inside another loop is known as Nested Loop.

Nested While Loop

- **Syntax**:

```
while(Test Condition1)
{
    while(Test Condition2)
    {
        //Body of Loop
    }
    Next-statements;
}
Next-statements;
```

Example:

```
void main()
{
        int i=1,j=1,n;
        clrscr();
        while(i<=10)
        {
                j=1;
                while(j<=i)
                {
                                printf("*");
                                j++;
                }
                printf("\n");
                i++;
        }
        getch();
}
```

Nested Do…While Loop

- **Syntax**:

```
do
{
    do
    {
        //body of loop
    }while(Test-Condition1);
}while(Test-Condition2);
```

Example:

```
void main()
{
        int i=1,j=1,n;
        clrscr();
        do
        {
                j=1;
                do
                {
                                printf("*");
                                j++;
                }while(j<=i);
                printf("\n");
                i++;
        }while(i<=10);
        getch();
}
```

Nested For Loop

- **Syntax:**

```
for ( initialization; Test- Condition1; Increment/Decrement )
{
   for ( initialization; Test- Condition2; Increment/Decrement )
          {
                 Body of Loop;
          }
}
```

Example:

```
void main()
{
        int i=1,j=1,n;
        clrscr();
        for(i=1;i<=10;i++)
        {
                for(j=1;j<=i;j++)
                {
                        printf("*");
                }
                printf("\n");
        }
        getch();
}
```

Infinite Loop

- Never Ending Loop is known as Infinite Loop.
- Example:
  - for(;;)

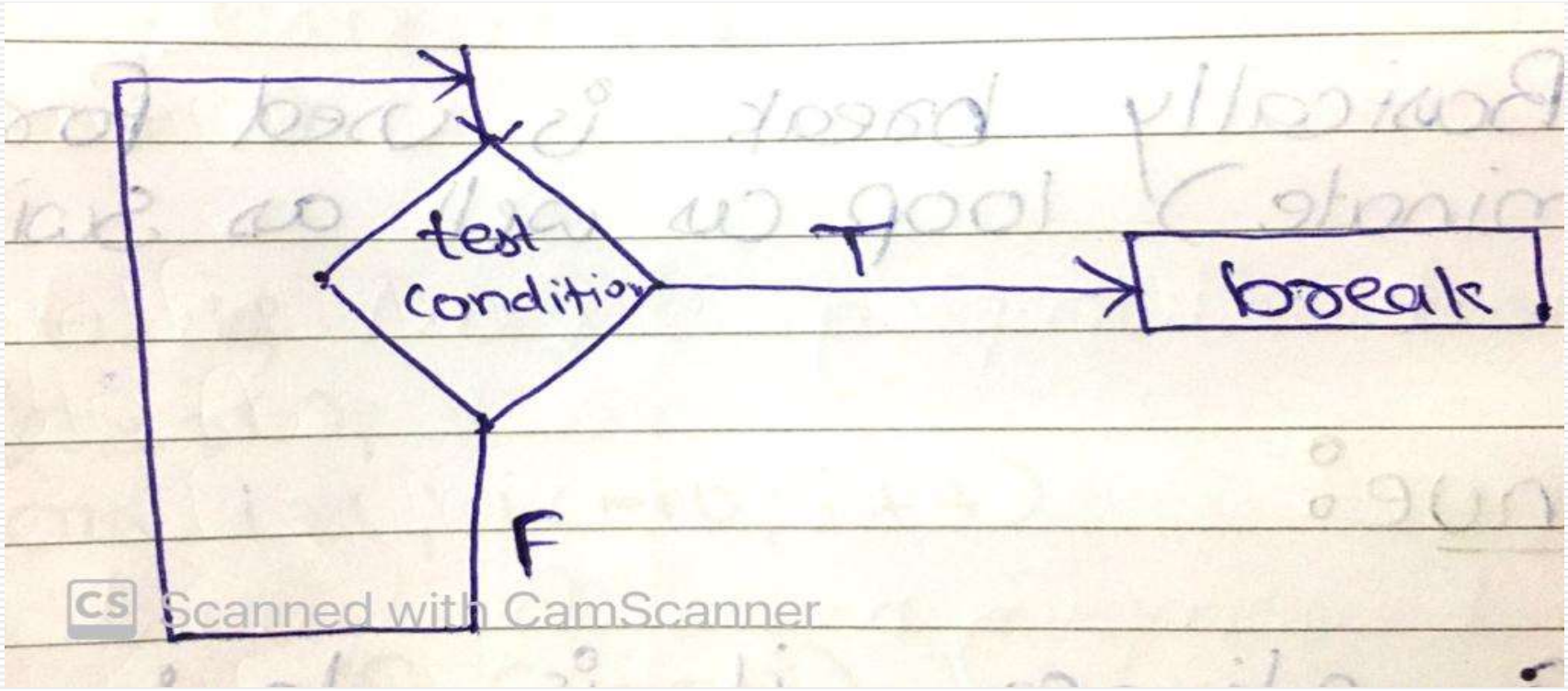
    printf(“VNSGU”);

4.6 Jumping statement: (break)

- When the break statement is executed inside a loop, the loop will be terminated and program control returns at the next statement of the loop.
- Syntax:
  - break;

- Note: Basically break is used for breaking(terminate) the loop as well as switch case.

