Advanced Programming (C++)

BY

Dr. EMAD SAMI

http://www.bu.edu.eg/staff/emadattwa3

Course Chapters

3. Expressions and Statements

Chapter Objectives:

<u>3-1 Operators</u>

- Are symbols that cause a program to do something to variables. For example, the arithmetic operator (+) causes the program to **add** two numbers.
- There are three sections for operators, as shown in the THREE tables:

Table 1:

Arithmetic Operators			
+	**Addition**	+=	**Addition Assignment**
-	**Subtraction**	-=	**Subtraction Assignment**
*	**Multiplication**	*=	**Multiplication Assignment**
/	**Division**	/=	**Division Assignment**
++	**Prefix Increment**	%	**Reminder (Modulus)**
- -	**Prefix decrement**	%=	**Reminder Assignment**

3-1 Operators …

Table 2:

Relational Operators			
>	Grater than	<=	Less than or equal
<	Less than	= =	Equal
>=	Greater than or equal	!=	Not Equal

Table 3:

Logical Operators	
&&	Logical AND
||	Logical OR
!	Logical NOT

3-1 Operators …

- There is a certain priority (or precedence أولوية) to compute arithmetic phrase contains more than one operator, see table 4.
- Priority achieves from **Left** to **Right** as:

Brackets → Arithmetic → Relational → Logical

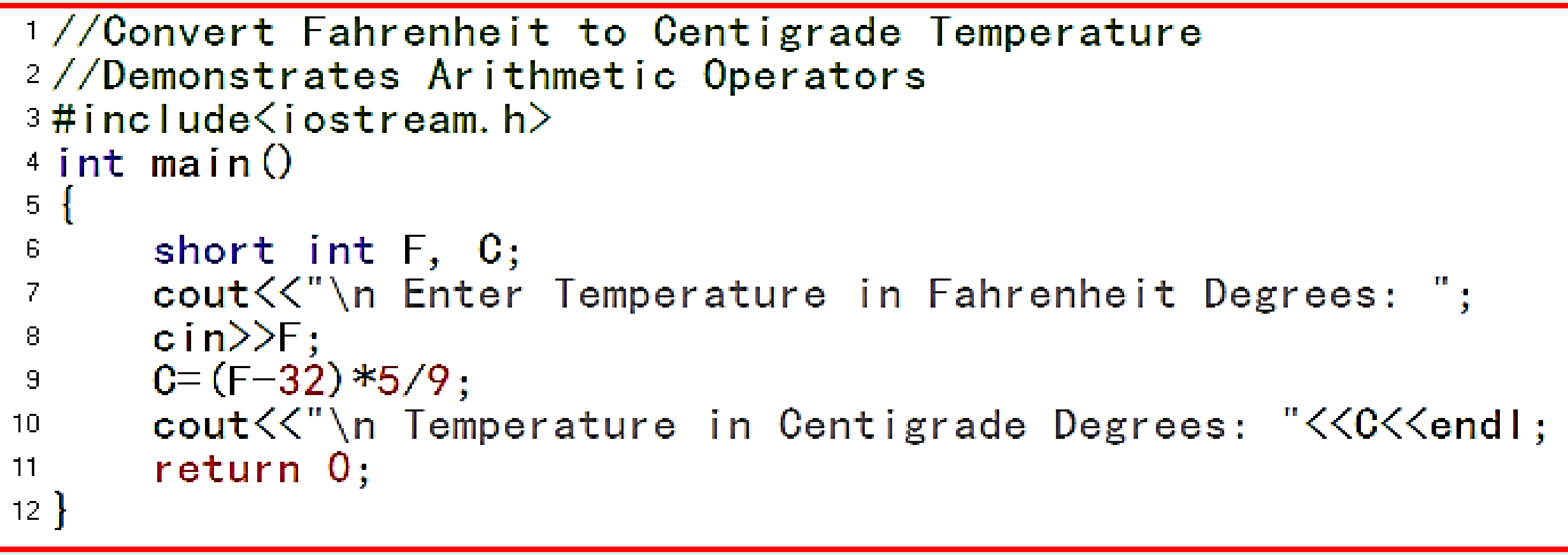
Table 4: **حفظ الجدول**

Priority of Operations		
1	**( )**	**Brackets الأقواس**
2	^	**Exponent الأس**
3	*** , / , % , *= , /= , %=**	
4	**+ , - , ++ , - - , + = , - =**	
5	**> , < , >= , <= , = = , !=**	
6	**&& , ||**	

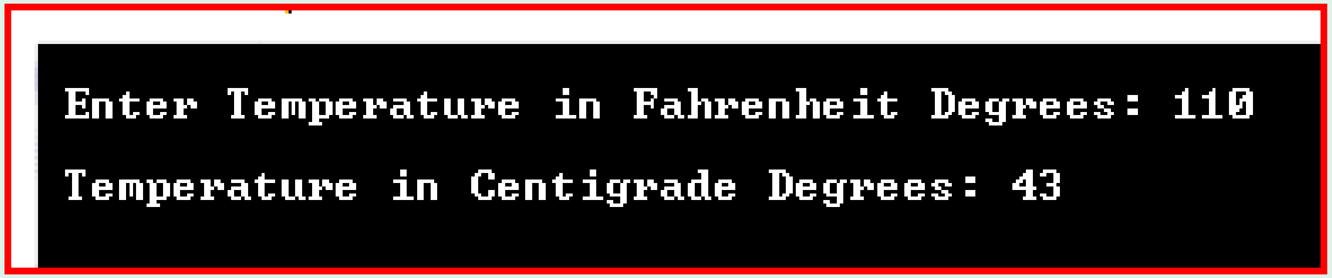
- Example: 5+3*6/3-(1^2+10/5) = ….. Ans.: 8

3-1-1 Arithmetic Operators

- Here is a program to demonstrates using of arithmetic operators. The program converts Fahrenheit to Centigrade temperature.
- FtoC.cpp



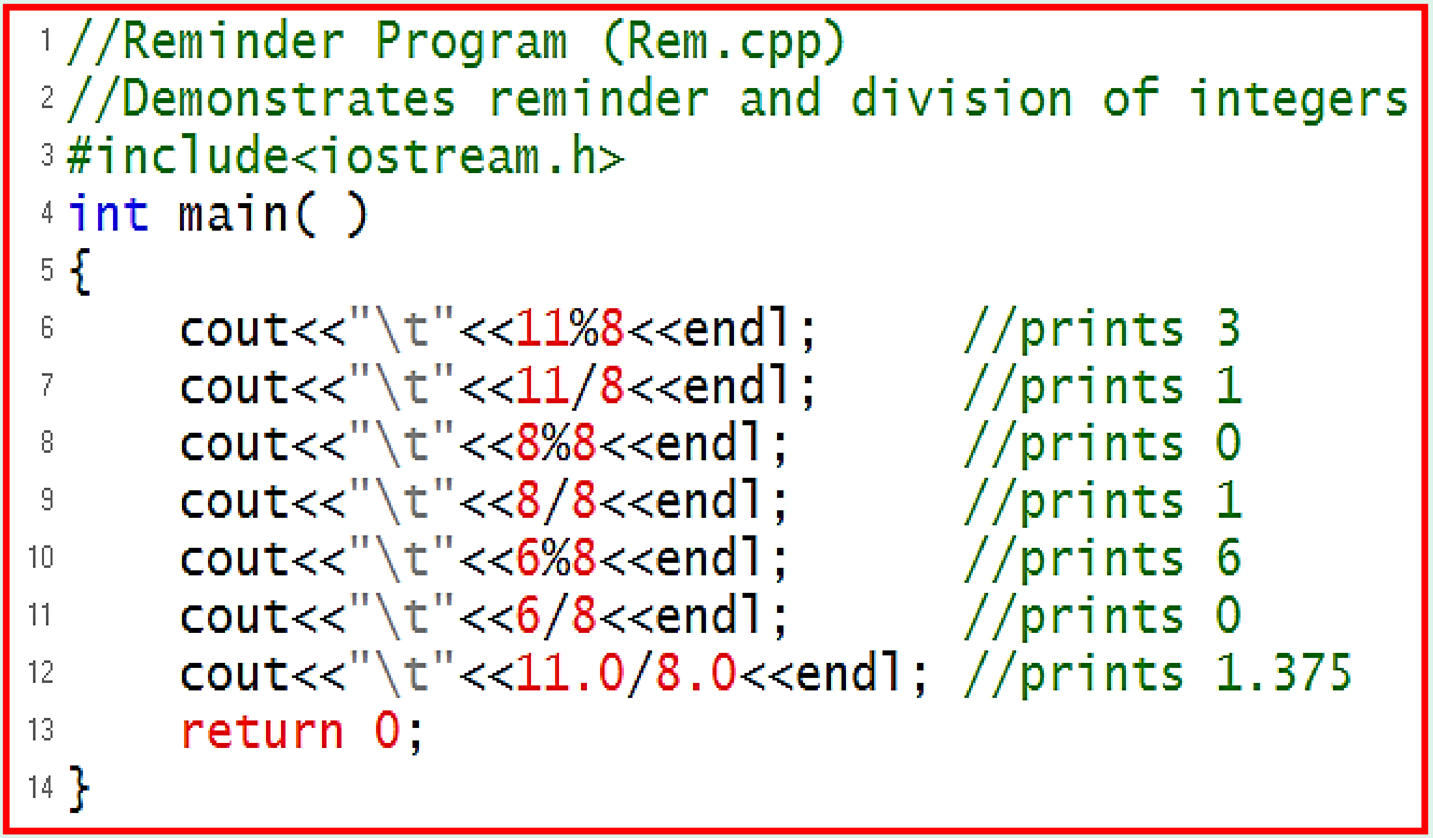
```
//Convert Fahrenheit to Centigrade Temperature
//Demonstrates Arithmetic Operators
#include<iostream.h>
int main()
{
     short int F, C;
     cout<<"\n Enter Temperature in Fahrenheit Degrees: ";
     cin>>F;
     C=(F-32)*5/9;
     cout<<"\n Temperature in Centigrade Degrees: "<<C<<endl;
     return 0;
}
```



