

---

# CONTROL STRUCTURES

## CONDITIONAL EXPRESSIONS

Take the construction like this:

```
if (conditional expression)
    statement(s) we do if the condition is true
else
    statement(s) we do if the condition is false
```

Simple example:

```
int grade=85;
if (x>=90)
    printf("Grade A");
else if (x>=80)
    printf("Grade B");
else if (x>70)
    printf("Uhm");
```

Another example:

```
int myheading = 10;
int myTargetBearing = 90;

printf("Are we there yet!?\n");

diff = myTargetBearing - myheading;
if (abs(diff) < 5)
{
    if (diff==0)
        printf("You are right on the target!");
    else
        printf("You are off by %d, but you are closed enough!", abs(diff) );
}
else
{
    printf("You need to turn about %d degrees ", abs(diff));

    if (diff>0)
        printf("clockwise");
    else
        printf("counter-clockwise");
}
```

The key is "<the condition>" will be evaluated down to become 1 or 0, i.e true or false. Therefore, the following sample also works:

```
if (1)
    printf("yeah!");
```

## CONDITIONAL LOOPS

Two ways to repeat a set of instructions: **while** clause and **for** clause

### WHILE CLAUSE

```
while (condition)
{
    statement(s)
}
```

Check out the difference the following code segments:

Sample 1	Sample 2	Sample 3
<pre>int x = 1; while (x &gt; 0) {     printf("number %d \n", x);     --x; }</pre>	<pre>int x = 1; while (--x &gt; 0)     printf("number %d \n", x);</pre>	<pre>int x = 1; while (x-- &gt; 0)     printf("number %d \n", x);</pre>

### FOR CLAUSE

```
for ( <initial setup>; <conditional expressions>; <update at the next iteration > ) {
    things we want to do a given
    number of times
}
```

Example:

```
for (x=100; x > 0; x--)
    printf("number %d\n", x);
```

**Summarize briefly:**

All these 3 loops do the same thing:

<pre>int i = 0; while (i &lt; 10) {     cout &lt;&lt; "value: " &lt;&lt; i;     i = i+1 }</pre>	<pre>int i = 0; while (i &lt; 10) {     cout &lt;&lt; "value: " &lt;&lt; i;     ++i; }</pre>	<pre>for (int i=0; i&lt;10; i++) {     cout &lt;&lt; "value: " &lt;&lt; i; }</pre>
---	--	--

### DO... WHILE CLAUSE

<pre>do {     statement(s) } while (condition);</pre>	<pre>int i=0; do {     cout &lt;&lt; "value: " &lt;&lt; i;     i = i+1; } while (i &lt; 10);</pre>
---	--

This may generate different result from while clause above. Do you know what that may be?

## BREAK OUT OF THE LOOPS (BREAK)

```
while ( conditional expression )
{
    do something...
    if (something is true) {
        break ;
    }
    ....
    more to do...
    ....
}
....
```

## JUMP BACK TO THE BEGINNING OF THE LOOP (CONTINUE)

```
while ( conditional expression )
{
    do something...
    if (something is true) {
        continue;
    }
    ....
    more to do...
    ....
}
....
```

## RELATIONAL AND LOGICAL OPERATORS

Relational operators : (will be further discussed on Day 2)

- == > < >= <= !

Logical operators: (will be further discussed on Day 2)

- && ||

## SWITCH

- substitute for long "if" statements
- compare a variable to several int/char values.
- programming structure:

```
switch ( <variable> ) {
    case value1:
        Code to execute if <variable> == this-value
        break;
    case value2:
        Code to execute if <variable> == that-value
        break;
    case ...:
        ....
    default:
        Code to execute if <variable> does not equal any value in the "cases"
        break;
}
```

### Example:

```
int age= 50;
int x = age / 10;
switch ( age ) {
    case 5:
        printf("Keep active!");
        break;
    case 4:
        printf("Still cool!");
        break;
    case 3:
        printf("Still young!");
        break;
    case 2:
        printf("Yeah! College is Cool!");
        break;
    case 1:
        printf("You can't drink! ");
        break;
    default:
        printf("Don't want to say\n");
        break;
}
```

## PRACTICE EXERCISES

1. Ask user to enter 4 numbers. Your program will produce the largest and the smallest number. You are allowed to use only 4 "if"s expressions though.

2. Write a function that outputs a right-side-up triangle of height  $n$  and width  $2n-1$ ; the output for  $n = 6$  would be:

```

*
***
*****
*****
*****
*****
*****

```

3. Write a program that prompts the user for a sentence, and prints each word on its own line.
4. Write a program to ask user to enter a number, and generate the factorial of that number. If user enters 5, answer = 120. If user enters 6, answer = 720.
5. Write a program to calculate a gcd with two numbers using Euclid Algorithm. This is how it works:

N	M	R= N % M
300	64	44 = 300 % 64
64	44	20 = 64 % 44
44	20	4 = 44 % 20
20	4	0 = 20 % 4

6. Write a program to calculate a square root of a number using Newton's method. This is how it works:  
e.g. user wishes to find out square root of 3:

N	N/M	M	Average of N/M and M
3	3	1	2
	1.5	2	1.75
	1.71429	1.75	1.73214
	1.73196	1.7314	1.73205
	1.73205	1.73205	1.73205

Your get your answer when:

$$| M_{\text{(current)}} - M_{\text{(previous)}} | \leq M * \frac{1}{10000}$$

7. There are two numbers with different number system, i.e.  $A_{b1} == B_{b2}$  where  $b1$  and  $b2$  are bases. Eg.  $419_{47} == 792_{35} == 8892$ . You will write a program which will be calculate  $b1$  and  $b2$  for each of the following set of numbers. Assuming all numbers here  $< 1000$ .

The numbers	b1	b2
$419_{b1}$ $792_{b2}$	47	35
$582_{b1}$ $232_{b2}$	2213	1565
$846_{b1}$ $598_{b2}$	918	1452
$252_{b1}$ $529_{b2}$	3275	4142
$365_{b1}$ $644_{b2}$	1849	1170
$473_{b1}$ $719_{b2}$	2241	1585

The following exercises are **Only** for those who know bit-wise operation. Write the following functions to print out all the bits of an data field:

8. Perform division with bits operation. No arithmetic operators, "+", "-", "/", "\*" is allowed. In order to come up with the solutions, you need to think about:
- Pattern
  - What operation will help you to know whether you need to carry over.
  - What operation will help you to do "\*" and "/"
  - How to know it is an odd or even

Hint for division:



