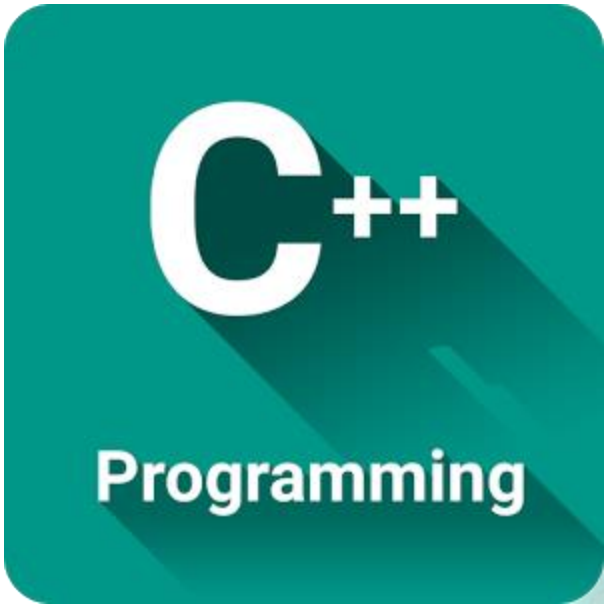
C++ Programming

Lecture 11



Problem: Even/Odd Number

```
int number;
cout << "Enter a number\n";
cin >> number;

if (number % 2 == 0)
   cout << "Even\n";
else
   cout << "Odd\n";
```

Exercise: Output?

```
if (x == 100)
      cout << "x is 100";
else
      cout << "x is not 100";
```

equivalent

```
if (x == 100)
cout << "x is 100";
cout << "x is not 100";
```

```
if (x == 100)
    cout << "x is 100";
cout << "x is not 100";
```

In case x equals 100
The first block will print one message only,
while the second and third blocks will print two messages

Logical Operators

- AND
- OR
- NOT

AND (&&)	true	false
true	true	false
false	false	false

OR (||)	true	false
true	true	true
false	true	false

NOT (!)	true	false
	false	true

XOR (^)	true	false
true	false	true
false	true	false

Exercise: Logical Operators

- Assume weight = 140 and age = 24.

Expression	Value
! (age > 18)	false
! (weight == 150)	true
(age > 18) || (weight >= 150)	true
(age > 18) && (weight >= 140)	true
(weight == 150) || (age < 45)	true
(age > 34) || (weight < 140)	false
(weight >=120) && (age >=30)	false

Problem: BMI

- Body Mass Index (BMI) is a measure of health on weight. It can be calculated by taking your weight in kilograms and dividing by the square of your height in meters. Program reads the person weight and height and displays his BMI. The interpretation of BMI for people 16 years or older is as follows:

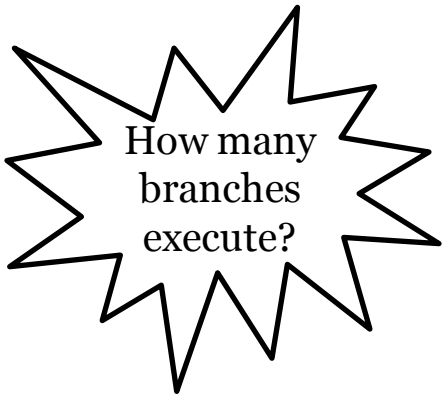
BMI	Interpretation
Below 18.5	underweight
Below 25	normal
Below 30	overweight
30 or more	Seriously overweight

Solution

```
float weight, height;

// Prompt the user to enter weight
cout << "Enter weight: ";
cin  >> weight;
// Prompt the user to enter height
cout << "Enter height: ";
cin  >> height;

float bmi = weight /(height * height);

cout << "BMI is " << bmi << endl;
if (bmi < 18.5)
   cout << "Underweight" << endl;
else if (bmi < 25)
   cout << "Normal" << endl;
else if (bmi < 30)
   cout << "Overweight" << endl;
else
   cout << "Seriously Overweight" << endl;
```

Grades Interpretation

- Program reads student grade and interprets it.




```
int main()
{
char grade;
cout << "Enter student grade\n";
cin >> grade;
if (grade ='A' || grade == 'a')
      cout << "Excellent\n";
else if (grade ='B' || grade == 'b')
      cout << "Very Good\n";
else if (grade ='C' || grade == 'c')
      cout << "Good\n";
else if (grade ='D' || grade == 'd')
      cout << "You can do better\n";
else if (grade ='E' || grade == 'e')
      cout << "Disappointing\n";
else
      cout << "Invalid Grade\n";
return 0;
}
```

Increase/Decrease Operators

- The increase operator (++) and the decrease operator (--) increase or reduce by one the value stored in a variable

```
x++;
x=x+1;
x+=1;
```

```
x--;
x=x-1;
x-=1;
```

are all equivalent in functionality

also called increment/decrement operators

unary operators: works on one operand

Increase/Decrease Operators

- This operator can be used as a prefix or as a suffix.

