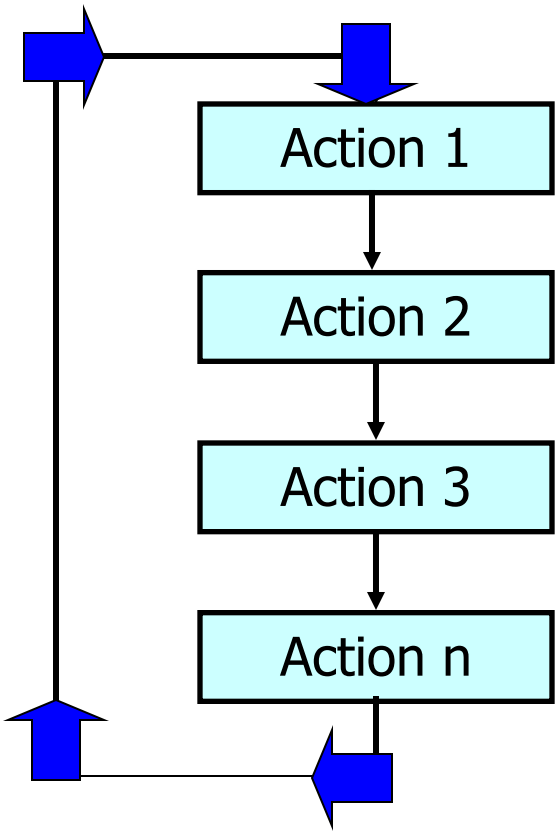
6.REPETITION

The real power of computer is their ability to repeat an operation or a sequence of operations many times. This operation is known as a looping operation. A repetition structure is used to perform any sequence of actions continuously.

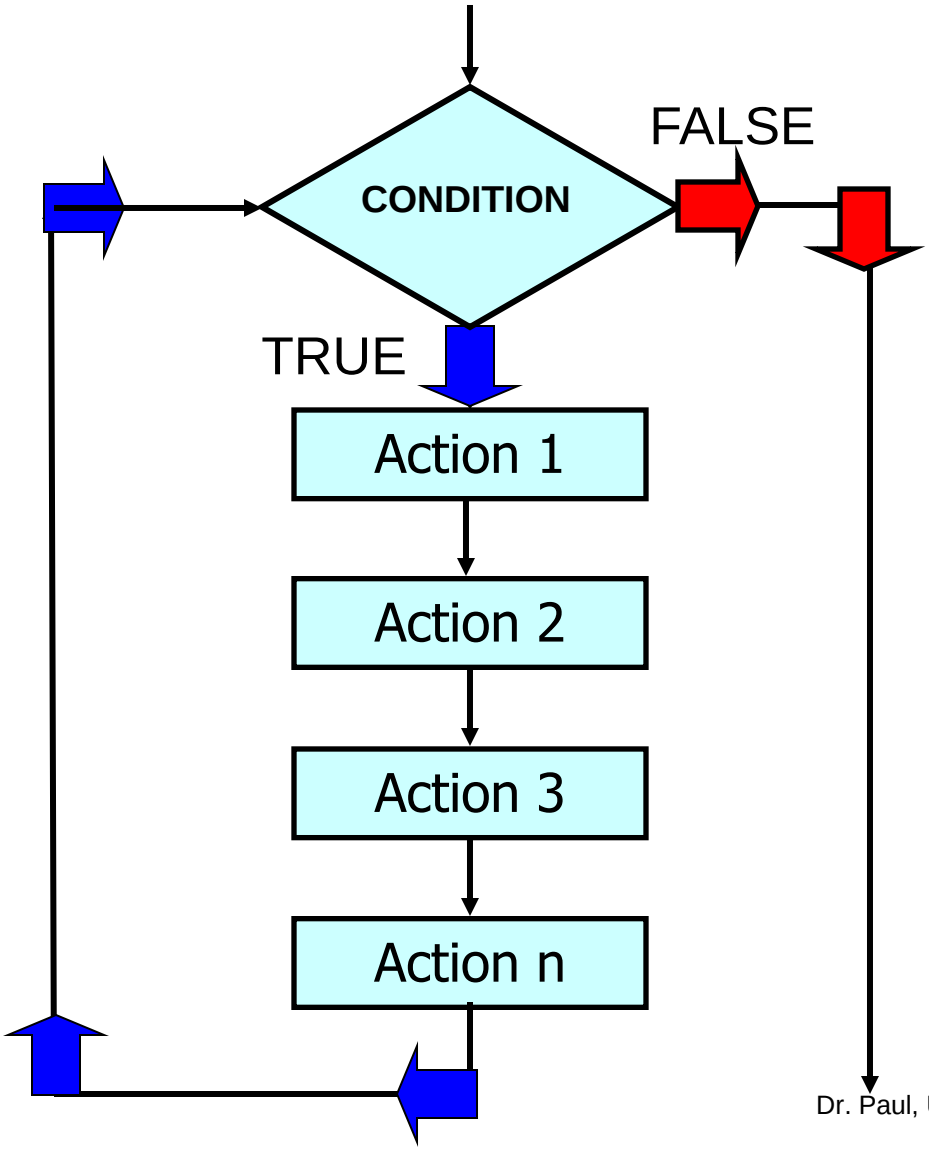


How to stop this infinite action?

How to come out of the loop when all the actions are performed?

We should have a condition that controls the loop and we must check the condition.

How to stop an infinite loop operation? Loops may be classified as pretest and post-test loops.

