Selection and Iterative/Repetitive control structures in C

Operator Basics

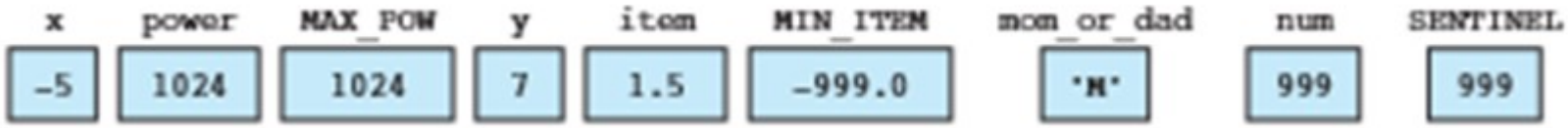
Operators used in condition checking
in control structures

Relational and Equality Operators

Operator	Meaning	Type
<	less than	relational
>	greater than	relational
<=	less than or equal to	relational
>=	greater than or equal to	relational
==	equal to	equality
!=	not equal to	equality

Examples

Memory with Values



Sample Conditions

Operator	Condition	English Meaning	Value
<=	x <= 0	x less than or equal to 0	1 (true)
<	power < MAX_POW	power less than MAX_POW	0 (false)
>=	x >= y	x greater than or equal to y	0 (false)
>	item > MIN_ITEM	item greater than MIN_ITEM	1 (true)
==	mom_or_dad == 'M'	mom_or_dad equal to 'M'	1 (true)
!=	num != SENTINEL	num not equal to SENTINEL	0 (false)

Logical Operators

- To form more complicated conditions or logical expressions
- Three operators:
  - And (&&)
  - Or (||)
  - Not(!)

Logical And

The && Operator (and)

operand1	operand2	operand1 && operand2
nonzero (true)	nonzero (true)	1 (true)
nonzero (true)	0 (false)	0 (false)
0 (false)	nonzero (true)	0 (false)
0 (false)	0 (false)	0 (false)

Logical Or

The || Operator (or)

operand1	operand2	operand1 || operand2
nonzero (true)	nonzero (true)	1 (true)
nonzero (true)	0 (false)	1 (true)
0 (false)	nonzero (true)	1 (true)
0 (false)	0 (false)	0 (false)

Logical Not

The ! Operator (not)

operand1	!operand1
nonzero (true)	0 (false)
0 (false)	1 (true)

When 'A‘ = 60 and 'B' =13

Operator	Description	Example
&	Binary AND Operator copies a bit to the result if it exists in both operands.	(A & B) = 12 i.e., 0000 1100
|	Binary OR Operator copies a bit if it exists in either operand.	(A | B) = 61 i.e., 0011 1101
^	Binary XOR Operator copies the bit if it is set in one operand but not both.	(A ^ B) = 49 i.e., 0011 0001
~	Binary Ones Complement Operator is unary and has the effect of 'flipping' bits.	(~A) =-61 i.e., 1100 0011 in 2's complement form.
<<	Binary Left Shift Operator. The left operands value is moved left by the number of bits specified by the right operand.	A << 2 = 240 i.e., 1111 0000
>>	Binary Right Shift Operator. The left operands value is moved right by the number of bits specified by the right operand.	A >> 2 = 15 i.e., 0000 1111

True/False Values

- For numbers all values except 0 is true
- For characters all values except ‘/0’ (Null Character) is true

Short Circuit Evaluation

- Stopping evaluation of a logical expression as soon as its value can be determined is called short-circuit evaluation
- Second part of ‘&&’ does not gets evaluated when first part is evaluated as False
- Second part of ‘| |’ does not gets evaluated when first part is evaluated as true

Short Circuit Evaluation

- (num % div == 0) – Runtime error if div = 0
- But prevented when written as
- (div != 0 && (num % div == 0))

Comparing Characters

- We can also compare characters in C using the relational and equality operators

Character Comparisons

Expression	Value
`'9' >= '0'`	1 (true)
`'a' < 'e'`	1 (true)
`'B' <= 'A'`	0 (false)
`'Z' == 'z'`	0 (false)
`'a' <= 'A'`	system dependent
`'a' <= ch && ch <= 'z'`	1 (true) if `ch` is a lowercase letter

Logical Assignment

- even = (n % 2 == 0);
- in_range = (n > -10 && n < 10);
- is_letter = ('A' <= ch && ch <= 'Z') || ('a' <= ch && ch <= 'z');
- Variable in_range gets 1 (true) if the value of n is between −10 and 10 excluding the endpoints;
- is_letter gets 1 (true) if ch is an uppercase or a lowercase letter.