Case 1: When variable is not used in expression

- Pre-incrementing and post-incrementing have same effect

```
++x;
cout << x;
```

Same value as

```
x++;
cout << x;
```

Case 2: When variable is used in expression

- **It is used as a prefix (++a)** the value is increased **before** the result of the expression is evaluated and therefore the increased value is considered in the outer expression.
- **It is used as a suffix (a++)** the value stored in a is increased **after** being evaluated and therefore the value stored before the increase operation is evaluated in the outer expression.

Increase/Decrease Operators

Example 1	Example 2
B=3; A=++B; // A contains 4, B contains 4	B=3; A=B++; // A contains 3, B contains 4

- In Example 1, B is increased before its value is copied to A.
- In Example 2, the value of B is copied to A and then B is increased.

Increase/Decrease Operators

- If `x = 5`, then

  ```
  cout << ++x;
  ```

  - `x` is changed to `6`, then printed out

- If `x = 5`, then

  ```
  cout << x++;
  ```

  - Prints out `5` (`cout` is executed before the increment).
  - `x` then becomes `6`

Increase/Decrease Operators

```
int i = 10;
int newNum = 10 * i++;
```

Same effect as

```
int newNum = 10 * i;
i = i + 1;
```

```
int i = 10;
int newNum = 10 * (++i);
```

Same effect as

```
i = i + 1;
int newNum = 10 * i;
```

Conditional Operator

- The only ternary operator in C++.
- Works on three operands.
- Simulates an IF-ELSE statement.

Conditional Operator

```
y = (x > 0)  ?  1  :  -1 ;
```

y = (BooleanExpression) ? expressio1 : expression2;

If BooleanExpression evaluates to true, then y= expression1.
If BooleanExpression evaluates to false, then y = expression2.

is equivalent to

```
if (x > 0)
        y = 1;
else
        y = -1;
```

Conditional Operator

(BooleanExpression) ? expressio1 : expression2;

- If BooleanExpression evaluates to true, then expression1 executes.
- If BooleanExpression evaluates to false, then expression2 executes.

(a > b) ? (c=25) : (c=45) ;

Equivalent to

c = (a > b) ? 25 : 45;

Problem

- Program reads two integers and displays the greater.

```
int num1, num2, greater;
cout<<"Enter two numbers:\n";
cin >> num1 >> num2;

greater = (num1 >= num2) ? num1 : num2;

cout<<"Greater is: " << greater << endl;
```

Problem

- Write a program that lets the user enter a year and checks whether it is a leap year.
- A year is a *leap year* if it is divisible by 4 but not by 100 , or if it is divisible by 400.

Solution: Leap Year

```
int year;
cout << "Enter a year: ";
cin >> year;

// Check if the year is a leap year
bool isLeapYear =
  (year%4 == 0 && year%100 != 0) || (year%400 == 0);

if (isLeapYear)
  cout << year << " is a leap year" << endl;
else
  cout << year << " is a not leap year" << endl;
```

Solution 2: Leap Year

```
int year;
cout << "Enter a year: ";
cin >> year;

// Check if the year is a leap year
if ((year % 400 == 0) || (year%4 == 0 && year%100 != 0))
    cout<< "Celebrate\n";
else
    cout<< "Ordinary Year\n";
```

Solution 3: Leap Year

```
int year;
cout << "Enter a year: ";
cin >> year;

// Check if the year is a leap year
((year % 400 == 0) || (year%4 == 0 && year%100 != 0)) ?
cout<< "Celebrate\n" : cout << "Ordinary Year\n";
```

Problem: Subtraction

Check result of subtraction of two input numbers.

```
int number1, number2, answer;
cout << "Enter two numbers: ";
cin >> number1 >> number2;
cout << "Enter the subtraction result: ";
cin >> answer;
if (number1 - number2 == answer)
  cout << "You are correct!\n";
else
  cout << "Wrong answer" << endl <<
  number1 << "-" << number2 << "=" <<
  number1 - number2 << endl;
```