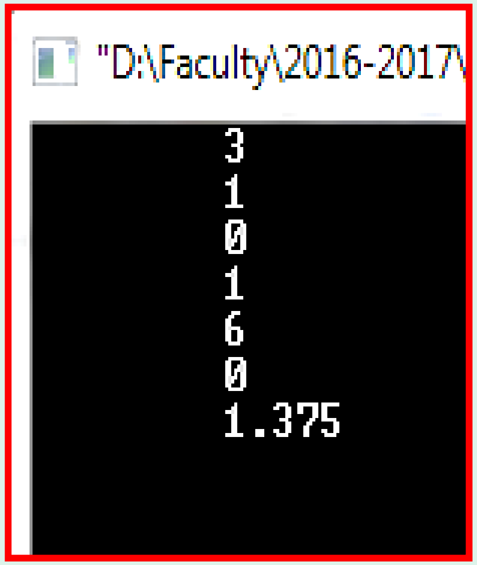
```
Enter Temperature in Fahrenheit Degrees: 110

Temperature in Centigrade Degrees: 43
```

3-1-2 Reminder (Modulus) Operator (%)

- The reminder operator is used to find the reminder الباقي when one *integer* number is divided by another *integer* number. It works only with **integer** variables.
- Rem.cpp



```
//Reminder Program (Rem.cpp)
//Demonstrates reminder and division of integers
#include<iostream.h>
int main( )
{
      cout<<"\t"<<11%8<<endl;      //prints 3
      cout<<"\t"<<11/8<<endl;      //prints 1
      cout<<"\t"<<8%8<<endl;       //prints 0
      cout<<"\t"<<8/8<<endl;       //prints 1
      cout<<"\t"<<6%8<<endl;       //prints 6
      cout<<"\t"<<6/8<<endl;       //prints 0
      cout<<"\t"<<11.0/8.0<<endl; //prints 1.375
      return 0;
}
```



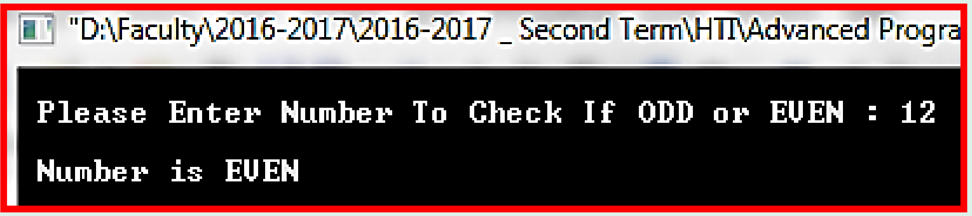
```
"D:\Faculty\2016-2017\
        3
        1
        0
        1
        6
        0
        1.375
```

3-1-2 Reminder (Modulus) Operator (%) …

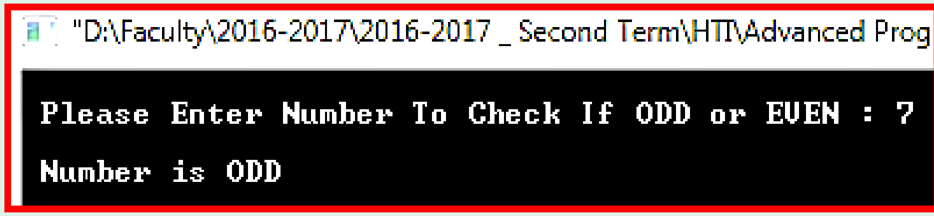
- Here is a program ask the user to enter a number and the program check if that number is odd or even.

OE.cpp

```cpp
//Reminder Program (OE.cpp)
//Demonstrates IF the input number is ODD or EVEN
#include<iostream.h>
int main( )
{
      int A;
      cout<<"\n Please Enter Number To Check If ODD or EVEN : ";
      cin>>A;
      if(A%2==0)
        cout<<"\n Number is EVEN \n";
      else
        cout<<"\n Number is ODD \n";
return 0;
}
```



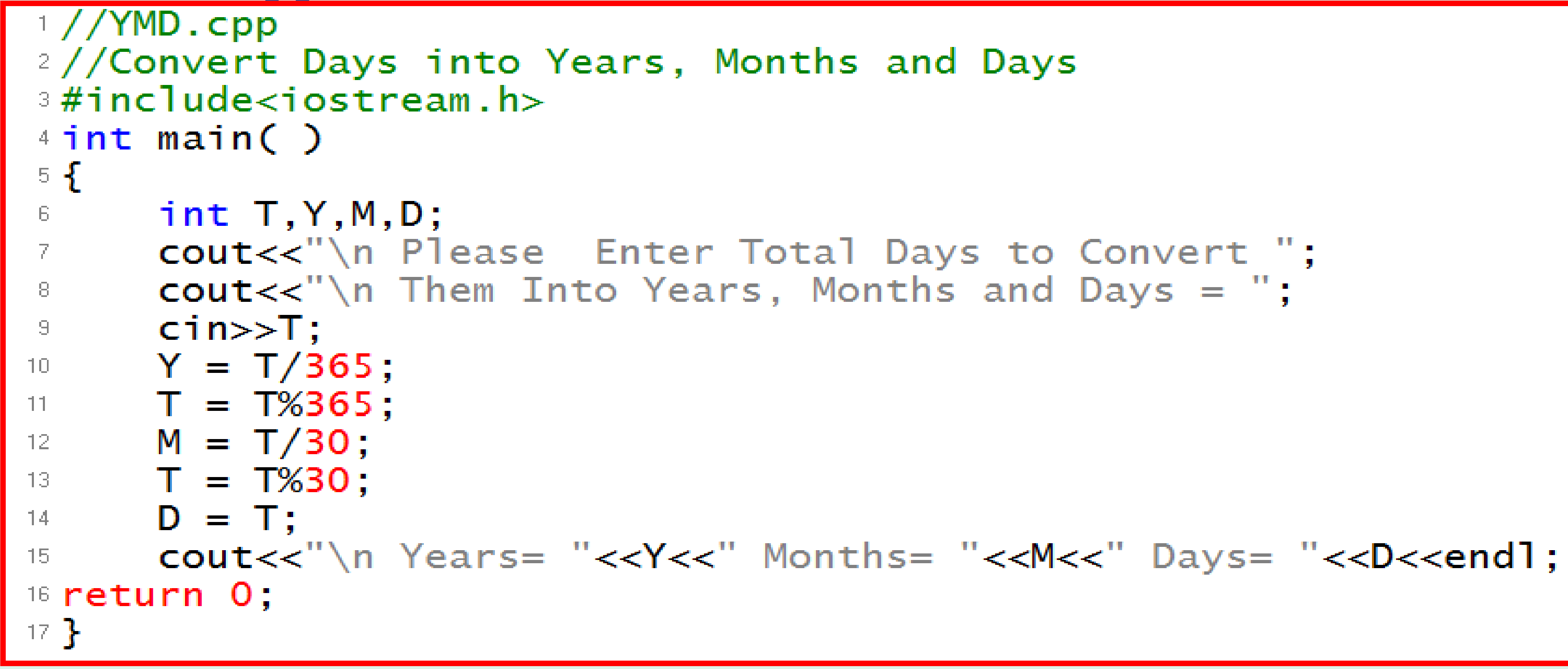
```
"D:\Faculty\2016-2017\2016-2017 _ Second Term\HTI\Advanced Progra

Please Enter Number To Check If ODD or EVEN : 12
Number is EVEN
```