To learn

- Selective Control Structures
  - if
  - if else
  - if else if
  - switch case
- Repetitive Control Structures
  - while
  - for
  - do while

Syntax of if Statement

if (condition)

    statement $_T$;

Eg:

if (x > 0.0)

    pos_prod = pos_prod * x;

If condition evaluates to true (a nonzero value), then statement $_T$ is executed; otherwise, statement $_T$ is skipped.

Syntax of if else

Form 2:

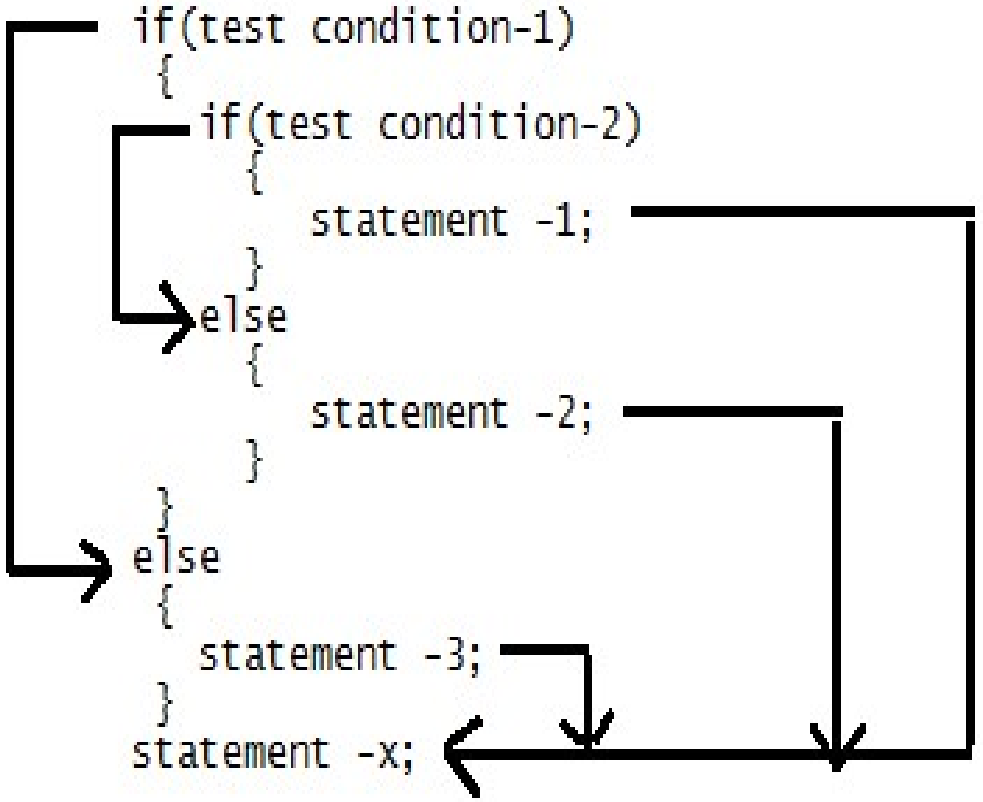
```
if ( condition )
    statement_T ;
else
    statement_F ;
```

If condition evaluates to true (a nonzero value), then $statement_T$ is executed; otherwise, $statement_F$ is executed

Example

```
if (x >= 0.0)
    printf("positive\n");
else
    printf("negative\n");
```

Nested If Statement in C



Compound Statements

- Until now we have been using only sequential flow
- A compound statement, written as a group of statements given within { and } , is used to specify sequential flow.
- {

```
    statement 1 ;
    statement 2 ;
    ...
    statement n ;
}
```

```c
#include <stdio.h>
void main()
{
    int hrs,min,amount;
    printf("Enter hours and minutes");
    scanf("%d%d",&hrs,&min);
    if (hrs>7)
    printf("Hours exceeded");
    else
    {
        if (hrs>=5)
        {
            amount+= 200;
            hrs-=5;
        }
        amount+=hrs*50;
        amount+=min;
        printf("Amount to be paid %d\n",amount);
    }

}
```