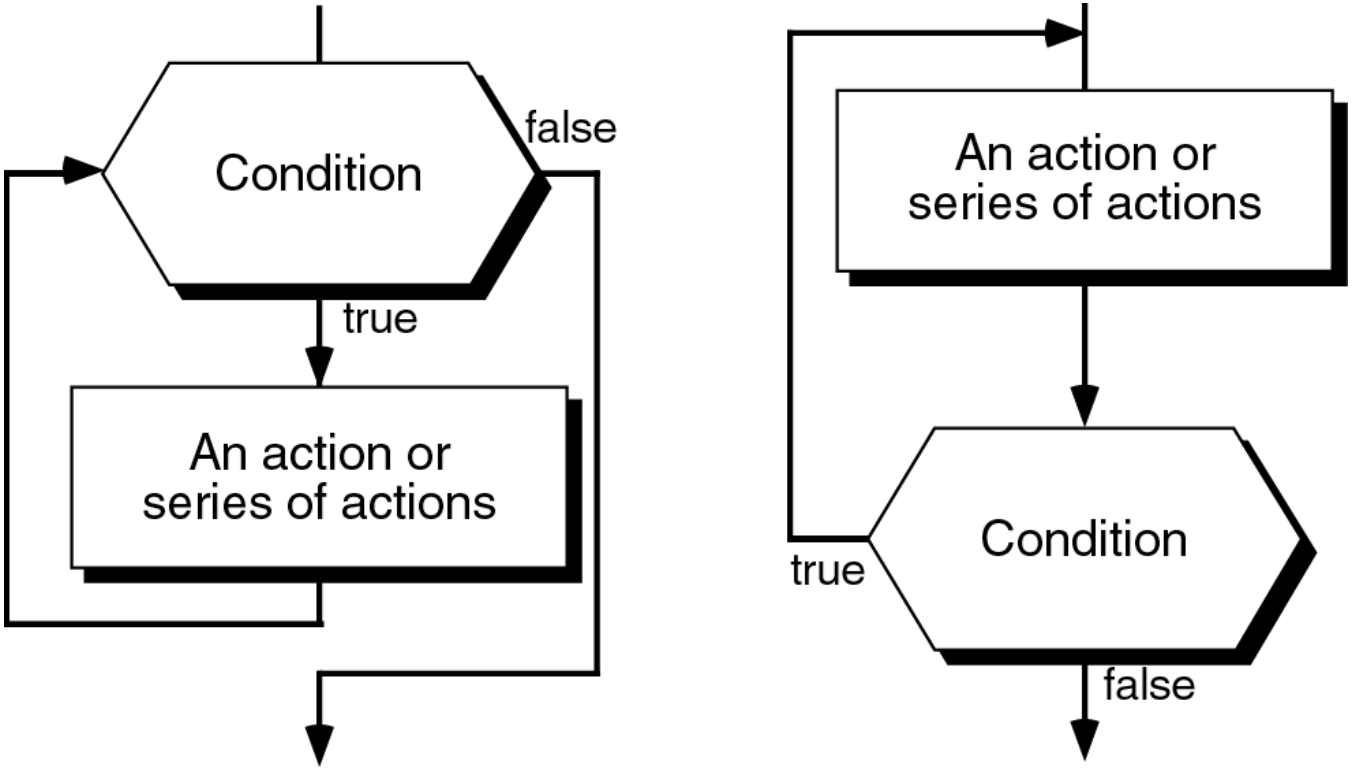


Types of loops:

Pretest loop
Post-test loop.

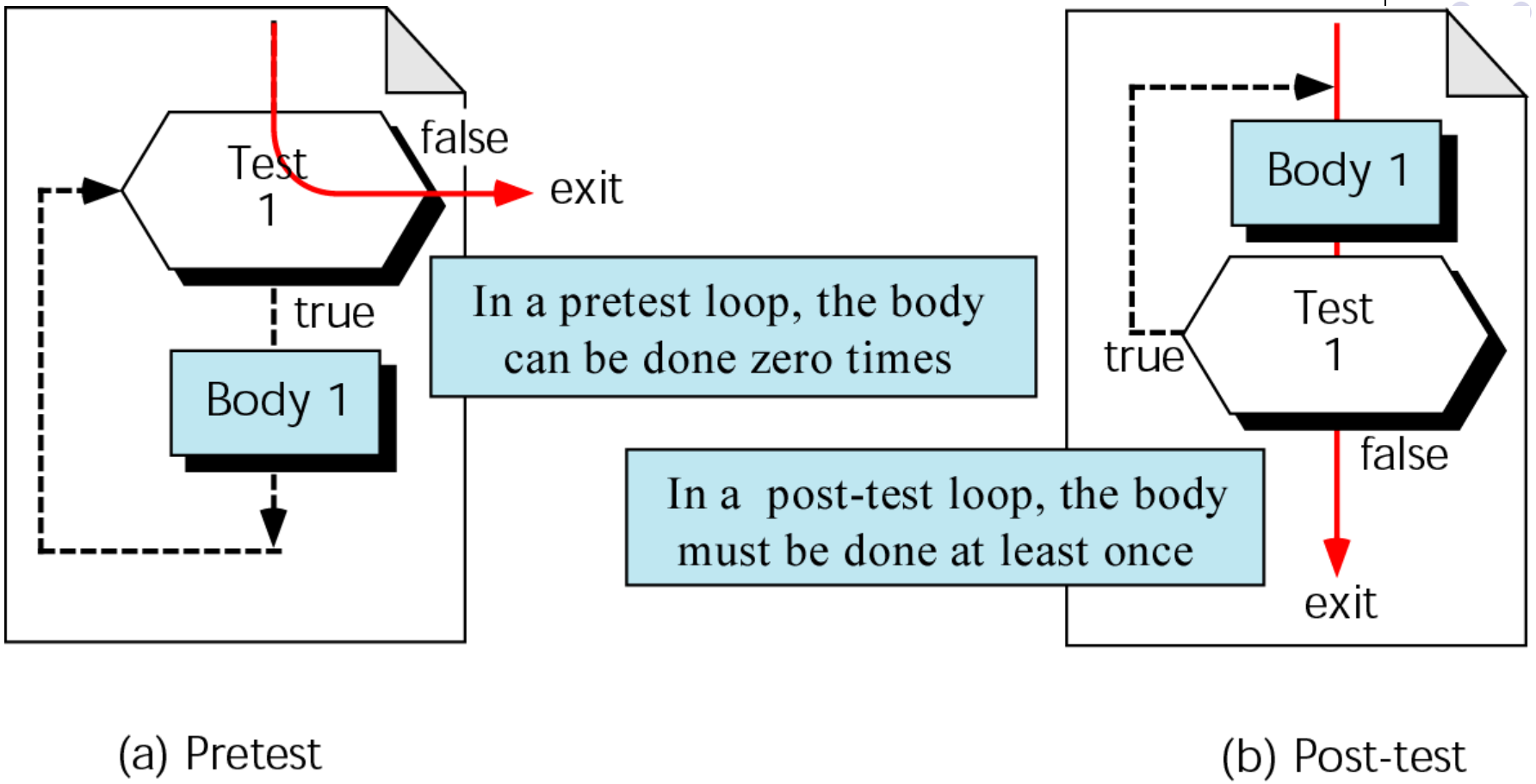
In a pretest loop, the loop control expression is tested first. If it is true then the loop actions are executed; if it is false the loop is terminated.

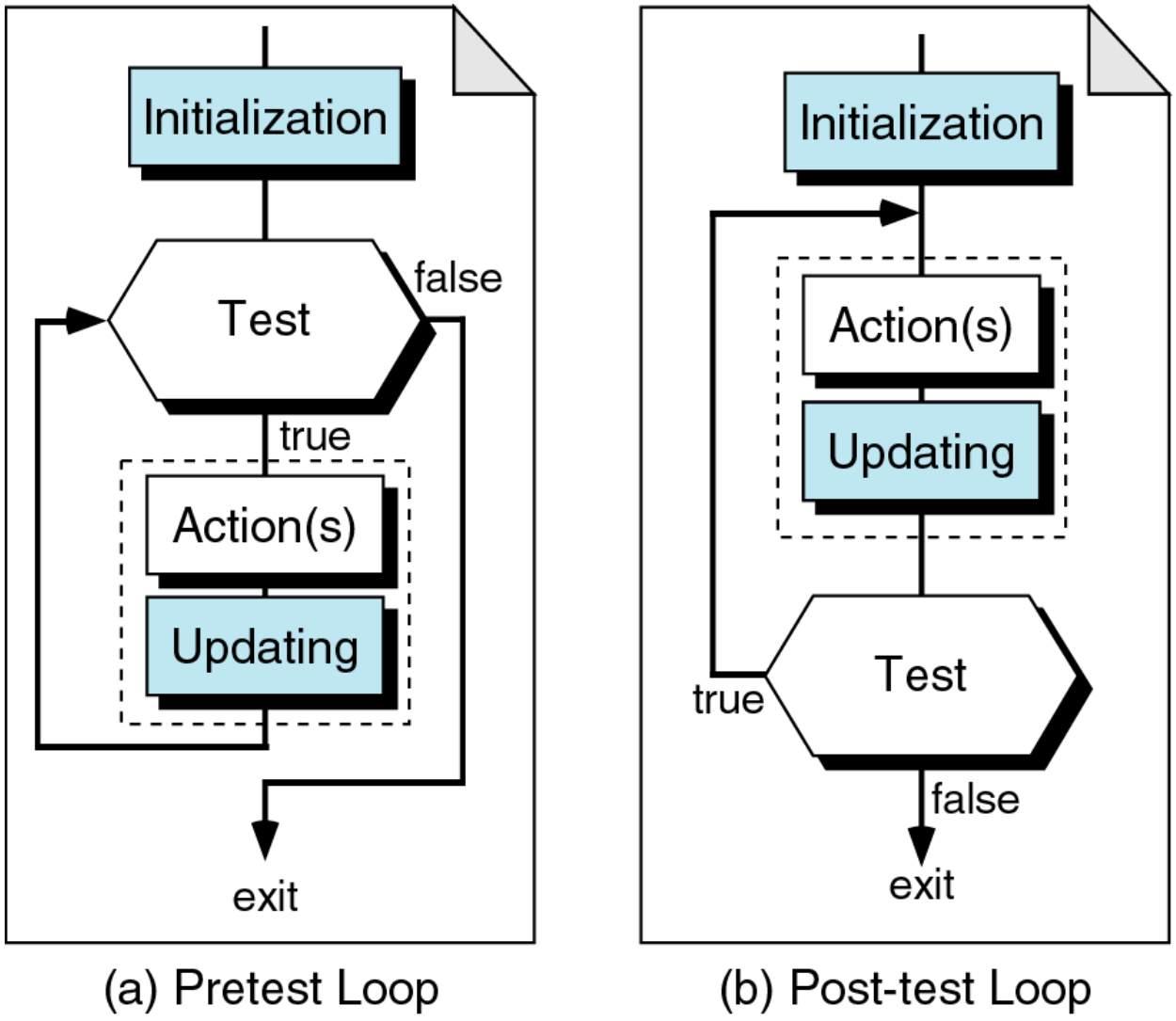
In a post-test loop, the loop actions are executed first. Then the loop control expression is tested. If it is true, a new iteration is started; otherwise, the loop terminates.



