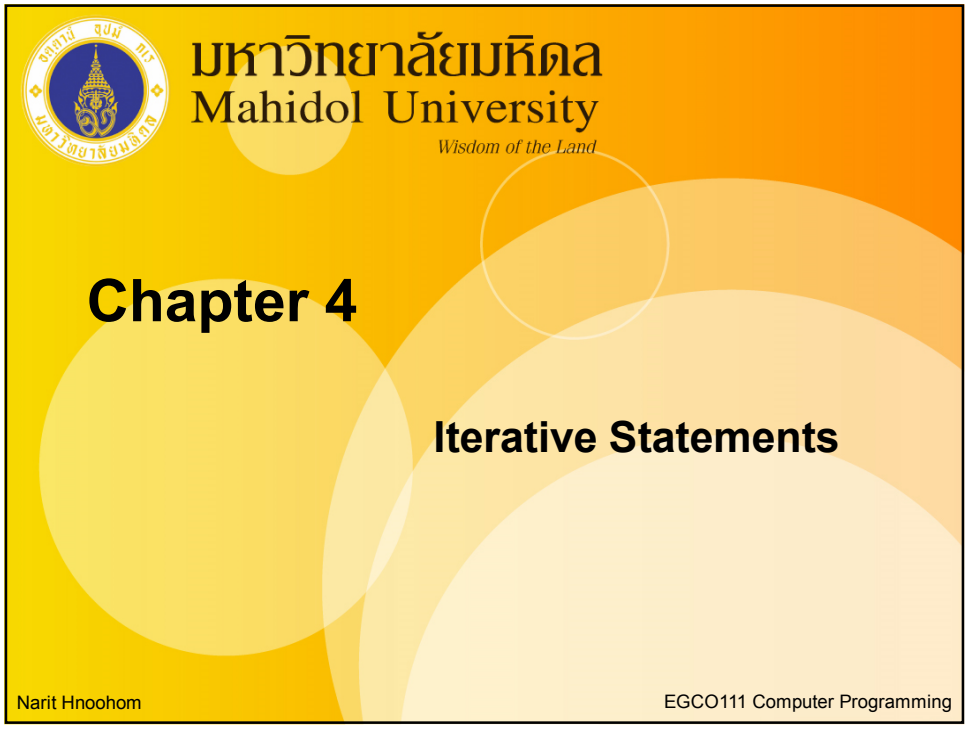


Compound Assignment Operator

Operator	Short form	Expand form

+=
-=
*=
/=
%=
	x += y;
	x -= y;
	x *= y;
	x /= y;
	x %= y;
		x = x + y;
		x = x - y;
		x = x * y;
		x = x / y;
		x = x % y;

- **Example**

Input	C Expression	Output

x=4, y=2;
x=4, y=2;
x=4, y=2;
x=4, y=2;
x=4, y=2;
	x += y;
	x -= y;
	x *= y;
	x /= y;
	x %= y;
		6
		2
		8
		2
		0

Boolean & Logical Operators

- **Boolean Operator**

Operator	Meaning	Example
==	equal	x==3
!=	not equal	x!=y
>	grater	x>2
<	less	x<5
>=	grater than or equal	x>=y
<=	less than or equal	x<=y

- **Logical Operator**

Operand		Not	AND	OR
A	B	A!	A&&B	A||B
1	1	0	1	1
1	0	0	0	1
0	1	1	0	1
0	0	1	0	0

- **Precedence**

1. Parentheses
2. Not, AND, OR

Boolean Expressions

- C uses "=" for assignments and "==" for comparisons.
- It returns 1 if operands are equal and 0 otherwise.

- Note:
  - There is no explicit Boolean type in traditional C.
  - Instead C uses integer values for Booleans, with the following meaning:
    false : value 0 (zero)
    true : any value except 0

- **Example**

```
printf("%d\n", 1==2); ⇒ 0
printf("%d\n", 1==1); ⇒ 1
```

Increment & Decrement Operators

- In C there is a short hand x++ for x=x+1 and x-- for x = x - 1.
- Also allowed are ++x and --x.