Flowchart



Example

```
void main()
{
	int i=1;
	clrscr();
	for(i=1;i<=10;i++)
	{
		if(i==5)
				break;
		printf("%d",i);
	}
	getch();
}
```

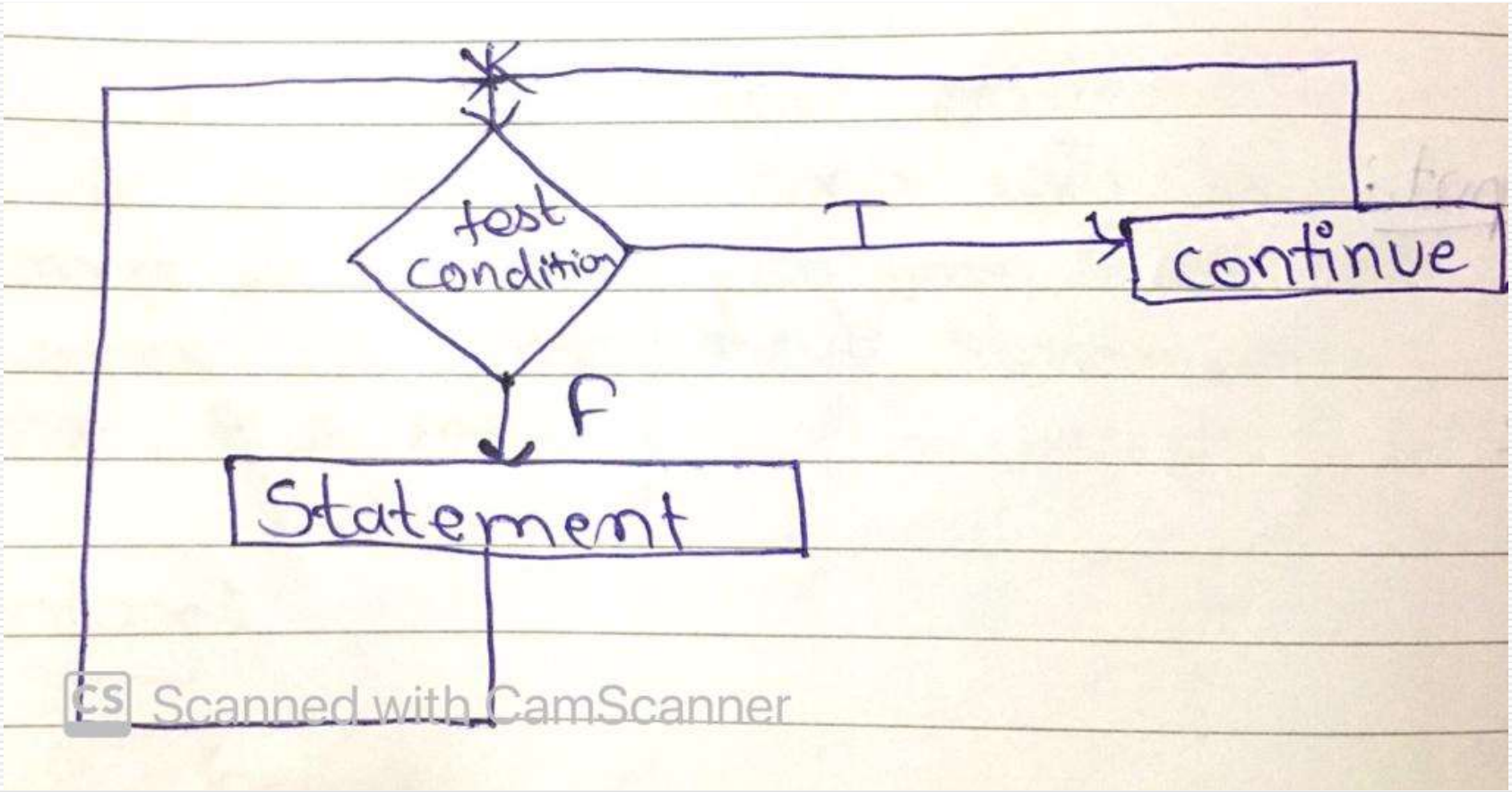
OUTPUT:

1234

4.6 Jumping statement: (continue)

- Sometimes it is desirable to skip some statement inside the loop, in such case **continue** statements are used.
- Continue forces the next iteration of the loop to take place and skipping any code in between.
- Syntax:
  - continue;

Flowchart


test condition
T
continue
F
Statement
Scanned with CamScanner

Example

```
void main()
{
        int i=1;
        clrscr();
        for(i=1;i<=10;i++)
        {
                if(i==5)
                        continue;
                printf("%d",i);
        }
        getch();
}
```

OUTPUT:

1234678910