



Repetition and Loop Statements

Comp230

Loops

- ▶ The repetition of steps in a program is called loop.
- ▶ Three C loop control statement:
 - ▶ while
 - ▶ for
 - ▶ do-while

Loops: Controlling Loop

- ▶ **Counter controlled loops:** control variable counting up/down (normal loops).
- ▶ **Event controlled loops:** until special value is encountered. (E.g., terminate loop when input is 'q' , or terminate loop when input is 0).
- ▶ **Result controlled loops:** continues until a test determines that the desired result is reached (e.g., numerical approximations)

Loop : While Loop

Initialization

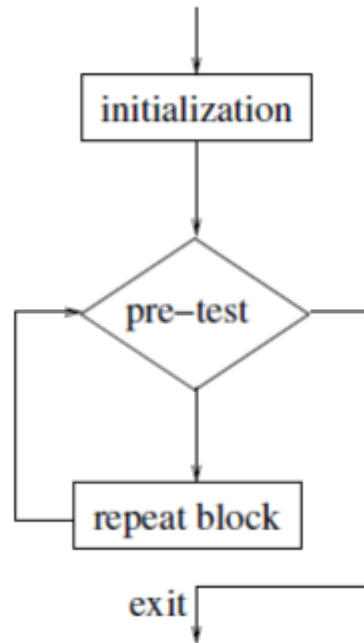
while (condition)

{

statement(s);

update statement;

}



Loop : Counter Controlled While

```
# 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; }
```

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

Loop : Event controlled While

```
# include <stdio.h>
int main ( ) {
int sum=0, x;
printf (" Please, enter value or zero to stop ");
scanf ("%d", &x); // Reading integer
while ( x != 0) // Exit the on reading a zero
{
    sum + = x; // add the value to sum
    printf (" Please, enter next value or zero to stop ");
    scanf ("%d", &x); // Reading integer
}
if (sum)
    printf (" Sum = %d ", sum);
else
    printf ("The first input is zero");
return 0;
}
```

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.

Loop : Result controlled while

```
# include <stdio.h>

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

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.

Compound Assignment Operators (Assignment Shorthands)

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

Pre and Post-Increment

- ▶ `++x` // Pre-increment x
- ▶ `x++` // Post-increment x

Example (Pre-increment):

- ▶ `a = ++x * b;` →

```
x = x + 1;  
a = x * b;
```

Pre and Post-Increment

- ▶ ++x // Pre-increment x
- ▶ x++ // Post-increment x

Example (Post-increment):

- ▶ $a = x++ * b; \rightarrow$

```
a = x * b;  
x = x + 1;
```

Pre and Post-Decrement

- ▶ `--x` // Pre-decrement x
- ▶ `x--` // Post-decrement x

Example (Pre-decrement):

- ▶ `a = --x * b;` →

```
x = x - 1;  
a = x * b;
```

Pre and Post-Decrement

- ▶ `--x` // Pre-decrement x
- ▶ `x--` // Post-decrement x

Example (Post-decrement):

- ▶ `a = x-- * b; →`

```
a = x * b;  
x = x - 1;
```

Examples

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

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

Examples

```
int a=2,b=3,c=0;  
c += --a * b++;
```

Find a,b,c ?

a=1 , b=4, and c = 3

Examples

```
int a=4,b=3,c=20;  
c /= ++a;  
Find a, b, c ?
```

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

Examples

```
int a=2,b=3,c=4;  
c *= ++a * b++;  
Find a, b, c ?
```

a=3 , b=4, and c = 36

Examples

```
int i = 1;  
while (i < 5)  
    printf ("%d " , i++);
```

- What is the output?
- What is the final value of i?

Output
1 2 3 4

Final value of i
i=5

Examples

- ▶ Write a program to find if an entered number is perfect or not?
- ▶ **Hint:** perfect number is a positive integer that is equal to the sum of its proper positive divisors, that is, the sum of its positive divisors excluding the number itself.
- ▶ **Example (1) :** The first perfect number is 6, because 1, 2, and 3 are its proper positive divisors, and $1 + 2 + 3 = 6$
- ▶ **Example (2) :** The next perfect number is $28 = 1 + 2 + 4 + 7 + 14$

```
#include <stdio.h>
// Function Prototype
int isPerfectNumber(int);
int main() {
    int user_input;
    printf("Enter a number: ");
    scanf("%d", &user_input);

    if (isPerfectNumber(user_input)) {
        printf("%d is a perfect number.\n", user_input);
    } else {
        printf("%d is not a perfect number.\n", user_input);
    }
    return 0;
}
// Function Definition
int isPerfectNumber(int number) {
    int i = 1, sum = 0;
    while (i < number) {
        if (number % i == 0) {
            sum += i;
        }
        i++;
    }
    return (sum == number);
}
```

Examples

- Write a program to find x^y ?
- Example: $2^3 = 8$

