BIRZEIT UNIVERSITY

Repetition and Loop Statements

Computer Science Department

Loops

- The *repetition of steps* in a program is called loop.

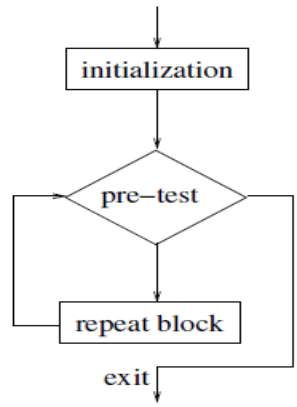
- Three C loop control statement:
  - While
  - for
  - do-while

Loops: Controlling Loop

- **Counter controlled loops: control variable counting up/down (normal loops).**
- **Sentinel - controlled loops (Event): until special value is encountered. (E.g., terminate loop when input is 'q' , or terminate loop when input is 0).**
- **Result - controlled loops : their cunt is unknown. Based on a certain calculation, the loop will be stopped.**

Loop : While Loop

```
while ( condition )
{
    body
}
```



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");
  return 0;
}
```

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

Loop : Sentinel 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

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.

```
# include  <stdio. h>
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;
}
```

Compound Assignment Operators

- C provides special assignment operators for instances of assignment statements of the form:

  variable = variable op expression;

  op is a C arithmetic operator (+, -, *, /, and %)
- Alternative form :

  variable op = expression;

- These include:
  - *increments and decrements of loop counters :*

    counter = counter + 1;

    time = time - 1;
  - Statements accumulating *a sum or computing a product* in a loop:

    total = total + pay;

    product = product * item;

(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
- 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;→

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

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

Examples

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

Find a,b,c ?
```

a = a -1;
c = c + a * b
b = b + 1

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

Examples

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

a = a + 1;
c = c / a;

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

Code

Code – perfect.c

```
#include <stdio.h>
int is_perfect (int);

int main()
{
    int number;
    printf("Please enter a number: ");
    scanf("%d",&number);
    if (is_perfect(number))
        printf("%d is perfect",number);
    else
        printf("%d is not perfect",number);

    return 0;
}
```

```
int is_perfect (int num)
{
    int sum=0;
    int i=1;
    while (num>i)
    {
        if (num%i==0)
            sum+=i;
        i++;

    }
    if (sum==num)
        return 1;
    else
        return 0;

}
```

Examples

Write a program to find x^y ?

Example: $2^3 = 8$

```c
#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: $2^3 = 8$

```c
#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

```c
#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

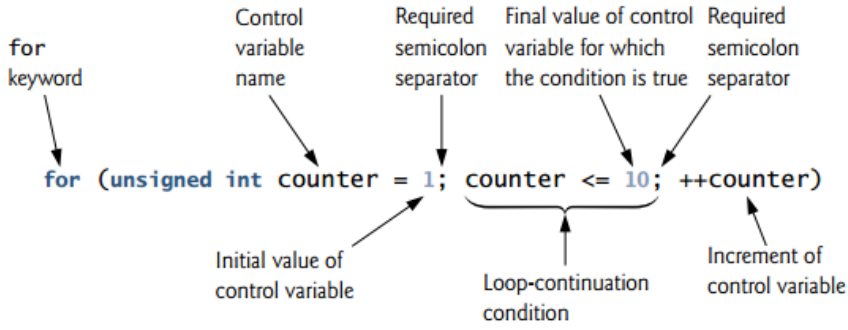
What would be displayed by the following program?

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

 int x=0 ;

 while(x++<=10) {

 if (x%2) break;

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

 }
    return 0;
}
```