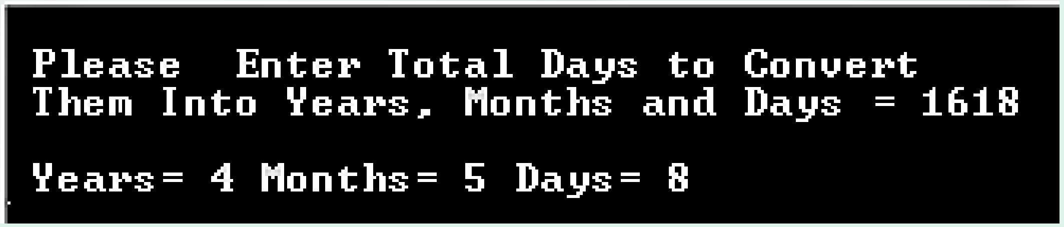
```
"D:\Faculty\2016-2017\2016-2017 _ Second Term\HTI\Advanced Prog

Please Enter Number To Check If ODD or EVEN : 7
Number is ODD
```

3-1-2 Reminder (Modulus) Operator (%) …

- Here is a program ask the user to input total number of days and convert them into Years, Months, Days.
- YMD.cpp



```
//YMD.cpp
//Convert Days into Years, Months and Days
#include<iostream.h>
int main( )
{
    int T,Y,M,D;
    cout<<"\n Please  Enter Total Days to Convert ";
    cout<<"\n Them Into Years, Months and Days = ";
    cin>>T;
    Y = T/365;
    T = T%365;
    M = T/30;
    T = T%30;
    D = T;
    cout<<"\n Years= "<<Y<<" Months= "<<M<<" Days= "<<D<<endl;
return 0;
}
```



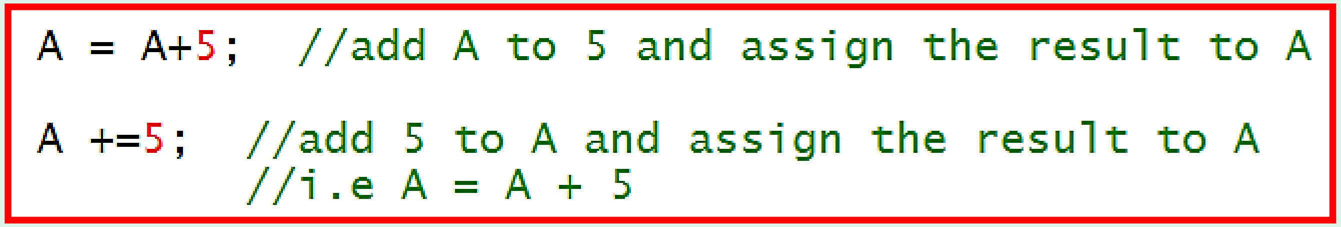
```
Please  Enter Total Days to Convert
Them Into Years, Months and Days = 1618

Years= 4 Months= 5 Days= 8
```

3-1-2 Reminder (Modulus) Operator (%) …

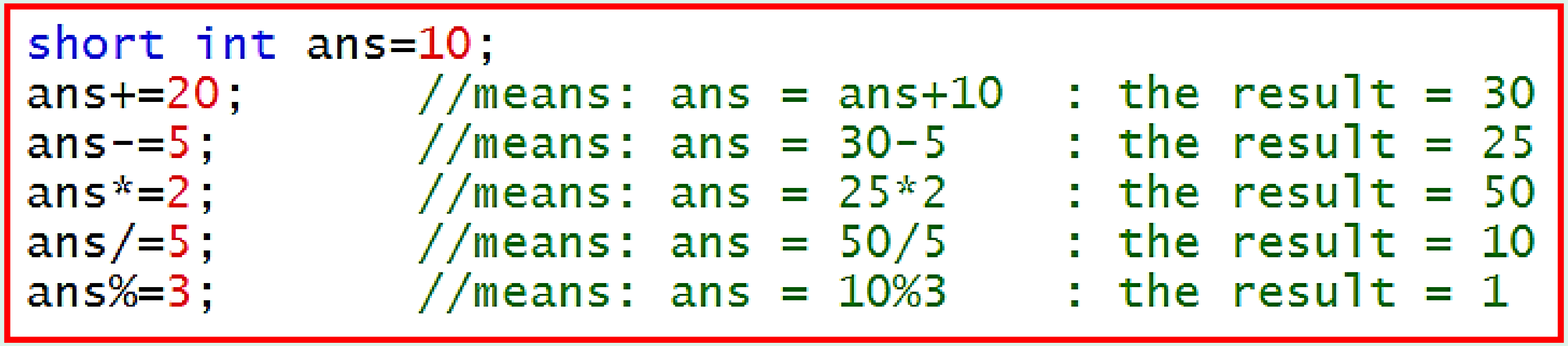
- Here is a program ask the user to input any amount of Pounds and convert them into Hundreds, Fifties, Twenties, Tens, Piasters. (***Home Work***)
- HFTTP.cpp

3-1-3 Arithmetic Assignment Operators

- C++ offers several ways to shorten and clarify your code by using these Assignment Operators (*=, /=, +=, - =, %=).
- Examples,



```
A = A+5;  //add A to 5 and assign the result to A

A +=5;  //add 5 to A and assign the result to A
        //i.e A = A + 5
```



```
short int ans=10;
ans+=20;       //means: ans = ans+10  : the result = 30
ans-=5;        //means: ans = 30-5    : the result = 25
ans*=2;        //means: ans = 25*2    : the result = 50
ans/=5;        //means: ans = 50/5    : the result = 10
ans%=3;        //means: ans = 10%3    : the result = 1
```

- You don’t need to use arithmetic assignment operators in your code, but they are a common feature of the language. They will appear in many examples.

3-1-4 Increment and Decrement

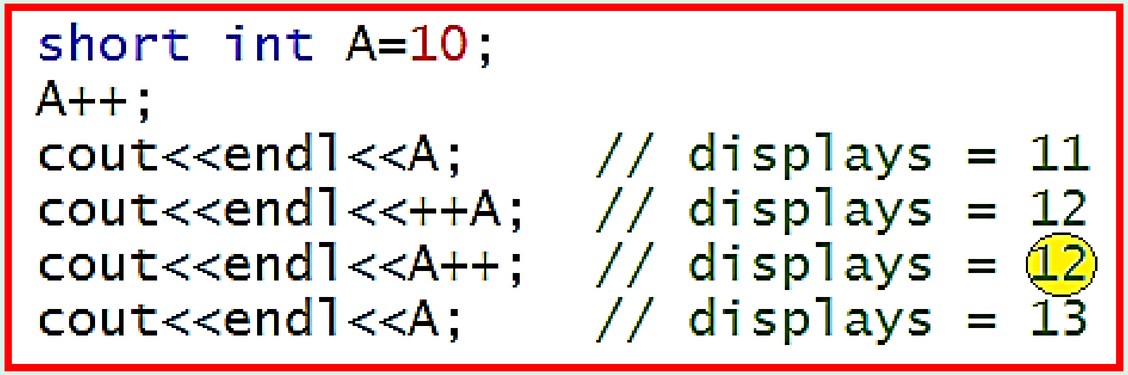
- In C++ increasing a value by 1 is called incrementing and decreasing it by 1 is called decrementing.
- The increment operator (++) increases the value by 1.
- The decrement operator (--) decreases the value by 1.
- Both the increment operator (++) and the decrement operator (--) comes in two ways, **prefix** and **postfix**.
- The **prefix** way is written before the variable name (++count or --count), the **postfix** way is written after the variable name (count++ or count--).
- Note that:

In the **prefix**:	increment the value and then use it. decrement the value and then use it.
In the **postfix**:	use the value and then increment it. use the value and then decrement it.