```
#include <stdio.h>
int main()
{
    int x,y;
    int result=1;
    printf("please enter x and y: ");
    scanf ("%d%d",&x,&y);
    while (y>=1)
    {
        result*=x;
        y--;
    }
    printf ("result is %d",result);

    return 0;
}
```

Examples

- Write a program to find x^y ?
- Example: $2^3 = 8$

```
#include <stdio.h>
int main()
{
    int x,y;
    int result=1;
    printf("please enter x and y: ");
    scanf ("%d%d",&x,&y);
    while (y-->=1)
    {
        result*=x;
    }
    printf ("result is %d",result);

    return 0;
}
```

Examples

- Write a program to find $n!$?
- Example: $4! = 24$

```
#include <stdio.h>
int main()
{
    int n;
    int result=1;
    printf("please enter a number: ");
    scanf ("%d",&n);
    while (n>=1)
    {
        result*=n;
        n--;
    }
    printf ("result is %d",result);

    return 0;
}
```

Break and Continue

- ▶ break statement
 - A break statement **takes the control out of the loop**.
 - When break is encountered inside any loop, control automatically passes to the first statement after the loop.
 - A break is usually associated with an if.
- ▶ continue statement
 - continue statement **take the control to the beginning of the loop**, bypassing the statements inside the loop, which have not yet been executed

Break and Continue: Examples

- break statement
- What would be displayed by the following program?

```
#include<stdio.h>
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: Examples

- ▶ continue statement
- ▶ What would be displayed by the following program?

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

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

Output

1
2
3
4
5
6
7
8
9
10

Break and Continue: Examples

- ▶ break statement
- ▶ 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: Examples

- ▶ continue statement
- ▶ 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 )
            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: Examples

- ▶ break statement
- ▶ What would be displayed by the following program?

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

    i = 1;
    while ( i++ < 5 )
    {
        printf("%d\n", ++i);
        if ( i == 3 )
            break;
        printf("%d\n", i);
    }
    printf("%d\n", ++i);
    return 0;
}
```

Output

3
4

Break and Continue: Examples

- ▶ continue statement
- ▶ What would be displayed by the following program?

```
#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

Break and Continue: Examples

- ▶ break statement
- ▶ What would be displayed by the following program?

```
#include<stdio.h>
int main()
{
    int x=0 ;

    while(x++<=10) {
        if (x%2) break;
        printf("%d\n" , x);
    }

    return 0;
}
```

Output

The for Statement

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

Normal forms are:

```
for(i = 0; i < 10; i++) {...}
```

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

expr1 : initialization expression

expr2 : loop repetition condition

expr3 : update statement

When **expr1** is omitted: loop index should be initialized before entry into loop.

When **expr3** is omitted, loop index should be incremented inside the loop.

When **expr2** is omitted, loop becomes infinite loop unless break occurs inside the loop.

The for Statement

```
1.  /*
2.  * Computes n!
3.  * Pre: n is greater than or equal to zero
4.  */
5.  int
6.  factorial(int n)
7.  {
8.      int i,          /* local variables */
9.      product;        /* accumulator for product computation */
10.
11.     product = 1;
12.     /* Computes the product n x (n-1) x (n-2) x ... x 2 x 1 */
13.     for (i = n; i > 1; --i) {
14.         product = product * i;
15.     }
16.
17.     /* Returns function result */
18.     return (product);
19. }
```


The for Statement

```
#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

The for Statement: Nested Loop

- What would be the output of the following code ?

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





int main()
{
    int i,j;

    for (i=0;i<3;i++)
        for (j=0;j<2;j++)
            printf("i=%d    j=%d\n",i,j);
    return 0;
}
```

```
output
i=0  j=0
i=0  j=1
i=1  j=0
i=1  j=1
i=2  j=0
i=2  j=1
```

The for Statement: Nested Loop

- Write a program to display the following outputs :

<pre> * *** ***** *****</pre>	<pre> * *** ***** *****</pre>	<pre>***** ***** ***** *** *</pre>	<pre>***** ***** ***** *** *</pre>
1	2	3	4
			

do-while loop

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

- ▶ Statement is executed first, and then expression is evaluated
- ▶ If expression is TRUE, the statement is executed again
- ▶ If expression is FALSE, the loop terminates
- ▶ In general, do-while loops are less frequently used

do-while loop

- Print all numbers between 1 and 100 that are divisible by 7

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

int main()
{
    int x =1;
    do
    {
        if ((x % 7) == 0)
            printf("%d\n", x);
        x++;
    }
    while (x<=100);
}
```

Example: while loop

- Write a c program to find out sum of digit of given number

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

int main()
{
    int num;
    int sum=0;
    printf("Please enter a number: ");
    scanf ("%d",&num);
    while (num>0)
    {
        sum+=num%10;
        num=num/10;
    }
    printf ("the sum is %d",sum);
    return 0;
}
```

