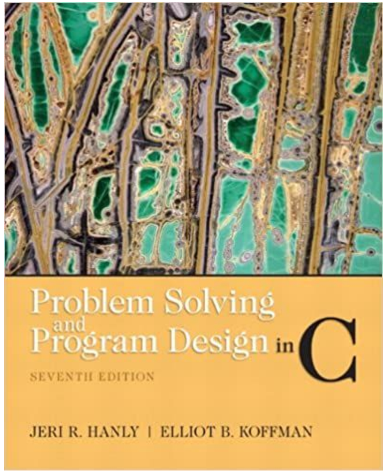
جامعة بيرزيت
BIRZEIT UNIVERSITY



Problem Solving and Program Design in C

SEVENTH EDITION

JERI R. HANLY | ELLIOT B. KOFFMAN

Faculty of Engineering and Technology
Department of Computer Science

Introduction to Computers and Programming (Comp 133)

References :
Book : Problem Solving and Program Design in C (7th Edition) 7th Edition
Slides : Dr. Radi Jarrar , Dr. Abdallah Karakra , Dr. Majdi Mafarja.

Repetition and Loop Statements

Chapter 5

Repetition and Loop

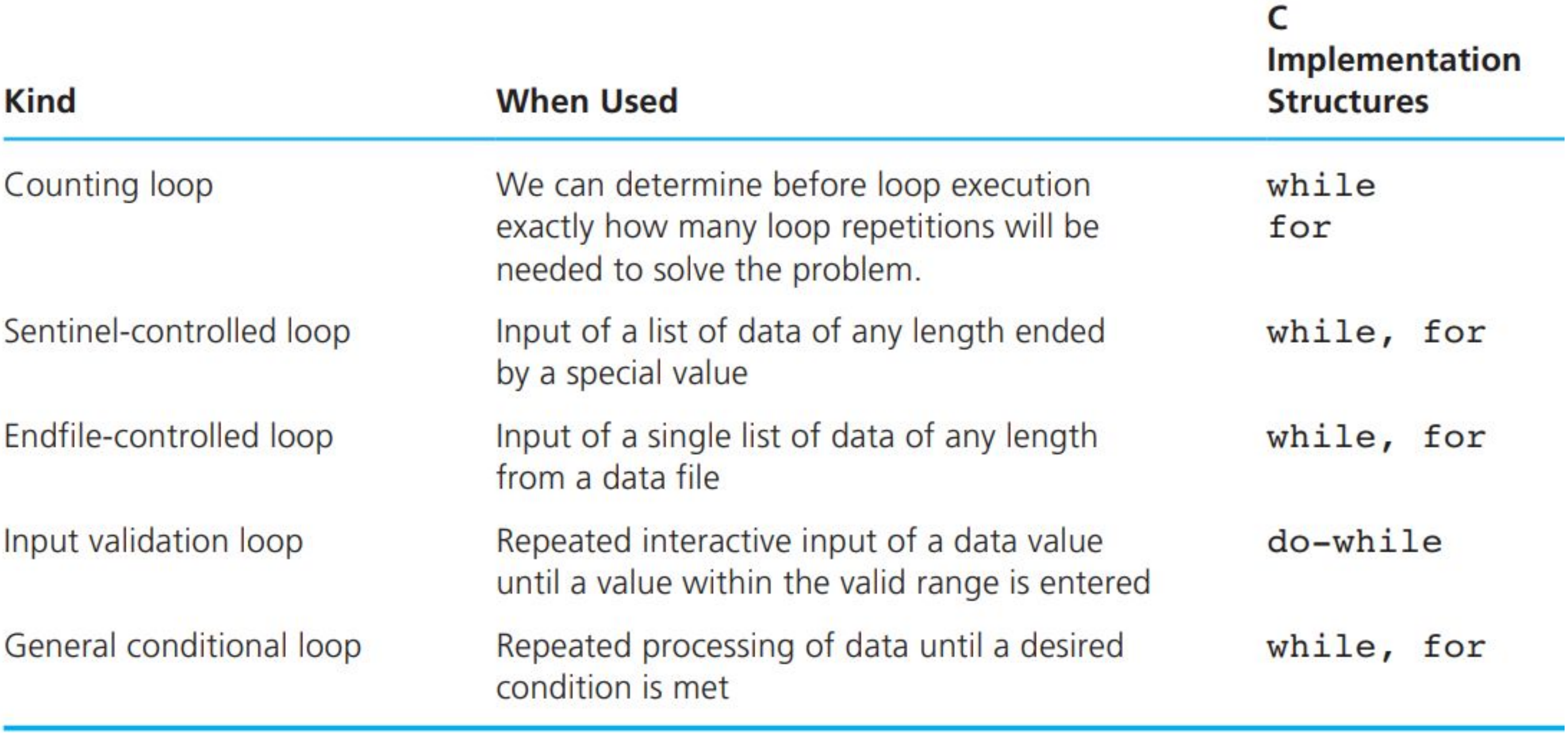
- **loop** a control structure that repeats a group of steps in a program.

- There are 3 types of loops in C
  - **while** — ①
  - **for** — ②
  - **do-while** — ③

Chapter 5

- Repetition in Programs

Loop Kinds

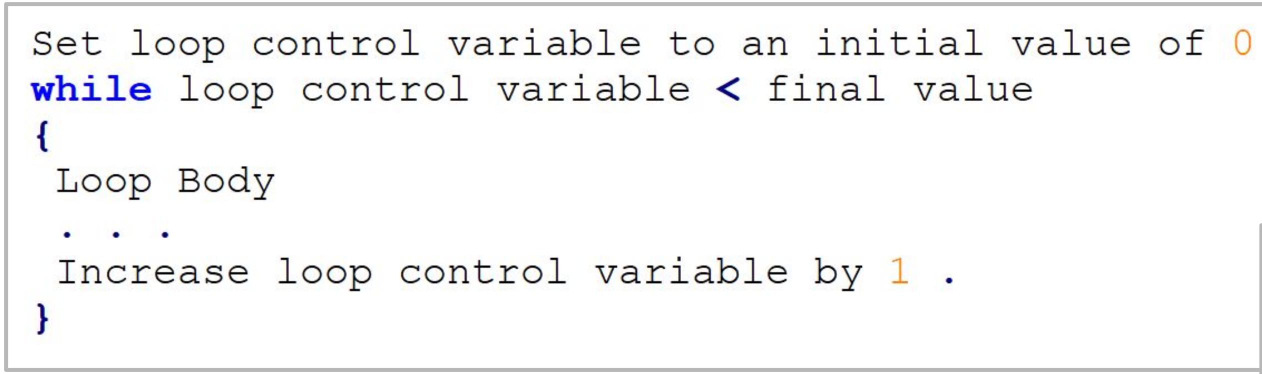


Kind	When Used	C Implementation Structures
Counting loop	We can determine before loop execution exactly how many loop repetitions will be needed to solve the problem.	`while` `for`
Sentinel-controlled loop	Input of a list of data of any length ended by a special value	`while, for`
Endfile-controlled loop	Input of a single list of data of any length from a data file	`while, for`
Input validation loop	Repeated interactive input of a data value until a value within the valid range is entered	`do-while`
General conditional loop	Repeated processing of data until a desired condition is met	`while, for`

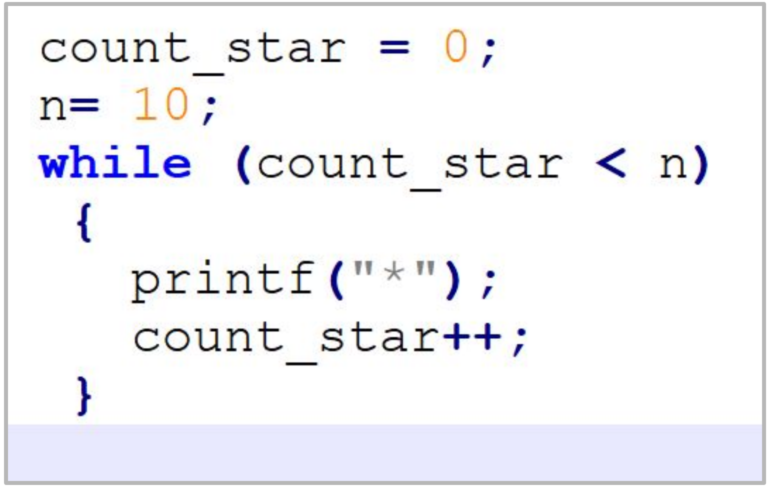
Controlling Loop

- **loop repetition condition:** the condition that controls loop repetition.
  - While(**count<10**)
- **Counter-controlled loop** : a loop whose required number of iterations can be determined before loop execution begins.
  - For(i=0;i<10;i++)
- **Event controlled loops:** stop when special value is encountered. (E.g., exit loop when input value is “E” , or stop a loop when input is -1).
  - While(X != -1)
- **Result controlled loops:** continues until a test determines that the desired result is reached (e.g., numerical approximations)
- **infinite loop** a loop that executes forever

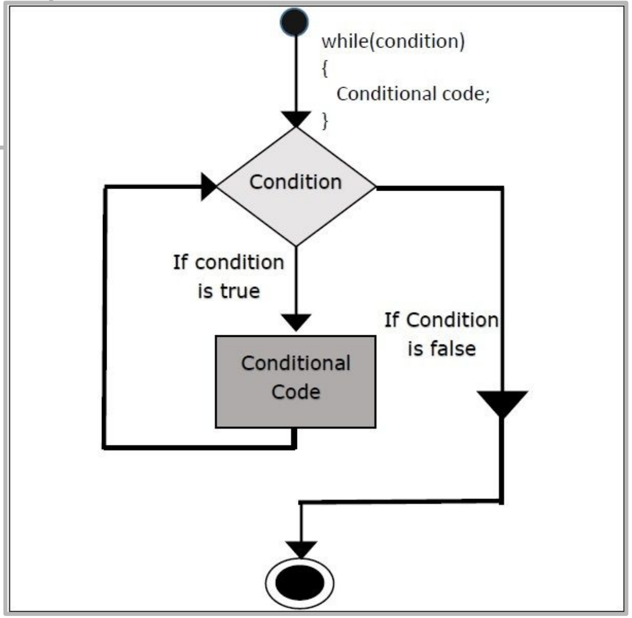
While Loop



```
Set loop control variable to an initial value of 0 .
while loop control variable < final value
{
 Loop Body
 . . .
 Increase loop control variable by 1 .
}
```



```
count_star = 0;
n= 10;
while (count_star < n)
 {
   printf("*");
   count_star++;
 }
```



```
while(condition)
{
  Conditional code;
}
```

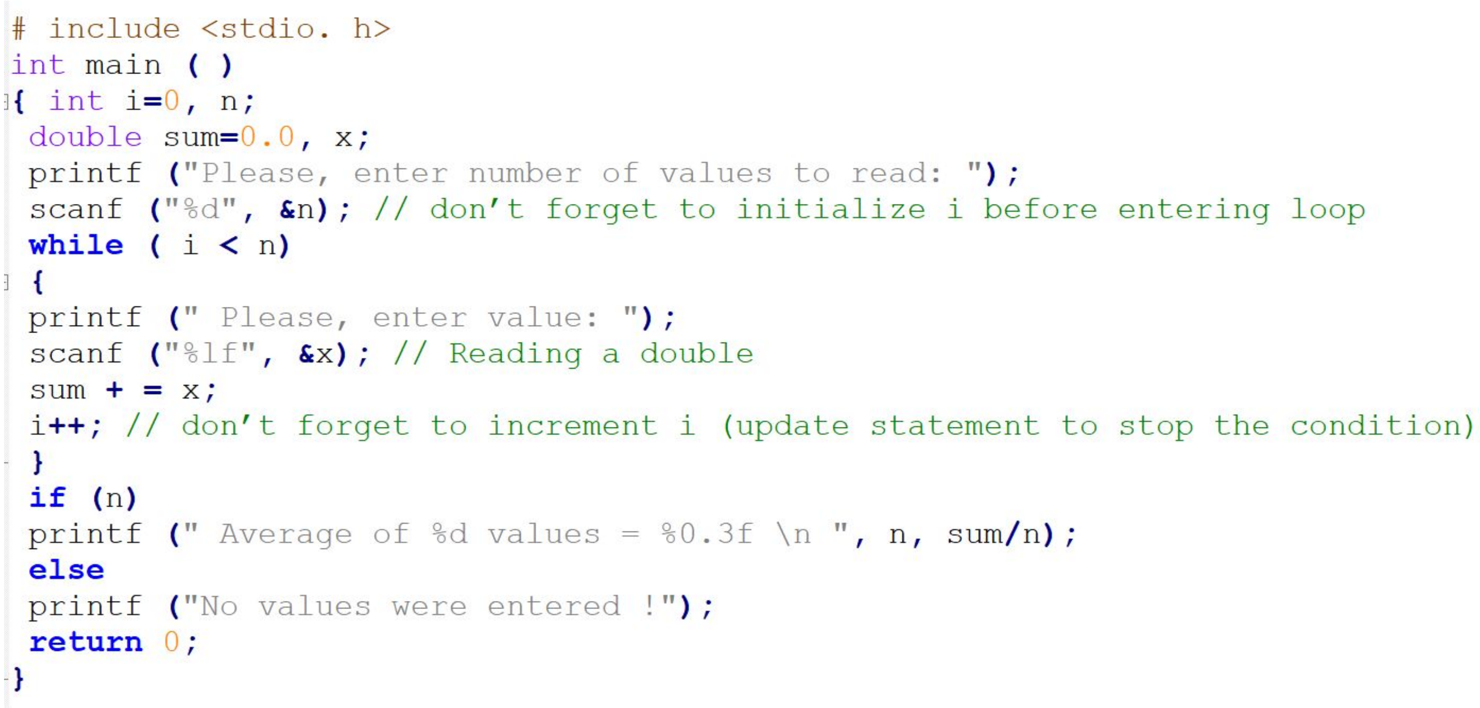
While Loop Example

- Write a program to print the first 100 positive integers.

```
#include<stdio.h>
int main(){
  int counter = 1;
  while( counter <= 100){
    printf(“%d\n”, counter);
    counter = counter + 1; //don’t forget
  }
  return 0;
}
```

While Loop Example

- Write a program to find and print the average of **n** values, where **n** is entered by the user.



```
# include <stdio. h>
int main ( )
{ int i=0, n;
 double sum=0.0, x;
 printf ("Please, enter number of values to read: ");
 scanf ("%d", &n); // don’t forget to initialize i before entering loop
 while ( i < n)
 {
 printf (" Please, enter value: ");
 scanf ("%lf", &x); // Reading a double
 sum + = x;
 i++; // don’t forget to increment i (update statement to stop the condition)
 }
 if (n)
 printf (" Average of %d values = %0.3f \n ", n, sum/n);
 else
 printf ("No values were entered !");
 return 0;
}
```

While Loop Example

Write a program that reads 10 grades and compute their average.

```
int main(){
  int counter = 0, grade, total = 0;
  float average;
  while( counter < 10 ){
    printf(“Please enter a grade”);
    scanf(“%d”, &grade);
    total = total + grade;
    counter = counter + 1;
  }
  average = total / counter;
  printf(“The average is %f\n”, average);
  return 0;
}
```

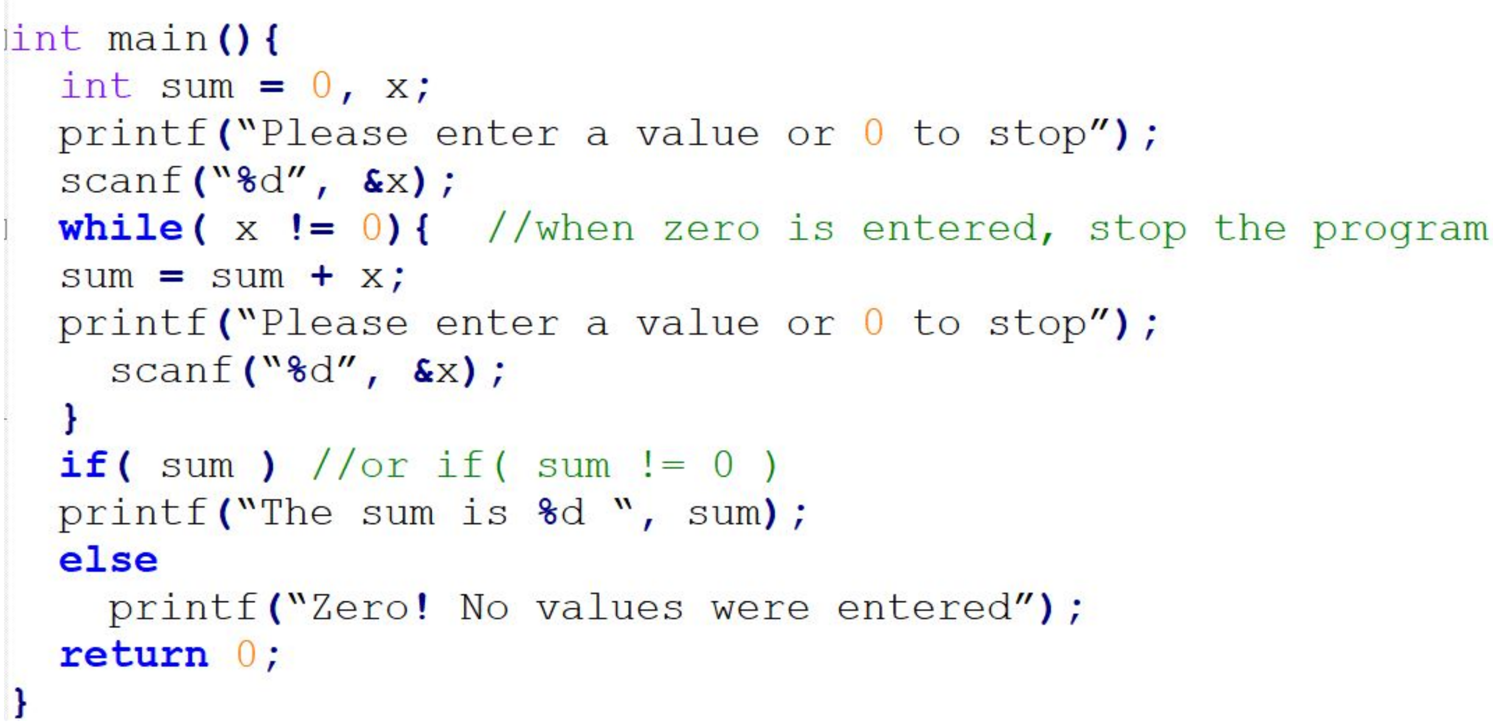
While Loop Example

Write a program that reads **n** grades and compute their average. When -1 is entered, stop.

```
int main(){
  int counter = 0, grade, total = 0;
  float average;
  printf(“Please enter a grade”);
  scanf(“%d”, &grade);
  while( grade != -1){
    total = total + grade;
    counter = counter + 1;

    printf(“Please enter a grade”);
  scanf(“%d”, &grade);
  }
  average = total / counter;
  printf(“The average is %f\n”, average);
  return 0;
}
```

While Loop Example

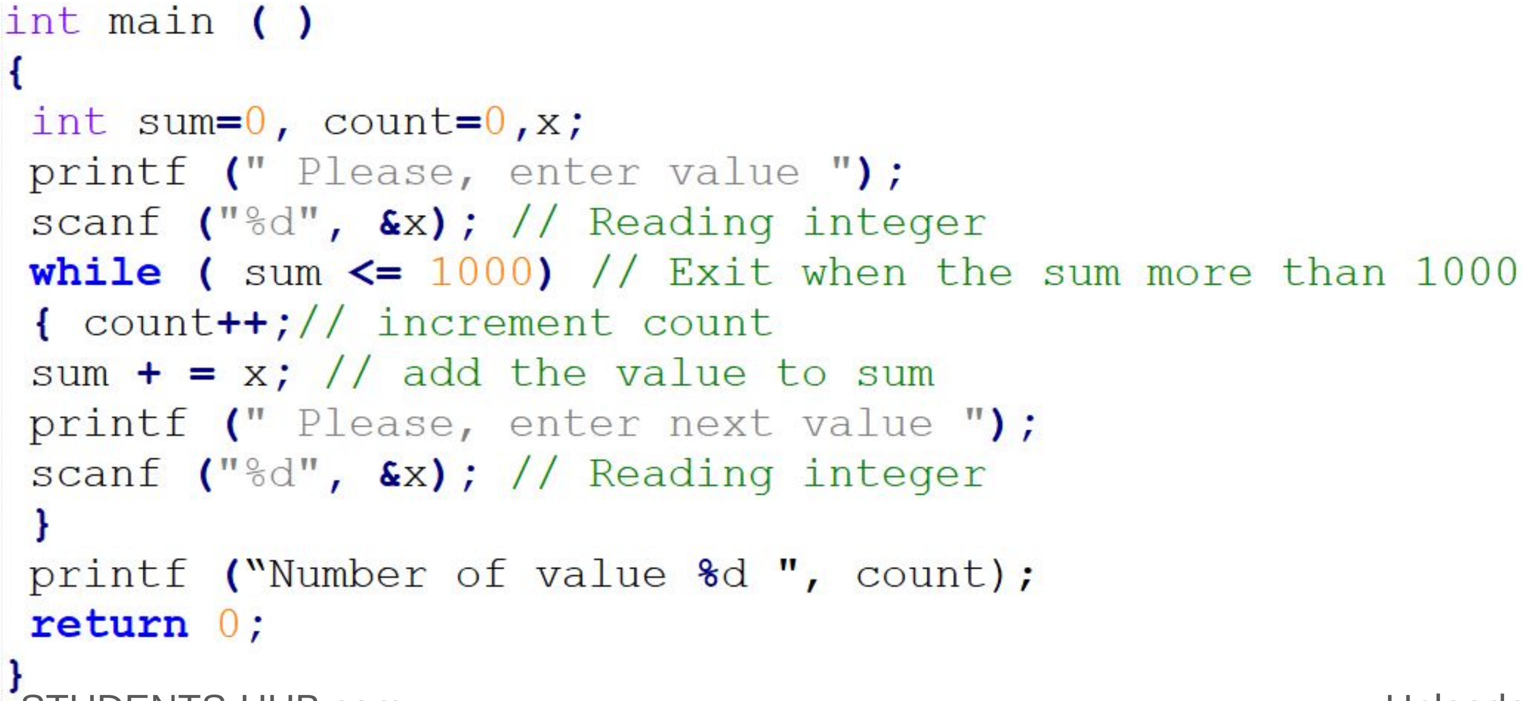
Write a program to calculate the sum of a set of values (we don’t know their count). When 0 is entered this means that program should stop receiving data, and print the sum.



```
int main(){
  int sum = 0, x;
  printf(“Please enter a value or 0 to stop”);
  scanf(“%d”, &x);
  while( x != 0){  //when zero is entered, stop the program
  sum = sum + x;
  printf(“Please enter a value or 0 to stop”);
    scanf(“%d”, &x);
  }
  if( sum ) //or if( sum != 0 )
  printf(“The sum is %d ”, sum);
  else
    printf(“Zero! No values were entered”);
  return 0;
}
```

While Loop Example

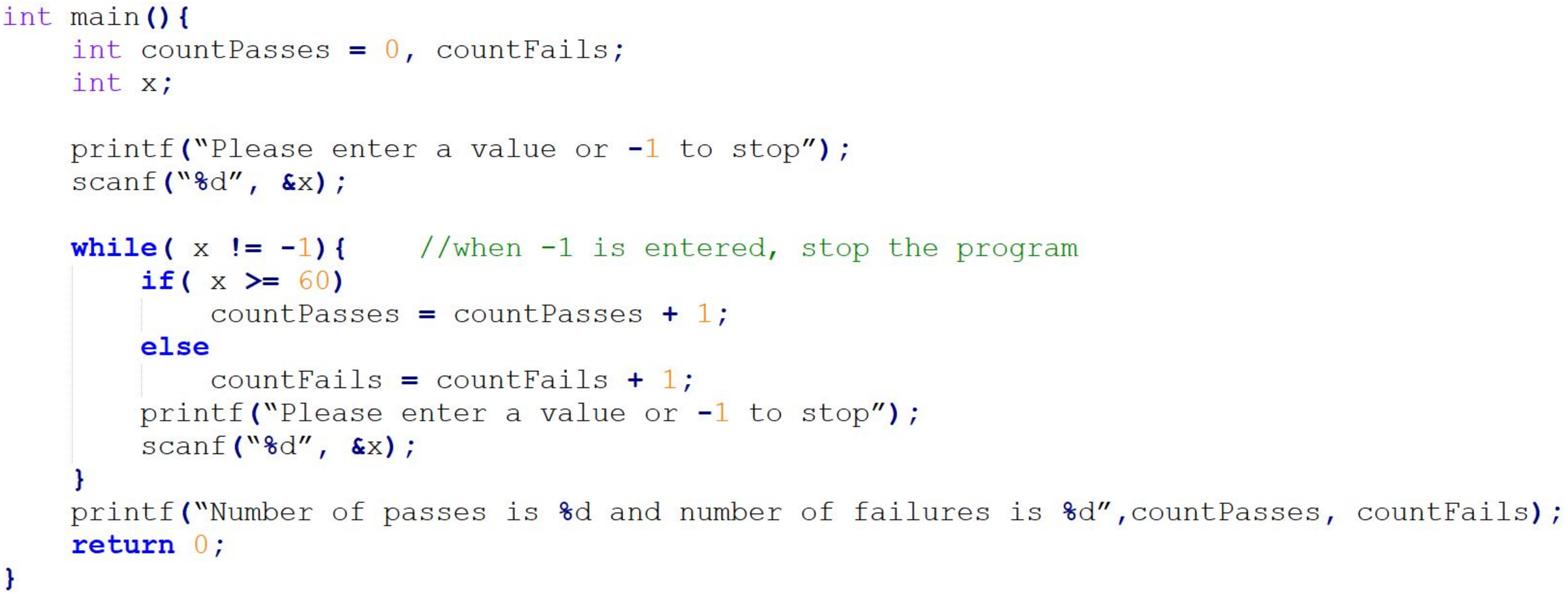
Write a program to calculate the sum of a set of values (we don’t know their count). When the sum exceeds 1000 this means that program should stop receiving data, and print the number of values were entered.



```
int main ( )
{
 int sum=0, count=0,x;
 printf (" Please, enter value ");
 scanf ("%d", &x); // Reading integer
 while ( sum <= 1000) // Exit when the sum more than 1000
 { count++;// increment count
 sum + = x; // add the value to sum
 printf (" Please, enter next value ");
 scanf ("%d", &x); // Reading integer
 }
 printf (“Number of value %d ", count);
 return 0;
}
```

While Loop Example

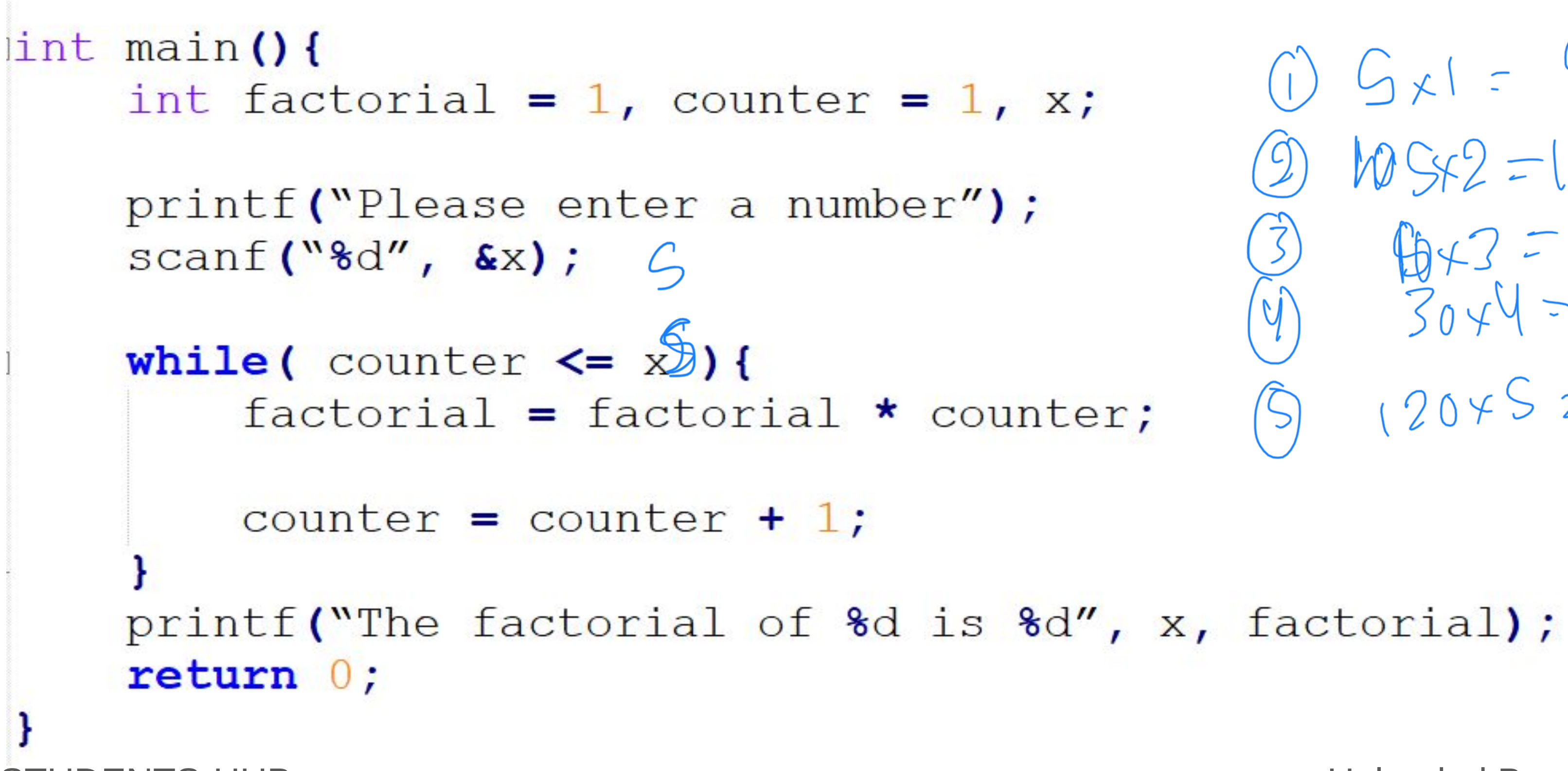
Write a program to print the number of passes and the number of failures in a set of n students. The user should enter -1 to stop.



```
int main(){
    int countPasses = 0, countFails;
    int x;

    printf(“Please enter a value or -1 to stop”);
    scanf(“%d”, &x);

    while( x != -1){     //when -1 is entered, stop the program
        if( x >= 60)
            countPasses = countPasses + 1;
        else
            countFails = countFails + 1;
        printf(“Please enter a value or -1 to stop”);
        scanf(“%d”, &x);
    }
    printf(“Number of passes is %d and number of failures is %d”,countPasses, countFails);
    return 0;
}
```

While Loop Example

Write a program to compute the factorial of a given number n.



```
int main(){
    int factorial = 1, counter = 1, x;

    printf(“Please enter a number”);
    scanf(“%d”, &x);

    while( counter <= x){
        factorial = factorial * counter;

        counter = counter + 1;
    }
    printf(“The factorial of %d is %d”, x, factorial);
    return 0;
}
```

5

5

① 5x1 = 5

② ~~10~~ 5x2 = 10

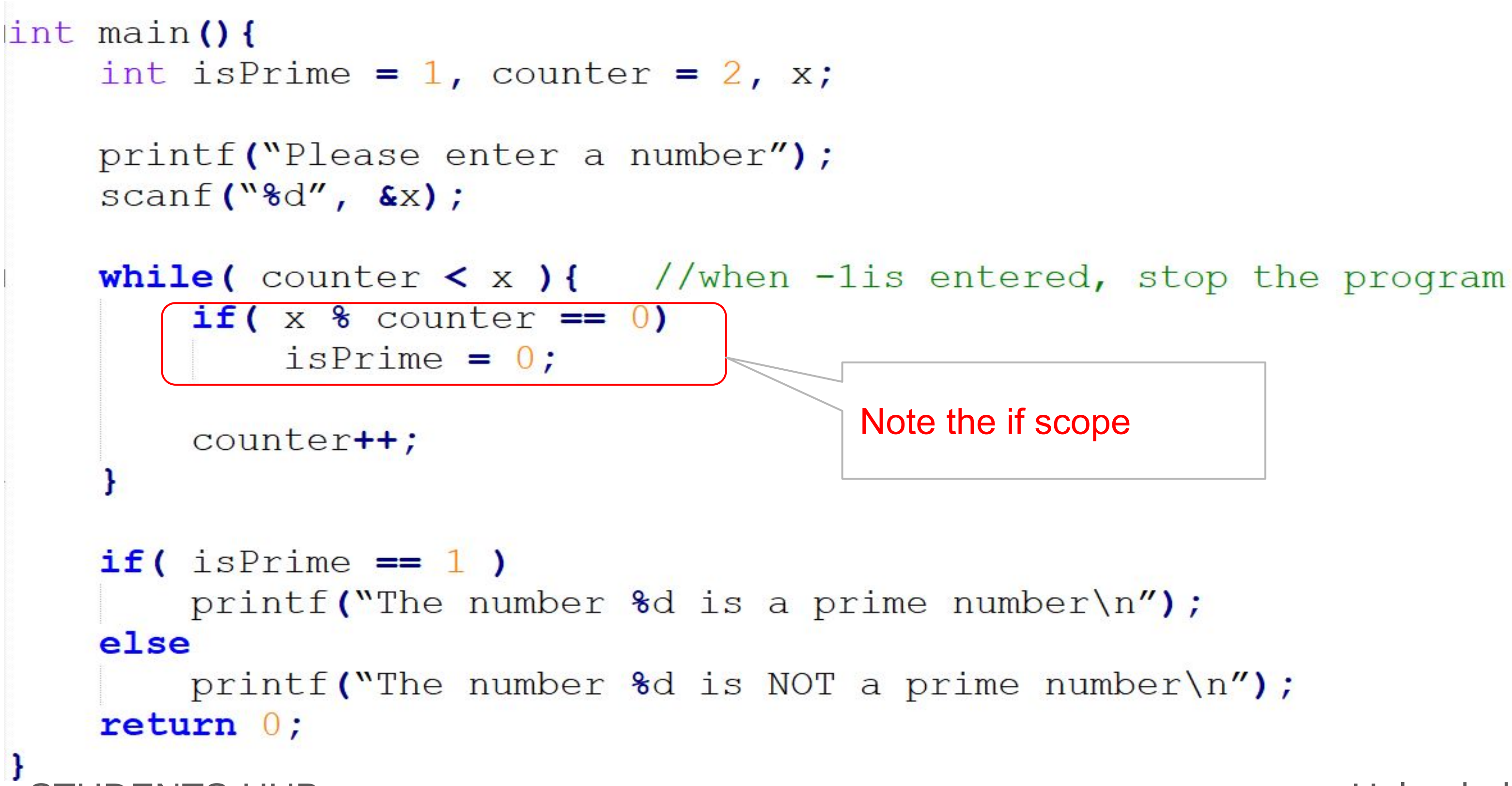
③ 10x3 = 70

④ 30x4 = 120

⑤ 120x5 = 600

While Loop Example

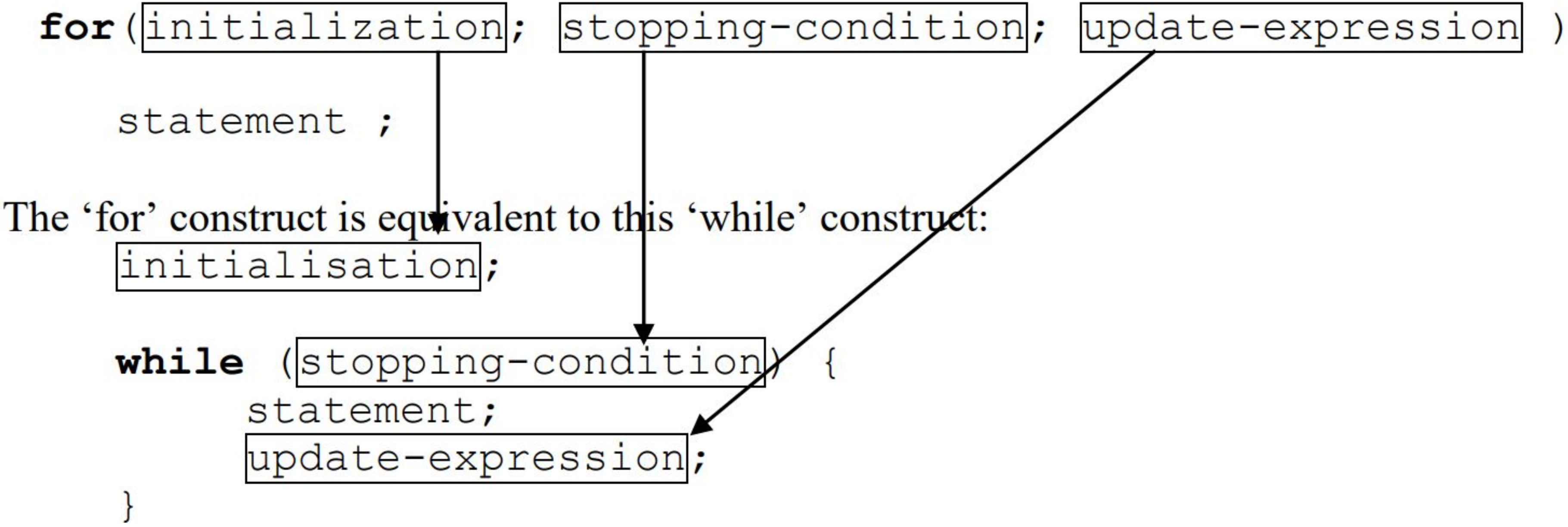
Write a program to check if an input number is prime or not.



```
int main(){
    int isPrime = 1, counter = 2, x;

    printf(“Please enter a number”);
    scanf(“%d”, &x);

    while( counter < x ){    //when -1is entered, stop the program
        if( x % counter == 0)
            isPrime = 0;

        counter++;
    }

    if( isPrime == 1 )
        printf(“The number %d is a prime number\n”);
    else
        printf(“The number %d is NOT a prime number\n”);
    return 0;
}
```

Note the if scope

For Loop



```
for (initialization; stopping-condition; update-expression )
    statement ;
```

The ‘for’ construct is equivalent to this ‘while’ construct:

```
    initialisation;

    while (stopping-condition) {
        statement;
        update-expression;
    }
```