(a) Pretest Loop

(b) Post-test Loop


Test
1
false
exit
true
Body 1
In a pretest loop, the body
can be done zero times
In a post-test loop, the body
must be done at least once
Body 1
Test
1
true
false
exit
(a) Pretest
(b) Post-test

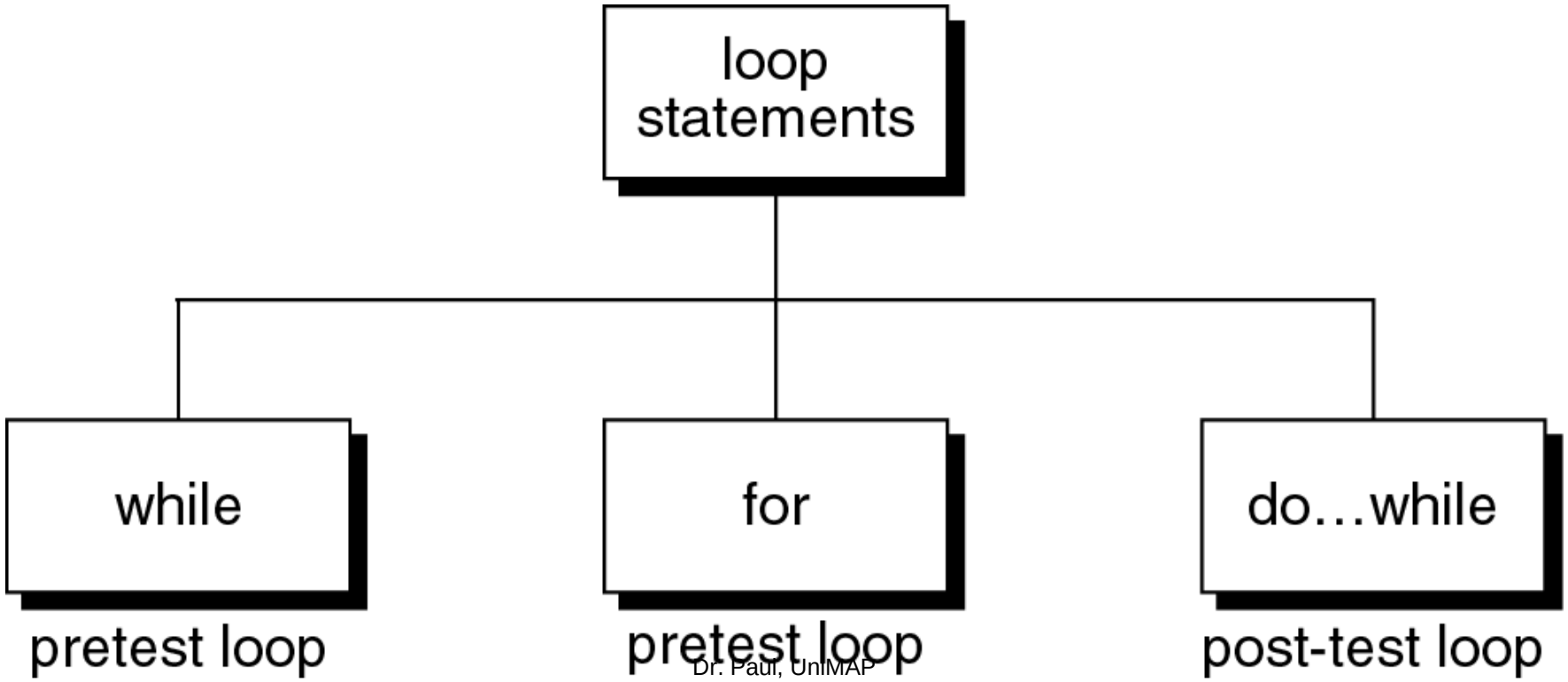


(a) Pretest Loop

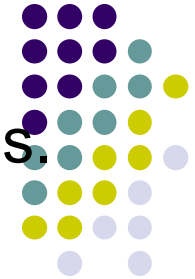
(b) Post-test Loop

‘C’ Language has three loop statements:

for statement	[Pretest Loop]
while statement	[Pretest Loop]
do while statement	[Post-Test Loop]



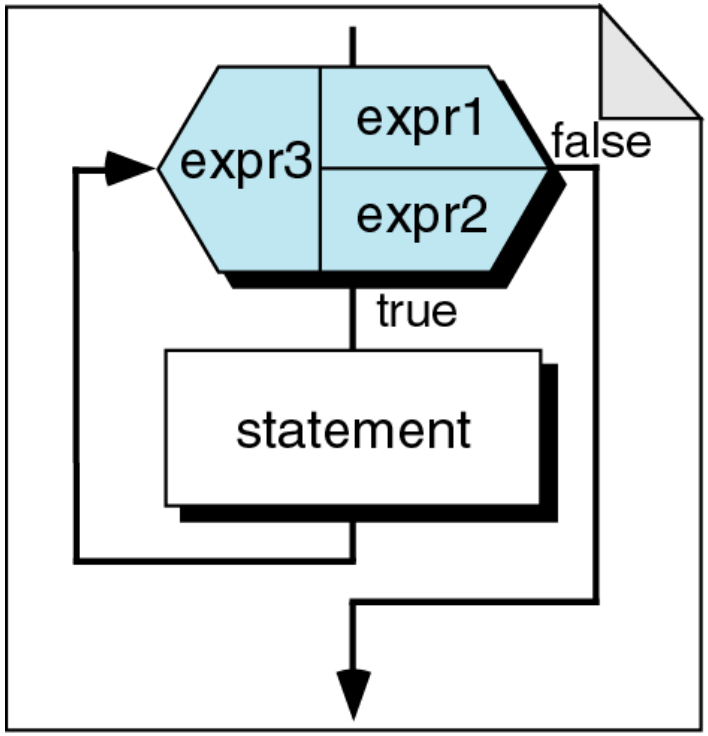
FOR STATEMENT



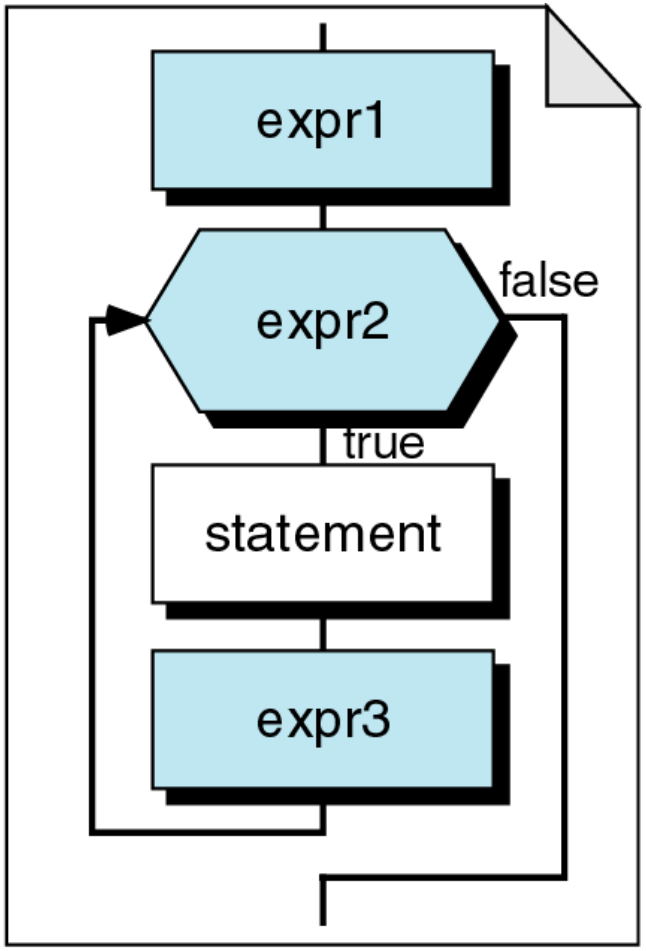
The ‘for’ loop is a pretest loop that uses three expressions.
The general form of ‘for’ statement is:

```
for   ( expression1;
                    expression2;
                                  expression3)
Statement;
```

Semicolons separate the three expressions.
Expression-1 is used to initialize the index parameter that controls the looping action.
Expression-2 represents a condition that must be satisfied for the loop to continue execution.
Expression-3 is used to alter the value of the parameter initially assigned by expression-1.

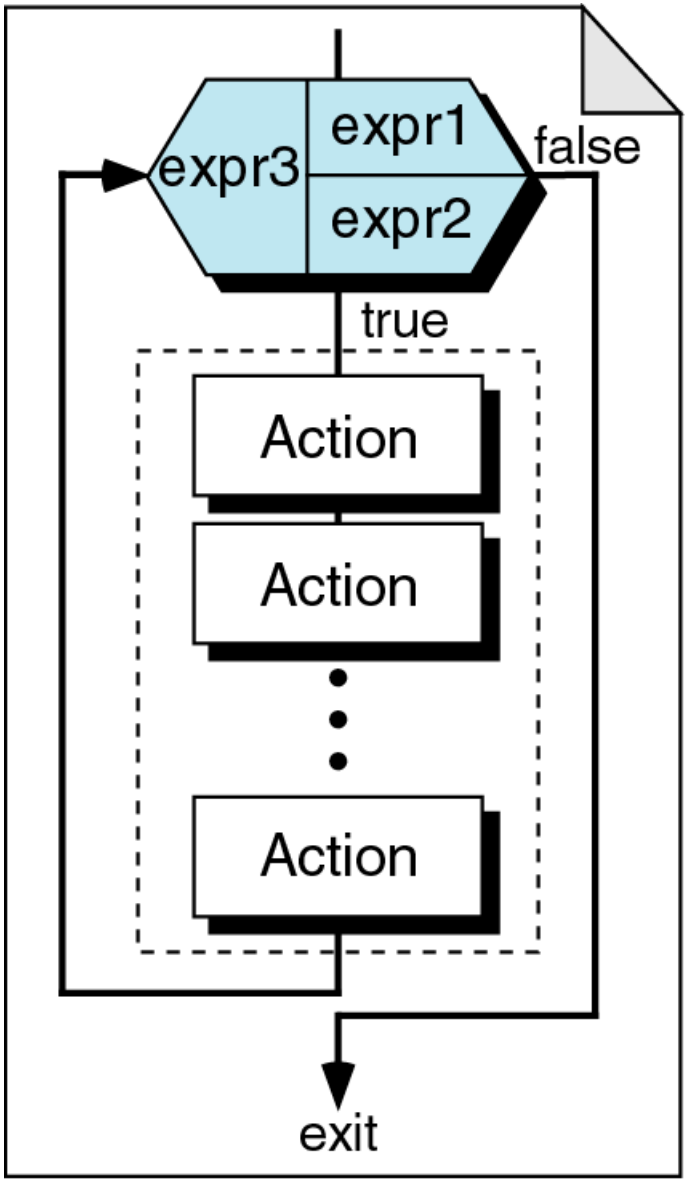


(a) Flowchart



(b) Expanded Flowchart

```
for (expr1; expr2; expr3)
      statement
```



(a) Flowchart

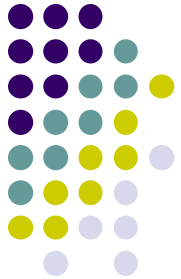
```
for (expr1;
     expr2;
     expr3)
  {
    Action
    Action
    Action
  } /* for */
```

(b) C Language

- The expression-1 is evaluated only once at the first time. Then expression-2 is evaluated. If the expression is-2 is true then the statement will be executed.

- Then the control transfers to the expression-3 and expression-3 is executed and transfers the control to expression-2. These operations are repeated as long as the expression-2 is evaluated to true. Once the expression-2 is evaluated to false, the control will come out of the loop.

- The statement in the for loop may be a simple statement or a complex statement or it may be another for statement.



