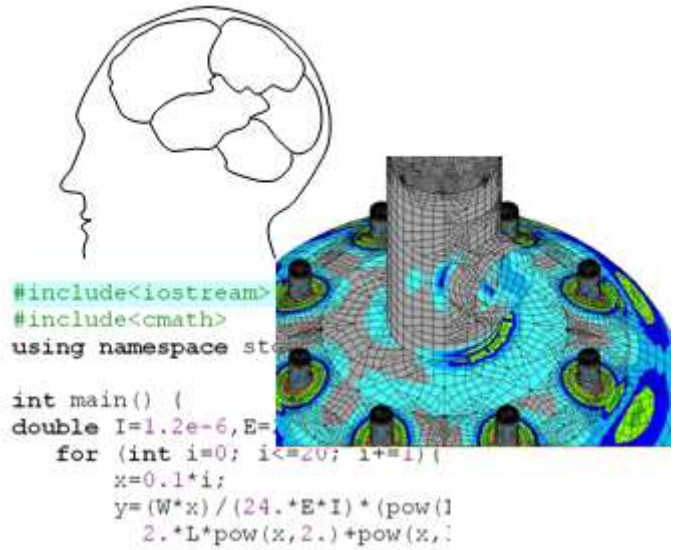
ME 240 Computation for Mechanical Engineering



Control structures:

Repetitive structures

Content

Extracted from `http://cpp.gantep.edu.tr`

Repetitive structures (*loops*) are control structures that cause a program to repeat (iterate) a block of code.

This week:

- The **`while`** structure.
- The **`do ... while`** structure.
- The **`for`** structure.
- **`continue`** and **`break`** statements
- Infinite loops
- Nested loops

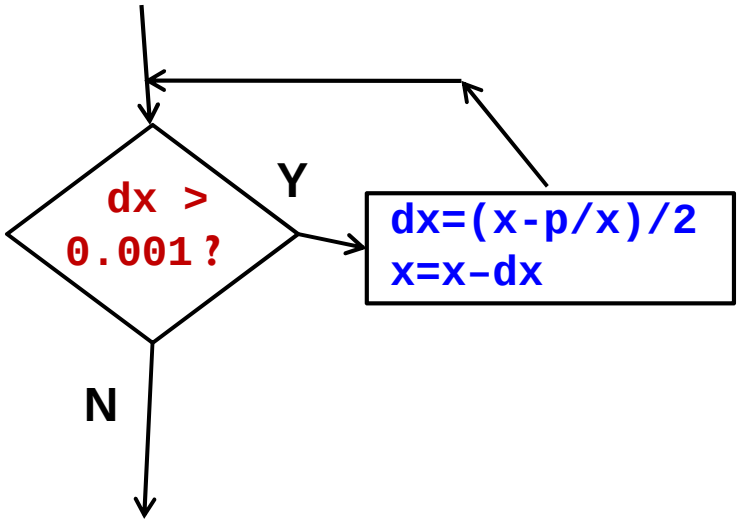
The while loop structure

The **while** loop has the general form:

```
while (condition) {
  statements
  .
  .
}
```

Here the block of statements is executed while **condition** is **true**.

Note that **condition** is tested at the top of the loop.



Example program section:

```
while ( dx > 0.001 ) {
  dx = (x-p/x)/2 ;
  x = x - dx;
}
```

Example while loop

This program calculates the series sum 1 + 2 + 3 + 4 + ... + n.

http://en.wikipedia.org/wiki/Sum_of_series

```
#include <iostream>
using namespace std;
int main() {

  int k=1, s=0, n;
  cout << "Input n: ";
  cin >> n;

  while (k<=n) {
    s = s + k;
    k++;
  }

  cout << "The series sum is "
       << s << endl;

}
```

Output

```
Input n: 8
The series sum is 36
```

Note that on the first iteration of the loop, **k**=1; and on the final execution **k**=**n**.

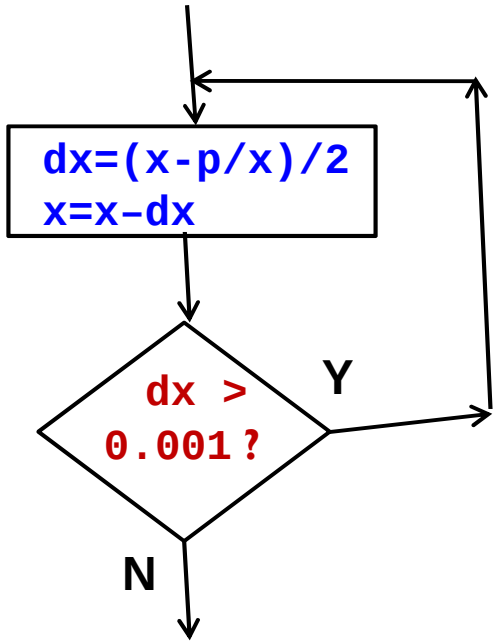
The do .. while loop structure

It has the general form:

```
do {
  statements
  .
  .
} while (condition);
```

Here the block of statements is executed while **condition** is **true**.

Note that **condition** is tested at the bottom of the loop.



Example program section:

```
do {
   dx = (x-p/x)/2 ;
   x = x - dx;
} while ( dx > 0.001 );
```

Example do .. while loop

This program determines whether a given number is prime or not.

```
#include <iostream>
using namespace std;

int main(){
    int number,sum=0,k=2;
    cout<<"input the number\n";
    cin>>number;
    do{
            if(number%k==0){
            cout<<"it is not prime number!\n";
            system("pause");
            return 0;
            };
            k++;
    }while(k<=number-1);
    cout<<"it is a prime number!\n";
system("pause"); }
```

Output

```
input the number
5
it is a prime number!
```

The for loop structure

The **for** statement allows you to execute a block of code a specified number of times. The general form is:

```
for (initialisation; condition; increment) {
  statements
  .
  .
}
```

Example program section:

```
for (int i=0; i<5; i++) {
  cout << i   << " squared = "
       << i*i << endl;
}
```

Output

```
0 squared = 0
1 squared = 1
2 squared = 4
3 squared = 9
4 squared = 16
```

Example for loop

This program calculates the series sum 1 + 2 + 3 + 4 + ... + n.

http://en.wikipedia.org/wiki/Sum_of_series

```
#include <iostream>
using namespace std;
int main() {

  int s=0, n;
  cout << "Input n: ";
  cin >> n;

  for (int k=1; k<=n; k++) {
    s = s + k;
  }

  cout << "The series sum is "
       << s << endl;

}
```

Output

```
Input n: 8
The series sum is 36
```

Note that on the first iteration of the loop, **k**=1; and on the final execution **k**=**n**.

Another example for loop

This program calculates and outputs sines and cosines of angles from 0 to 90° in steps of 10°.

```
#include <iostream>
#include <cmath>
using namespace std;

int main() {

  for (int deg=0; deg<=90; deg=deg+10)
  {
    double rad = deg * M_PI/180.0;
    cout << deg
         << "  " << sin(rad)
         << "  " << cos(rad)
         << endl;
  }
}
```

Output

```
0  0  1
10  0.173648  0.984808
20  0.34202  0.939693
30  0.5  0.866025
40  0.642788  0.766044
50  0.766044  0.642788
60  0.866025  0.5
70  0.939693  0.34202
80  0.984808  0.173648
90  1  6.12303e-17
```

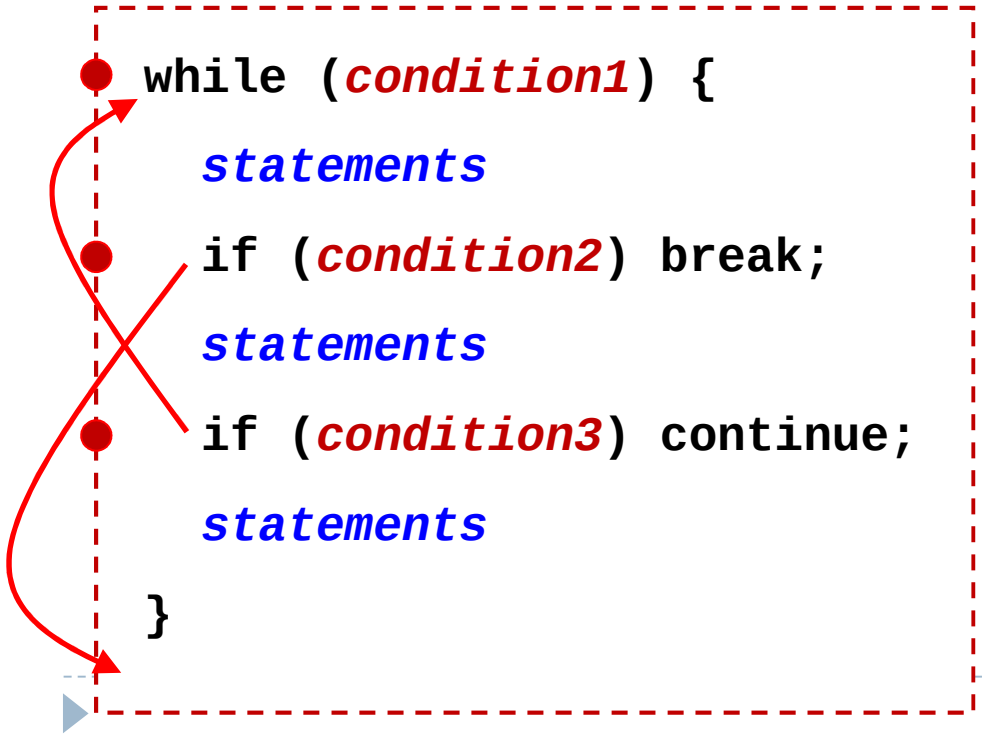
The output could be improved with some formatting (see week 14).