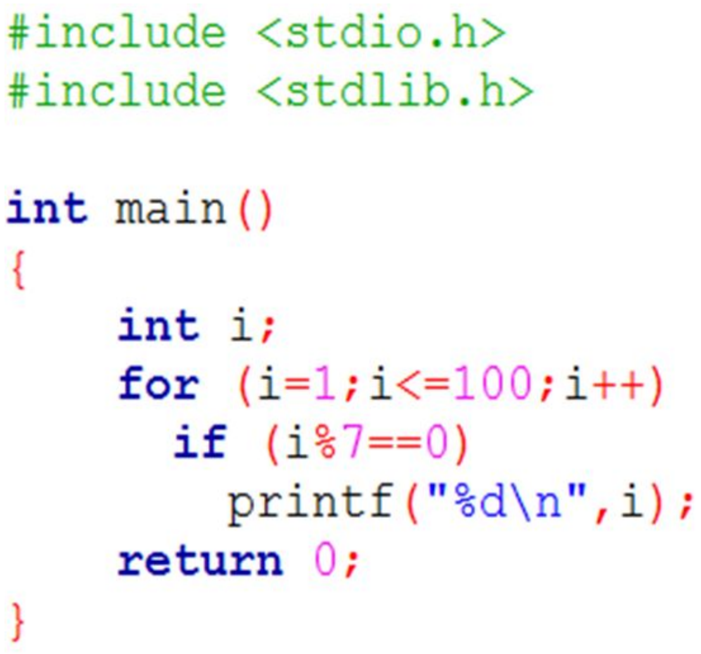
For Loop

```
for(expr1; expr2; expr3)
{
    body
}
```

Normal forms are:

```
for(i = 0; i < 10; i++) {...}
for(i = n-1; i >= 0; i--) {...}
```

For Loop



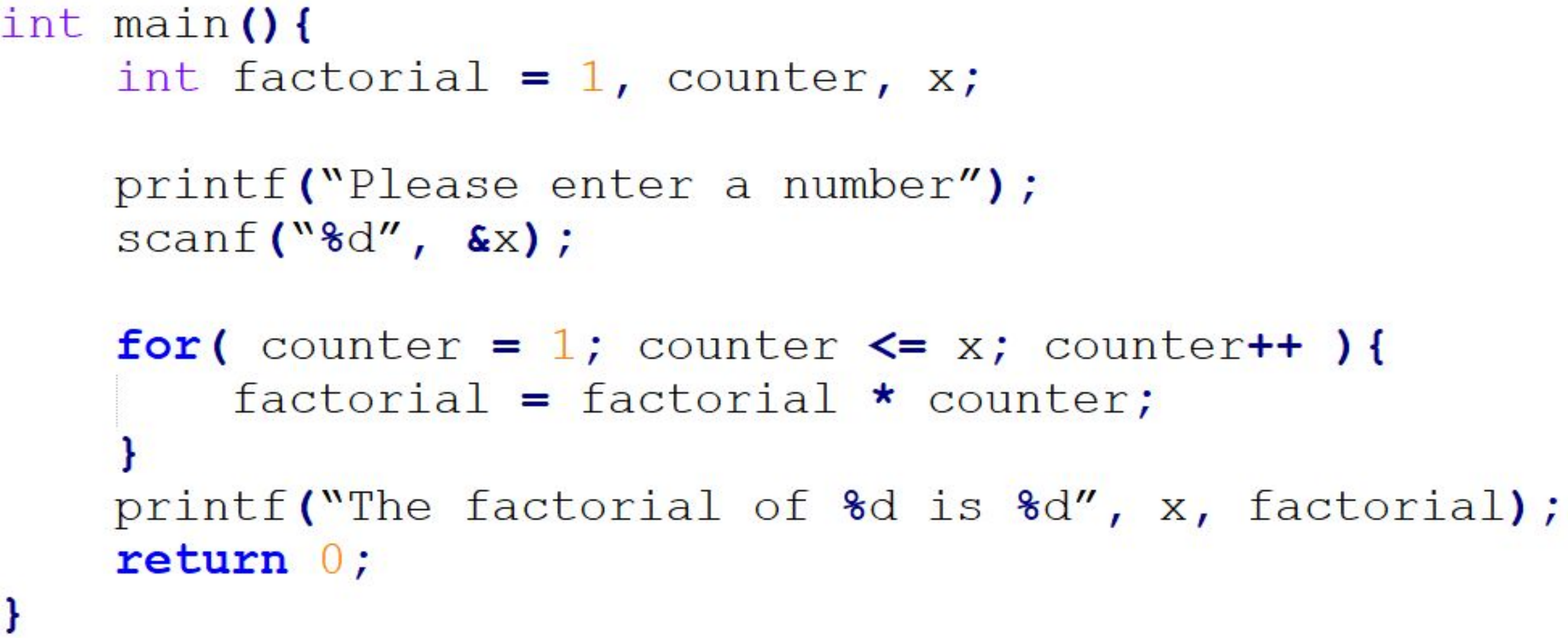
```c
#include <stdio.h>
#include <stdlib.h>

int main()
{
     int i;
     for (i=1;i<=100;i++)
       if (i%7==0)
         printf("%d\n",i);
     return 0;
}
```

Output

```
7
14
21
28
35
42
49
56
63
70
77
84
91
98
```

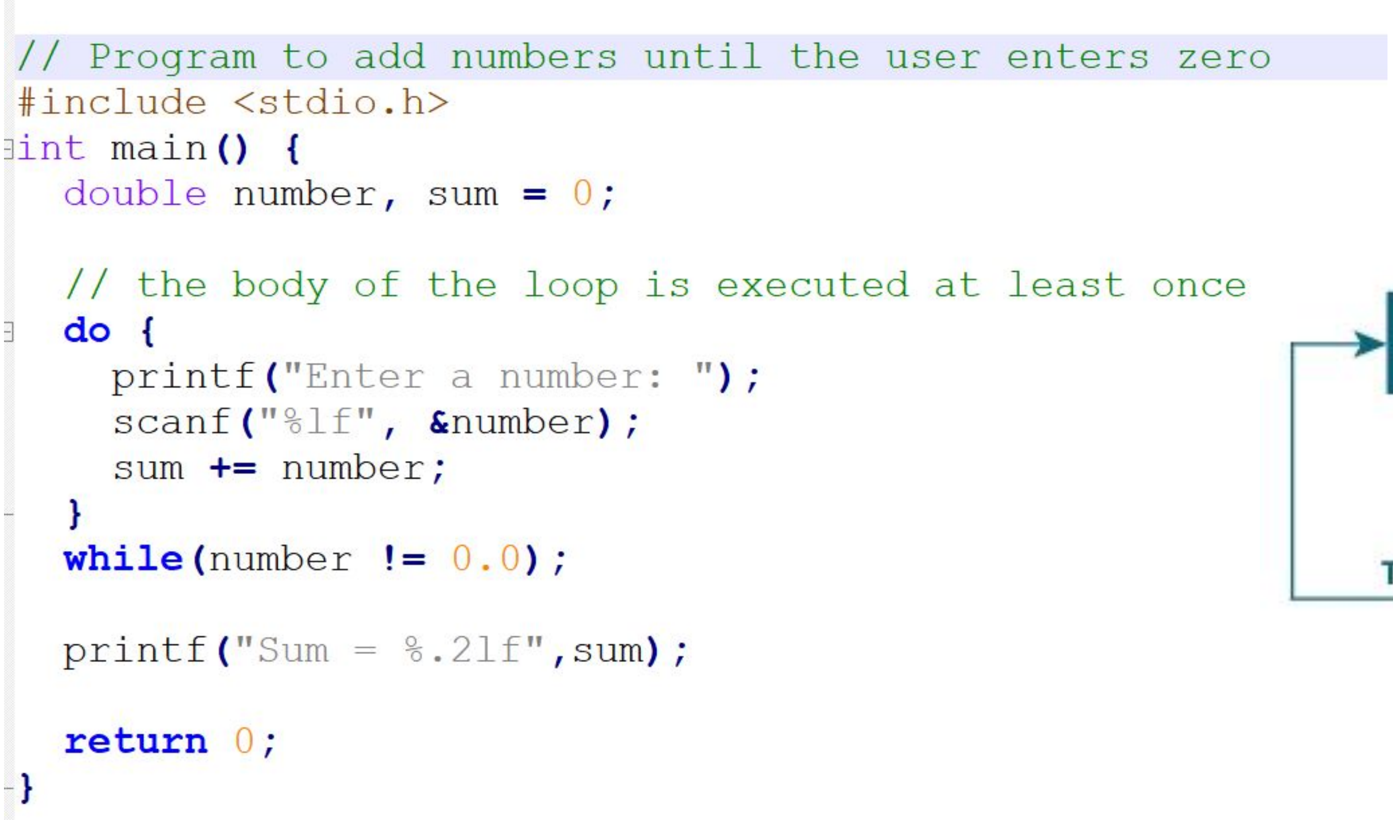
For Loop Example

Write a program to compute the factorial of a given number **n**.

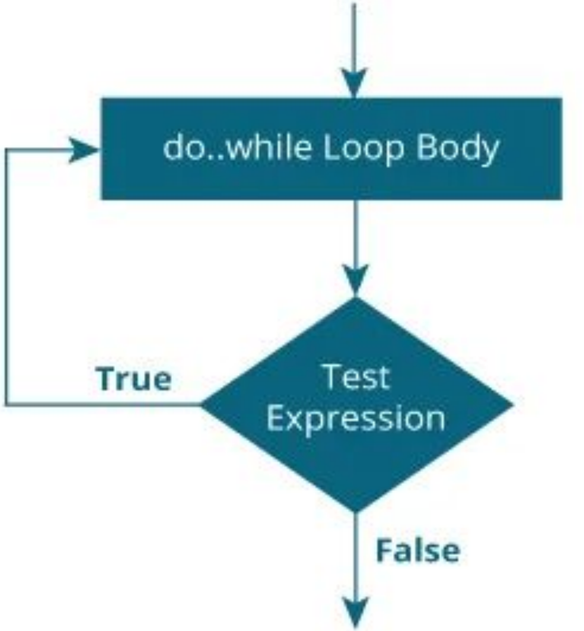


```
int main(){
    int factorial = 1, counter, x;

    printf(“Please enter a number”);
    scanf(“%d”, &x);

    for( counter = 1; counter <= x; counter++ ){
        factorial = factorial * counter;
    }
    printf(“The factorial of %d is %d”, x, factorial);
    return 0;
}
```

Do-While Loop Example



```
// Program to add numbers until the user enters zero
#include <stdio.h>
int main() {
  double number, sum = 0;

  // the body of the loop is executed at least once
  do {
    printf("Enter a number: ");
    scanf("%lf", &number);
    sum += number;
  }
  while(number != 0.0);

  printf("Sum = %.2lf",sum);

  return 0;
}
```



https://www.programiz.com/c-programming/c-do-while-loops

Chapter 5

- Logical and relational operators

Relational & Equality Operators

Operator	Meaning	Type
<	Less than	Relational
<=	Less than or equal	Relational
>	Greater than	Relational
>=	Greater than or equal	Relational
==	Equals	Equality
!=	Not equal	Equality

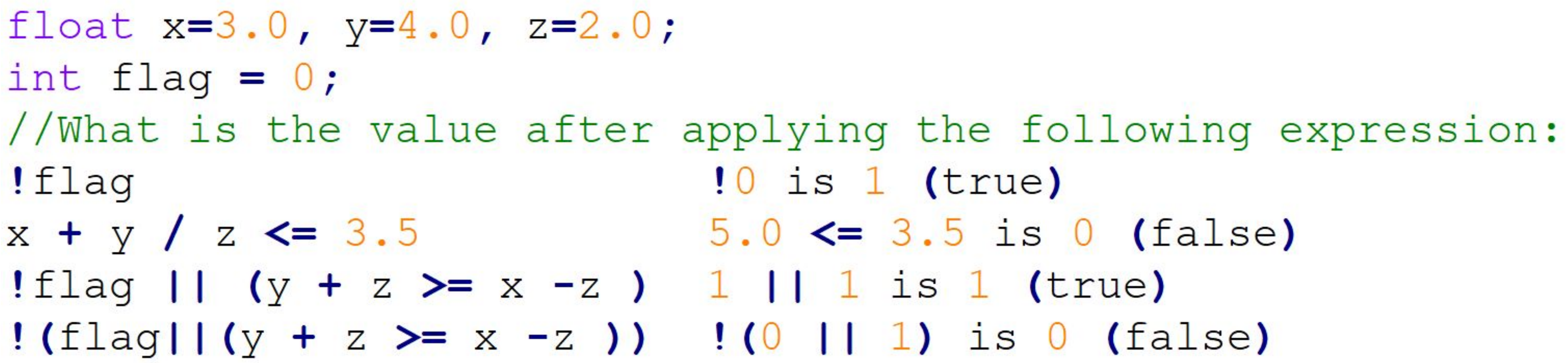
Logical Operators

Operator	Meaning
&&	And
||	Or
!	Negation (not)

Operator Precedence

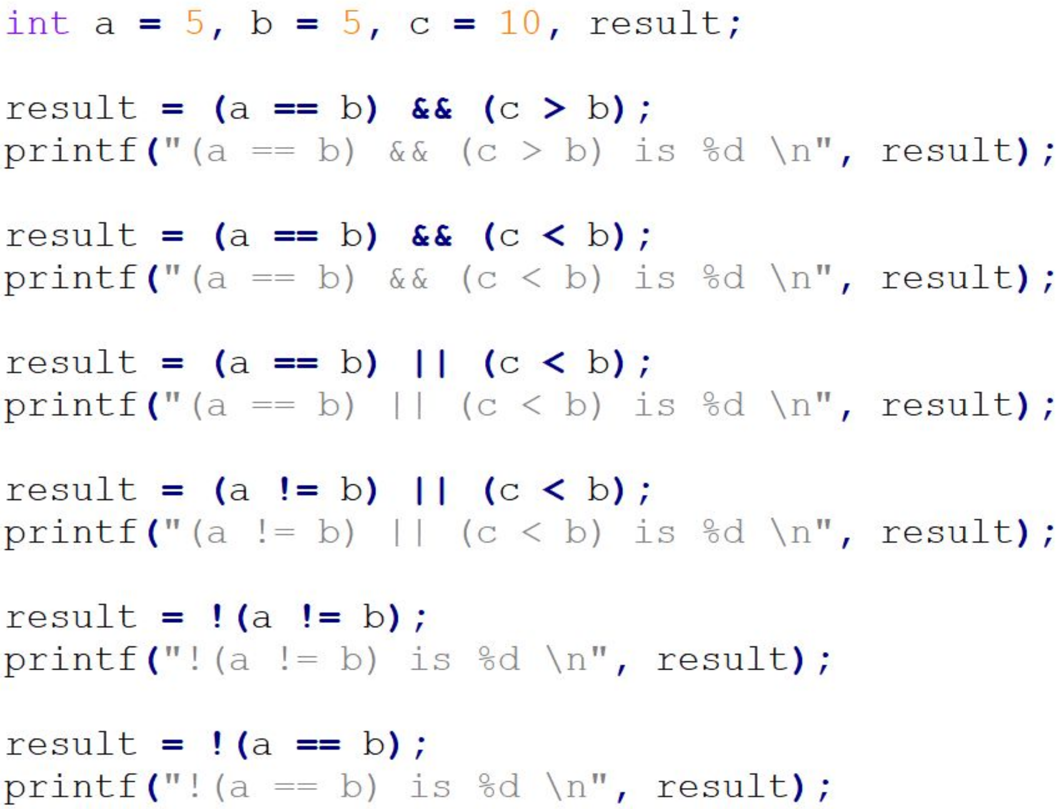
Operator	Precedence
!, +, -, & (unary operators)	Highest
*, /, %	
+, -	
<, <=, >, >=	
==, !=	
&&	
||	
=	Lowest

Logical Operators

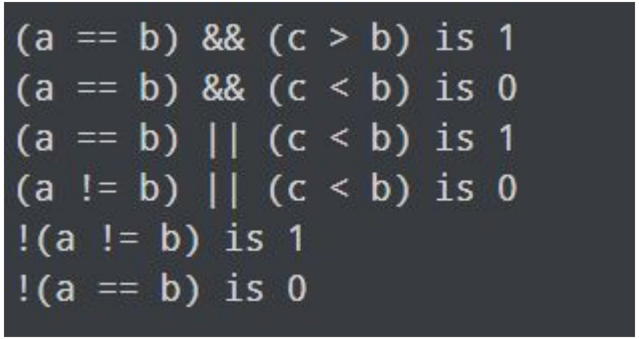


```
float x=3.0, y=4.0, z=2.0;
int flag = 0;
//What is the value after applying the following expression:
!flag                      !0 is 1 (true)
x + y / z <= 3.5           5.0 <= 3.5 is 0 (false)
!flag || (y + z >= x -z )  1 || 1 is 1 (true)
!(flag||(y + z >= x -z ))  !(0 || 1) is 0 (false)
```

Logical Operators



```c
int a = 5, b = 5, c = 10, result;

result = (a == b) && (c > b);
printf("(a == b) && (c > b) is %d \n", result);

result = (a == b) && (c < b);
printf("(a == b) && (c < b) is %d \n", result);

result = (a == b) || (c < b);
printf("(a == b) || (c < b) is %d \n", result);

result = (a != b) || (c < b);
printf("(a != b) || (c < b) is %d \n", result);

result = !(a != b);
printf("!(a != b) is %d \n", result);

result = !(a == b);
printf("!(a == b) is %d \n", result);
```



```
(a == b) && (c > b) is 1
(a == b) && (c < b) is 0
(a == b) || (c < b) is 1
(a != b) || (c < b) is 0
!(a != b) is 1
!(a == b) is 0
```

https://www.programiz.com/c-programming/c-operators

Assignment Shorthands

Simple Assignment Operators	Compound Assignment Operators
x = x + 1;	**x += 1;**
x= x -1;	**x -= 1;**
x = x * y;	**x *= y;**
x= x / y;	**x /= y;**
n = n % (x+1);	**n %= x+1;**

Pre and Post-Increment

- **++x** : Pre-increment **x**
  - **a = ++x * b;**
    - **x = x + 1;**
    - **a = x * b;**
- **x++** : Post-increment **x**
  - **a = x++ * b;**
    - **a = x * b;**
    - **x = x + 1;**

- **--x** : Pre-decrement **x**
  - **a = --x * b;**
    - **x = x - 1;**
    - **a = x * b;**
- **x--** : Post-decrement **x**
  - **a = x-- * b;**
    - **a = x * b;**
    - **x = x - 1;**

Pre and Post-Increment

```
int a=2, b=3, c;
c = ++a * b++;
```

→ 3 * 3 = 9

→ b = 3+1 → 4

Find a,b,c ?

a=2	b=3	c=
a=3	b=3	c=
a=3	b=3	c=9
a=3	b=4	c=9

a=3 , b=4, and c = 9

Pre and Post-Increment

- Find x,y,z ?

```
int x=2,y=3,z=0;

z += --x * y++;
```

= 1 * 3 = 3

*

Result : x=1 , y=4, and $\boxed{z = 3}$

- Find a,b,c ?

5

3

```
int a=4,b=3,c=20;

c /= ++a;
```

→ c = (c / ++a)

= (20 / 5)

= 4

Result : a=5 , b=3, and c = 4

- Find x,y,z ?

```
int x=2,y=3,z=4;

z *= ++x * y++;
```

→ z = z * (++x * y++)

= 4(3 * 3)

= 36

* $\boxed{x=3}$

* $\boxed{y=4}$

Result : x=3 , y=4, and $\boxed{z = 36}$

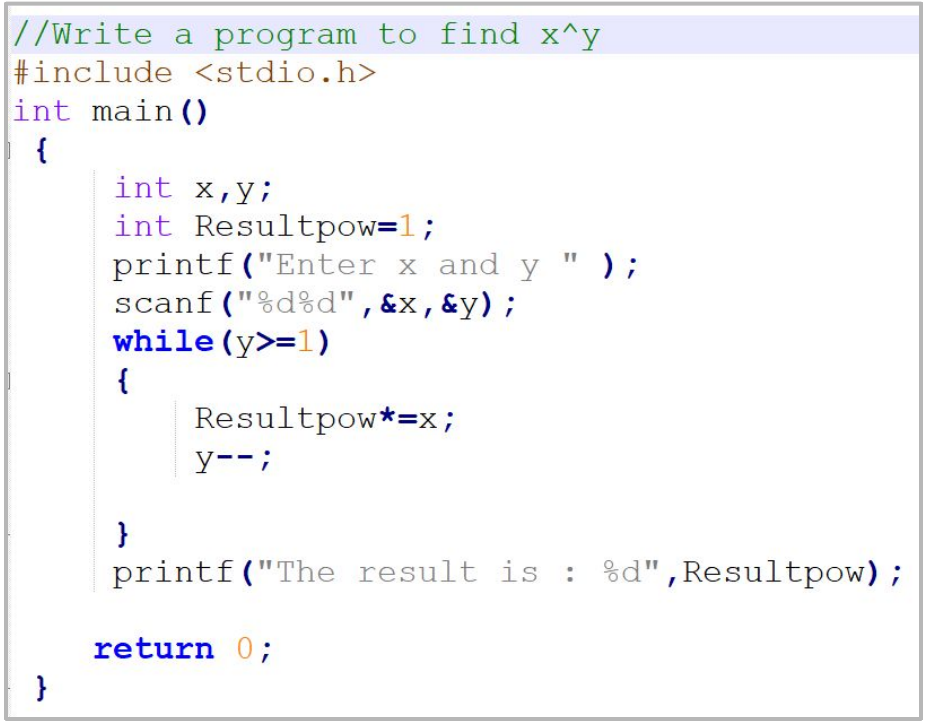
Pre and Post-Increment

```
int i = 1;

while (i < 5)

printf ("%d " , i++);
```

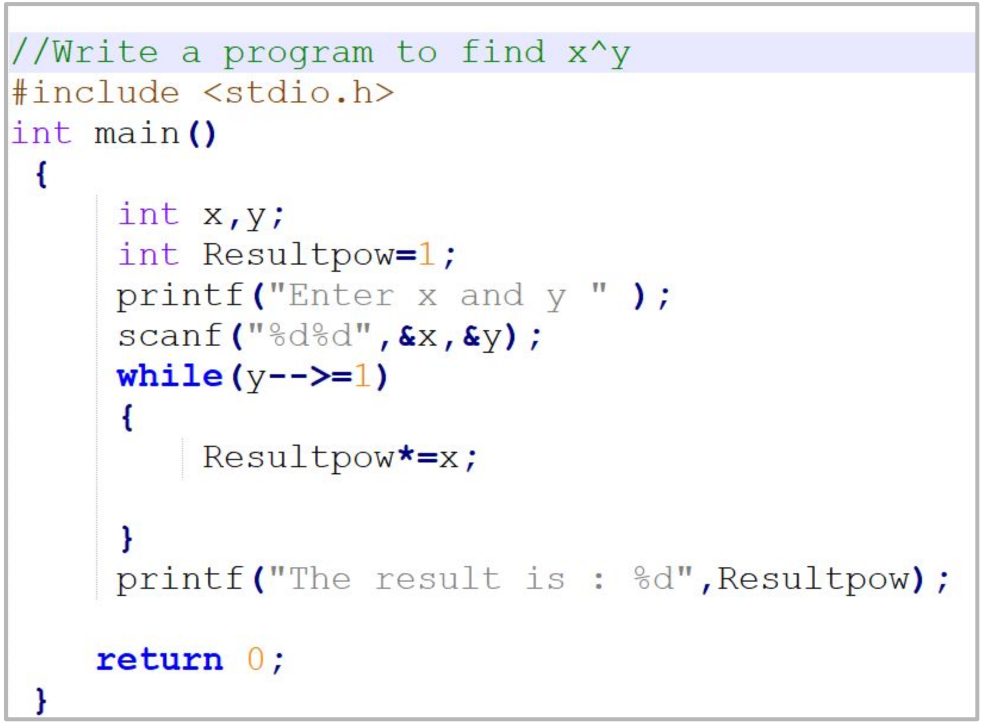
- What is the output?
  - 1 2 3 4
- What is the final value of **i** ?
  - i=5

Pre and Post-Increment

- Write a program to find x^y



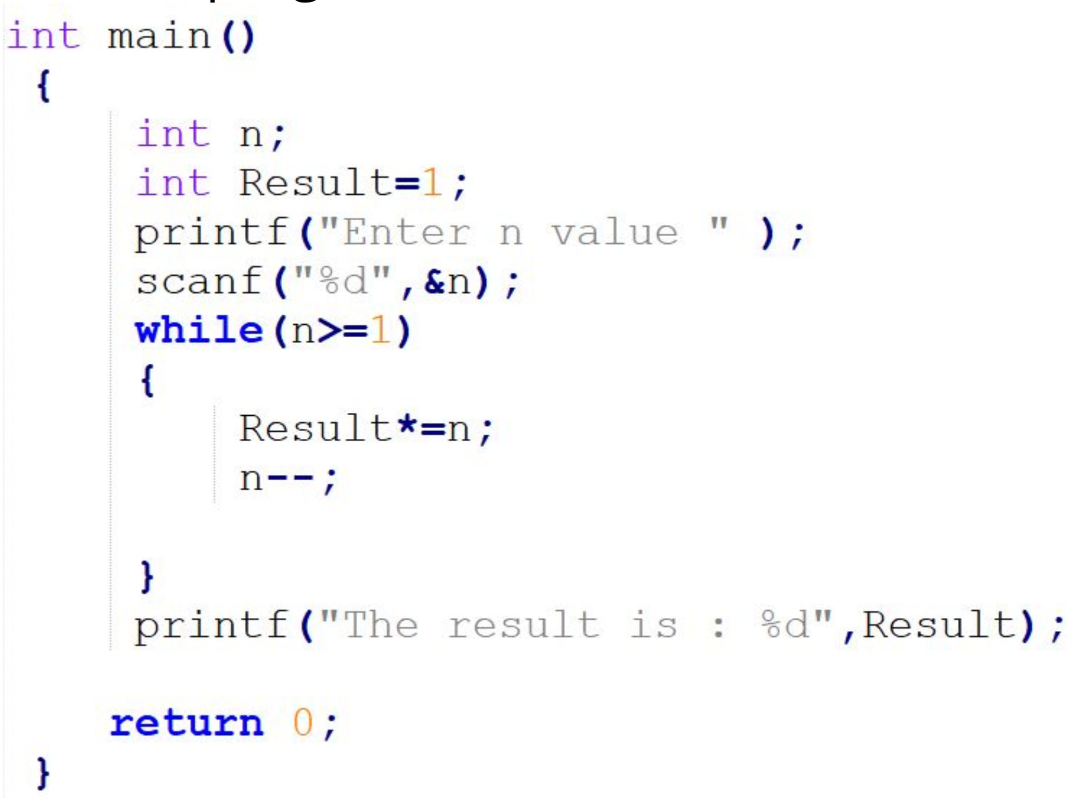
```c
//Write a program to find x^y
#include <stdio.h>
int main()
 {
     int x,y;
     int Resultpow=1;
     printf("Enter x and y " );
     scanf("%d%d",&x,&y);
     while(y>=1)
     {
         Resultpow*=x;
         y--;

     }
     printf("The result is : %d",Resultpow);

    return 0;
 }
```



```c
//Write a program to find x^y
#include <stdio.h>
int main()
 {
     int x,y;
     int Resultpow=1;
     printf("Enter x and y " );
     scanf("%d%d",&x,&y);
     while(y-->=1)
     {
         Resultpow*=x;

     }
     printf("The result is : %d",Resultpow);

    return 0;
 }
```

Pre and Post-Increment

- Write a program to find n!