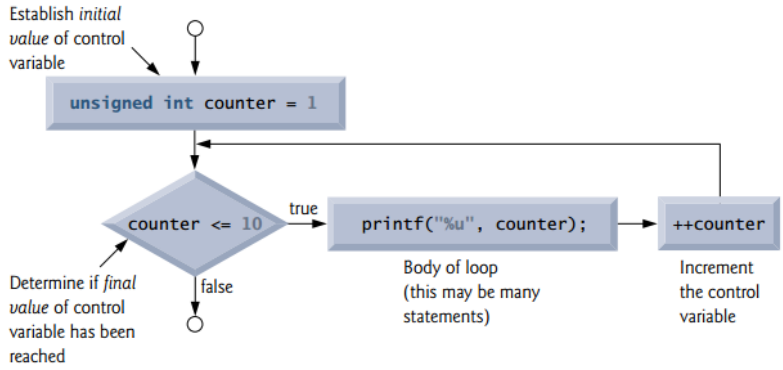
Output

For Statement

For Statement Header Components



Flowchart of For-Loop



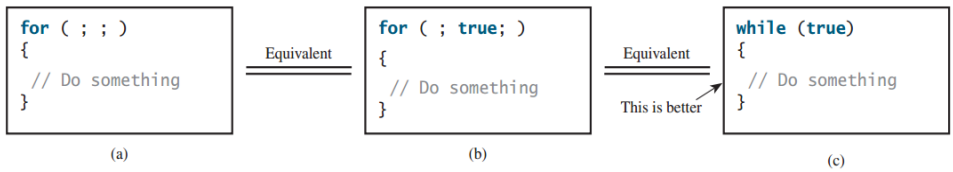
The for Statement

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

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.

If **expr2** is omitted, it is implicitly true. (a) is an *infinite loop*, is the same as in (b).
It is better to use the equivalent loop in (c).



(a) (b) (c)

Examples Using the for Statement

1.Vary the control variable from 1to 100 in increments of 1.

```
for (unsigned int i = 1; i <= 100; ++i)
```

2.Vary the control variable from 100to 1 in increments of -1(i.e., decrements of 1).

```
for (unsigned int i = 100; i >= 1; --i)
```

3.Vary the control variable from 7to 77 in increments of 7.

```
for (unsigned int i = 7; i <= 77; i += 7)
```

4.Vary the control variable from 20to 2 in increments of -2.

```
for (unsigned int i = 20; i >= 2; i -= 2)
```

5.Vary the control variable over the following sequence of values: 2, 5, 8, 11, 14, 17.

```
for (unsigned int j = 2; j <= 17; j += 3)
```

6.Vary the control variable over the following sequence of values: 44, 33, 22, 11, 0.

```
for (unsigned int j = 44; j >= 0; j -= 11)
```

5-37

An Example of the `for` Loop

```
1.  /* Process payroll for all emp
2.  total_pay = 0.0;
3.  for  (count_emp = 0;
4.        count_emp < number_emp;
5.        count_emp += 1) {                    /* update                    */
6.       printf("Hours> ");
7.       scanf("%lf", &hours);
8.       printf("Rate > $");
9.       scanf("%lf", &rate);
10.      pay = hours * rate;
11.      printf("Pay is $%6.2f\n\n", pay);
12.      total_pay = total_pay + pay;
13. }
14. printf("All employees processed\n");
15. printf("Total payroll is $%8.2f\n", total_pay);
```

Initialization Expression

Loop repetition condition

Update Expression

`count_emp` is set to 0 initially.

`count_emp` should not exceed the value of `number_emp`.

`count_emp` is increased by one after each iteration.

Example 2 for 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. }
```

main.c

```
11  }
12  int
13  factorial(int n)
14  {
15  int i, /* local variables */
16   product; /* accumulator for product computation */
17
18   product = 1;
19   /* Computes the product n x (n-1) x (n-2) x . . . x 2 x 1 */
20    for (i = n; i > 1; --i) {
21          printf(" i = %d  ",i);
22          product = product * i;
23   }
24
25   /* Returns function result */
26   return (product);
27   }
```

Watches (new)

Function arguments		
n	5	
Locals		
i	5	
product	1	
n	5	int
i	5	int
product	1	int

Watches (new)

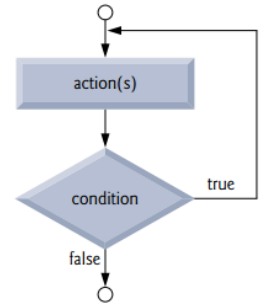
Function arguments		
n	5	
Locals		
i	2	
product	120	
n	5	int
i	2	int
product	120	int

```
Hello world!
 i = 5   i = 4   i = 3   i = 2
Result is 120.
```

do-while loop

do-while loop

```
do
Actions;
while (Condition);
```



- **Actions are executed first, and then condition in evaluated**
- **If condition is TRUE, the actions are executed again**
- **If condition is FALSE, the loop terminates**
- **In general, do-while loops are *less frequently used***

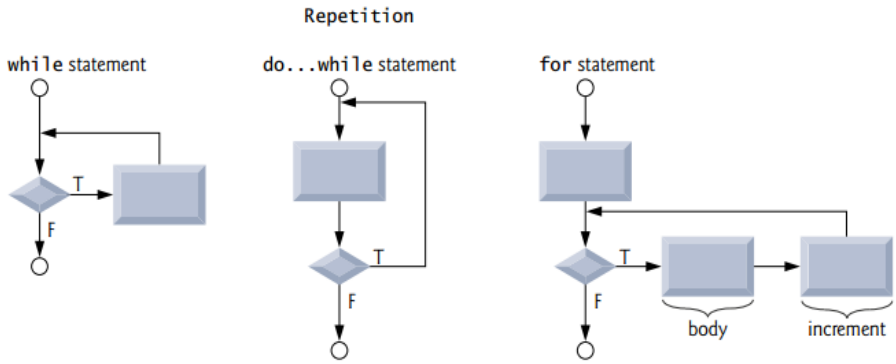
Do-while - Example