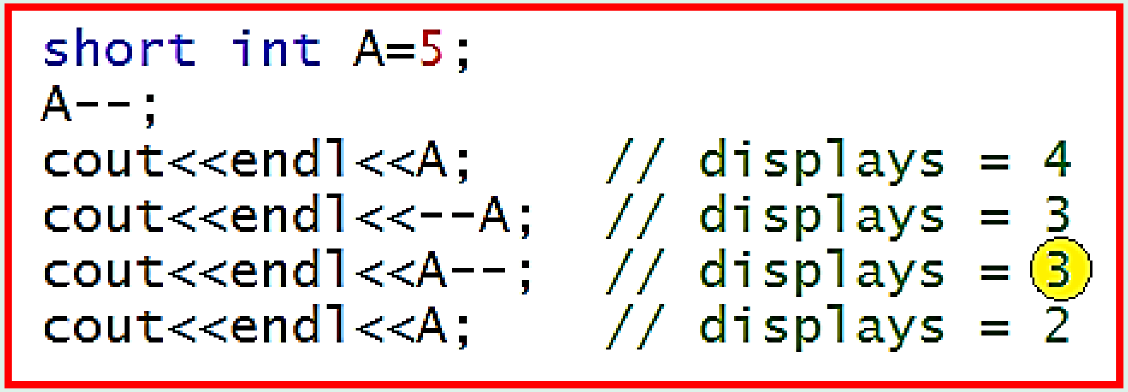
3-1-4 Increment and Decrement …

prefix	++count; // count = count +1
	--count; // count = count - 1
postfix	count++; // count+=1
	count--; // count-=1

• Here is two programs to illustrate this:



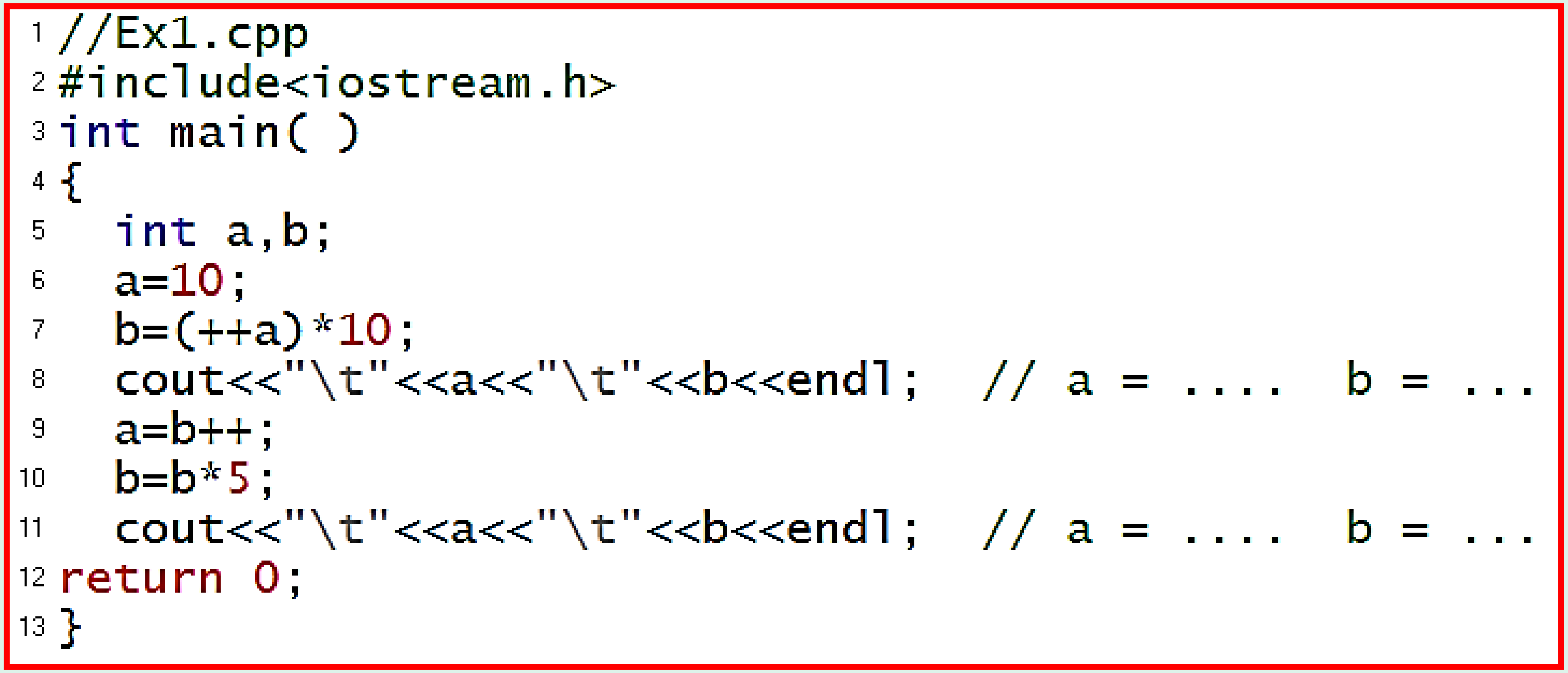
```
short int A=10;
A++;
cout<<endl<<A;     // displays = 11
cout<<endl<<++A;   // displays = 12
cout<<endl<<A++;   // displays = 12
cout<<endl<<A;     // displays = 13
```



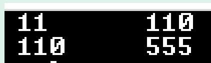
```
short int A=5;
A--;
cout<<endl<<A;     // displays = 4
cout<<endl<<--A;   // displays = 3
cout<<endl<<A--;   // displays = 3
cout<<endl<<A;     // displays = 2
```

3-1-4 Increment and Decrement …

- Discuss the output of the following code:



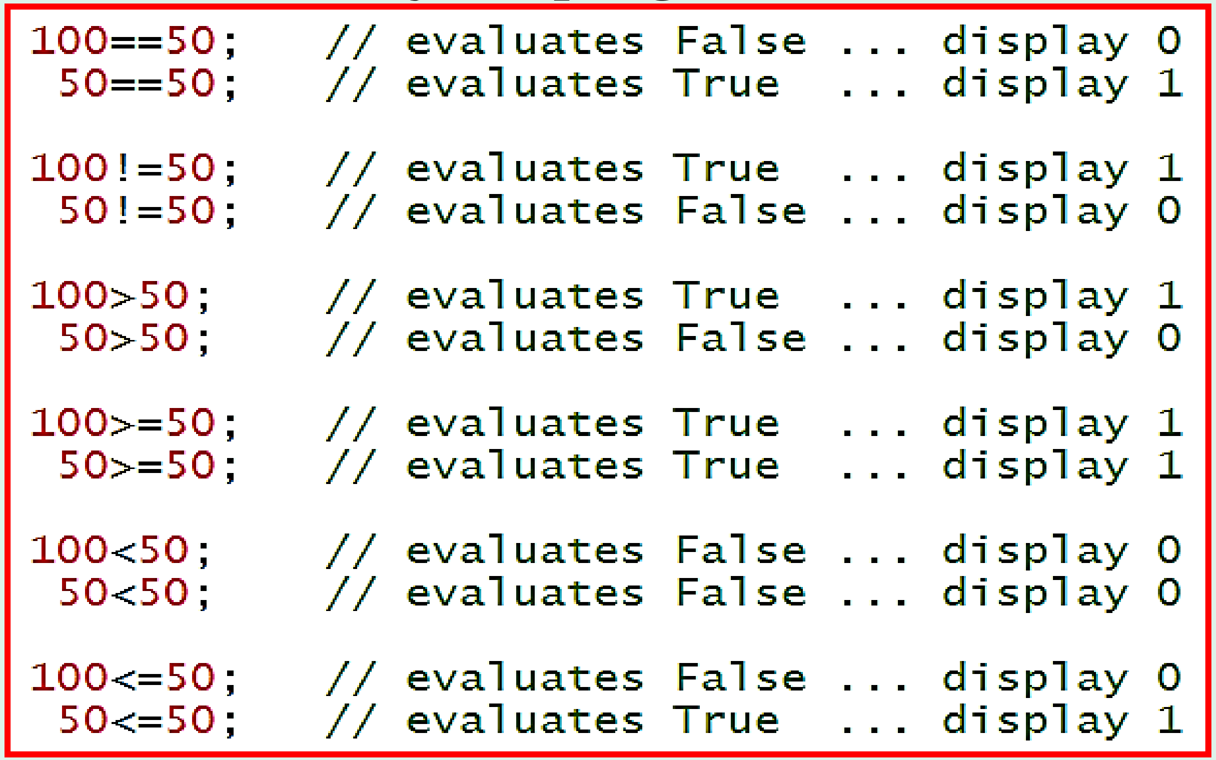
```
//Ex1.cpp
#include<iostream.h>
int main( )
{
  int a,b;
  a=10;
  b=(++a)*10;
  cout<<"\t"<<a<<"\t"<<b<<endl;  // a = ....  b = ...
  a=b++;
  b=b*5;
  cout<<"\t"<<a<<"\t"<<b<<endl;  // a = ....  b = ...
return 0;
}
```



```
 11      110
 110     555
```

3-1-5 Relational Operators

- Every relational statement evaluates to either **1** (**True**) or **0** (**False**).
- In C++, **zero** is considered **False**, and **all other values** are considered **True**, although **True** is usually represented by **1**.
- ***Warnings***: Many new C++ programmers confuse the assignment operator (=) with the equality operator (==). This creates a bad ***error*** in your program.