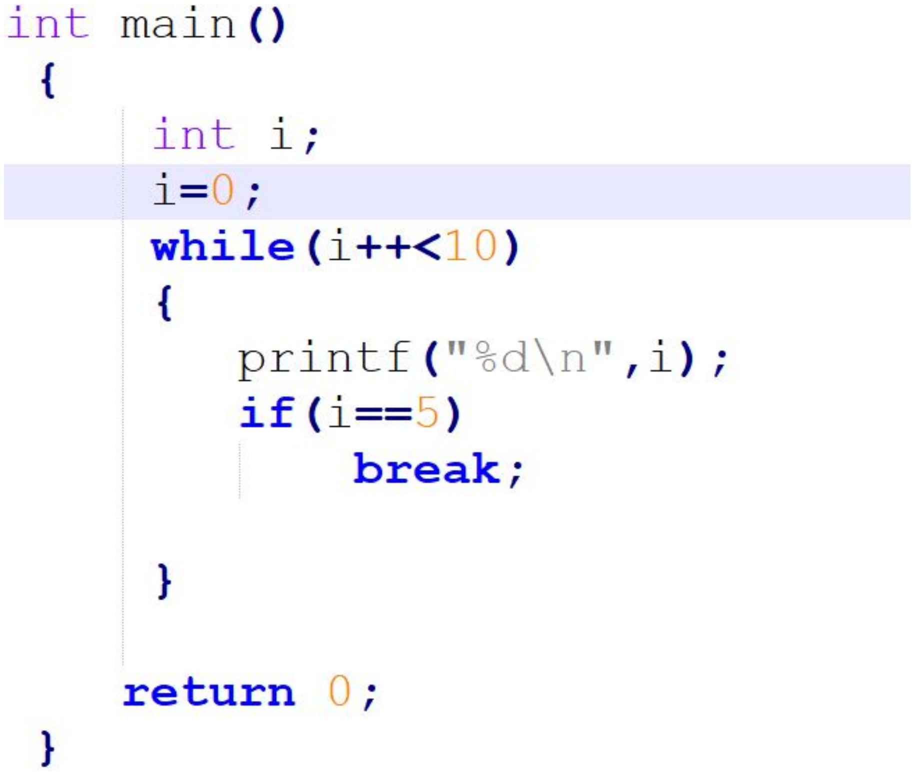
```
int main()
 {
     int n;
     int Result=1;
     printf("Enter n value " );
     scanf("%d",&n);
     while(n>=1)
     {
         Result*=n;
         n--;

     }
     printf("The result is : %d",Result);

    return 0;
 }
```

Break and Continue

- The break and continue statements are used to alter the flow of control.
- The ‘break’ statement: terminates a loop under some special condition
- The ‘continue’ statement: skips a section of the loop body in an iteration.
- The ‘break’ statement in a ‘switch’, ‘while’, ‘do-while’ or ‘for’ structure causes immediate exit from the structure

Break and Continue

- What would be displayed by the following program?



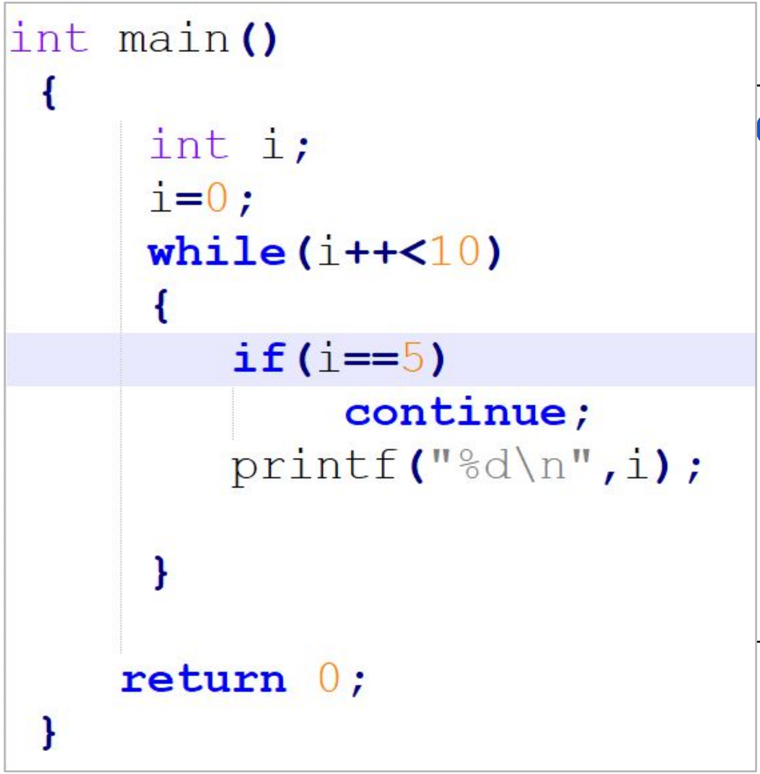
```
int main()
 {
     int i;
     i=0;
     while(i++<10)
     {
        printf("%d\n",i);
        if(i==5)
            break;

     }

    return 0;
 }
```

Output:

```
1
2
3
4
5
```

Break and Continue

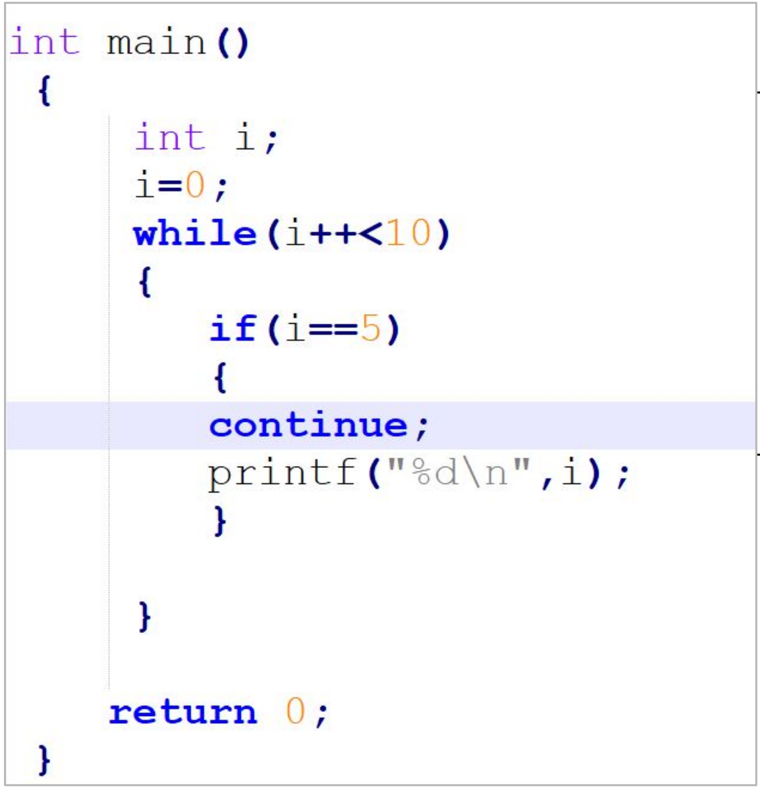
- What would be displayed by the following program?



```c
int main()
 {
      int i;
      i=0;
      while(i++<10)
      {
          if(i==5)
              continue;
          printf("%d\n",i);

      }

    return 0;
 }
```

Output:

```
1
2
3
4
6
7
8
9
10
```



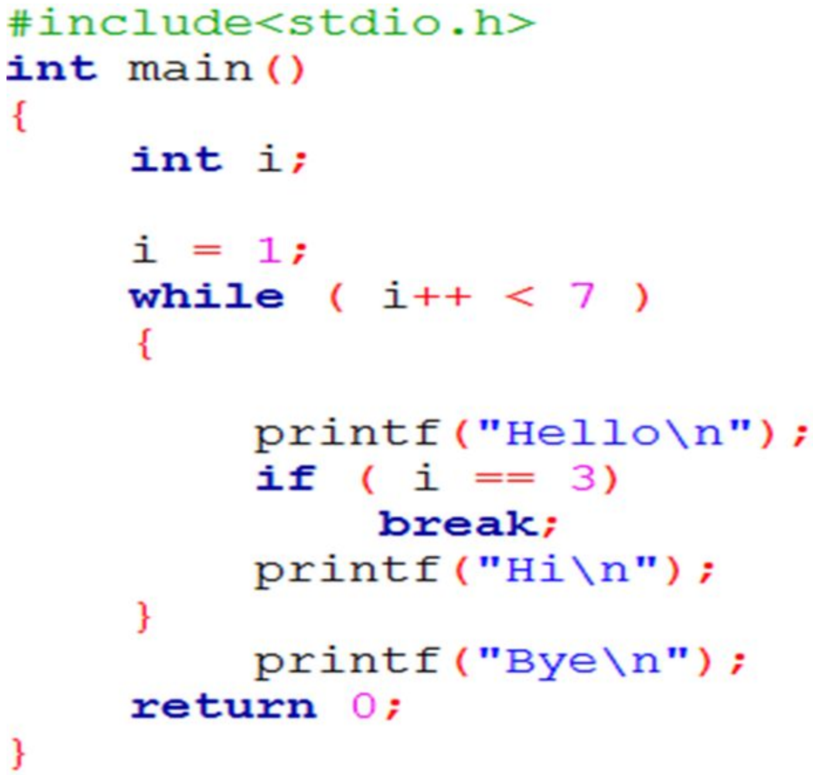
```c
int main()
 {
      int i;
      i=0;
      while(i++<10)
      {
          if(i==5)
          {
          continue;
          printf("%d\n",i);
          }

      }

    return 0;
 }
```

Output:

????

Break and Continue

- What would be displayed by the following program?



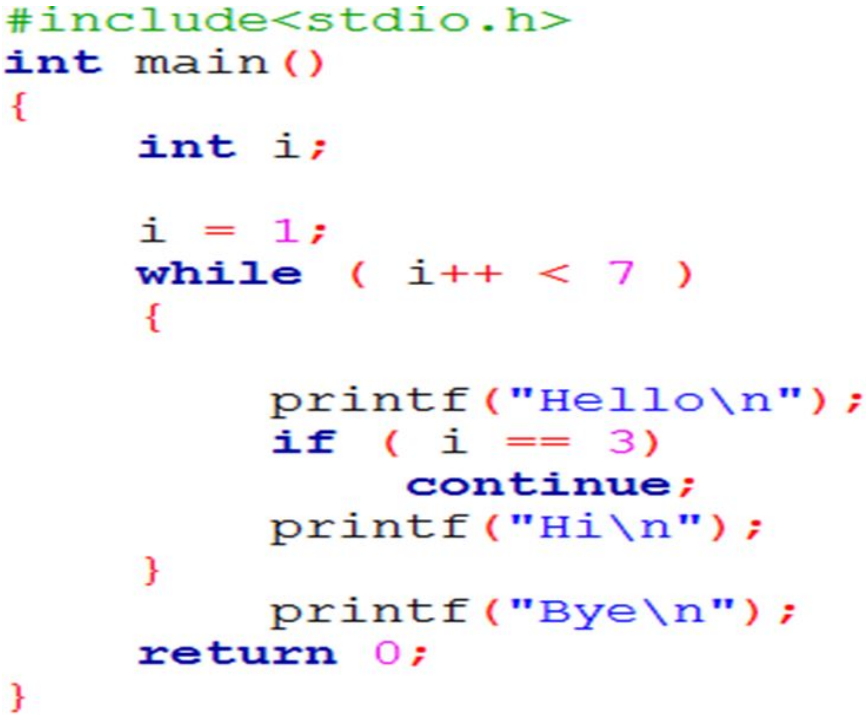
```
#include<stdio.h>
int main()
{
    int i;

    i = 1;
    while ( i++ < 7 )
    {

        printf("Hello\n");
        if ( i == 3)
            break;
        printf("Hi\n");
    }
        printf("Bye\n");
    return 0;
}
```

Output:

```
Hello
Hi
Hello
Bye
```

Break and Continue

- What would be displayed by the following program?



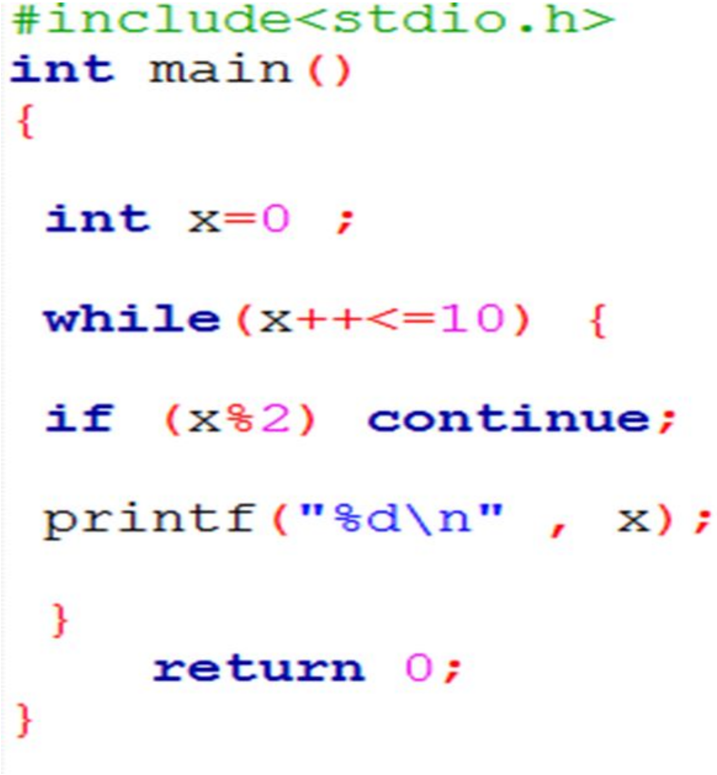
```c
#include<stdio.h>
int main()
{
    int i;

    i = 1;
    while ( i++ < 7 )
    {

        printf("Hello\n");
        if ( i == 3)
            continue;
        printf("Hi\n");
    }
        printf("Bye\n");
    return 0;
}
```

Output:

```
Hello
Hi
Hello
Hello
Hi
Hello
Hi
Hello
Hi
Hello
Hi
Bye
```

Break and Continue

- What would be displayed by the following program?



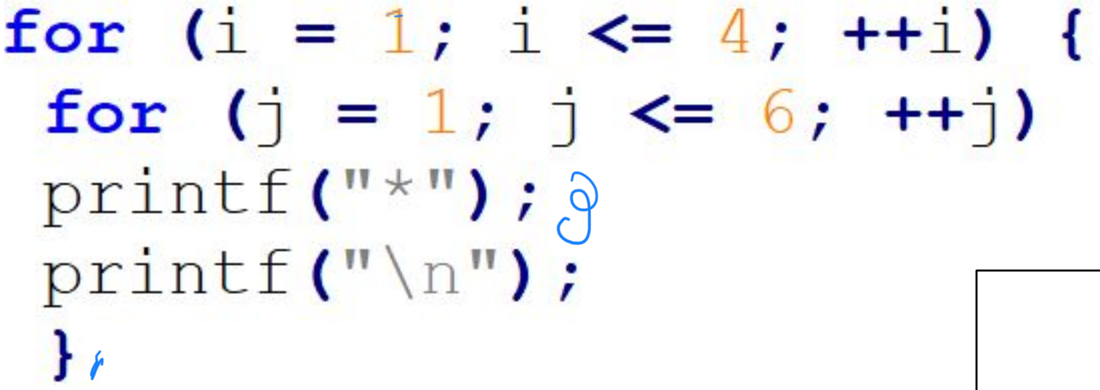
```c
#include<stdio.h>
int main()
{

 int x=0 ;

 while(x++<=10) {

 if (x%2) continue;

 printf("%d\n" , x);

  }
     return 0;
}
```

Output:

```
2
4
6
8
10
```

Nested Loop

- What would be displayed by the following program?



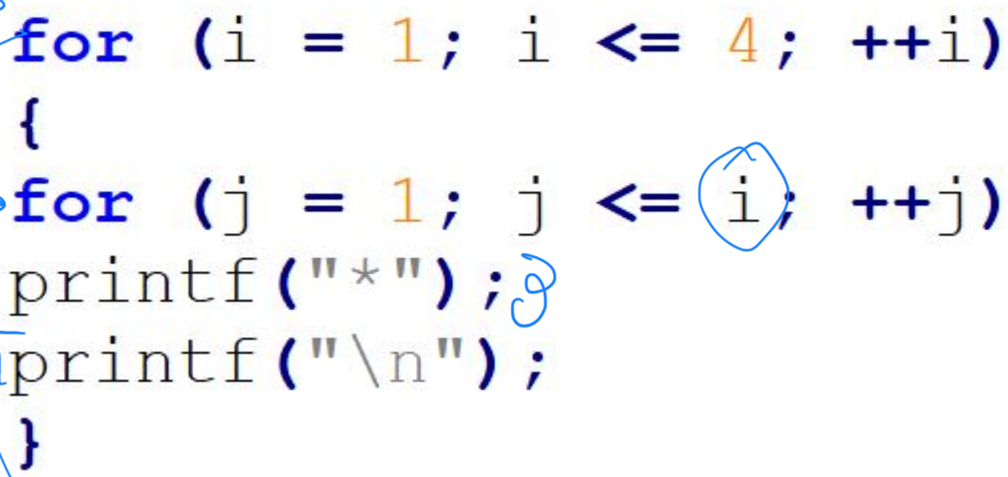
```
for (i = 1; i <= 4; ++i) {
 for (j = 1; j <= 6; ++j)
 printf("*");
 printf("\n");
 }
```

Output:

```
1 ******
2 ******
3 ******
4 ******
```

Nested Loop

- What would be displayed by the following program?



```
for (i = 1; i <= 4; ++i)
{
for (j = 1; j <= i; ++j){
printf("*");}
printf("\n");
}
```

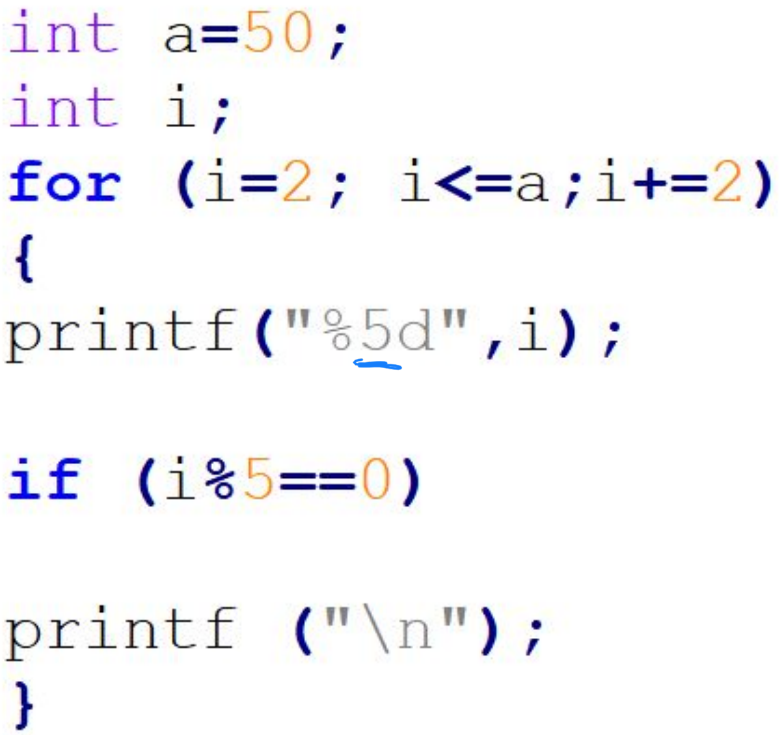
Output:

```
*       (1) /n
**      (2) /n
***     (3) /n
****    (4) /n
```

Nested Loop

- What would be displayed by the following program?

i = i+2



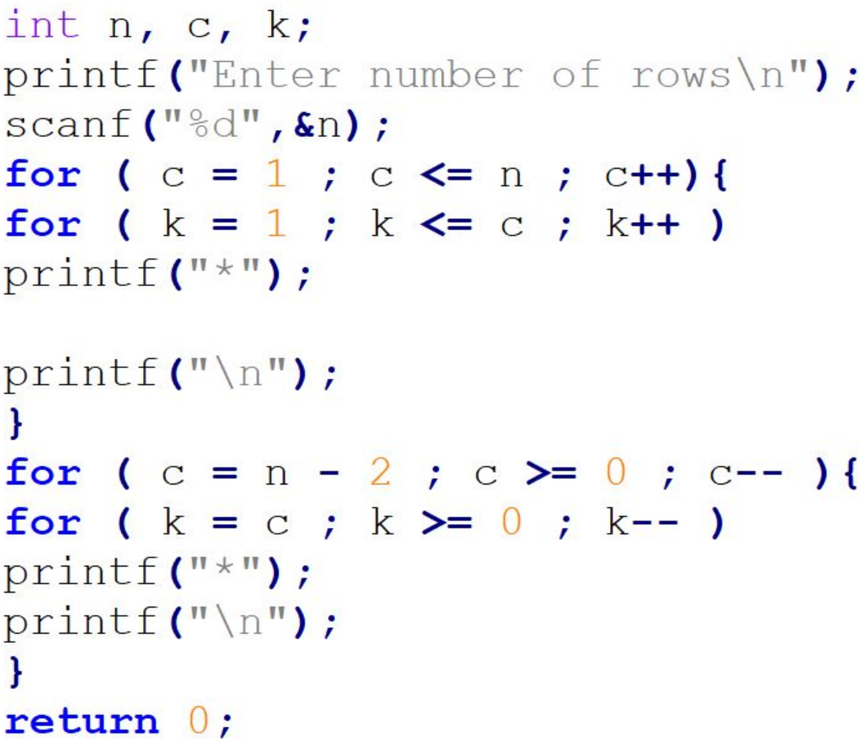
```
int a=50;
int i;
for (i=2; i<=a;i+=2)
{
printf("%5d",i);

if (i%5==0)

printf ("\n");
}
```

Output:

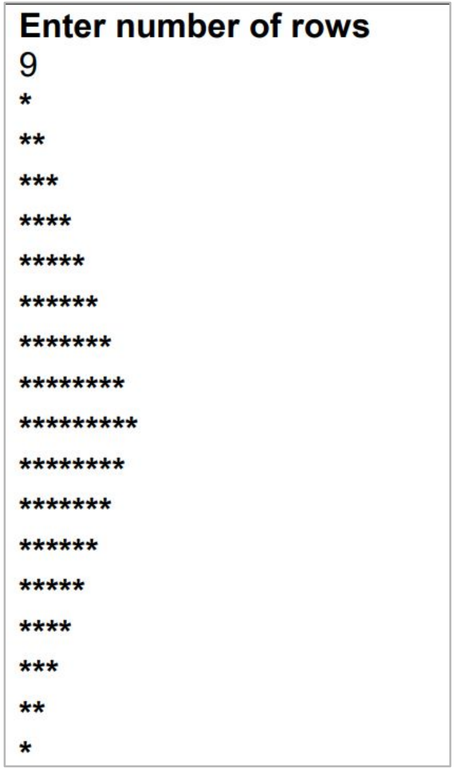
```
2 4 6 8 10
12 14 16 18 20
22 24 26 28 30
32 34 36 38 40
42 44 46 48 50
```

Nested Loop

- What would be displayed by the following program?