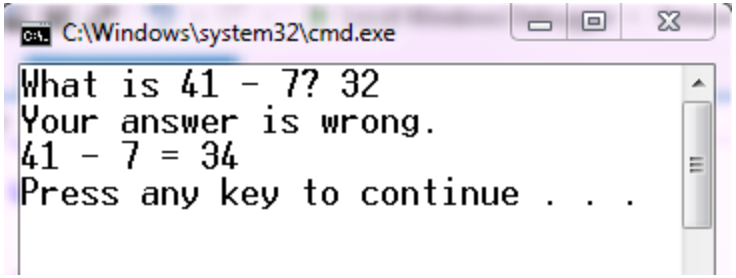
Problem

Perform subtractions of two randomly generated numbers.

Function rand() generates random number.

int x = rand() % n;

Generates a random number from 0 to n-1.



Solution

Perform subtraction of two randomly generated numbers.

```
int number1 = rand() % 50; //generates random num from 0 to 49
int number2 = rand() % 10; //generates random num from 0 to 9

 cout << "What is " << number1 << "-" << number2 << "? ";
 int answer;
 cin >> answer;
 if (number1 - number2 == answer)
   cout << "You are correct!\n";
 else
   cout << "Wrong answer" << endl <<
   number1 << "-" << number2 << "=" <<
   number1 - number2 << endl;
```

Note

- Remember to include necessary braces

```
if (radius >= 0)
  area = radius * radius * PI;
  cout << "The area "
    << " is " << area;
```

(a) Wrong

```
if (radius >= 0)
{
  area = radius * radius * PI;
  cout << "The area "
    <<  " is " <<  area;
}
```

(b) Correct

Note

Logic Error

```
if (radius >= 0);
{
  area = radius * radius * PI;
  cout << "The area "
    << " is " << area;
}
```

(a)

Equivalent

Empty Body

```
if (radius >= 0) { };
{
  area = radius * radius * PI;
  cout << "The area "
    <<  " is " <<  area;
}
```

(b)

Wrong Semicolon at the <u>if</u> Line

```
int i = 1;
int j = 2;
int k = 3;

if (i > j)
  if (i > k)
    cout << "A";
else
    cout << "B";
```

(a)

Equivalent

This is better with correct indentation

```
int i = 1;
int j = 2;
int k = 3;

if (i > j)
  if (i > k)
    cout << "A";
  else
    cout << "B";
```

(b)

Note

To force the else clause to match the first if clause, you must add a pair of braces:

```
int i = 1; int j = 2; int k = 3;
if (i > j)
{
    if (i > k)
       cout << "A";
}
else
    cout << "B";
```

Output?

This statement prints B.

Note

- Variable of type bool has one of two values: true/false.
- Remember that the assignment operator (=) is a binary operator. It deals with a variable on the left side and a value on the right side.

bool isEven; **if (number % 2 == 0)** **isEven = true;** **else** **isEven = false;**	**bool isEven = number % 2 == 0;**

Note

- Redundant Testing of Boolean Values

```
if (even == true)
  cout <<"It is even.";
```

(a)

Equivalent

```
if (even)
  cout << "It is even.";
```

(b)

This is better

Selective Structure: Switch

```
switch (expression)
{
   case constant1:
         group of statements 1;
         break;
   case constant2:
         group of statements 2;
         break;
         . . .
   default:
         default group of statements;
}
```

- The switch statement uses *break* statement after the group of statements to be executed for a specific condition.
- Otherwise, the remainder cases will also be checked until the end of the switch selective block.
- *Default* case is optional.

Switch vs. IF-ELSE

- Switch can only be used to compare an expression against **constants**. We cannot put variables as labels or ranges because they are not valid C++ constants.

switch example	if-else equivalent

`switch (x) {`
`  case 1:`
`  cout << "x is 1";`
`  break;`
`  case 2:`
`  cout << "x is 2";`
`  break;`
`  default:`
`  cout << "value of x unknown";`
`  }`
	`if (x == 1) {`
	`  cout << "x is 1";`
	`  }`
	`else if (x == 2) {`
	`  cout << "x is 2";`
	`  }`
	`else {`
	`  cout << "value of x unknown";`
	`  }`

Grade Interpretation

```
char grade;
cout << "Enter student grade\n";
cin >> grade;
switch (grade)
{
case 'A':
      cout << "Excellent\n"; break;
case 'B':
      cout << "Very Good\n"; break;
case 'C':
      cout << "Good\n"; break;
case 'D':
      cout << "You can do better\n"; break;
case 'E':
      cout << "Disappointing\n"; break;
default:
     cout << "Invalid Grade\n";
}
```

`break` causes `switch` to end and the program continues with the first statement after the `switch` structure.