Example 1:

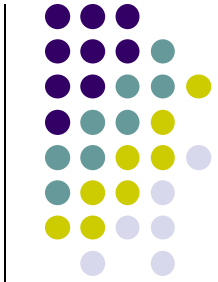
```
int  i;
for( i=1; i<=10; i++)
        printf("Good Morning\n");
```

Example 2:

```
int index;
for(index = 1; index <=10; ++index)
                    printf("%d\n", index);
```

Example 3:

```
/* program to find the sum of n numbers*/
#include<stdio.h>
void main(void) {
float sum = 0.0;
float num;
int index;
int n;
printf(“Enter number of data “);
scanf(“%d”,&n);
for(index = 1; index <=n; ++index)
        {
                scanf(“%f”,&num);
                sum += num;
        }
printf(“The sum = %f\n”,sum);
printf(“The average = %f”,sum\n);
return; }
```



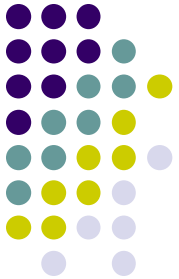
Example 4:

/* program to read 10 numbers and find how many numbers are greater than 10 but less than 100 */

```
#include<stdio.h>
void main(void) {
int count10 = 0;
int index; int number;
for(index = 1;index <=10;index++)
        {
                scanf("%d",&number);
                if(number > 10 && number < 100)
                        ++count10;
        }
printf("Numbers within the range = %d\n",count10);
return; }
```

Example 5:

```
#include<stdio.h>
void main(void) { float num;
int i, sumodd, sumeven, noodd, noeven;
sumodd = 0;  sumeven = 0; noodd = 0;  noeven = 0;
for(i=1;i<=100;i++)
        {
        scanf("%d",&num);
        if(num%2 == 0)
                {
                sumeven+=num;
                ++noeven;
                }
        else
                {
                sumodd+=num;
                ++noodd;
                }
        }
printf("Sum even = %d\n",sumeven);
printf("Sum odd = %d\n",sumodd);
printf("no of even = %d\n",noeven);
printf("no of odd = %d\n",noodd);
return;   }
```



Example 6:

/*program to find the largest number */

```
#include<stdio.h>
#define nsize 10
void main(void)
{
int index;
float number,big;
scanf("%f", &big);
for(index = 1;index <= nsize - 1; index ++)
{
        scanf("%f",&number);
        if(number > big) big = number;
        }
printf("The largest number is %f\n",big);
return; }
```

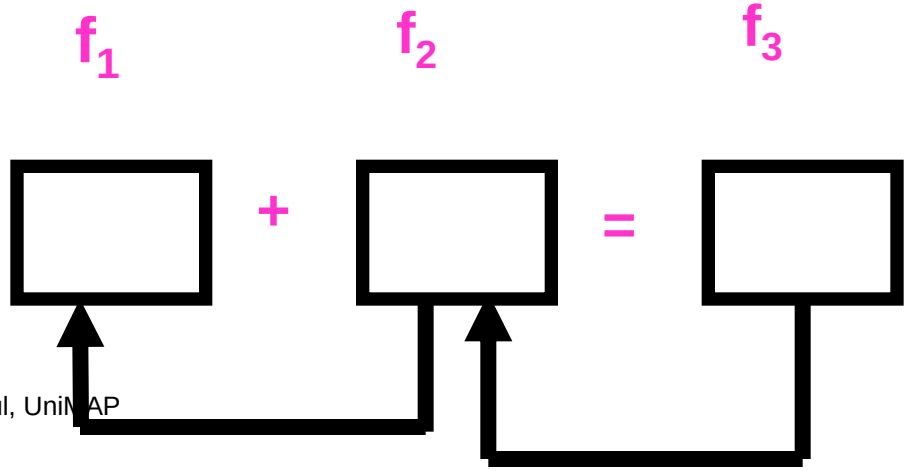
Note: The nsize can be redefined for any number of inputs.

Example 7: [Fibonacci Series]

Given F1 = F2 = 1; F3 = F2 + F1; F4 = F3 + F2; Fn = Fn-1 + Fn-2

Generate the sequence 1,1,2,3,5,8,...

```
#include<stdio.h>
#define term 10
void main(void)
{
int f1,f2,f3,index;
f1 = 1;
f2 = 1;
printf("%d\t%d\t",f1,f2);
for(index = 1;index <= term –2; index ++)
        {       f3 = f2+f1;
                printf("%d\t",f3);
                f1 = f2;
                f2 = f3;
 }
return;
}
```



Example 8: What is the output?

```
for(i=1; i<=5;  printf("%d\t",i))
i++;
```

Result :2 3 4 5 6

Example 9: What is the output?

```
for(i=1;i<=5;i++);
        printf("%d\t",i);
printf("%d\n",i);
```

Result : 6

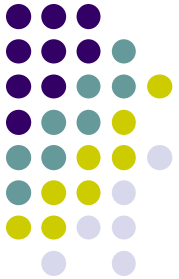
Example 10: What is the output?

```
j  = 5;
for(;j<=10;)    {
printf("%d\t",j);
j+=5;}
```

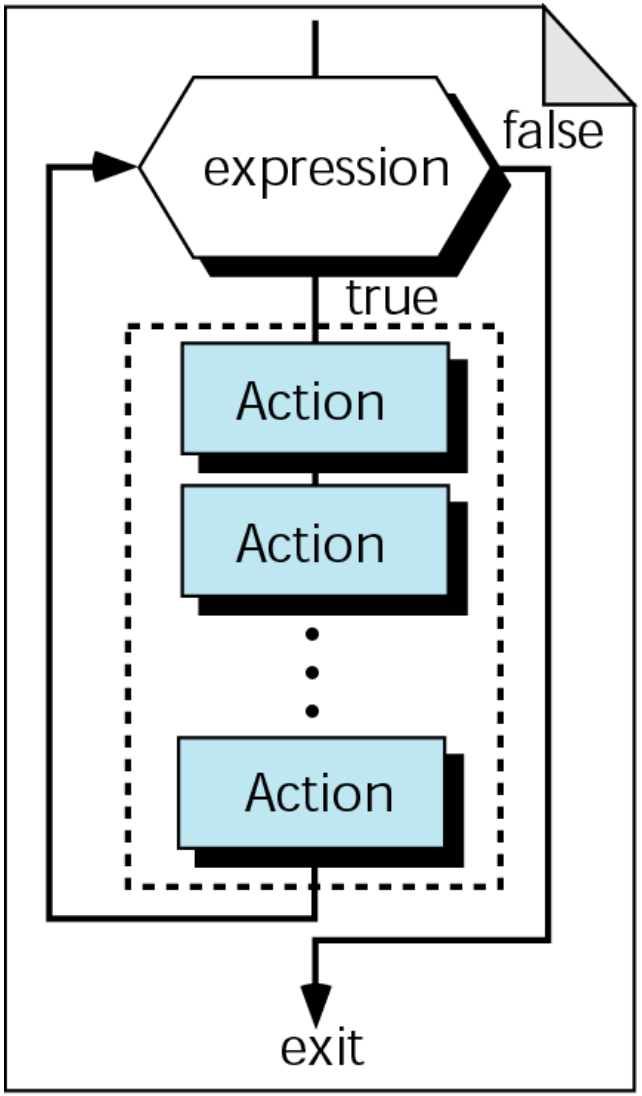
Result: 5 10

Example 11:

```
/* program in C to find the value of phi */
#include<stdio.h>
#include<math.h>
#define nmax 100
void main(void)
{
        int i;
        float sum,term, eps= 0.000001;
        sum = 1.0;
        for(i=2;i<=namx;i++)
                {
                term = 1.0/(float) (i*i);
                if(term < eps)
                        break;
                else
                sum += term;
                }
        sum = sqrt(sum);
        printf("The sum is %f\n",sum);
        return;
}
```



WHILE STATEMENT



(a) Flowchart

```
while ( expression )
  {
        Action
        Action
          .
          .
          .
        Action
  }  /* while */
```

