Programming fundamentals

Lecture 8: Loops, Nested loops, pattern printing with nested loops

Recap

- Switch statement
- While loop

Agenda

- Do while loop
- For loop
- Nested loops

Typecasting: implicit

```
double d = 3.4;
int i = 3;
int i1 = 0; double d1 = 0;
//double to int
i1 = d;
cout <<"i1: "<< i1 << endl;
//int to double
d1 = i;
cout <<"d1: "<< d1 << endl;
```

Microsoft Visual Studio Debug Console

```
i1: 3
d1: 3
```

Typecasting: explicit

```
double d = 3.4;
int i = 3;
int i1 = 0; double d1 = 0;
i1 = (int)d;
i1 = static_cast<int>(d);
cout <<"i1: "<< i1 << endl;

d1 = (double)i;
cout <<"d1: "<< d1 << endl;
```

Microsoft Visual Studio Debug Console

```
i1: 3
d1: 3
```

DIVISION RULES

```
double d = 3.4;
int a = 5, b = 2;
int res_int = a/b; double res_double = a/b;
cout <<"res_int: "<< res_int << endl;
cout << "res_double: " << res_int << endl;
```

Microsoft Visual Studio Debug Console

```
res_int: 2
res_double: 2
```

```
double d = 3.4;
double a = 5; double b = 2;
int res_int = a/b;
double res_double = a/b;
cout <<"res_int: "<< res_int << endl;
cout << "res_double: " << res_double << endl;
```

Microsoft Visual Studio Debug Console

```
res_int: 2
res_double: 2.5
```

One of the variables must be double to retain the decimal part while dividing

```
double d = 3.4;
int a = 5; double b = 2;
int res_int = a/b;
double res_double = a/b;
cout <<"res_int: "<< res_int << endl;
cout << "res_double: " << res_double << endl;
```

Microsoft Visual Studio Debug Console

```
res_int: 2
res_double: 2.5
```

Some examples..

- cout<< (char)65; //prints letter A
- cout<< static_cast<char> (65);

```
for (int x = 65; x < 90; x++)
{
    cout << x << " " << (char)x << "";

}
```

Microsoft Visual Studio Debug Console

```
65 A        66 B        67 C        68 D        69 E        70 F        71 G        72 H
73 I        74 J        75 K        76 L        77 M        78 N        79 O        80 P
81 Q        82 R        83 S        84 T        85 U        86 V        87 W        88 X
89 Y
```

Comparing int with double

```
double d = 3.4;
int i = 3;
if (i == d)//d= 3.3999999999999999   i=3
    cout << "same" << endl;
else
    cout << "different" << endl;
```

Microsoft Visual Studio Debug Console

```
different
```

```
double d = 3.4;
int i = 3;
if (i == (int)d)//d= 3   i=3
    cout << "same" << endl;
else
    cout << "different" << endl;
```

Microsoft Visual Studio Debug Console

```
same
```

```
double d = 3.4;
int i = 3;
if ((double)i == d)//d= 3   i=3
    cout << "same" << endl;
else
    cout << "different" << endl;
```

Microsoft Visual Studio Debug Console

```
different
```

Int to other types

- Int to char

```
int res_int = 65;
char c;
c = static_cast<char>(res_int);
cout <<"c: "<< c << endl;
```

Microsoft Visual Studio Debug Console

```
c: A
```

- Int to string (NOT ALLOWED)

double to other types

- double to char

```
int res_double = 65;
char c;
c = static_cast<char>(res_double);
cout <<"c: "<< c << endl;
```

Microsoft Visual Studio Debug Console

```
c: A
```

- double to string (NOT ALLOWED)

Char to other types

- Char to double

```
double res_double = 65.5;
char c=65;
res_double = static_cast<double>(c);
cout <<"res_double: "<< res_double << endl;
```

Microsoft Visual Studio Debug Console

```
res_double: 65
```

- Char to int

```
int res_int = 65.5;
char c=65;
res_int = static_cast<int>(c);
cout <<"res_int: "<< res_int << endl;
```

Microsoft Visual Studio Debug Console

```
res_int: 65
```

- char to string (NOT ALLOWED)

Find the number of digits after a decimal part

```
1.  double no = 3.44;
2.  int count = 0;
3.  while (no != ((int)no)){
4.          count++;
5.          no = no * 10;}
6.  cout<< count;
```

Line#	no	count	(int)no	no!=(int)no ?
1	3.44			
2	3.44	0		
3	3.44	0	3	3.44 ! = 3
4	3.44	1	3	
5	34.4	1	3	
3	34.4	1	34	34.4 ! = 34
4	34.4	2	34	
5	344.0	2	34	
3	344.0	2	344	344.0 ! = 344
6	Print 2			