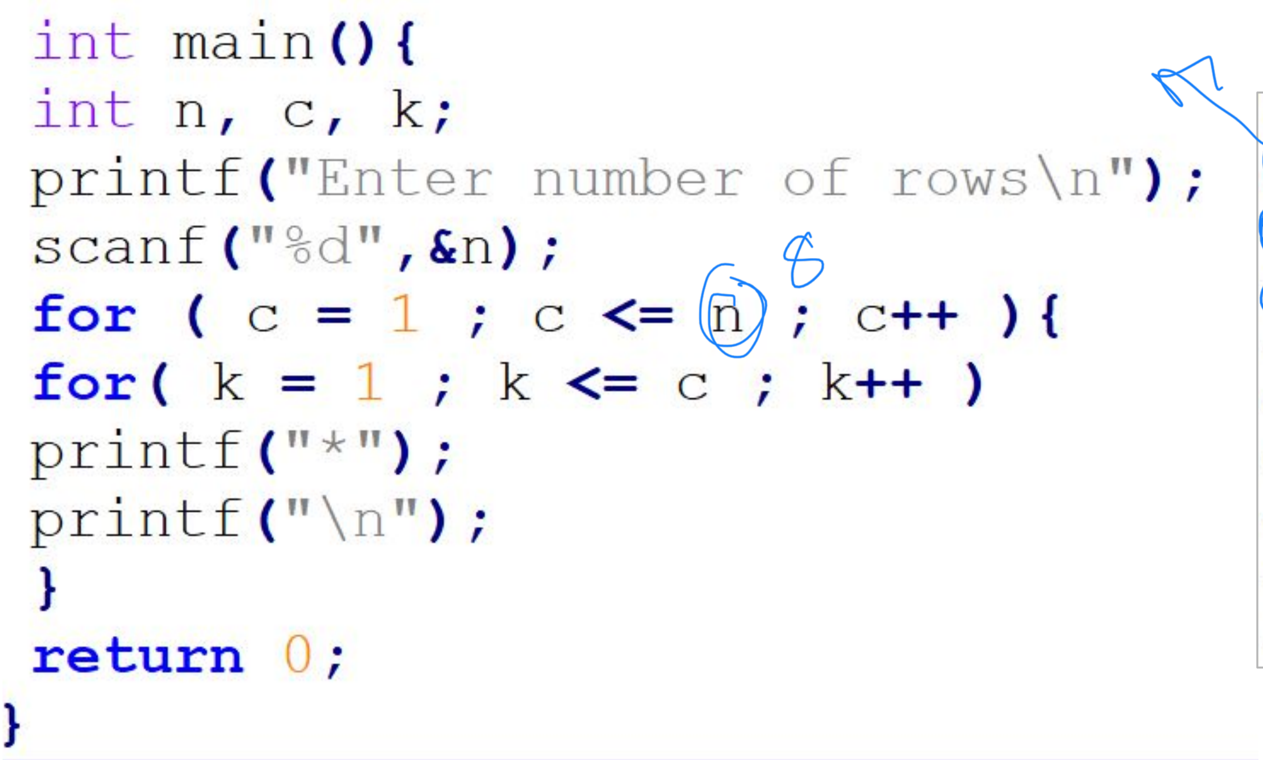
```
int n, c, k;
printf("Enter number of rows\n");
scanf("%d",&n);
for ( c = 1 ; c <= n ; c++){
for ( k = 1 ; k <= c ; k++ )
printf("*");

printf("\n");
}
for ( c = n - 2 ; c >= 0 ; c-- ){
for ( k = c ; k >= 0 ; k-- )
printf("*");
printf("\n");
}
return 0;
```



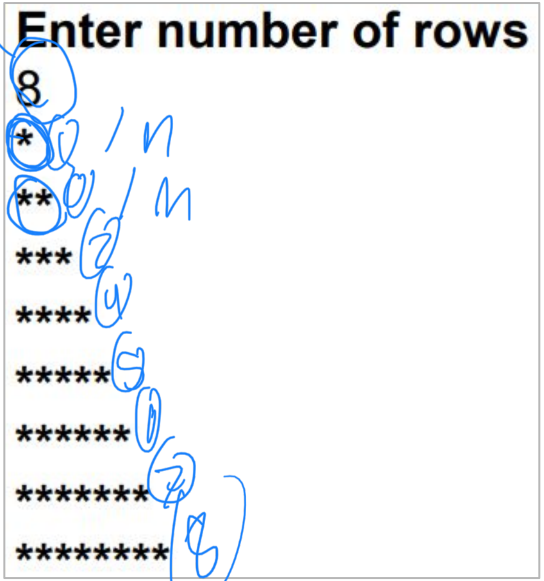
```
Enter number of rows
9
*
**
***
****
*****
******
*******
********
*********
********
*******
******
*****
****
***
**
*
```

Nested Loop

- What would be displayed by the following program?



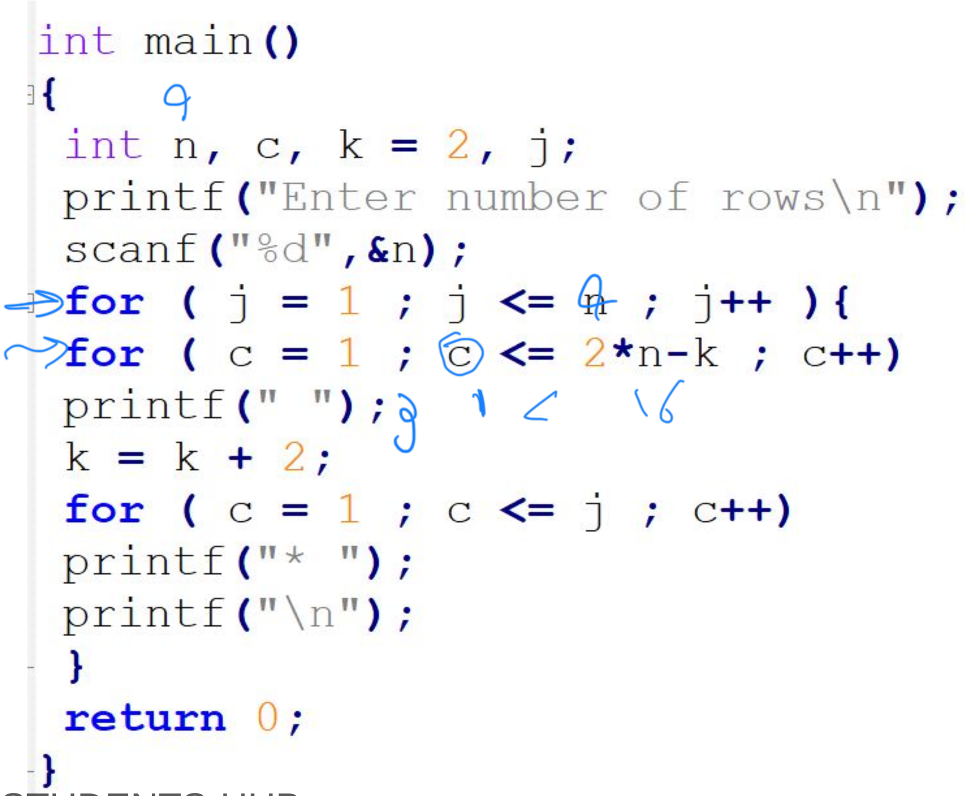
```
int main(){
int n, c, k;
printf("Enter number of rows\n");
scanf("%d",&n);
for ( c = 1 ; c <= n ; c++ ){
for( k = 1 ; k <= c ; k++ )
printf("*");
printf("\n");
}
return 0;
}
```



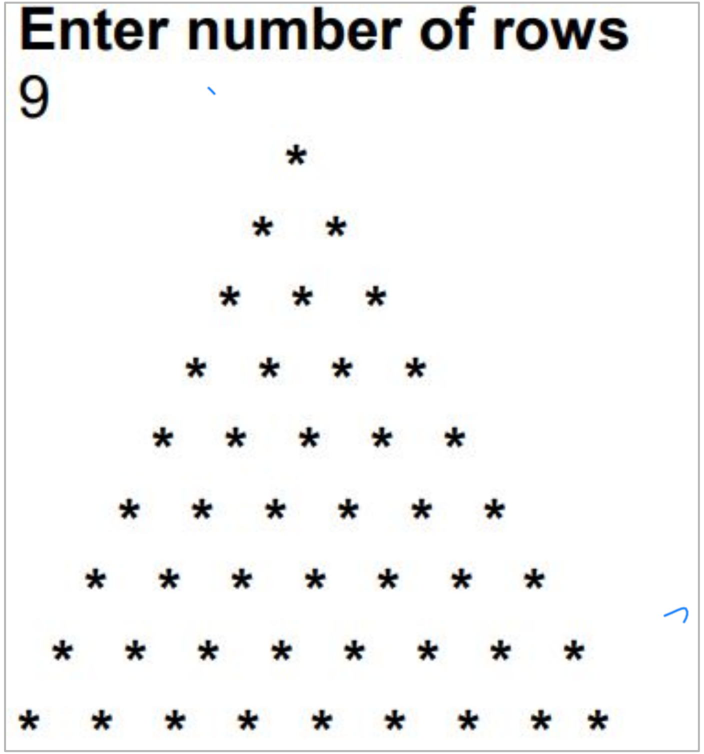
```
Enter number of rows
8
*
**
***
****
*****
******
*******
********
```

Nested Loop

- What would be displayed by the following program?

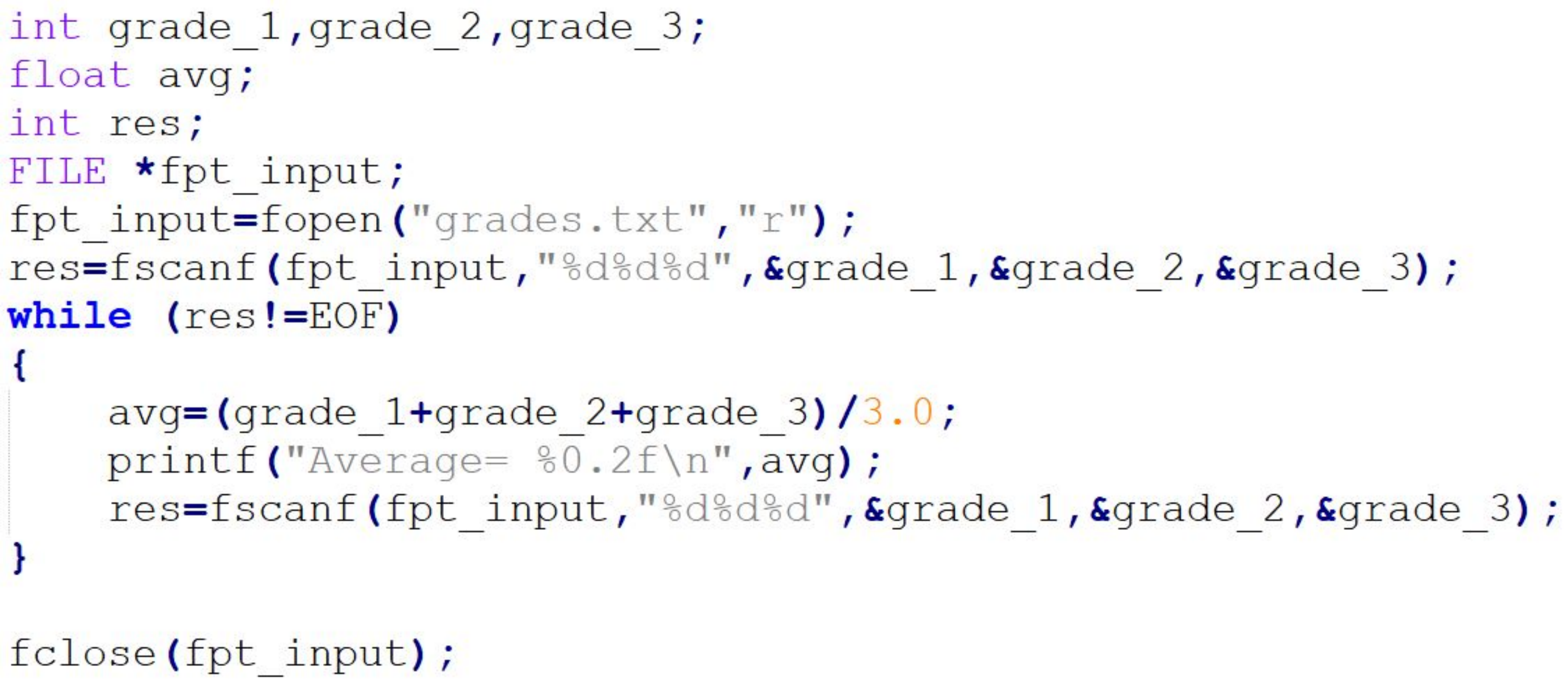


```
int main()
{
 int n, c, k = 2, j;
 printf("Enter number of rows\n");
 scanf("%d",&n);
 for ( j = 1 ; j <= n ; j++ ){
 for ( c = 1 ; c <= 2*n-k ; c++)
 printf(" ");
 k = k + 2;
 for ( c = 1 ; c <= j ; c++)
 printf("* ");
 printf("\n");
 }
 return 0;
}
```



```
Enter number of rows
9
        *
       * *
      * * *
     * * * *
    * * * * *
   * * * * * *
  * * * * * * *
 * * * * * * * *
* * * * * * * * *
```

End Of File



```
int grade_1,grade_2,grade_3;
float avg;
int res;
FILE *fpt_input;
fpt_input=fopen("grades.txt","r");
res=fscanf(fpt_input,"%d%d%d",&grade_1,&grade_2,&grade_3);
while (res!=EOF)
{
    avg=(grade_1+grade_2+grade_3)/3.0;
    printf("Average= %0.2f\n",avg);
    res=fscanf(fpt_input,"%d%d%d",&grade_1,&grade_2,&grade_3);
}

fclose(fpt_input);
```

Thank You.

BIRZEIT UNIVERSITY