If else if ladder

Syntax :

```
if ( condition )
{
        statements;
}
else if( condition )
{
        statements;
}
else if(condition)
{
        statements;
}
else
{
        statements;
}
```

Example

```
/* increment num_pos, num_neg, or num_zero depending on x */
if (x > 0)
    num_pos = num_pos + 1;
else if (x < 0)
    num_neg = num_neg + 1;
else /* x equals 0 */
    num_zero = num_zero + 1;
```

Class of the Ship

- Each ship serial number begins with a letter indicating the class of the ship. Write a program that reads a ship's first character of serial number and displays the class of the ship.

Class ID	Ship Class
B or b	Battleship
C or c	Cruiser
D or d	Destroyer
F or f	Frigate

If else if construct

```
#include<stdio.h>
void main()
{
char c;
scanf("%c",&c);
if ((c=='b')||(c=='B'))
printf("Battleship");
else if ((c=='c')||(c=='C'))
printf("Cruiser");
else if ((c=='d')||(c=='D'))
printf("Destroyer");
else if ((c=='f')||(c=='F'))
printf("Frigate");
}
```

Switch Statement

- Useful when the selection is based on the value of a single variable or of a simple expression (called the controlling expression)
- Value of this expression may be of type int or char , but not of type double

Syntax of Switch Statement

```
switch ( control expression )
{
    label set1 :
        statements1
        break;
    label setn :
        statementsn
        break;
    default:
        statementsd
}
```

Syntax of Switch Statement

- When a match between the value of the controlling expression and a case label value is found, the statements following the case label are executed until a break statement is encountered.
- Then the rest of the switch statement is skipped.
- If no case label value matches the controlling expression, the entire switch statement body is skipped unless it contains a default label.

```c
#include<stdio.h>
void main()
{
char c;
scanf("%c",&c);
switch(c)
{
case 'b':
case 'B':
printf("Battleship");
break;
case 'c':
case 'C':
printf("Cruiser");
break;
case 'd':
case 'D':
printf("Destroyer");
break;
case 'f':
case 'F':
printf("Frigate");
}
}
```

```c
#include<stdio.h>
void main()
{
char c;
scanf("%c",&c);
switch(c)
{
case 'b':
case 'B':
printf("Battleship");
case 'c':
case 'C':
printf("Cruiser");
case 'd':
case 'D':
printf("Destroyer");
case 'f':
case 'F':
printf("Frigate");
default:
printf("No match");
}
}
```

Output

BattleshipCruiserDestroyerFrigateNo match

When input is b

While loop

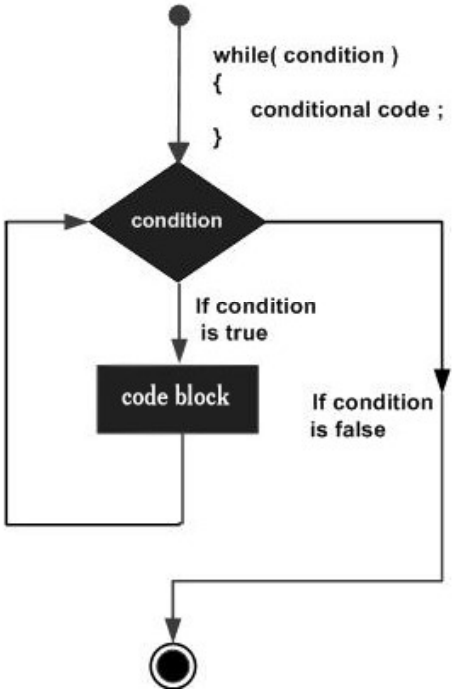
To repeat a set of statements either while a condition is met or till a condition is met

while (loop control variable < final value) . . .
Change value of loop control variable

Syntax: While Statement in C

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

Flow Diagram



•Here, **statement(s)** may be a single statement or a block of statements.

•The **condition** may be any expression, and true is any nonzero value.

• The loop iterates while the condition is true.

•When the condition becomes false, the program control passes to the line immediately following the loop.