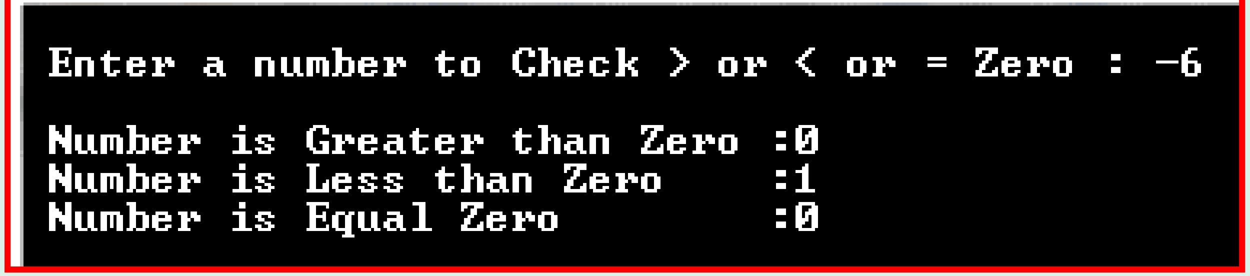


```
100==50;   // evaluates False ... display 0
 50==50;   // evaluates True  ... display 1

100!=50;   // evaluates True  ... display 1
 50!=50;   // evaluates False ... display 0

100>50;    // evaluates True  ... display 1
 50>50;    // evaluates False ... display 0

100>=50;   // evaluates True  ... display 1
 50>=50;   // evaluates True  ... display 1

100<50;    // evaluates False ... display 0
 50<50;    // evaluates False ... display 0

100<=50;   // evaluates False ... display 0
 50<=50;   // evaluates True  ... display 1
```

3-1-5 Relational Operators …

- Write a program ask the user to an input number and the programs displays that if the number is *greater than* or *less than* or *equal* **0**. Relation.cpp

```cpp
//Relation.cpp
//demonstrates > or < or = zero
#include<iostream.h>
int main( )
{
  int x;
  cout<<"\n Enter a number to Check > or < or = Zero : ";
  cin>>x;
  cout<<" \n Number is Greater than Zero :"<<(x>0);
  cout<<" \n Number is Less than Zero    :"<<(x<0);
  cout<<" \n Number is Equal Zero        :"<<(x==0);
  cout<<endl;
return 0;
}
```



```
Enter a number to Check > or < or = Zero : -6

Number is Greater than Zero :0
Number is Less than Zero    :1
Number is Equal Zero        :0
```

3-1-6 Logical Operators

- The logical operators that found in C++ are :

  AND → **&&**　　　　OR→ | |　　　　NOT→ **!**

- Often you want to ask more than one relational question at a time.
- A program might need to determine that both or any of these conditions are true in order to make an action.
- Logical AND operator (**&&**):

if (condition1 && condition2)

if the two conditions are True then the result condition is True.

Condition1	condition2	Result
T	**T**	**T**
T	**F**	**F**
F	**T**	**F**
F	**F**	**F**

3-1-6 Logical Operators …

- Logical OR operator (| |):

if (condition1 | | condition2)

if one condition is True then the result condition is True.

- Logical NOT operator (**!**):

if (!condition)

if the condition is False then the result condition is True.

again, if the value of the test is False then the result condition is True.

- ***Example***: if X=3, Y=5 and Z=7. What is the result condition, True or False?

if (X==3 && Y==5)	→ The result is True
if (X==4 && Y==5)	→ The result is False
if (Y==1 | | Z==7)	→ The result is True
if (!(Y==1)) ↔ if (Y!=1))	→ The result is True
if (X!=6 && Y==5 | | Z==7)	→ The result is True
if (!(X==3) && Y==5 | | Z==7)	→ The result is True
if (X==5 | | Y==5 && Z==6)	→ The result is False

3-1-6 Logical Operators …

- To construct a program gives the values of the previous example, you must use one of these programming procedure:
- Note the difference; the two cases are the same:

Conditional (Ternary) Operator	**if …else statement**

(expression1) ? (expression2) : (expression3)
 	
Means: if expressio1 is True, return the value of expressio2; otherwise, return the value of expression3.
	if (……… < ………)
	……………;
	else
	……………;

- Note the difference; the two conditions are the same:

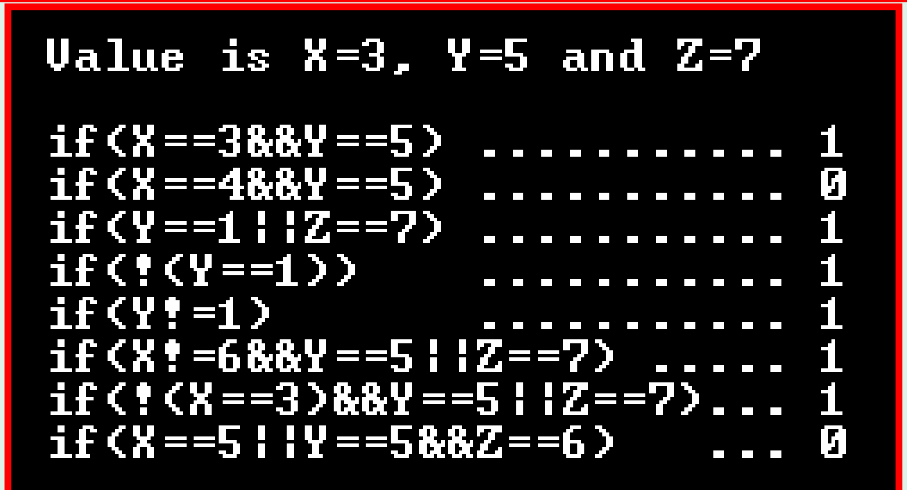
Case (1)	**if (**Numb1 < Numb2**)**
	Minimum = Numb1**;**
	else
	Minimum = Numb2**;**