The break and continue statements

The **break** statement jumps to the end of a loop and exits the loop.

The **continue** statement jumps to the beginning of a loop and the loop continues.

Consider the **while** loop with a **break** and **continue** statement.



```
while (condition1) {
    statements
    if (condition2) break;
    statements
    if (condition3) continue;
    statements
}
```

These statements are placed inside the loop thus adding more conditioning to the loop structure.

This **while** loop is now controlled by three conditions.

Example use of the `continue` statement

This program calculates the mean of 9 values **excluding zeros**.

```
#include <iostream>
using namespace std;
int main() {

  double v, sum=0.0;
  int i=0, j=0;

  while (i<9) {

    i++;
    cout << "Input value " << i << ": ";
    cin >> v;
    if ( v == 0.0 ) continue;

    j++;
    sum = sum + v;

  }

  cout << j << " non-zero values." << endl;
  cout << "Their mean is " << sum/j << endl;

}
```

Output

```
Input value 1: 3.4
Input value 2: 6.1
Input value 3: 0
Input value 4: 0
Input value 5: 3.3
Input value 6: 0
Input value 7: 2.9
Input value 8: 0
Input value 9: 4.1
5 non-zero values.
Their mean is 3.96
```

Check:
(3.4+6.1+3.3+2.9+4.1) / 5
= 3.96

The continue statement in a for loop

The previous program is a little more concise with a **for** loop.

```
#include <iostream>
using namespace std;
int main() {

  double v, sum=0.0;
  int j=0;

  for (int i=1; i<=9; i++) {

    cout << "Input value " << i << ": ";
    cin >> v;
    if ( v == 0.0 ) continue;

    j++;
    sum = sum + v;

  }

  cout << j << " non-zero values." << endl;
  cout << "Their mean is " << sum/j << endl;

}
```

Output

```
Input value 1: 3.4
Input value 2: 6.1
Input value 3: 0
Input value 4: 0
Input value 5: 3.3
Input value 6: 0
Input value 7: 2.9
Input value 8: 0
Input value 9: 4.1
5 non-zero values.
Their mean is 3.96
```

Infinite loops

If the ***condition*** of a loop is always **true**, then the loop will Iterate *infinitely*, i.e. it will loop forever!

```
while ( true ) {
  cout << "infinite loop!" << endl;
}
```

```
while ( 1 ) {
  cout << "infinite loop!" << endl;
}
```

```
do {
  cout << "infinite loop!" << endl;
} while ( 7>3 );
```

```
for ( ; ; ) {
  cout << "infinite loop!" << endl;
}
```

It is sometimes useful to create infinite loops like these, but with the addition of a ***condition*** for breaking out of the loop.

A "break out" can be achieved with the **break** statement together with an **if** structure.....

Example use of the break statement in an infinite loop

This program continually inputs values and outputs their reciprocal.
The program terminates when the input is zero.

```
#include <iostream>
using namespace std;

int main() {

  double x;

  while(1) {
    cout << "Input x: ";
    cin >> x;
    if ( x==0. ) break;
    cout << "The reciprocal is "
         << 1/x << endl;
  }
  cout << "Bye." << endl;

}
```

The important property of this kind of loop is that ***condition*** is tested somewhere between the beginning and end of the loop.

```
Input x: 34.2
The reciprocal is 0.0292398
Input x: 0.8
The reciprocal is 1.25
Input x: 3.4
The reciprocal is 0.294118
Input x: 3.0
The reciprocal is 0.333333
Input x: 0.2
The reciprocal is 5
Input x: 0
Bye.
```

Example of break and continue in an infinite loop

```
#include <iostream>
#include <cmath>
using namespace std;

int main() {

  double x;

  while (true) {

    cout << "Input a value (zero to exit): ";
    cin  >> x;

    if (x<0.0) {
      cout << "Negative square root!" << endl;
      continue;
    } else if (x==0.0) {
      cout << "Bye." << endl;
      break;
    }

    cout << "sqrt(x) = "
         <<  sqrt(x) << endl;

  }
}
```

This program continually inputs values and outputs their square-root for positive values only.

