Control Statements - Loop

Gagan Deep

Founder & Director

Rozy Computech Services

Kurukshetra, rozygag@yahoo.com

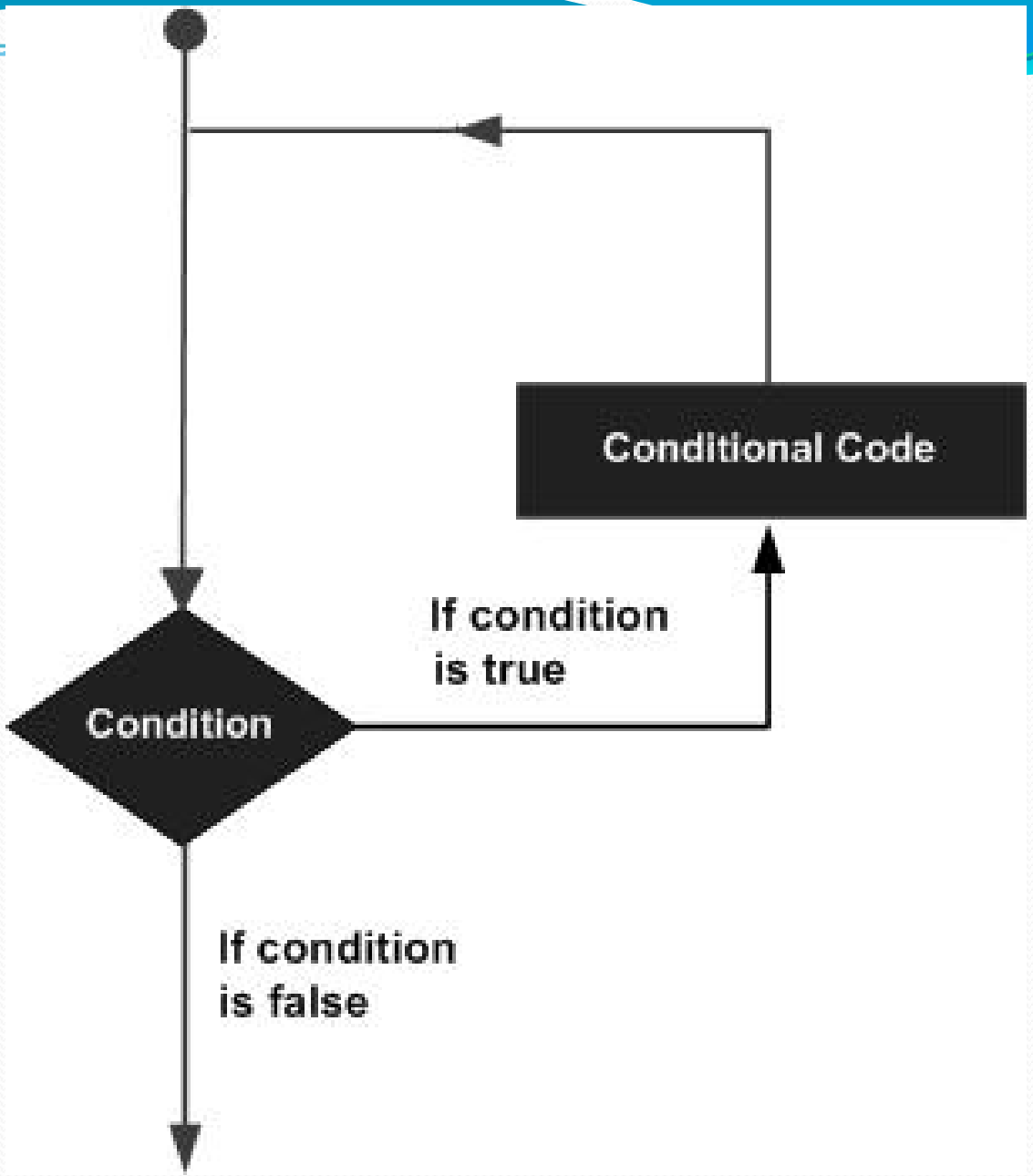
www.rozyph.com

Looping

- Suppose we want to display hello on output screen five times in five different lines.
- We might think of writing either five printf statements or one printf statement consisting of constant "hello\n" five times.
- What if we want to display hello 500 times?
- Should we write 500 printf statement or equivalent ?
- Obviously not.
- It means that we need some programming facility to repeat certain works.
- Such facility in Programming Languages is available in the form Looping Statements.

Looping

- Loops are used to repeat something.
- Repetition of block of code.
- If program requires that group of instructions be executed repeatedly is known as looping.
- Looping is of two types
  - Conditional
  - Unconditional
- Conditional Looping: Computation continues indefinitely until the logical condition is true.
- Unconditional Looping : The number of repetition known in advance or repetition is infinite.