Case (2)	**(**Minimum = (Numb1<Numb2)**) ?** Numb1 **:** Numb2**;**

3-1-6 Logical Operators …

- LogicA.cpp   using ternary operator

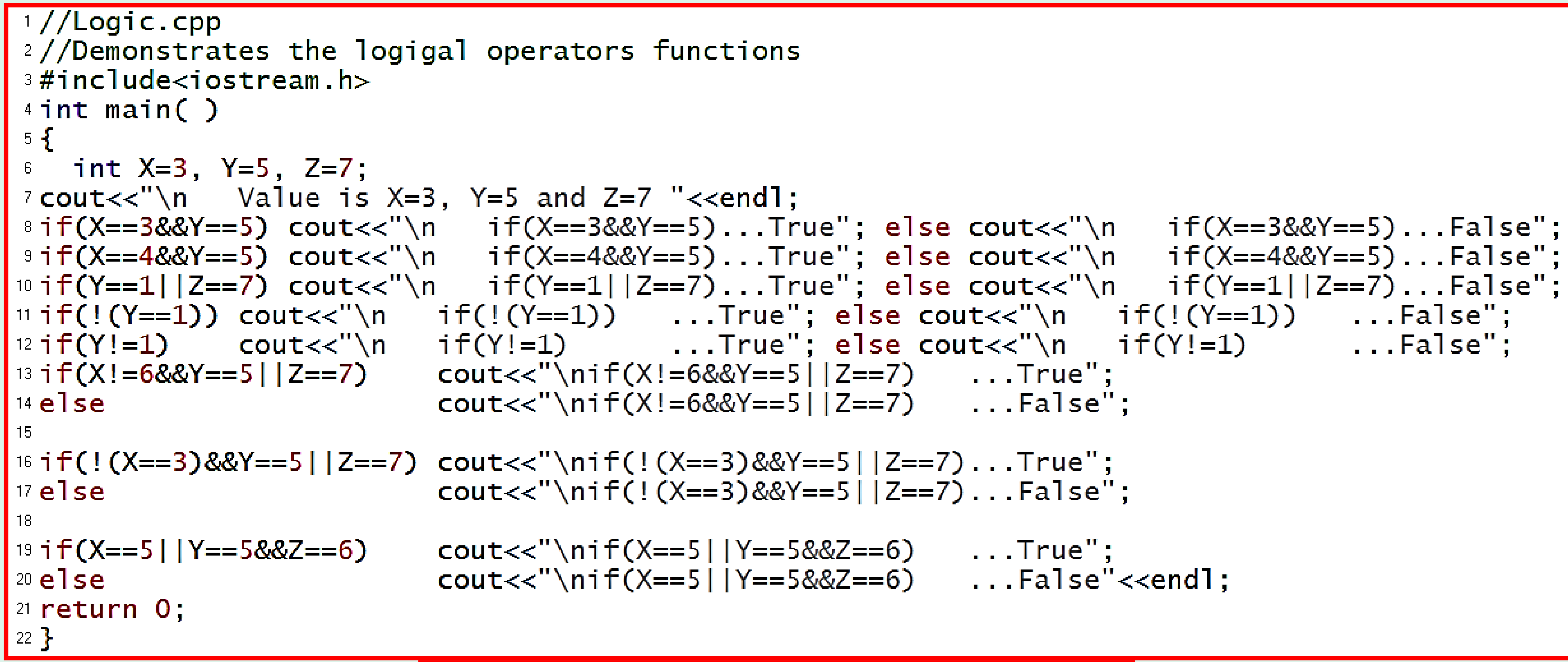
```cpp
//LogicA.cpp
//Demonstrates the logigal operators functions
#include<iostream.h>
int main( )
{
  int X=3, Y=5, Z=7;
cout<<"\n   Value is X=3, Y=5 and Z=7 "<<endl;
cout<<"\n   if(X==3&&Y==5) ........... "<<(X==3&&Y==5)?(1):(0);
cout<<"\n   if(X==4&&Y==5) ........... "<<(X==4&&Y==5)?(1):(0);
cout<<"\n   if(Y==1||Z==7) ........... "<<(Y==1||Z==7)?(1):(0);
cout<<"\n   if(!(Y==1))    ........... "<<(!(Y==1))?(1):(0);
cout<<"\n   if(Y!=1)       ........... "<<(Y!=1)?(1):(0);
cout<<"\n   if(X!=6&&Y==5||Z==7) ..... "<<(X!=6&&Y==5||Z==7)?(1):(0);
cout<<"\n   if(!(X==3)&&Y==5||Z==7)... "<<(!(X==3)&&Y==5||Z==7)?(1):(0);
cout<<"\n   if(X==5||Y==5&&Z==6)   ... "<<(X==5||Y==5&&Z==6)?(1):(0);
cout<<"\n ";
return 0;
}
```



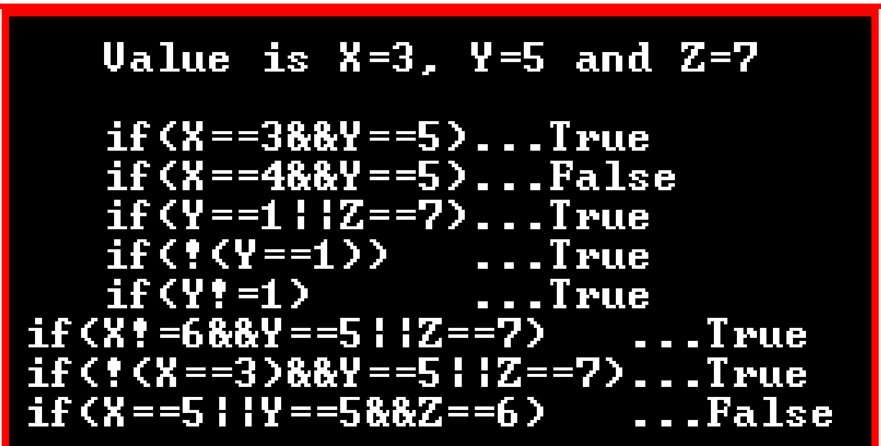
```
Value is X=3, Y=5 and Z=7

if(X==3&&Y==5) ............ 1
if(X==4&&Y==5) ............ 0
if(Y==1¦¦Z==7) ............ 1
if(!(Y==1))    ............ 1
if(Y!=1)       ............ 1
if(X!=6&&Y==5¦¦Z==7) ...... 1
if(!(X==3)&&Y==5¦¦Z==7).... 1
if(X==5¦¦Y==5&&Z==6)    ... 0
```

3-1-6 Logical Operators …

- Logic.cpp using if --- else (if …. else سوف يتم لاحقا شرح قاعدة)



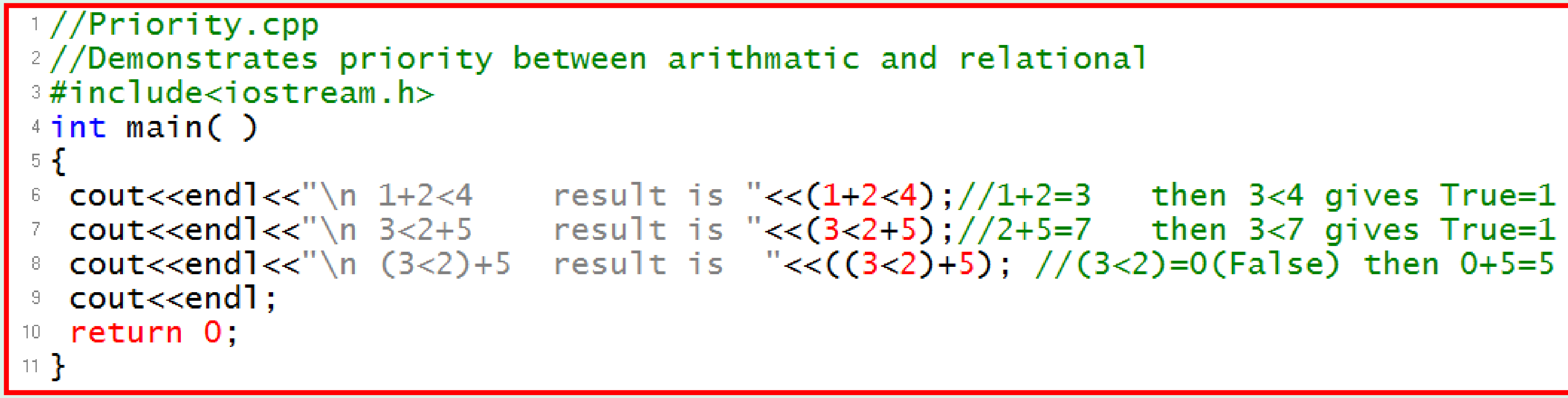
```
//Logic.cpp
//Demonstrates the logigal operators functions
#include<iostream.h>
int main( )
{
  int X=3, Y=5, Z=7;
cout<<"\n   Value is X=3, Y=5 and Z=7 "<<endl;
if(X==3&&Y==5) cout<<"\n   if(X==3&&Y==5)...True"; else cout<<"\n   if(X==3&&Y==5)...False";
if(X==4&&Y==5) cout<<"\n   if(X==4&&Y==5)...True"; else cout<<"\n   if(X==4&&Y==5)...False";
if(Y==1||Z==7) cout<<"\n   if(Y==1||Z==7)...True"; else cout<<"\n   if(Y==1||Z==7)...False";
if(!(Y==1)) cout<<"\n   if(!(Y==1))   ...True"; else cout<<"\n   if(!(Y==1))   ...False";
if(Y!=1)    cout<<"\n   if(Y!=1)      ...True"; else cout<<"\n   if(Y!=1)      ...False";
if(X!=6&&Y==5||Z==7)    cout<<"\nif(X!=6&&Y==5||Z==7)   ...True";
else                    cout<<"\nif(X!=6&&Y==5||Z==7)   ...False";

if(!(X==3)&&Y==5||Z==7) cout<<"\nif(!(X==3)&&Y==5||Z==7)...True";
else                    cout<<"\nif(!(X==3)&&Y==5||Z==7)...False";

if(X==5||Y==5&&Z==6)    cout<<"\nif(X==5||Y==5&&Z==6)   ...True";
else                    cout<<"\nif(X==5||Y==5&&Z==6)   ...False"<<endl;
return 0;
}
```



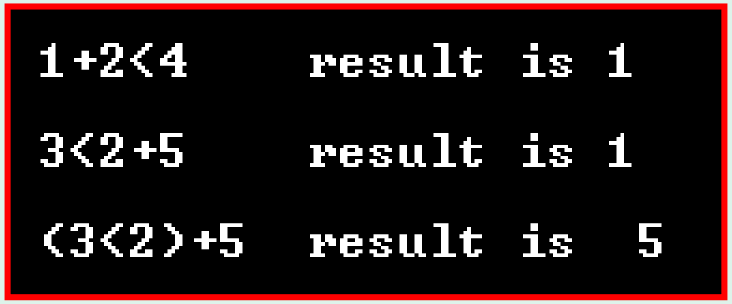
```
   Value is X=3, Y=5 and Z=7

   if(X==3&&Y==5)...True
   if(X==4&&Y==5)...False
   if(Y==1||Z==7)...True
   if(!(Y==1))   ...True
   if(Y!=1)      ...True
if(X!=6&&Y==5||Z==7)   ...True
if(!(X==3)&&Y==5||Z==7)...True
if(X==5||Y==5&&Z==6)   ...False
```