Example:

```c
#include <stdio.h>

int main () {

   /* local variable definition */
   int a = 10;

   /* while loop execution */
   while( a < 20 ) {
      printf("value of a: %d\n", a);
      a++;
   }

   return 0;
}
```

Output:

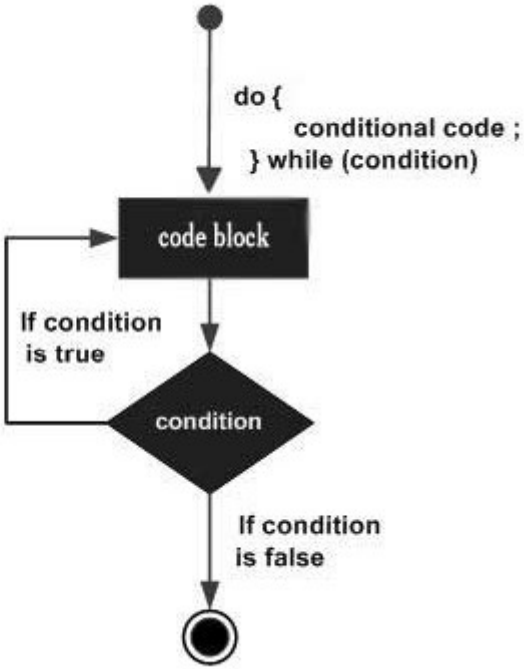
```
value of a: 10
value of a: 11
value of a: 12
value of a: 13
value of a: 14
value of a: 15
value of a: 16
value of a: 17
value of a: 18
value of a: 19
```

Do..While Loop

Syntax:

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

Flow Diagram



•A **do...while** loop is similar to a while loop, except the fact that it is guaranteed to execute at least one time.

•If the condition is true, the flow of control jumps back up to do, and the statement(s) in the loop executes again. This process repeats until the given condition becomes false.

•Condition is checked at the end of execution

Example:

```c
#include <stdio.h>

int main () {

   /* local variable definition */
   int a = 10;

   /* do loop execution */
   do {
      printf("value of a: %d\n", a);
      a = a + 1;
   }while( a < 20 );

   return 0;
}
```

Output:

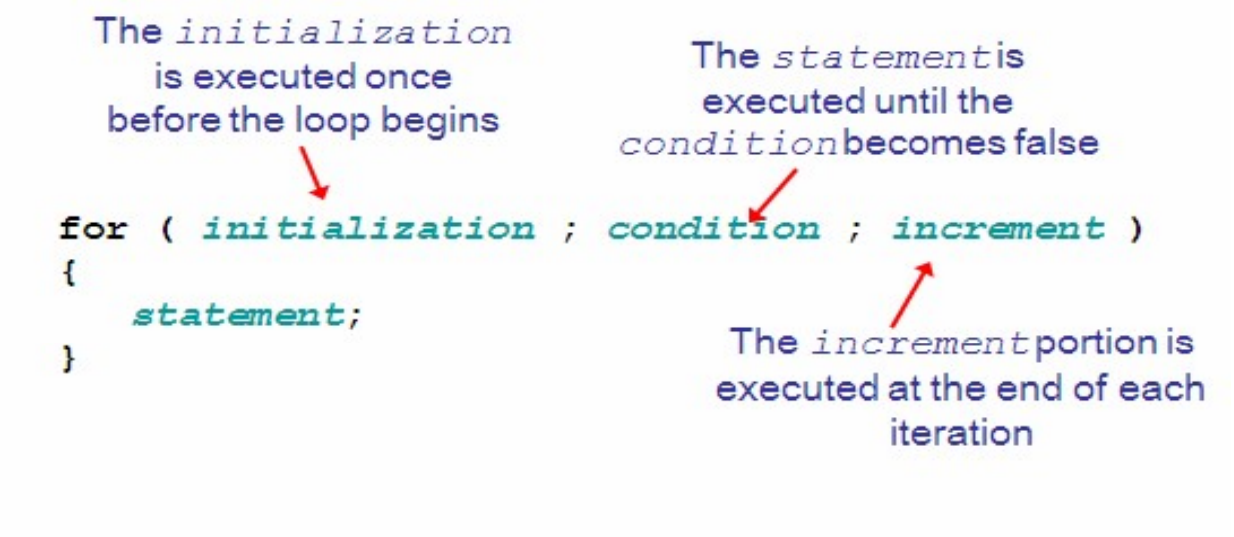
```
value of a: 10
value of a: 11
value of a: 12
value of a: 13
value of a: 14
value of a: 15
value of a: 16
value of a: 17
value of a: 18
value of a: 19
```