Grade Interpretation

```
char grade;
cout << "Enter student grade\n";
cin >> grade;
switch (grade)
{
case 'A': // Fall to through to next case
case 'a':
     cout << "Excellent\n"; break;
case 'B':
case 'b':
     cout << "Very Good\n"; break;
case 'C':
case 'c':
     cout << "Good\n"; break;
case 'D':
case 'd':
     cout << "You can do better\n"; break;
case 'E':
case 'e':
     cout << "Disappointing\n"; break;
default:
     cout << "Invalid Grade\n";
}
```

Problem

- Decide if input day is weekday or part of weekend based on its number.

```
int day;
cout << "Enter day\n";
cin >> day;
switch (day)
{
case 1: // Fall to through to the next case
case 2:
case 3:
case 4:
case 5:
    cout << "Weekday\n";    break;
case 6:
case 7:
    cout << "Weekend\n";    break;
default:
    cout << "Invalid day\n";
}
```

Design Structures

Sequence

One statement is executed after another

Selection/Decision

Statements can be executed or skipped depending on whether a condition evaluates to True or False

Repetition

Statements are executed repeatedly until a condition evaluates to True or False

While Loop

```
while (condition)
        statement;
```

```
while (condition)
{
        statement1;
        statement2;
}
```

Its functionality is simply to repeat statements as long as the condition is true.

```
1
2
3   #include <iostream>
4
5   using std::cout;
6   using std::endl;
7
8   // function main begins program execution
9   int main()
10  {
11     int number = 1;              // initialization
12
13     while (number <= 10 ) {     // repetition condition
14        cout << number << endl;  // display number
15        ++number;                // increment
16
17     } // end while
18
19     return 0;   // indicate successful termination
20
21  } // end function main
```

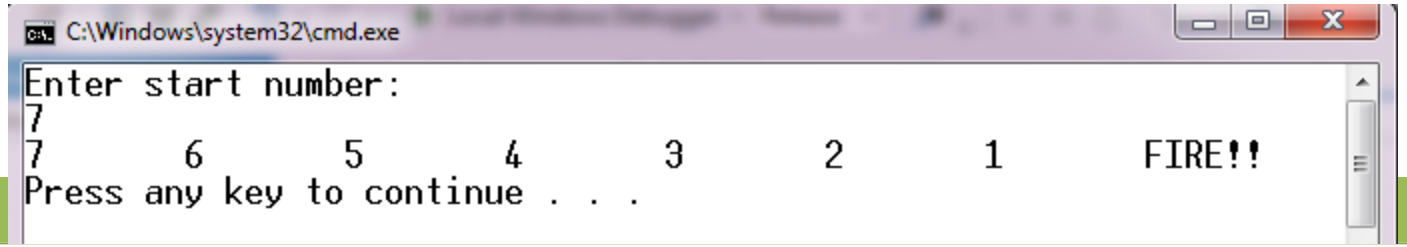
Output:

1
2
3
4
5
6
7
8
9
10

While Loop: Count Down

Program counts down from a user input value to 1 then prints FIRE!

```
int num;
cout << "Enter start number:\n";
cin >> num;
while (num > 0) // while (num >= 1)
{
  cout << num << '\t';
  num--;
}
cout << "FIRE!!\n";
```



Class Average (5 std.)

```
C:\Windows\system32\cmd.exe
Enter grade: 4
Enter grade: 4
Enter grade: 4
Enter grade: 4
Enter grade: 1
Class average is 3
Press any key to continue . . .
```

```
int total;          // sum of grades
int gradeCounter;   // number of grades to be entered
int grade;          // grade value
int average;        // average of grades

total = 0;          // initialize total
gradeCounter = 1;   // initialize loop counter

while (gradeCounter <= 5)            //loop 5 times
{
  cout << "Enter grade: ";          // prompt for input
  cin >> grade;                     //read grade from user
  total = total + grade;            // add grade to total
  gradeCounter++;                   // increment counter
}
average = total / 5;                // integer division
cout << "Class average is " << average << endl;
```