Conditional Code
If condition
is true
Condition
If condition
is false

Types of Loops

- Conditional Loops
  - While Loop – Pre Condition Loop – Entry Time Conditional Loop
  - Do-While Loop – Post Condition Loop – Exit Time Conditional Loop
- Unconditional Loop
  - For Loop – Counted Loops
  - Uncounted Loop – Infinite loop
  - Uncounted Loop –infinite loop controlled by control statements.

while Statement

- The while statement is used to carry out looping operations, in which a group of statement is executed repeatedly, until some condition has been satisfied.
- The general form of statement is

```
while (test expression)
statement to be executed;
or
while (test expression)
{ statements to be executed }
```

- The statement will be executed repeatedly as long as the test expression is true.
- The statement can be simple or compound, though it is usually a compound statement.
- It must include some features that eventually alters the value of the expression, thus providing stopping condition for the loop.
- However, an empty condition is not legal for a while loop as it is with a for loop(discussed in next slides).

Program 1-4 First 10 Natural Numbers

```
#include <stdio.h>
void main()
{ int i=0;
while(i<=9) // i<10
printf("%d", i++);
}/*here we can't use ++i – that gives different result*/
//only a statement
```

```
#include <stdio.h>
void main()
{ int i=0;
while(i<=9) // i<10
{ printf("%d\n", i);
i++; }//compound statement
} //we also can use ++i
```

```
#include <stdio.h>
void main()
{ int i=10;
while(i<=9) // i<10
{ printf("%d", i);
i++; }
// doesn't work body of loop
}
```

```
#include <stdio.h>
void main()
{ int i=9;
while(i>=0) // i>-1
{ printf("%d", i);
i--; } //--i
}//decrement – decreasing order
```

Program 5-8

```
#include<stdio.h>
void main()
{ int n=10;
 while(n>1);
//; at the end of loop
// while(n>1){ };
 printf("HELLO"); }
//It won't Print anything
```

```
#include<stdio.h>
void main()
{ while(1) //non Zero - True
printf("HELLO"); }

//Infinite Time "HELLO"
word
```

```
#include <stdio.h>
void main()
{ int i=0;
while(i<=9)
printf("%d", i); }
//infinite loop
```

```
#include<stdio.h
 void main()
{ while('C')
//ASCII Value is Non zero
printf("HELLO");  }
//Infinite Time "HELLO"
word
```

Program 9-12

```
#include <stdio.h>
void main()
{ int i=0;
while( )
//invalid statement
printf("%d", i); }
```

```
#include <stdio.h>
void main()
{ int i=1,sum=0;
while(i<=10) // i<10
{sum+=i;
i++; } printf("%d", sum); }
```

```
#include <stdio.h>
void main()
{ int i=1,p=1;
while(i<=10)
{p*=i;
++i;}
printf("%d", p); }
//gives garbage value
```

```
#include <stdio.h>
void main()
{ int i=1; long int p=1;
while(i<=9)
{p*=i;
i++;  }
 printf("Product  is=%ld",  p);
}
```

Program 13-16

```
#include <stdio.h>
void main()
{ int i=1, n; long int fact=1
scanf("%d", &n);
while(i<=n) / i<10
{fact=fact*i;
i++; } //Factorial of n
printf("factorial is %ld", fact);}
```

```
#include <stdio.h>
void main()
{ int i=0;
while(i<=9) // i<10
{scanf("&d",&a) ;
sum+=a;
i++; } printf("%d", sum);
}//sum of 10 numbers
```

```
#include <stdio.h>
void main()
{ int i=0,sum=0,a;
while(i<=9)
{ scanf("&d",&a) ;
sum+=a; } //infinte loop
printf("%d", sum); }
//never terminate, i = 0
```

```
#include <stdio.h>
void main()
{ int i=0,sum=0,a;
while(i<=9)
{scanf("&d",&a) ;
if(a<0)break;//break whn a<0
sum+=a; } /*infinite, but breaks
when you want*/
printf("%d", sum); }
```

do.....while Statement

- When a loop is constructed using the while statement, the test for continuation of the loop carried out at the beginning of each pass.
- Sometimes, however it is desirable to have a loop with the test for continuation at the end the end of each pass. This can be accomplished by do..while statement. The general form of statement is

```
do     Statement;   //single statement
while (expression);    or
do
{ statements to be executed }
while (expression);
```