- Note difference between prefix and postfix form:

Prefix

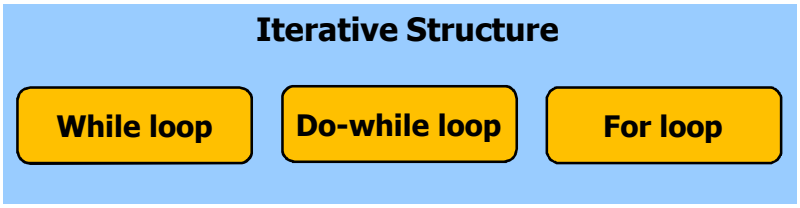
```
int x = 3;
int y;
y = x++;
printf("%d\n", x); ⇒ 4
printf("%d\n", y); ⇒ 3
```

Postfix

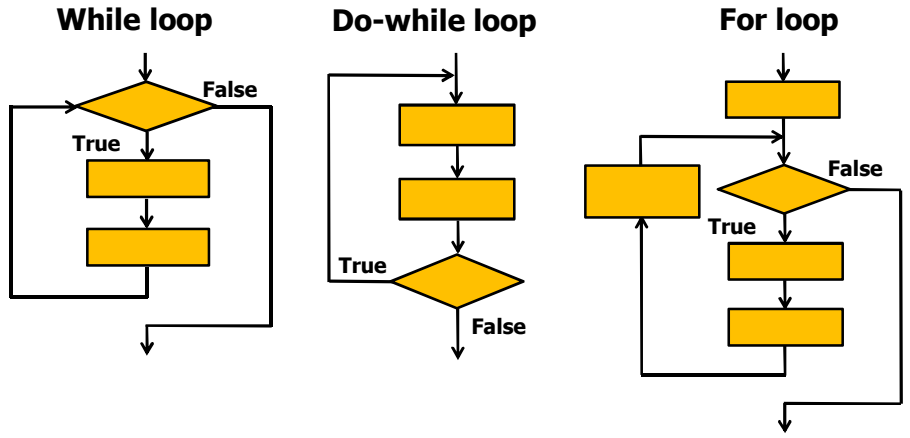
```
int x = 3;
int y;
y = ++x;
printf("%d\n", x); ⇒ 4
printf("%d\n", y); ⇒ 4
```

- Differences:
  - x++ returns the current value of x and then increments x.
  - ++x increments first and then returns new value of x.

Iterative Structure (Loops)



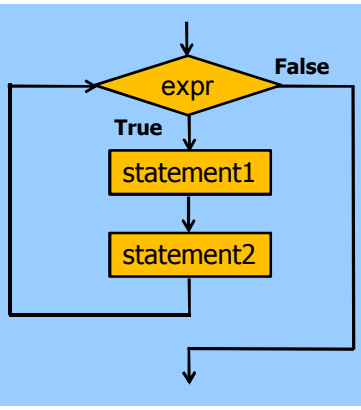
Iterative Structure (Loops)



While Loop

- **Syntax**

```
while (Boolean_expression)
 {
        statement1;
        statement2;
        ..................;
 }
```



- While-loops are executed as long as the expression is true.
- The expression is tested before entering the loop.

Statement and Statement-Blocks

- **Syntax**

```
while (Boolean_expression)
        statement;
```

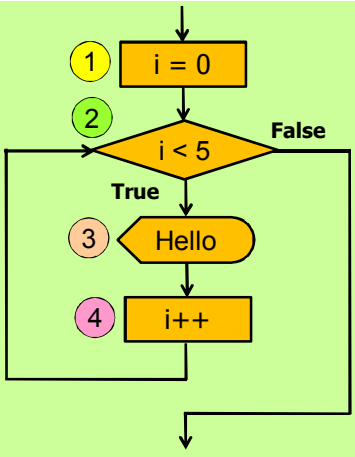
- **Syntax**

```
while (Boolean_expression)
 {
        statement1;
        statement2;
        ……..;
 }
```


While Loop

- **Example**

```
   int i;
1  i = 0;  2
   while ( i < 5)
   {
    3 printf("Hello\t");
    4 i++;
   }
```



- Note: Increment and decrement operators are often used to control loops.

- **Result:**

Hello Hello Hello Hello Hello

Example of While Loop

- **Example1**

```
1   #include <stdio.h>
2   main()
3   {
4     int i =1;                  // Initialization
5     while (i <= 3)             // Test condition
6     {
7       printf("%d\n", i);       // Body
8       i++;                     // Counter
9     }                          // End
10      printf("Finished, but why is the count %d?\n", i);
11  }
```

- **Result:**

```
1
2
3
Finished, but why is the count 4?
```


Exercise of While Loop

- **Write a program to find the sum of the weights of five people.**
  - Get weights from keyboard.
  - Sum the weights of five people.
- **Example of result:**

```
Enter weight : 65.5
Enter weight : 70.2
Enter weight : 67.8
Enter weight : 72.5
Enter weight : 80.3

Sum of the weights is 356.30 Kilograms
```