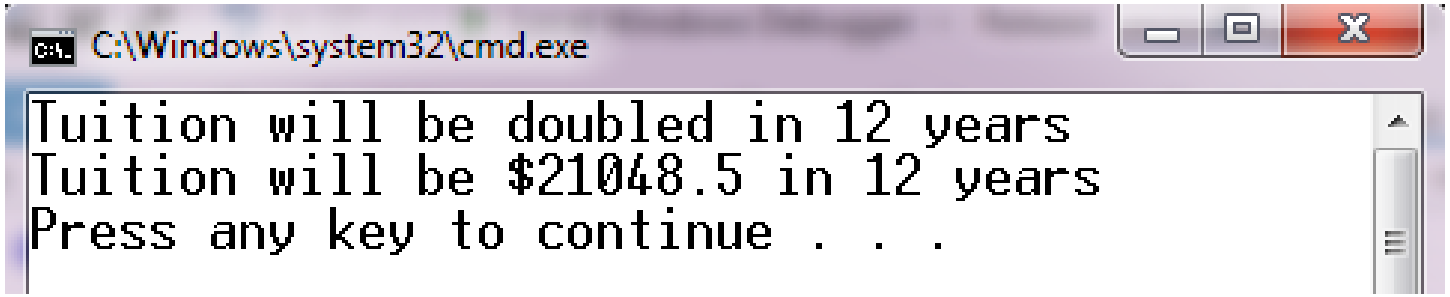
How to Display Correct Average?

```
Enter grade: 4
Enter grade: 4
Enter grade: 4
Enter grade: 4
Enter grade: 1
Class average is 3.4
Press any key to continue . . .
```

```
float total;
int gradeCounter;
int grade;
float average;

total = 0;
gradeCounter = 1;

while (gradeCounter <= 5)
{
  cout << "Enter grade: ";
  cin >> grade;
  total = total + grade;
  gradeCounter = gradeCounter + 1;
}
average = total / 5;
cout << "Class average is " << average << endl;
```

Problem

- Suppose that the tuition for a university is $10,000 this year and that the tuition increases 7% every year. In how many years will the tuition be doubled or more?



Solution

```
int year = 1;
float tuition = 10000;   // Year 1
while (tuition < 20000)
{
   year++;
   tuition *= 1.07;  //tuition = tuition + tuition * 0.07

}

cout << "Tuition will be doubled in " << year << " years"
<< endl;
cout << "Tuition will be $" << tuition << " in "
<< year << " years" << endl;
```

Do while Loop

```
do
    statement;
while (condition);
```

```
do
{
    statement1;
    statement2;
} while (condition);
```

Its functionality is simply to execute set of statements and then repeat them as long as the condition is true.

```cpp
// Fig. 2.24: fig02_24.cpp
// Using the do/while repetition structure.
#include <iostream>

using std::cout;
using std::endl;

// function main begins program execution
int main()
{
   int counter = 1;              // initialize counter

   do {
      cout << counter << "  ";   // display counter
   } while ( ++counter <= 10 );  // end do/while

   cout << endl;

   return 0;   // indicate successful termination

} // end function main
```

Notice the preincrement in loop-continuation test.

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

Do While: Count Down

```
int num;
cout << "Enter start number:\n";
cin >> num;
do
{
    cout << num << '\t';
    num--;
} while (num > 0);
cout << "FIRE!!\n";
```

Problem: Counting Grades

- Read student grades.
- User chooses when to stop.
- Display count of each grade.

- Variables needed?

```
char grade, answer;
int aCount=0, bCount=0,
cCount=0, dCount=0, eCount=0;
```

C:\Windows\system32\cmd.exe

```
Enter student grade
A
Do you want to continue?
y
Enter student grade
H
Invalid Grade
Do you want to continue?
y
Enter student grade
C
Do you want to continue?
y
Enter student grade
e
Do you want to continue?
y
Enter student grade
B
Do you want to continue?
y
Enter student grade
a
Do you want to continue?
n
2 students got an A
1 students got B
1 students got C
0 students got D
1 students got E
Press any key to continue . . .
```

Problem: Counting Grades

```
do
{
cout << "Enter student grade\n";
cin >> grade;
switch (grade)
{
     case 'A':
     case 'a': aCount++; break;
     case 'B':
     case 'b': bCount++; break;
     case 'C':
     case 'c': cCount++; break;
     case 'D':
     case 'd': dCount++; break;
     case 'E':
     case 'e': eCount++; break;
     default:
            cout << "Invalid Grade\n";
}
cout << "Do you want to continue?\n";
cin >> answer;
} while (answer == 'Y' || answer =='y');
```