3-1-7 Priority (Precedence)

- If an expression contains both arithmetic and relational, then arithmetic operators have a higher priority.
- Priority.cpp



```
//Priority.cpp
//Demonstrates priority between arithmatic and relational
#include<iostream.h>
int main( )
{
 cout<<endl<<"\n 1+2<4    result is "<<(1+2<4);//1+2=3   then 3<4 gives True=1
 cout<<endl<<"\n 3<2+5    result is "<<(3<2+5);//2+5=7   then 3<7 gives True=1
 cout<<endl<<"\n (3<2)+5  result is  "<<((3<2)+5); //(3<2)=0(False) then 0+5=5
 cout<<endl;
 return 0;
}
```



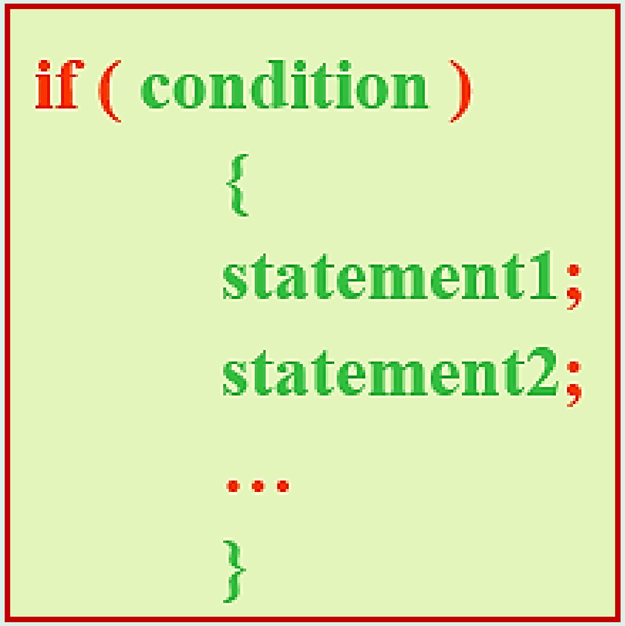
```
1+2<4    result is 1
3<2+5    result is 1
(3<2)+5  result is  5
```

3-2 Branching

- The conditional statements can be made using one of the three:
  - ***If* statement**
  - ***Switch…case* statement**
  - ***Conditional Operator* statement**

<u>3-2-1 *If* statement …</u>

- It is used for making decision.
- The general form of the **if** statement is :



```
if ( condition )
        {
        statement1;
        statement2;
        ...
        }
```

- Here if condition is logical **TRUE**, the statements inside the braces are executed.
- If condition is logical **FALSE**, then the statements are skipped.

3-2-1 *If* statement …

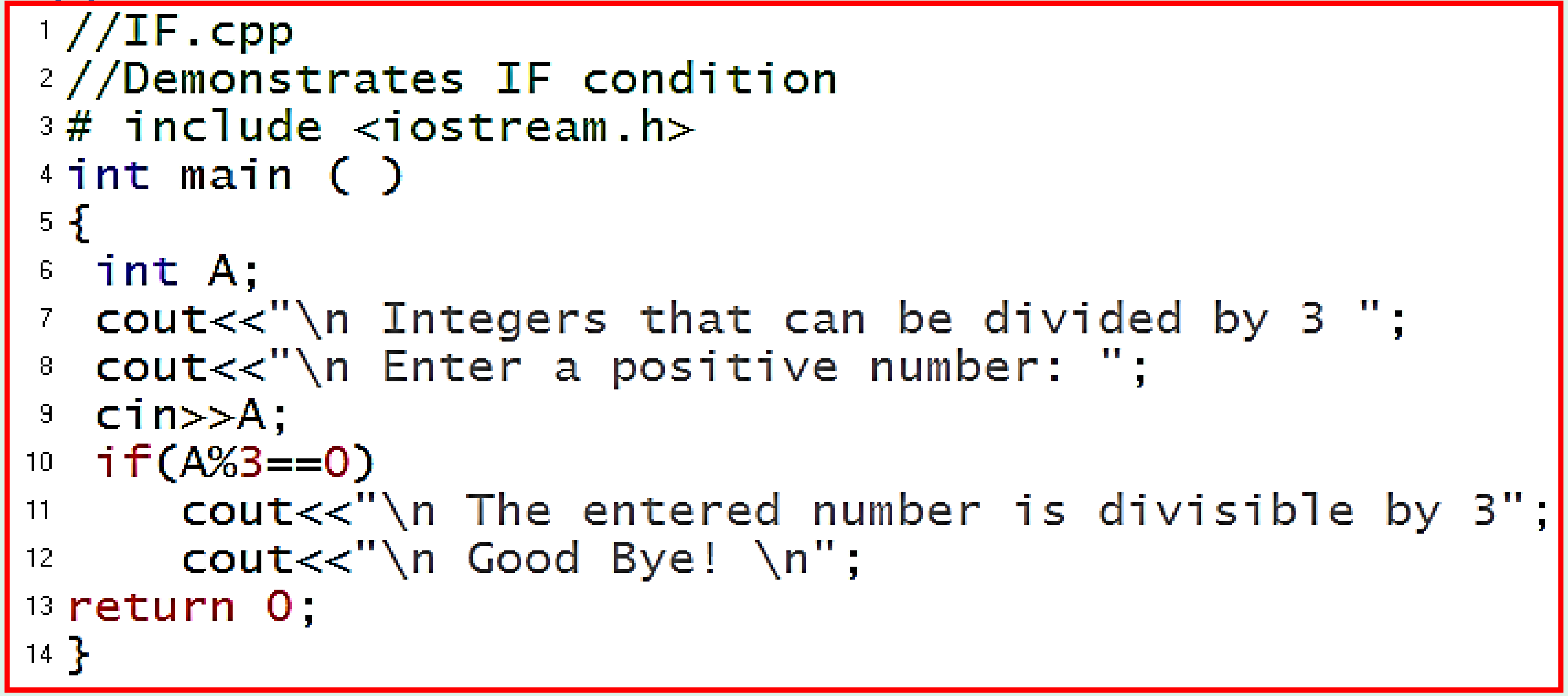
- The parentheses **( )**, however, must always be used to enclose the conditional expression.
- Note that: the braces **{ }** form a block of statements that is under the control of the **if** statement.
- *If there is **only one statement** inside the block, the **braces can be ignored**.*
- For example, the following expression

```
if (x > 0)
cout<<“The square root of x is= ”<< sqrt (x);
```

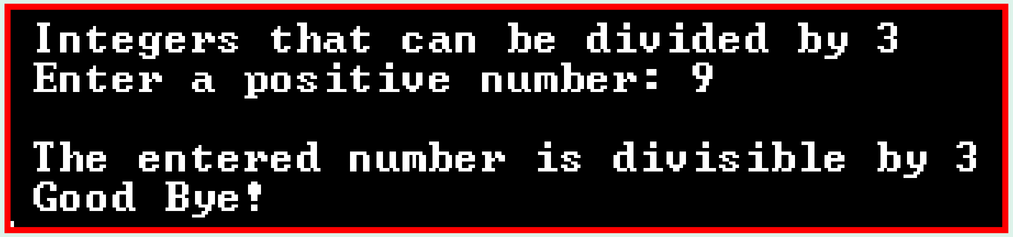
Tells the computer that if the value of x > zero, it should calculate the square root of x and then print the result. But, if the value of x ≤ zero, then execution ignores the statements inside *if* statement.