Syntax of a For loop

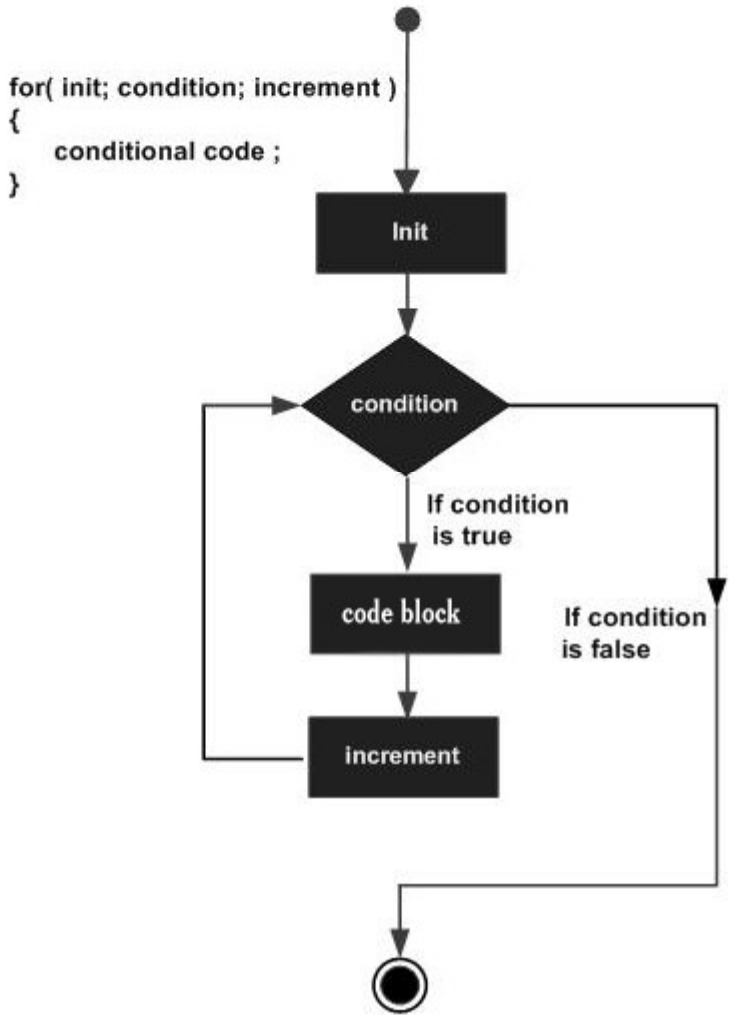
for (loop control variable initialization; loop terminating condition; loop control variable update)

-All three components are optional
-But semicolons are mandatory

For Statement in C



Flow Diagram



Example:

```c
#include <stdio.h>

int main () {

   int a;

   /* for loop execution */
   for( a = 10; a < 20; a = a + 1 ){
      printf("value of a: %d\n", a);
   }

   return 0;
}
```

Output:

```
value of a: 10
value of a: 11
value of a: 12
value of a: 13
value of a: 14
value of a: 15
value of a: 16
value of a: 17
value of a: 18
value of a: 19
```

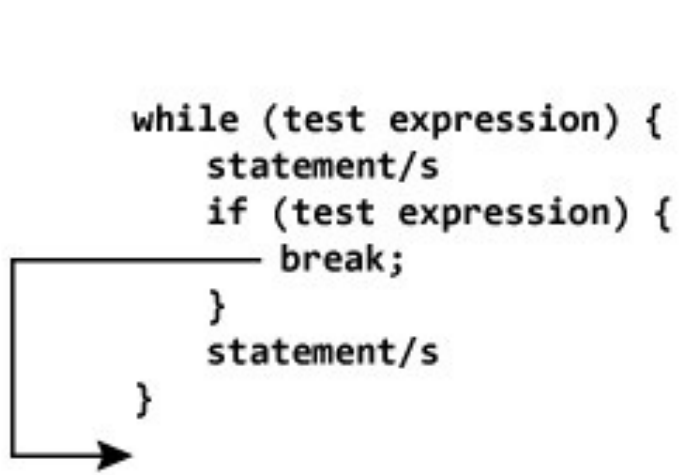
```c
#include<stdio.h>
void main()
{
int i = 0;
for(;i<10;)
{
printf("Hello");
i++;
}
}
```