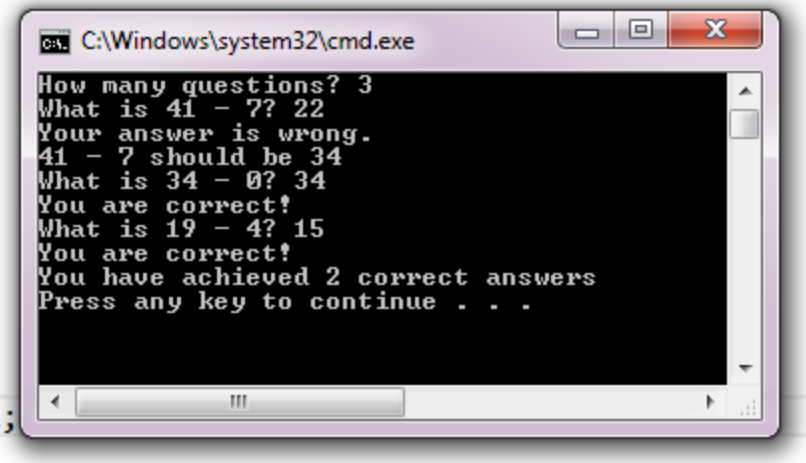
Problem: Practice Subtraction

```
int number1,number2,answer,numQuestions,trials=0;
cout << "How many questions? ";
cin >> numQuestions;
do
{
     number1 = rand() % 50;
     number2 = rand() % 10;
     cout << "What is " << number1 << " - " << number2 << "? ";
     cin >> answer;

     if (number1 - number2 == answer)
            cout << "You are correct!\n";

     else
            cout << "Your answer is wrong." << endl << number1 <<
            "-" << number2 << "=" << (number1 - number2) << endl;
     trials++;
} while (trials < numQuestions);
```

Problem: Practice Subtraction 2

- Display count of correct trials

```
int number1, number2, answer, numQuestions, trials=0, correctTrials=0;
cout << "How many questions? ";
cin >> numQuestions;

do
{
number1 = rand() % 50;
number2 = rand() % 10;
cout << "What is " << number1 << " - " << number2 << "? ";
cin >> answer;

if (number1 - number2 == answer)
{
  cout << "You are correct!\n";
  correctTrials++;
}
else
  cout << "Your answer is wrong." << endl
  << number1 << " - " << number2 <<
  " should be " << (number1 - number2) << endl;

trials++;
} while (trials < numQuestions);
cout<<"You have achieved "<< correctTrials<< " correct answers\n";
```



Problem

- Display number of passed and failed students.

```
int passed=0;
int failed=0;
float grade;
char answer;
do
{
    cout << "Enter grade\n";
    cin >> grade;
    (grade >= 60) ? passed++ : failed++ ;
    cout << "More students?\n";
    cin >> answer;
} while (answer == 'Y' || answer =='y');
cout << passed << " students have passed while " <<
failed << " failed" << endl;
```

Conditional operators can be used to perform process Such as increment variable

For Loop

```
for (initialization; condition; increment)
        statement;

for (initialization; condition; increment)
{
        statement1;
        statement2;
}
```

Repeat statements for a specified number of times.

Parameters are separated using ;

```cpp

// Counter-controlled repetition with the for structure
#include <iostream>

using std::cout;
using std::endl;

// function main begins program execution
int main()
{
   // Initialization, repetition condition and incrementing
   // are all included in the for structure header.

   for ( int counter = 1; counter <= 10; counter++ )
      cout << counter << endl;

   return 0;   // indicate successful termination

} // end function main
```

Output:

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

For Loop: Count Down

```
int main()
{
for (int i = 10; i > 0 ; i--)
    cout << i << '\t';
cout << "FIRE!!\n";
return 0;
}
```

```
C:\Windows\system32\cmd.exe
10      9       8       7       6       5       4       3       2       1
FIRE!!
Press any key to continue . . .
```

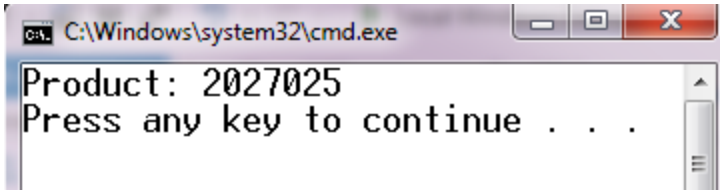
```
1
2   // Sum even integers in range from 2 to 100
3   #include <iostream>
4
5   using namespace std;
6
7
8   // function main begins program execution
9   int main()
10  {
11     int sum = 0;                          // initialize sum
12
13     // sum even integers from 2 through 100
14     for ( int number = 2; number <= 100; number += 2 )
15        sum += number;                     // add number to sum
16
17     cout << "Sum is " << sum << endl;   // output sum
18     return 0;                             // successful termination
19
20  } // end function main
```

```
Sum is 2550
```