Exercise of While Loop

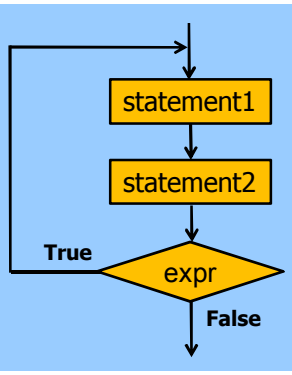
- **Write a program to find the average of numbers.**
  - Get integer numbers from keyboard.
  - Average the sum of numbers.
- **Example of result:**

```
**Exit program by enter negative number**

Enter integer number : 20
Enter integer number : 30
Enter integer number : 40
Enter integer number : 50
Enter integer number : -1

Average of 140/4 is 35.00
```


Do-while Loop

- **Syntax**

```
do {
        statement1;
        statement2;
        ..................;
} while (Boolean_expression);
```



- Do-while loops are executed as long as the expression is true.
- The expression is tested at the end of the loop body.
- The body do-while loops is executed at least once.

Statement and Statement-Blocks

- **Syntax**

```
do
        statement;
while (Boolean_expression);
```

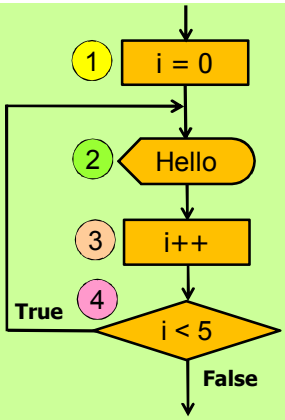
- **Syntax**

```
do {
        statement1;
        statement2;
        ……..;
} while (Boolean_expression);
```


Do-while Loop

- **Example**

```
   int i;
1  i=0;
   do {
2     printf("Hello\t");
3     i++;
4  } while ( i < 5);
```



- **Result:**

  Hello Hello Hello Hello Hello

While Loop vs. Do-while Loop

Example: while loop

```
while ( a < 5)
  {
     printf("Hello");
     a ++;
}
```

Example : do-while loop

```
do {
     printf("Hello");
     a ++;
   } while ( a < 5);
```


Exercise of Do-while Loop

- **Write a program to find the sum of the weights of five people.**
  - Get weights from keyboard.
  - Sum the weights of five people.
- **Example of result:**

```
Enter weight : 65.5
Enter weight : 70.2
Enter weight : 67.8
Enter weight : 72.5
Enter weight : 80.3

Sum of the weights is 356.30 Kilograms
```

Exercise of Do-while Loop

- **Write a program to find the sum of integers.**
  - Get integer numbers from keyboard.
  - Count the number times of entered.
- **Example of result:**

```
**Exit program(Summation is over than 100)**
Enter integer number : 30
Enter integer number : 30
Enter integer number : 10
Enter integer number : 50

You enter numbers 4 times
```


Exercise of Do-while Loop

- **Write a program to find the average of integers.**
  - Get integer numbers from keyboard.
  - Average the sum of numbers.
- **Example of result:**

```
**Exit program (3333)**
Enter integer number : 20
Enter integer number : 30
Enter integer number : 40
Enter integer number : 50
Enter integer number : 60
Enter integer number : 3333

Average of 200/5 is 40.00
```

Example of Do-while Loop

- **Example1: Do-while loop**

```
1  #include <stdio.h>
2  main()
3  {
4     float price;
5     int quantity;
6     char answer;
7     do{
8           printf("Enter 'price quantity': ");
9           scanf("%f %d", &price, &quantity);
10          printf("The total for this item is $%6.2f.\n", price*quantity);
11          printf("Another (Y/N)? ");
12          scanf(" %c", &answer);
13    }while (answer == 'Y' || answer == 'y');
14    printf("Thank you for your patronage.\n");
15 }
```


Example of Do-while Loop

- **Result:**

Enter 'price quantity': **10.50 3**
The total for this item is $ 31.50.
Another (Y/N)? **Y**
Enter 'price quantity': **4.75 6**
The total for this item is $ 28.50.
Another (Y/N)? **y**
Enter 'price quantity': **20.25 5**
The total for this item is $101.25.
Another (Y/N)? **N**
Thank you for your patronage.

Exercise of Do-while Loop

- **Reverse the previous program with while loop.**
- **Example of result:**

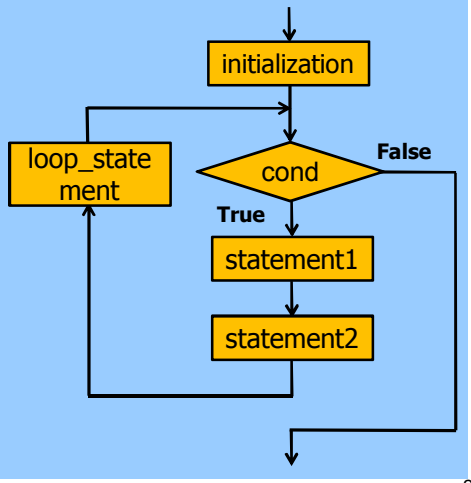
```
**Enter 0 0 to quit.**
Enter 'price quantity': 2.2 3
The total for this item is   6.60.
Enter 'price quantity': 1.5 4
The total for this item is   6.00.
Enter 'price quantity': 0 0
Thank you for your patronage.
```


For Loop

- **Syntax**

```
for (intialization; condition; loop_statement)
{
     statement1;
     statement2;
     ..............;
}
```

- Initialization executed before loop is entered the first time.
- Condition is a condition for loop continuation:
  - if condition is true then loop continues.
  - if condition is false then loop is terminated.
- Loop statement will be executed after the last statement of the loop body.



Statement and Statement-Blocks

- **Syntax**

```
for (intialization; condition; loop_statement)
      statement;
```

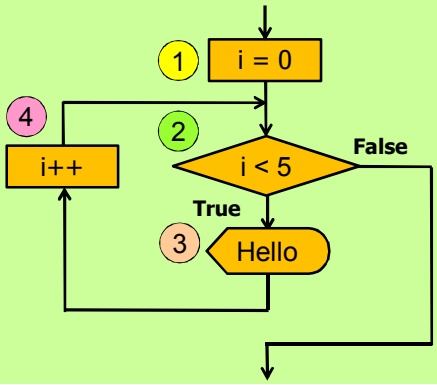
- **Syntax**

```
for (intialization; condition; loop_statement)
{
      statement1;
      statement2;
      ……..;
}
```


For Loop

- **Example**

```
int i;
for ( i = 0;  i < 5;  i++) {
  printf("Hello\t");
 }
```



- **Result:**

Hello Hello Hello Hello Hello

Example of For Loop

- **Example1**

```
1  #include <stdio.h>
2  main()
3  {
4     int i;
5     for (i = 1; i <= 3; i++) {
6           printf("%d\n", i);   // Braces not required
7     }
8     printf("Finished, but why is the count %d?\n", count);
9  }
```

- **Result:**

```
1
2
3
Finished, but why is the count 4?
```


Example of For Loop

- **Example2**

```
1  #include <stdio.h>
2  main()
3  {
4    char ch;
5    for (ch = 'a'; ch <= 'z'; ch++)
6    printf("%2c", ch);
7  }
```

- **Result:**

a b c d e f g h i j k l m n o p q r s t u v w x y z

ASCII Table

Below is the standard ASCII characters.

Dec	Char	Dec	Char	Dec	Char	Dec	Char	Dec	Char	Dec	Char
33	!	49	1	65	A	81	Q	97	a	113	q
34	"	50	2	66	B	82	R	98	b	114	r
35	#	51	3	67	C	83	S	99	c	115	s
36	$	52	4	68	D	84	T	100	d	116	t
37	%	53	5	69	E	85	U	101	e	117	u
38	&	54	6	70	F	86	V	102	f	118	v
39	'	55	7	71	G	87	W	103	g	119	w
40	(	56	8	72	H	88	X	104	h	120	x
41	)	57	9	73	I	89	Y	105	i	121	y
42	*	58	:	74	J	90	Z	106	j	122	z
43	+	59	;	75	K	91	[	107	k	123	{
44	,	60	<	76	L	92	\	108	l	124	|
45	-	61	=	77	M	93	]	109	m	125	}
46	.	62	>	78	N	94	^	110	n	126	~
47	/	63	?	79	O	95	_	111	o	127	
48	0	64	@	80	P	96	`	112	p		

Caution of For Loop

Source code	Output
1. for(i = 1; i <= 3; i++) printf("%d\n", i);	
2. for(i = 1; i <= 3; i++); printf("%d\n", i);	

3. for(i = 1; i <= 3; i++)
{
multiple = 3 * i;
printf("%d\n", multiple);
}	

4. for(i = 1; i <= 3; i++)
multiple = 3 * i;
printf("%d\n", multiple);	

Exercise of For Loop

- **Exercise**

Source code	Loop number	Last of loop number
1. for(i = 1; i <= 10; i ++)=> 2. for(j = 13; j <= 17; j ++)=> 3. for(k = 20; k > 5; k --)=> 4. for(l = 4; l < 40; l +=4)=> 5. for(m = 2; m < 20; m *=2)=>		

Exercise of For Loop

- **Write a program to find the multiple of two numbers.**
- **Example of result:**

  5 * 1 = 5
  5 * 2 = 10
  5 * 3 = 15
  5 * 4 = 20
  5 * 5 = 25
  5 * 6 = 30

Exercise of For Loop

- **Write a program to find the average of five integers.**
  - Get integer number five times from keyboard.
  - Average the sum of five numbers.
- **Example of result:**

Enter integer number 1 : 5
Enter integer number 2 : 10
Enter integer number 3 : 15
Enter integer number 4 : 20
Enter integer number 5 : 25
Average of 75/5 is 15.00

Exercise of For Loop

- **Write a program to find the factorial of n.**
  - Get n from keyboard.
  - n!=1*2*3*…*n
- **Example of result:**

Enter number of Factorial : 6
Factorial of 6! is 720

Break Statement

- **Syntax**

  break;

- Break is used in terminating the loop immediately after it is encountered. The break statement is used with conditional if statement.
- The break statement can be used in terminating all four loops switch…case, for, while and do...while loops.

Example of Break Statement

- **Example1**

```
1   #include <stdio.h>
2   main()
3   {
4      int i;
5      for (i = 1; i <= 10; i++)
6      {
7           printf("i = %d\n", i);
8           if (i == 2) break;
9      }
10     printf("Finish!\n");
11  }
```

- **Result:**

  1
  2
  Finish!

Nested Loops

- A loop within another loop is referred to as a nested loop.
- Nesting may be as deep as needed – a loop within a loop within …..

- **Result:**

Enter a positive integer: **5**

Integer	Factorial
5	120
4	24
3	6
2	2
1	1

Example of Nested Loops

- **Example1: Factorial with nested loops**

```
1   #include <stdio.h>
2   main()
3   {
4      int n, i, factorial;
5      printf("Enter a positive integer: ");
6      scanf("%d", &n);
7      printf("Integer    Factorial\n");
8      for ( ; n >= 1; n--)
9     {
10        printf("%7d", n);       // n initialized by scanf()
11        factorial = 1;
12        for (i = 1; i <= n; i++)
13           factorial *= i;      //Only 1 statement, no braces needed
14        printf("    %d\n", factorial);
15     }
16  }
```

Example of Nested Loops

- **Example2:**

```
1  #include <stdio.h>
2  main()
3  {
4     int out, in;
5     for (out = 1; out <= 2; out++)
6     {
7          printf("Loop out %d\n", out);
8          for (in = 1; in <= 4; in++)
9               printf( "Loop in %d\n", in);
10         printf("\n");
11    }
12 }
```

- **Result:**

```
Loop out 1
Loop in 1
Loop in 2
Loop in 3
Loop in 4

Loop out 2
Loop in 1
Loop in 2
Loop in 3
Loop in 4
```


Exercise of Nested Loops

- **Write a program to display the numbers counting down from 5 to 0 and counting up from 1 to 5.**
- **Example of result:**

Count-up	Count-down
1	5
2	4
3	3
4	2
5	1

Exercise of Nested Loops

- **Write a program to find the different strings of xyz letters.**
- **Example of result:**

  1.xxx 2.xxy 3.xxz 4.xyx 5.xyy 6.xyz 7.xzx 8.xzy 9.xzz 10.yxx 11.yxy 12.yxz 13.yyx 14.yyy 15.yyz 16.yzx 17.yzy 18.yzz 19.zxx 20.zxy 21.zxz 22.zyx 23.zyy 24.zyz 25.zzx 26.zzy 27.zzz

  There are 27 different strings of letters.

scanf vs. getchar

- **Scanf**

```
1  #include <stdio.h>
2  main()
3  {
4    char a;
5    printf(" Enter a character: ");
6    scanf("%c", &a);
7    printf("%c", a);
8  }
```

- **Result:**

  Enter a character: **c**

- **Getchar**

```
1  #include <stdio.h>
2  #include <conio.h>
3  main()
4  {
5    char a;
6    printf("Enter a character: ");
7    a=getchar();
8    putchar(a);
9  }
```

- **Result:**

  Enter a character: **c**

Getch vs. Getche

- **Getch**

```
1  #include <stdio.h>
2  #include <conio.h>
3  main()
4  {
5   char a;
6   printf(" Enter a character: ");
7   do{
8          a=getch();
9    } while (a != 'E');
10 }
```

- **Result:**
  Enter a character:

- **Getche**

```
1  #include <stdio.h>
2  #include <conio.h>
3  main()
4  {
5    char a;
6    printf("Enter a character: ");
7    do{
8          a=getche();
9    } while (a != 'E');
10 }
```

- **Result:**
  Enter a character: **cccE**