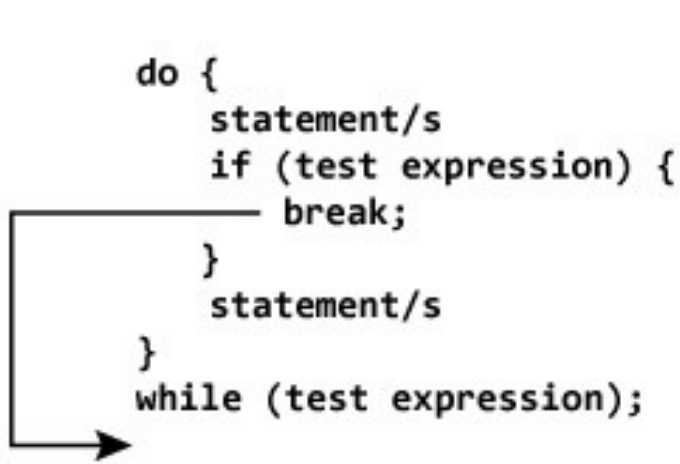
```c
#include<stdio.h>
void main()
{
for(;;)
printf("Hello");
}
```

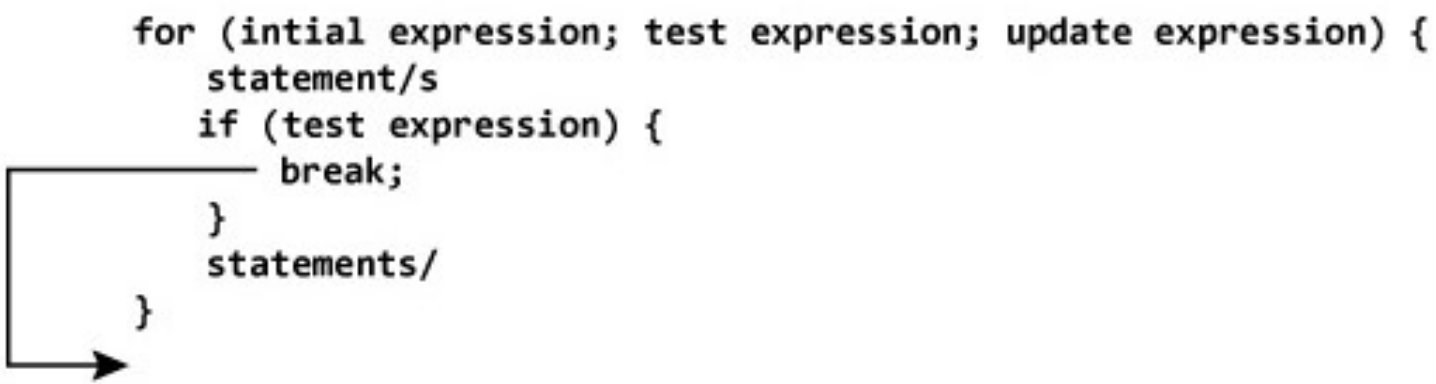
```c
#include<stdio.h>
void main()
{
int i = 0;
for()
{
printf("Hello");
i++;
}
}
```

Break and Continue Statements

- Interrupt iterative flow of control in loops
- Break causes a loop to end
- Continue stops the current iteration and begin the next iteration


while (test expression) {
statement/s
if (test expression) {
break;
}
statement/s
}


do {
statement/s
if (test expression) {
break;
}
statement/s
}
while (test expression);


for (intial expression; test expression; update expression) {
statement/s
if (test expression) {
break;
}
statements/
}

```c
//Program to count non digits
#include<stdio.h>
#define MAX 10
void main()
{
int counter,non_Digits=0;
char ch;
for(counter=0;counter<MAX;counter++)
{
        //Read a character
        scanf("%c\n",&ch);
        //Check if the character is not digit
        if(isdigit(ch))
        {
        //Not a digit continue to read next character
        continue;
        }
        //If it is not a digit then increment the counter for non_Digits
        else
        non_Digits++;
}
printf("%d",non_Digits);
}
```