(b) C Language

The syntax of a while statement is:

```
while (expression)
statement;
```

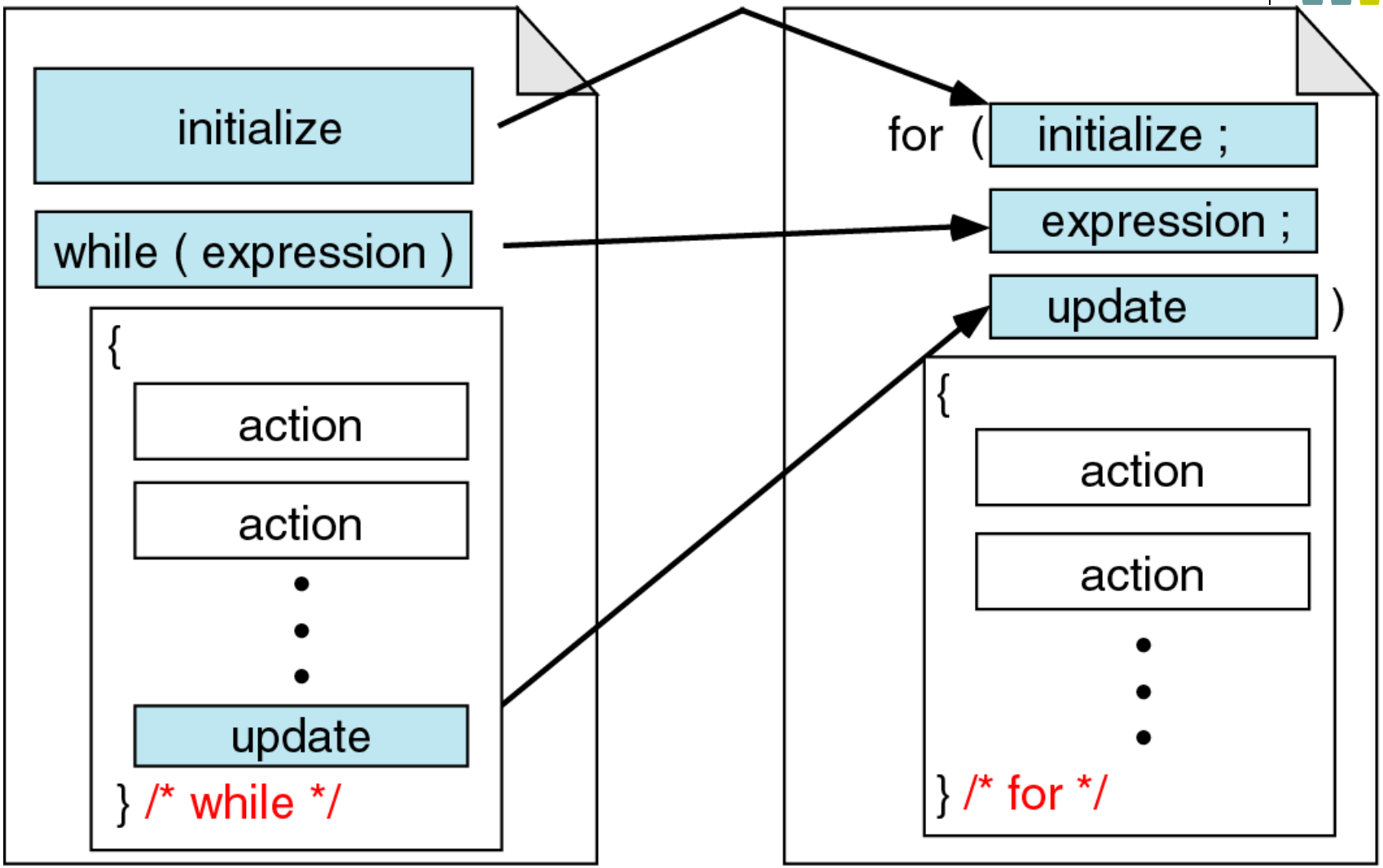
❖The statement may be a simple statement or a compound statement.

❖The statement will be repeatedly executed as long as the expression is true.

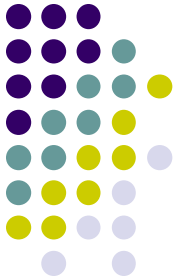
❖At first the expression will be evaluated. If the expression is true then the statement will be executed and control will return to the top of the loop and the process is repeated. But if any time when the expression becomes false then the statement will be skipped and control will transfer to the statement immediately after the while statement.

Note: The expression must be enclosed in braces.

Comparison of a 'while' and a 'for' statement



Comparison of a ‘while’ and a ‘do-while’ statement



Pretest
nothing prints

```
while (0)
  {
   printf("Hello World");
  } /* while */
```

```
do
  {
   printf("Hello World");
  } while (0)
```

Posttest
“Hello...” prints

Example 12: What is the output?

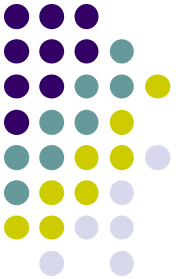
```
int x;
x = 7;                          /* expr initialization */
while (x >=0)                   /* expr evaluation */
        {
        printf("%d\t",x);
        x = x – 2;              /* expr updation */
        }
```

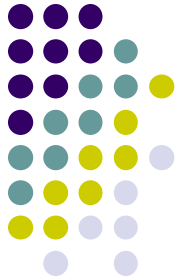
Result : 7 5 3 1

Example 13: What is the output?

```
int x;
x =7;
while(x >=0)
{
        x = x –2;
        printf("%d\t",x);
}
```

Result : 5 3 1 -1





Example 14: What is the output?

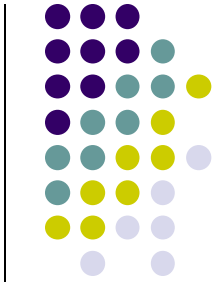
```
int x; x = 7;
while(x>=0)
        x = x-2;
        printf("%d\n",x);
```

Result: -1

Example 15: What is the output?

```
int  x;
x = 1;
while(x==1)
{
        x = x-1;
        printf("%d\t",x);
}
```

Result: 0



Example 16: What is the output?

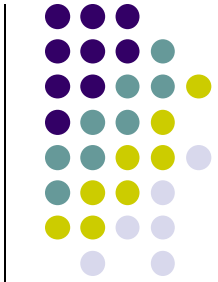
```
int x;
x = 1;
while(x ==1)  x = x-1;
printf("%d\t",x);
```

Result : 0

Example 17: Find the error.

```
while(x>0) do
        x *=1;
printf("%d\n",x);
```

The error : "do" is not a key word.



Example 18:

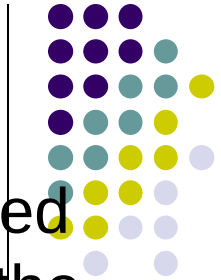
The following while loop executes indefinitely. (Why?)

```
while(1)
        printf("This is an infinite loop\n");
while('c')
        printf("Wish u a happy New year\n");
while(- 3.2)
        printf("How are you?\n");
```

Example 19:

The body of the following while statement will not be executed.

```
count = 0;
while(count)
printf("Tomorrow is a holiday Enjoy\n");
```



Note:

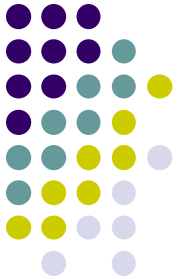
1. EOF: The definition of named constant EOF is included in the stdio.h file. The EOF signal can be given from the keyboard by typing the control character ctrl-z.

2. ASCII value: Using ASCII each character has been an assigned value.
   The character ‘A’ has an ASCII value of 65 and ‘a” has an ASCII value of 97.