Continue in while loop

- To bypass the loop's normal control structure
- The **continue** statement forces the next iteration of the loop to take place, skipping any code between itself and the conditional expression that controls the loop

```
#include <iostream>
using namespace std;
int main()
{
int x = 0; int num = 0;
while(x <= 5)
{
        x++;
        cout << "before if iteration#" << x << endl;
        if ((num++) % 2==1) continue;
        cout << x << endl;
        cout << "after if iteration#" << x << endl;

}
return 0;
}
```

These statements will be skipped if **continue** gets executed

Microsoft Visual Studio Debug Console

```
before if iteration#1
1
after if iteration#1
before if iteration#2
before if iteration#3
3
after if iteration#3
before if iteration#4
before if iteration#5
5
after if iteration#5
before if iteration#6
```

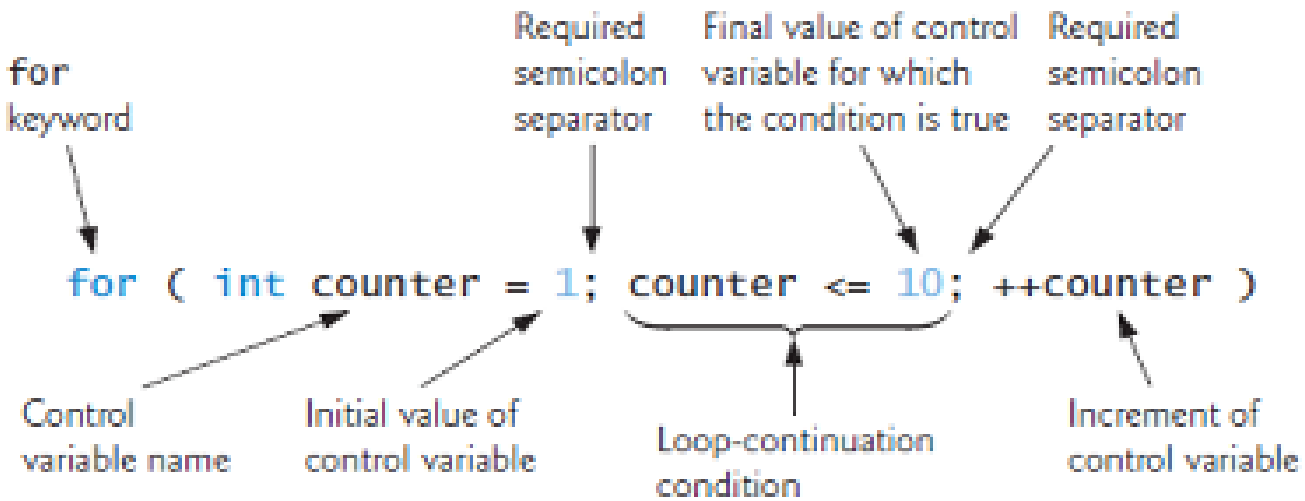
Do while loop

- The do..while loop is similar to the while loop with one important difference. The body of do...while loop is executed at least once. Only then, the test expression is evaluated.
- The syntax of the do...while loop is:

```
do
{
   // statements inside the body of the loop
}
while (testExpression);
```

```
do{
    bake 5 cookies;
} while (thereAreFriends);
```

For loop



Comparison... print first 10 numbers

```
for (int i=1;i<10;i++)
{
    cout << i << endl;
}
```

```
int i = 1;
while (i<=10)
{
        cout << i << endl;
        i = i + 1;
}
```

```
int i = 1;
do
{
        cout << i << endl;
        i = i + 1;
} while (i <= 10);
```

Nested loops

- Loop within a loop

Pattern practice

```
*****
*   *
*   *
*   *
*****
```

```
@@@@@
$$$$$
&&&&&
```

Division and modulus operator

Number= 12345	
12345 / 10000 = 1	12345 % 10 = 5
Number= 2345	
2345 / 1000 = 2	2345 % 10 = 5
Number= 6547	
6547 / 1000 = 6	6547 % 10 = 7

Extracting each digit of an integer

Number=12345	**d=10000**
Number / d	Number % d
1	2345
Number=2345	**d=1000**
Number / d	Number % d
2	345
Number=345	**d=100**
Number / d	Number % d
3	45
Number=45	**d=10**
Number / d	Number % d
4	5

Palindrome

```
int num=12345;
int d = 10000;
while (num % d!=0)
{
        cout << "Digit: " <<num/d<< endl;
        num = num % d;
        d = d / 10;
}
cout << "Digit: " << num / d << endl;
```

Multiple cases in a switch

```
switch (grade) // switch statement nested in while
{
        case 'A': // grade was uppercase A
        case 'a': // or lowercase a
                ++aCount; // increment aCount
                break; // necessary to exit switch
        case 'B': // grade was uppercase B
        case 'b': // or lowercase b
                ++bCount; // increment bCount
                break; // exit switch
        default: // catch all other characters
                cout << "Incorrect letter grade entered."
                << " Enter a new grade." << endl;
                break; // optional; will exit switch anyway
} // end switch
```

Recommended reads

- Dietal & Dietal
  - Chapter 5