The program terminates when the input is zero.

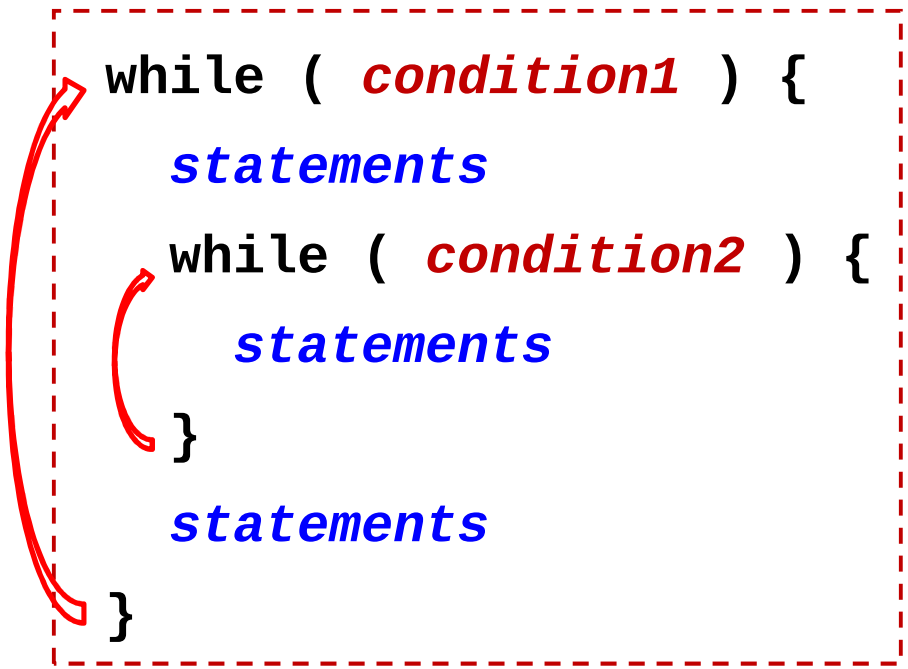
Output

```
Input a value (zero to exit): 2.4
sqrt(x) = 1.54919
Input a value (zero to exit): -6.7
Negative square root!
Input a value (zero to exit): 12.6
sqrt(x) = 3.54965
Input a value (zero to exit): 0
Bye.
```

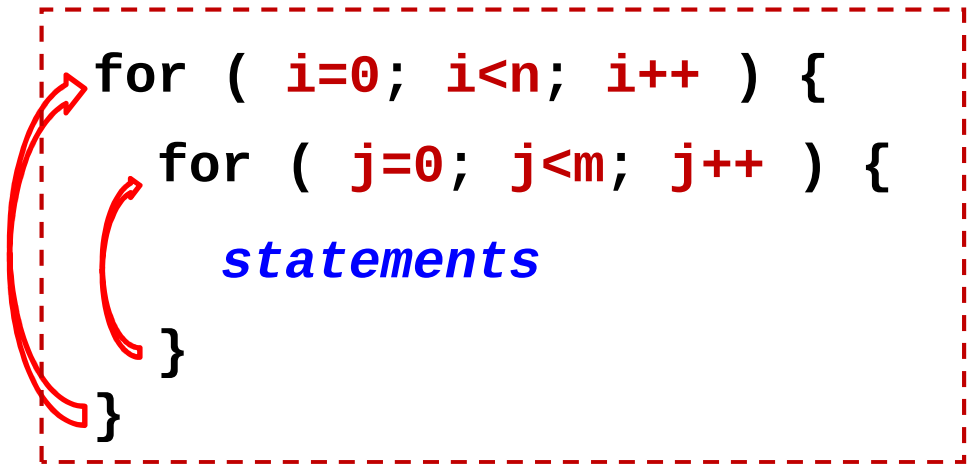
Nested loops

Nested loops are *loops within loops*

Nested **`while`** loops



```
while ( condition1 ) {
   statements
   while ( condition2 ) {
      statements
   }
   statements
}
```




```
i=0;   j=0, 1, 2, 3, ..., m-1
i=1;   j=0, 1, 2, 3, ..., m-1
i=2;   j=0, 1, 2, 3, ..., m-1
.
.
i=n-1; j=0, 1, 2, 3, ..., m-1
```

There are a total of **m**×**n** iterations of this nested loop.

```
#include <iostream>
using namespace std;

int main() {

  for ( int i=1; i<=8; i++ )
  {
    for ( int j=1; j<=6; j++ )
    {
      cout << i*j << "\t";
    }
    cout << endl;
  }

}
```

In this example **i** loops over rows and **j** loops over columns.