Example 20:

```
char c;
c = ‘A’;
printf(“%c\t%d”, c,c);
```

Result: A 65



Example 21:

Convert the upper case characters in the following statement into upper case characters.

```
#include <stdio.h>
void main(void)
{
        char c;
        while(scanf("%c",&c) != '\n') {
                if(c>='A' && c<='Z')
                printf("%c",c+32);
                else
                printf("%c",c); }
        return;
}
```

Input: My name is paul^z

Result : my name is paul

Example 22:

```
while(expr)
action
```

=

```
for(;expr;)action
```

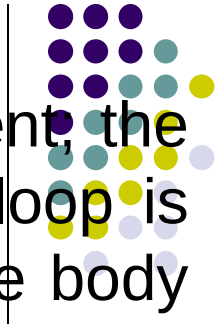
Example 23:

```
expr1;
while(expr2)
 {
action
expr3
}
```

=

```
for(expr1;
          expr2;
                    expr3)
action
```

DO WHILE STATEMENT

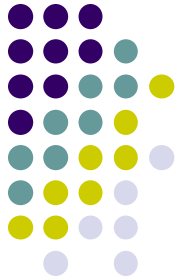


The do while statement is similar to the while statement, the only difference is that the expression controlling the loop is executed at the bottom of the loop. For this reason the body of the loop is always executed at least once.

The syntax of a do while statement is:

```
do
        action statement
while(expression);
```

First the action statement is executed. Then the expression is evaluated. If the expression is true then the control is returned to the top of the loop and the process is repeated. If at any time, when the expression is evaluated to false, then the action statement is skipped and control is transferred to the statement immediately following the while (expression) statement.



Example 24:

```
int i = 0;
do      {  i++;
printf("%d\t",i); }
while(i<5);
```

Result: 1 2 3 4 5

Example 25: What is the output?

```
int x;
x  = 4;
do
{
x-=2;
printf("%d\t",x);
}
while(x>=1);
```

Result: 2 0

Example 26: What is the output?

```
int x = 2;
do
printf(%d\t”,x);
while(--x > 0);
```

Result: 2 1

Example 27: What is the output?

```
int x = 2;
do  printf(%d\t”,x);
while( x-- > 0);
```

Result: 2 1 0

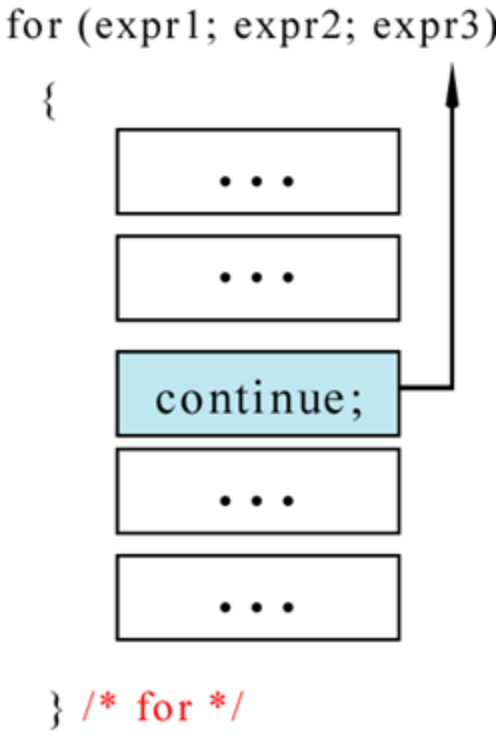
Example 28:

```
do{
printf(“Play again ? 1 = Yes, 2 = No”);
scanf(“%d”,&resp);
printf(“\n”);     }
while(resp !=1 && resp !=0);
```

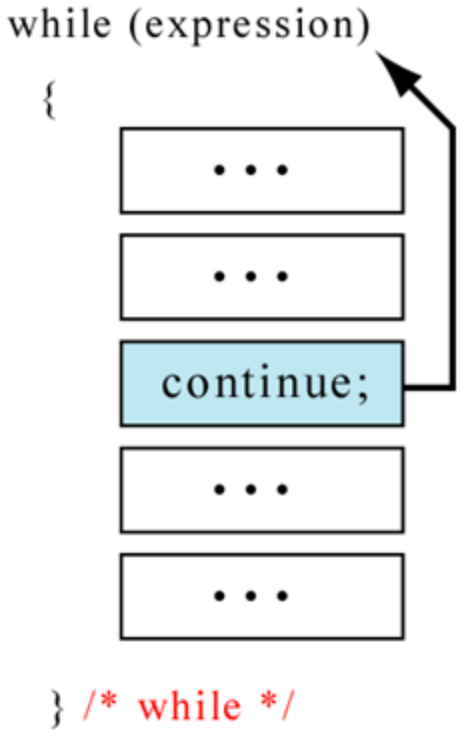
CONTINUE STATEMENT

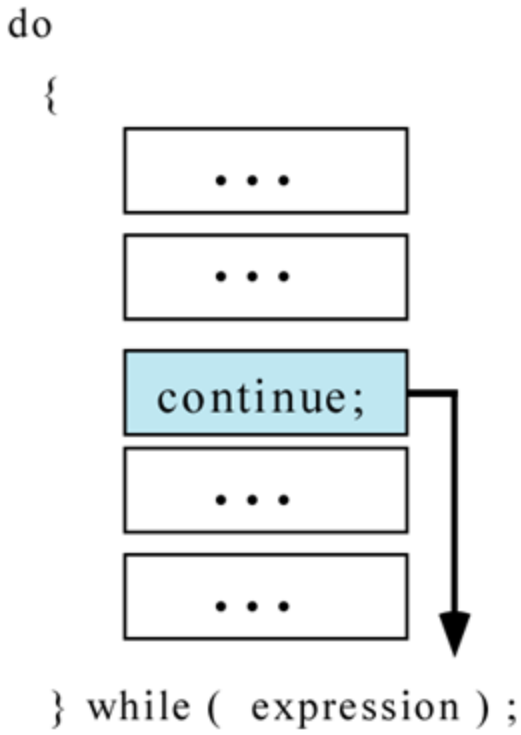
Continue statement is similar to the break statement in that the continue statement also terminates the body of the loop, but instead of exiting the loop, consider resuming execution of the loop.

If a continue statement is encountered in a for loop, then control will immediately transfer to the expression 3 and then the expression 2 is tested to determine whether to continue the loop.

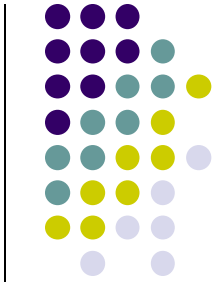


If a continue statement is encountered in a **while loop**, control will immediately jump to the top of the loop and the expression is tested to determine whether to execute the body of the loop again.





If a continue statement is encountered in a **do while** statement, then the control will immediately move to the bottom of the loop and the expression is tested to determine whether to continue the loop.



Example 29: What is the output?

```
int i = 0;
do
if(i<3) { i+=2;
          printf("%d\n"i);
          continue;
          }
else      {
          printf("%d\n",++i);
          break;
          } }
while(i<5);
```

Result:

2

4

5

COMMA OPERATOR

- This operator is used in conjunction with the for statement. Two or more expressions can be separated be means of a comma operator.
- When a sequence of expressions have to be evaluated then the expressions separated by a comma operator. The comma operator has the lowest precedence level.

Example:

```
for(i=0,j=3;i<3 && j<= 5; i++, j+=2)
        printf("%d\t%d\t"I,j);
```

3 1 5

CONDITIONAL OPERATOR

✓A conditional operator is used to evaluate expressions depending on conditions.

✓A conditional operator is equal to a simple if else statement.

The general syntax is :

expr1 ? expr2 : expr3;

First expr1 will be evaluated. If it is true then the value of the expression is expr2 else the value of the expression is expr3.

Example:

```
int  x = 5;
int y = 7;
int z;
/* z will have the larger value */
z = x>y?x:y;
printf("Larger value is %d\n",z);
```

The statement z = x>y ? x:y; is equivalent to the following if statement:

```
if(x>y) z = x; else z =y;
```

The formula for converting centigrade temperature to Fahrenheit is:

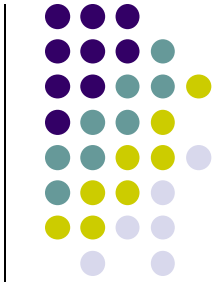
$$F = 32.0 + (C * 9.0 / 5.0)$$

Write a program that prints out conversion table for Celsius to Fahrenheit (0^0 to 100^0) in steps of 5^0.

```
#include<stdio.h>

void main(void)
{
float celsius, fahren;
printf("Celsius\t\tFahrenheit\n");
for(celsius = 0.0; celsius <= 100.0; celsius+=5.0)
    {
        fahren = 32.0   + (celsius*9.0/5.0);
        printf("%6.2f\t\t%8.2f\n", celsius, fahren);
     }
return;
}
```

Factorial of a number (n!) : n! = 1 * 2 * 3 * 4 *…*(n-1) * n

```
#include<stdio.h>

void main(void)
{
long int fact  = 1;
int i, n;
printf("Enter an Integer ");
scanf("%d",&n);
for(i = 1; i  <= n; i++)
     fact = fact*i;
printf("Factorial of %d is %Ld\n", n, fact);
return;
}
```



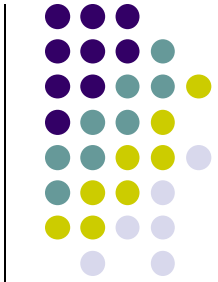
```
/* Program to compute LCM */
#include<stdio.h>
 int main(void)
{
int a, b, index, prod, lcm;
 printf(“Enter two integers ");
scanf("%d%d",&a,&b);
  for(index = 1; index <= a*b; index++)
    if(index % a == 0 && index % b == 0)
      {
       printf("LCM is %d\n", index);
       break;
       }
return 0;
}
```

$$\cos(x) = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \ldots$$

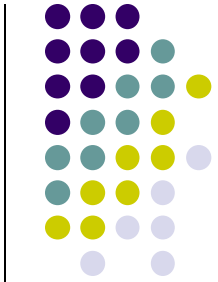
$$t_n = \frac{(-1)^{n+1} x^{2n-2}}{(2n-2)!}$$

$$t_{n+1} = \frac{(-1)^{n+1+1} x^{2(n+1)-2}}{(2(n+1)-2)!} = \frac{(-1)^{n+2} x^{2n}}{(2n)!}$$

$$\frac{t_{n+1}}{t_n} = \frac{(-1)^{n+2} x^{2n}}{(2n)!} \frac{(2n-2)!}{(-1)^{n+1} x^{2n-2}} = \frac{(-1)x^{2n}}{(2n)(2n\text{-}1)}$$



```
/* Program to compute cos(x) */
/* cos(x) = 1 - x^2/2! + x^4/4! + ... */
#include<stdio.h>
#include<math.h>
#include<float.h>
int main(void)   {
        float xd,x,term,sum;
        int n;
        printf("Enter the angle in degrees ");
        scanf("%f",&xd);
        x = (float) ((22.0/7.0)/180.0)*xd;

        term = 1.0;
        sum = 0.0;
        n = 1;
```



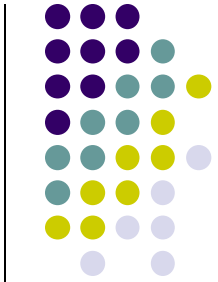
```
while(fabs(term) > 1.0e-5)
{
sum = sum+term;
term = ((-1)*x*x/((2*n-1)*(2*n)))*term;
n = n+1;
}

printf("cos(%8.2f) = %8.2f\n",xd,sum);
return 0;
}
```

$$\sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \ldots$$

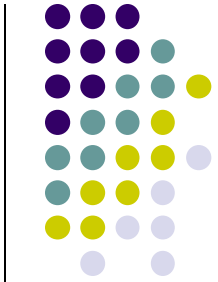
$$t_n = \frac{(-1)^{n+1} x^{2n-1}}{(2n-1)!}$$

$$t_{n+1} = \frac{(-1)^{n+1+1} x^{2(n+1)-1}}{(2(n+1)-1)!} = \frac{(-1)^{n+2} x^{2n+1}}{(2n+1)!}$$

$$\frac{t_{n+1}}{t_n} = \frac{(-1)^{n+2} x^{2n+1}}{(2n+1)!} \frac{(2n-1)!}{(-1)^{n+1} x^{2n-1}} = \frac{(-1)x^2}{(2n)(2n+1)}$$



```
/* Program to compute sin(x) */
/* sin(x) = x – x^3/3! + x^5/5! + ... */
#include<stdio.h>
#include<math.h>
#include<float.h>
int main(void)   {
        float xd,x,term,sum;
        int n;
        printf("Enter the angle in degrees ");
        scanf("%f",&xd);
        x = (float) ((22.0/7.0)/180.0)*xd;

        term = x;
        sum = 0.0;
        n = 1;
```



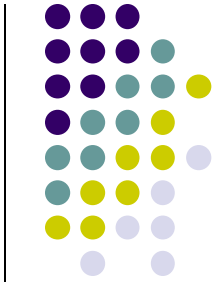
```
while(fabs(term) > 1.0e-5)
{
sum = sum+term;
term = ((-1)*x*x/((2*n+1)*(2*n)))*term;
n = n+1;
}

printf(“sin(%8.2f) = %8.2f\n",xd,sum);
return 0;
}
```

$$\cosh x = 1 + \frac{x^2}{2!} + \frac{x^4}{4!} + \frac{x^6}{6!} + \ldots +$$

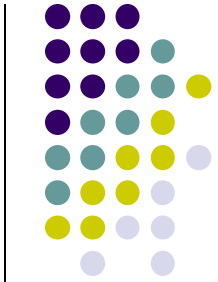
$$t_n = \frac{x^{2n-2}}{(2n-2)!}$$

$$t_{n+1} = \frac{x^{2n}}{(2n)!}$$

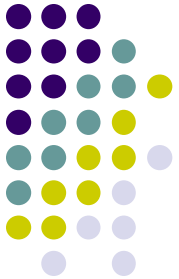
$$\frac{t_{n+1}}{t_n} = \frac{x^{2n}}{(2n)!} \frac{(2n-2)!}{x^{2n-2}} = \frac{x^2}{(2n-1)!}$$



```
/* Program to compute cosh (x) */
/* cosh (x) = 1 + x^2/2! + x^4/4! + ... */
#include<stdio.h>
#include<math.h>
#include<float.h>
int main(void)   {
        float xd,x,term,sum;
        int n;
        printf("Enter the angle in degrees ");
        scanf("%f",&xd);
        x = (float) ((22.0/7.0)/180.0)*xd;

        term = 1.0;
        sum = 0.0;
        n = 1;
```



```
while(fabs(term) > 1.0e-5)
{
sum = sum+term;
term = (x*x/((2*n-1)*(2*n)))*term;
n = n+1;
}

printf("cosh (%8.2f) = %8.2f\n",xd,sum);
return 0;
}
```



THANK YOU