```
unsigned int counter = 1; // initialize counter

do {
   printf("%u  ", counter);
} while (++counter <= 10);
```

Output:

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

Repetition Statements



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

What would be the output of the following code ?

```c
#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

Example: while loop

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

```c
#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 - While

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

End-file-Controlled Loops

End-file-Controlled Loops

Repetition statement is very similar to the sentinel controlled loop that uses the status value returned by the scanning function to control repetition rather than using the values scanned.

1. Get the first *data value* and save *input status*
2. while *input status* does not indicate that end of file has been reached
    3. Process *data value*
    4. Get next *data value* and save *input status*

The loop repetition condition: **input_status != EOF**

input_status = scanf("%d%d%lf", &part_id, &num_avail, &cost);

scanf function returns as its value the number of data items scanned Here 3

Example: Write a C program that reads the integers stored in a text file

```c
#include <stdio.h>
int
main()
{
    int m = 0, n, k = 0;
    FILE *fptr;
    fptr = fopen("c:\\Code\\numbers.dat", "r");
    if (fptr != NULL)
    {
        printf("\nFile numbers.dat is opened successfully.");
        printf("\nContents of file numbers.dat:");
        m = fscanf(fptr, "%d", &n);

        while(m != EOF)
        {
            printf("%d ", n);
            m = fscanf(fptr, "%d", &n);
        }
        printf("\n");
        k = fclose(fptr);
        if(k == -1)
            printf("\nFile-closing failed");
        if(k == 0)
            printf("\nFile is closed successfully.");
    }
    else
        printf("\nFile-opening failed");
    return(0);
}
```

Nested Loop

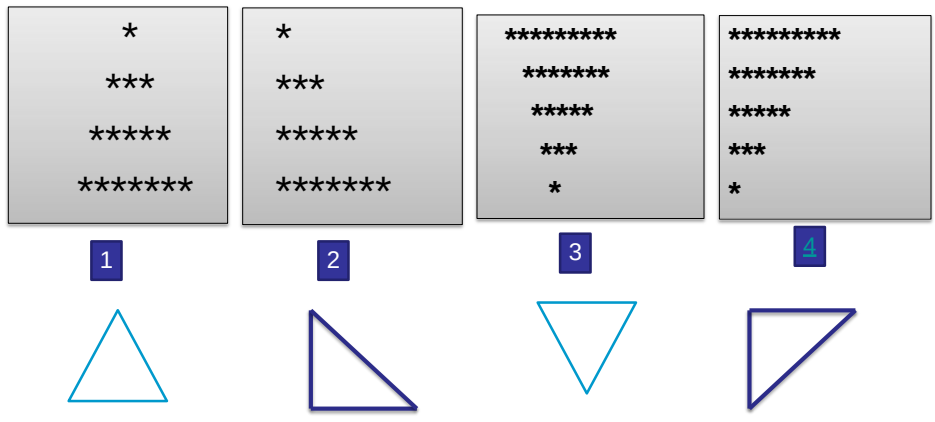
Nested Counting Loop Program

```
1.  /*
2.   * Illustrates a pair of nested counting loops
3.   */
4.
5.  #include <stdio.h>
6.
7.  int
8.  main(void)
9.  {
10.       int i, j;   /* loop control variables */
11.
12.       printf("           I    J\n");          /* prints column labels          */
13.
14.       for  (i = 1;  i < 4;  ++i) {            /* heading of outer for loop     */
15.           printf("Outer %6d\n", i);
16.           for  (j = 0;  j < i;  ++j) {        /* heading of inner loop         */
17.               printf("  Inner%9d\n", j);
18.           }  /* end of inner loop */
19.       }  /*  end of outer loop */
20.
21.       return (0);
22. }
```

```
            I   J
Outer       1
  Inner         0
Outer       2
  Inner         0
  Inner         1
Outer       3
  Inner         0
  Inner         1
  Inner         2
```

Exercises:
The for Statement: **Nested Loop**

Write a program to display the following outputs :



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.

Answer

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

Answer

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

Answer