Output

```
1	2	3	4	5	6
2	4	6	8	10	12
3	6	9	12	15	18
4	8	12	16	20	24
5	10	15	20	25	30
6	12	18	24	30	36
7	14	21	28	35	42
8	16	24	32	40	48
```

Solved problem

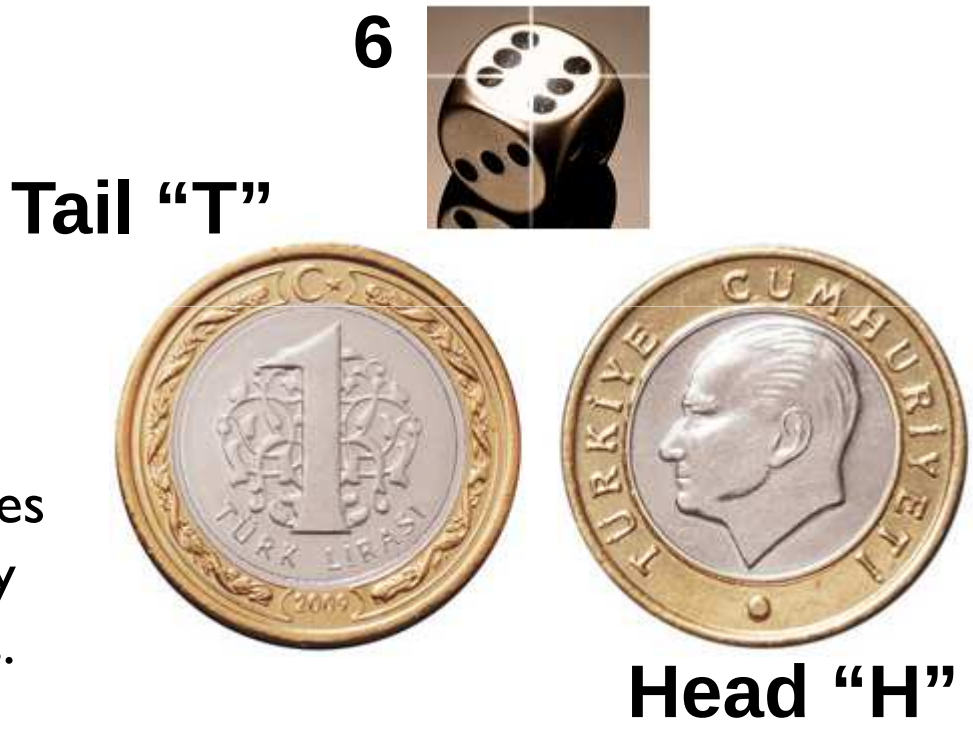
Consider the tossing of two coins and one die;

Question:

What is the probability of obtaining "a Tail", "a Head" and a "6" (in any order)?

Solution:

Count the number of outcomes "T", "H", and "6" and divide by the total number of outcomes.



We can write a C++ program that displays all possible outcomes…

```
#include <iostream>
using namespace std;

int main(){

  for ( int c1=0; c1<=1; c1++ ) {
  for ( int c2=0; c2<=1; c2++ ) {
  for ( int di=1; di<=6; di++ ) {
    cout << di << ",";
    if (c1) { cout << "H,"; }
    else    { cout << "T,"; }
    if (c2) { cout << "H\n";  }
    else    { cout << "T\n";}
  }
  }
  }

}
```

1,T,T	1,H,T
2,T,T	2,H,T
3,T,T	3,H,T
4,T,T	4,H,T
5,T,T	5,H,T
6,T,T	**6,H,T**
1,T,H	1,H,H
2,T,H	2,H,H
3,T,H	3,H,H
4,T,H	4,H,H
5,T,H	5,H,H
6,T,H	6,H,H

That's 2 out of 24 $\Rightarrow$
P("6", "T", "H") = 1/12