Program 17-20 First 10 Natural Numbers

```
#include <stdio.h>
void main()
{ int i=0;
do
printf("%d", i++);
while(i<=9);      }
```

```
#include <stdio.h>
void main()
{ int i=0;
do
{printf("%d\n", i);
i++; }
while(i<=9) ;   }
```

```
#include <stdio.h>
void main()
{ int i=10;
do  //first execute then check
{ printf("%d", i);
i++; } while(i<=9) ;    }
// prints 10(execute one time)
```

```
#include <stdio.h>
void main()
{ int i=9;
do{ printf("%d", i);
i--; }
while(i>=0) ;  }
```

for Statement

- The for statement is the third and perhaps the most commonly used looping statement in C.
- This statement includes an expression

  that specifies an initial value for an index,
- another expression

  that determines whether or not the loop is continued,
- and a third expression

  that allows the index to be modified at the end of each pass.
- The general form of statement is

  for (expr1; expr2; expr3)

  statement; or { statements }

for Statement

- First expr1 is used to initialize some parameter (called an index) that controls the looping action. Typically expr1 is an assignment statement.
- 2nd expr2 represents a condition that must be true for the loop to continue execution and executed at the beginning of each pass. Expr2 is a logical expression.
- 3rd expr3 is used to alter the value of the parameter initially assigned by expr1 and executed at the end of each pass. Expr3 is a unary expression or an assignment expression.

for(initial;condition/logical; increment/decrement)

Print 1 to 10

Program 21-24

```
#include<stdio.h>
void main()
{int i;
for(i=1;i<=10;i++)
printf("%d",i); }
```

```
#include<stdio.h>
void main()
{
for(int i=1;i<=10;++i)
printf("%d",i); }
```

```
#include<stdio.h>
void main()
{
for(int i=1;i<=10;i=i+1)
printf("%d",i);
}
```

```
#include<stdio.h>
void main()
{int i=1;
for(  ; i<=10;    )
 { printf("%d",i);
i++ ; }
}//you can also use ++i
```

for Statement – Infinite loop

- A loop becomes infinite loop if a condition never becomes false. The **for** loop is traditionally used for this purpose. Since none of the three expressions that form the for loop are required, you can make an endless loop by leaving the conditional expression empty.

```
#include <stdio.h>
void main ()
{   for( ; ; )  { printf("This loop will run forever.\n"); }   }
```

- When the conditional expression is absent, it is assumed to be true. You may have an initialization and increment expression, but C programmers more commonly use the for(;;) construct to signify an infinite loop.
- However, in while and do..while statements you cant skip condition that is invalid statement.

Program 25-28

```
#include<stdio.h>
void main()
{for(int i=0;i<5;i++)
{ }
} // i goes to 5 only
```

```
#include<stdio.h>
void main()
{
for(i=0;i<5;i++);
}  // i goes to 5 only
```

Comma Operator

```
#include<stdio.h>
void main()
{
for(i=0,j=0;i<5;i++, j++) {
statement1; statement2;
statement3; }
```

```
#include<stdio.h>
void main()
{int i = 0;
for( ; ;)
{ statements;
if(breaking condition)
break;  i++; }
```

Reverse Number

N=1234 => RN =4321 ?

RN=0

R=N%10

N=N/10

RN=RN*10+R

Pass 1

R=4

N=123

RN=4

Pass 2

R=3

N=12

RN=43

Pass 3

R=2

N=1

RN=432

Pass 4

R=1

N=0

RN=4321

Our passes runs until N=0

Hence our required revere Number Comes RN=4321

Program 29-30

Reverse Number

```
#include<stdio.h>
void main()
{int n, r, rn=0;
scanf("%d", &n);
while (n!=0) {
r=n%10;     n=n/10 ;
rn=rn*10+r; }
printf("Reverese     Number
=%d", rn);
}
```

Palindrome Number

```
#include<stdio.h>
void main()
{int n, r, T, rn=0;
scanf("%d", &n);   T=n;
while (T!=0) {
r=T%10;     T=T/10 ;
rn=rn*10+r; }
if (rn==n)
printf("Palindrome Number);
else
printf("Non-Palindrome
Number);
}
```

Program 31-32

Count Number of Digits in a Number.

```
#include<stdio.h>
void main()
{n,  count=0;
scanf("%d", &n);
while (n!=0) {
n=n/10 ;
count++ }
printf("No. of Digits =%d",
count);
}
```

Sum of digits of a number

```
#include<stdio.h>
void main()
{n, r, sum=0,count=0;
scanf("%d", &n);
while (n!=0) {
r=n%10;
n=n/10 ;
sum=sum+r;
count++ }
printf("Sum of Digits =%d &
Number of digits = %d", sum,
count); }
```

Program 33

Prime Number

```
#include<stdio.h>
void main()
{int n,  i=2;
scanf("%d", &n);
while (n % i != 0)
i++;
if  (n==i)
 printf("%d is prime", n);
else
 printf("%d is non-prime", n);

}
```

Do Yourself

- Count even and odd digits of a number
- Sum of even and odd digits of a number
- Check for Armstrong Number
- Fibonacci Sequence

Next Lecture

- Nested Loops
- Control Statements

THANKS!

If you have any queries you can contact me at :

rozygag@yahoo.com