Example: for loop

- Write a c program to find out sum of digit of given number

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

int main()
{
    int num;
    int sum=0;
    printf("Please enter a number: ");
    scanf ("%d",&num);
    for (;num>0; num=num/10)
    {
        sum+=num%10;
    }
    printf ("the sum is %d",sum);
    return 0;
}
```

Example

- Convert the following while loop to a for loop

```
int x = 5;
while ( x < 50 )
{
    printf("%d",x);
    x++;
}
```

```
for (x = 5; x < 50; x++)
    printf("%d",x);
```


Example

- Convert a following for loop to a while loop

```
for (x = 50; x > 5; x--)  
    printf("%d",x);
```

```
x = 50;  
while ( x > 5)  
{  
    printf("%d",x);  
    x--;  
}
```

Example

- What would be the output of the following code ?

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

int main()
{
    int balance = 29;
    while ( 5 )
    {
        if (balance < 9)
            break;
        balance = balance - 9;
    }
    printf("%d", balance);
    return 0;
}
```

Output
2

Example

- What would be the output of the following code ?

```
#include <stdio.h>

int main()
{
    int i = 10;

    do
    {
        printf("Hello %d\n", i );
        i = i -1;
    }
    while ( i > 0 );
    return 0;
}
```

Output

```
Hello 10
Hello 9
Hello 8
Hello 7
Hello 6
Hello 5
Hello 4
Hello 3
Hello 2
Hello 1
```

Extra Exercises

- Input a range from user and print all the magic numbers in that range. A number is magical if repeated adding of its digit gives 1. Example 19 is magical as $1 + 9 = 10$, $1 + 0 = 1$ hence magical. So is 991 as $9 + 9 + 1 = 19$, $1 + 9 = 10$, $1 + 0 = 1$. However 224 is not.

```
#include <stdio.h>

int sumOfDigits(int);
int isMagicNumber(int);
int main() {
    int start, end;
    printf("Enter the starting number of the range: ");
    scanf("%d", &start);
    printf("Enter the ending number of the range: ");
    scanf("%d", &end);
    printf("Magic numbers in the range %d to %d are:\n", start, end);
    for (int i = start; i <= end; i++) {
        if (isMagicNumber(i)) {
            printf("%d\n", i);
        }
    }
    return 0;
}
```

```
int sumOfDigits(int num) {
    int sum = 0;
    while (num > 0) {
        sum += num % 10;
        num /= 10;
    }
    return sum;
}

int isMagicNumber(int num) {
    while (num >= 10) {
        num = sumOfDigits(num);
    }
    return (num == 1);
}
```

Extra Exercises

- Input a range from user and print all the narcissistic number in that range.
Hint: A number is called narcissistic if each of its digits raised to the power of the number of digits equals the number. Example : 153 is a narcissistic number since $1^3 + 5^3 + 3^3 = 1 + 125 + 27 = 153$. $1634 = 1^4 + 6^4 + 3^4 + 4^4$

```

#include <stdio.h>
#include <math.h>
int countDigits(int);
int isNarcissistic(int);
int main() {
    int start, end;
    printf("Enter the starting number of the range: ");
    scanf("%d", &start);
    printf("Enter the ending number of the range: ");
    scanf("%d", &end);
    printf("Narcissistic numbers in the range %d to %d
are:\n", start, end);
    for (int i = start; i <= end; i++) {
        if (isNarcissistic(i)) {
            printf("%d\n", i);
        }
    }
    return 0;
}

```

```

int countDigits(int num) {
    int count = 0;
    while (num != 0) {
        num /= 10;
        count++;
    }
    return count;
}

int isNarcissistic(int num) {
    int originalNum = num;
    int numDigits = countDigits(num);
    int sum = 0;
    while (num > 0) {
        int digit = num % 10;
        sum += pow(digit, numDigits);
        num /= 10;
    }
    return (sum == originalNum);
}

```

Extra Exercises

- Write a program that will read an unspecified numbers of integers from keyboard, determine how many even and how many odd numbers have been read. The program should also compute the average of the integers read. The program should display the number of odd integers, the number of even integers; and the average. Your program should stop when user enters 0


```
#include <stdio.h>

int main() {
    int num, sum = 0, countEven = 0, countOdd = 0;
    printf("Enter integers (enter 0 to stop):\n");
    do {
        scanf("%d", &num);
        if (num != 0) {
            sum += num;
            if (num % 2 == 0) {
                countEven++;
            } else {
                countOdd++;
            }
        }
    } while (num != 0);
    printf("Number of even integers: %d\n", countEven);
    printf("Number of odd integers: %d\n", countOdd);
    if (countEven + countOdd > 0) {
        double average = (double)sum / (countEven + countOdd);
        printf("Average of integers: %.2f\n", average);
    } else {
        printf("No numbers entered.\n");
    }
    return 0;}
}
```