Problem: Product of Numbers

- Calculate product of odd numbers from 1 to 15.

```
int main()
{
int product = 1;
for (int i = 1; i <= 15; i = i+2)
    product *= i;
cout << "Product: " << product << endl;
return 0;
}
```



Problem: Compound Interest

- A person invests $1000.00 in a savings account yielding 5 percent interest. Assuming that all interest is left on deposit in the account, calculate and print the amount of money in the account **at the end of each year for 10 years**. Use the following formula for determining these amounts:

  $a = p\,(1+r)^{n}$

  p is the original amount invested (i.e., the principal),
  r is the annual interest rate,
  n is the number of years and
  a is the amount on deposit at the end of the nth year.

Problem: Compound Interest

```
float amount;                // amount on deposit
float principal = 1000.0;    // starting principal
float rate = 0.05;           // interest rate

cout << "Year\t Amount on deposit" << endl;

// amount on deposit for each of ten years
for ( int year = 1; year <= 10; year++ )
{
     // calculate new amount for specified year
     amount = principal * pow( 1.0 + rate, year );

     cout << year << '\t' << amount << endl;
}
```

pow(x,y) = x^y

Note

- In for loop, the initialization and increase fields are optional. They can remain empty, but in all cases the semicolon signs between them must be written.
- For example we could write:

```
for (   ; n<10 ;  n++)
```

if we wanted to include an increase field but no initialization (maybe because the variable was already initialized before).

```
for (   ; n<10 ;   )
```

if we wanted to specify no initialization and no increase.

Note

- If the condition section in a for loop is omitted, it is implicitly evaluated to true.
- Thus the statement given below in (a), which is an infinite loop, is correct.
- However, it is better to use the equivalent loop in (b) to avoid confusion.

```
for ( ; ; )
{
   // Do something
}
```

(a)

Equivalent

This is better

```
while (true)
{
   // Do something
}
```

(b)

Exercise: Trace/What is Output

```
int k =3;
int m =2;
for (int j=3 ;  j <=10;  j=j+m)
   k = k + j;
   k = k * 3;
```

Remember to include necessary braces

j	k	m
	3	2
3	6	
5	11	
7	18	
9	27	
11	81	

Exercise: Trace/What is Output

```
int k =3;
int m =2;
for (int j=3 ;  j <=10;  j=j+m)
{
   k = k + j;
   k = k * 3;
}
```

j	k	m
	3	2
3	6	
	١٨	
5	23	
	69	
7	76	
	228	
9	237	
	711	
11		

Recommendation

- In general, a for loop may be used if the number of repetitions is counter-controlled, as, for example, when you need to do a process 100 times.
- A while loop may be used if the number of repetitions is sentinel-controlled, which means use an input value to signify the end of the loop. As in the case of reading the numbers until the input is 0.
- A do-while loop can be used if the loop body has to be executed before testing the continuation condition.

Jump Statements

- Break
- Continue

Break Statement

- **`break`** statement
  - Immediate exit from **`while`**, **`for`**, **`do/while`**, **`switch`**
  - Program continues with first statement after structure

- Common uses
  - Escape early from a loop

    Using break, we can leave a loop even if the condition for its end is not fulfilled. It can be used to end an infinite loop, or to force it to end before its natural end.

  - Skip the remainder of **`switch`**

Break Statement

```
while (test expression) {
    statement/s
    if (test expression) {
        break;
    }
    statement/s
}
```

```
do {
    statement/s
    if (test expression) {
        break;
    }
    statement/s
}
while (test expression);
```

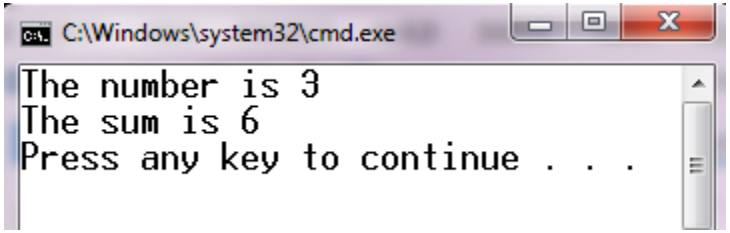
```
for (intial expression; test expression; update expression) {
    statement/s
    if (test expression) {
        break;
    }
    statements/
}
```

NOTE: The break statment may also be used inside body of else statement.

Break Statement

- Calculate the sum of integers starting from 0 until the sum is an even number.

```
int sum = 0;
int number = 0;
while (true)
{
  number++;
  sum += number;
  if (sum % 2 == 0)
    break;
}
cout << "The number is " << number << endl;
cout << "The sum is " << sum << endl;
```



Continue Statement

- Continue statement:
  - Causes the program to skip the rest of the loop in the current iteration as if the end of the statement block had been reached.
  - Jumps to the start of the following iteration.

Continue Statement

```
while (test expression) {
    statement/s
    if (test expression) {
        continue;
    }
    statement/s
}
```

```
do {
    statement/s
    if (test expression) {
        continue;
    }
    statement/s
}
while (test expression);
```

```
for (intial expression; test expression; update expression) {
    statement/s
    if (test expression) {
        continue;
    }
    statements/
}
```

NOTE: The continue statment may also be used inside body of else statement.

Continue Statement

- **`continue`** statement
  - Used in **`while`**, **`for`**, **`do/while`**
  - Skips remainder of loop body for current iteration
  - Proceeds with next iteration of loop
- **`while`** and **`do/while`** structure
  - Loop-continuation test evaluated immediately after the **`continue`** statement
- **`for`** structure
  - Increment expression executed
  - Next, loop-continuation test evaluated

Continue Statement

Display the sum of integers less than 20, do not include numbers 10 and 11 in the summation process.

```
int sum = 0;

for (int number = 0; number < 20; number++)
{
  if (number == 10 || number == 11)
     continue;
  sum += number;
}

cout << "The sum: " << sum << endl;
```

Exercise: Break

```
int x = 0;
int y = 1;
do
{
  x += 2;
  for (int i=1; i<4; i++)
  {
      y= y+ 2*i;
      if (y > 25 )
            break;
      y += 3;
  }
  y = y + x;
} while (x < 6 && y < 100);
```

x	y	i
0	1	
2	3	1
	6	
	10	2
	13	
	19	3
	22	
	24	
4	26	1
	30	
6	32	1
	38	

Exercise: Continue

```
int x = 0;
int y = 1;
do
{
  x += 2;
  for (int i=1; i<4; i++)
  {
      y= y+ 2*i;
      if (y > 25 )
            continue;
      y += 3;
  }
  y = y + x;
} while (x < 6 && y < 100);
```

x	y	i
0	1	
2	3	1
	6	
	10	2
	13	
	19	3
	22	
	24	
4	26	1
	30	2
	36	3
	40	
6	42	1
	46	2
	52	3
	58	

Exercise: Correct Code

```
char done = 'Y';
while (done = 'Y')
{
    //...
    cout << "Continue? (Y/N)";
    cin >> done;
}
```

"Why doesn't my loop ever end?"

```
char done = 'Y';
while (done == 'Y')
{
    //...
    cout << "Continue? (Y/N)";
    cin >> done;
}
```

If you use a single equal sign to check equality, your program will instead assign the value on the right side of the expression to the variable on the left hand side, and the result of this statement is TRUE. Therefore, the loop will never end.

Exercise: Correct Code

```
int x;
for (x = 0; x < 100; x++) ;
    cout << x;
```

"Why does it just output 100?"

```
int x;
for (x = 0; x < 100; x++)
    cout << x;
```

Removed extra semicolon

Remember, semicolons don't go after FOR loop definition.
If you put it, your program will function improperly.

Exercise: Complete

1. Every C++ program begins execution at function …
2. All variables must be given a … when they are declared.
3. The … statement is used to make a decision.
4. The … operator can be used as a prefix or as a suffix.
5. The … loop is sentinel-controlled.
6. In case statement, the … is optional.
7. … is the only ternary operator in C++.
8. The … statement causes the program to skip the rest of the loop in the current iteration.

Exercise: True/False

1. C++ considers the variables *number* and *Number* to be identical.
2. An expression containing the || operator evaluates to true if either or both of its operands are true.
3. The default case is required in the switch statement.
4. Operators *, / and % have the same evaluation precedence.
5. The break statement jumps to the start of the following iteration.
6. Any IF..ELSE statement can be represented using switch case.

Exercises

if (isBool = true)

if (isBool)

- Are these statements equivalent?
- Differentiate between: Continue and Break.
- What happens when you omit the default case?
- Which loop can be used if the loop body has to be executed before testing the continuation condition?

Thank You