3-2-1 *If* statement …

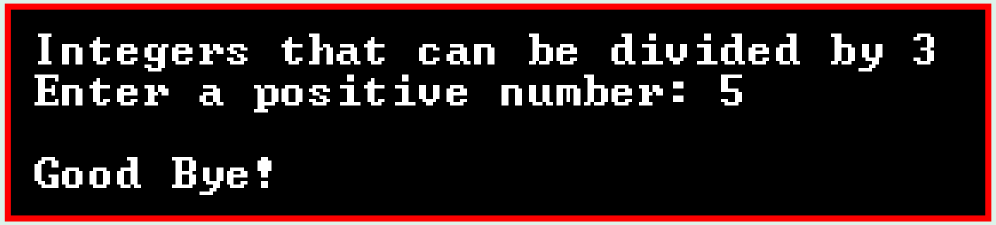
- IF.cpp



```cpp
//IF.cpp
//Demonstrates IF condition
# include <iostream.h>
int main ( )
{
 int A;
 cout<<"\n Integers that can be divided by 3 ";
 cout<<"\n Enter a positive number: ";
 cin>>A;
 if(A%3==0)
    cout<<"\n The entered number is divisible by 3";
    cout<<"\n Good Bye! \n";
return 0;
}
```



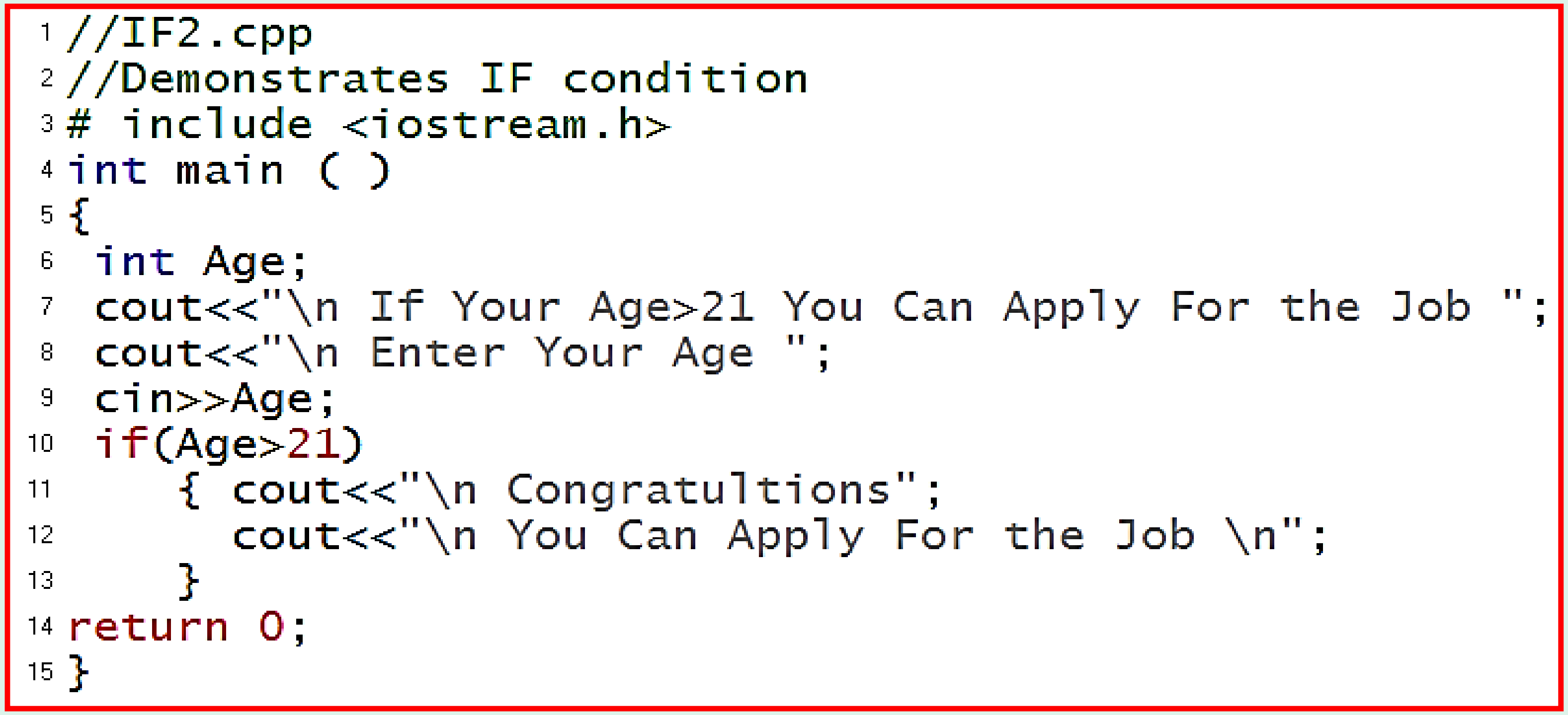
```
Integers that can be divided by 3
Enter a positive number: 9

The entered number is divisible by 3
Good Bye!
```



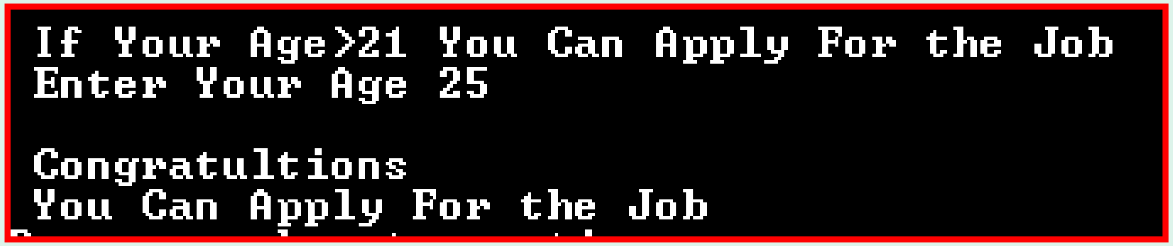
```
Integers that can be divided by 3
Enter a positive number: 5

Good Bye!
```

3-2-1 *If* statement …

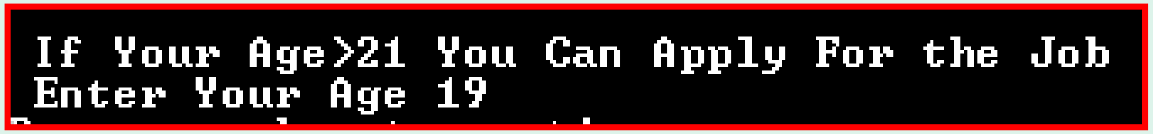
- IF2.cpp



```cpp
//IF2.cpp
//Demonstrates IF condition
# include <iostream.h>
int main ( )
{
 int Age;
 cout<<"\n If Your Age>21 You Can Apply For the Job ";
 cout<<"\n Enter Your Age ";
 cin>>Age;
 if(Age>21)
     { cout<<"\n Congratultions";
       cout<<"\n You Can Apply For the Job \n";
     }
return 0;
}
```



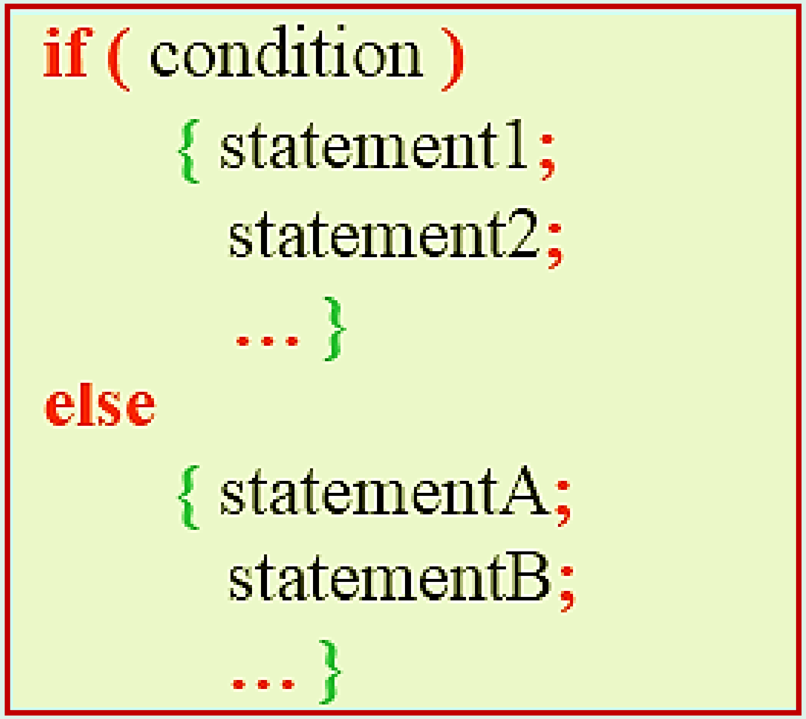
```
 If Your Age>21 You Can Apply For the Job
 Enter Your Age 25

 Congratultions
 You Can Apply For the Job
```



```
 If Your Age>21 You Can Apply For the Job
 Enter Your Age 19
```

3-2-1 *If…else* statement

- The general form of the ***if – else*** statement is :



```
if ( condition )
      { statement1;
        statement2;
        … }
else
      { statementA;
        statementB;
        … }
```

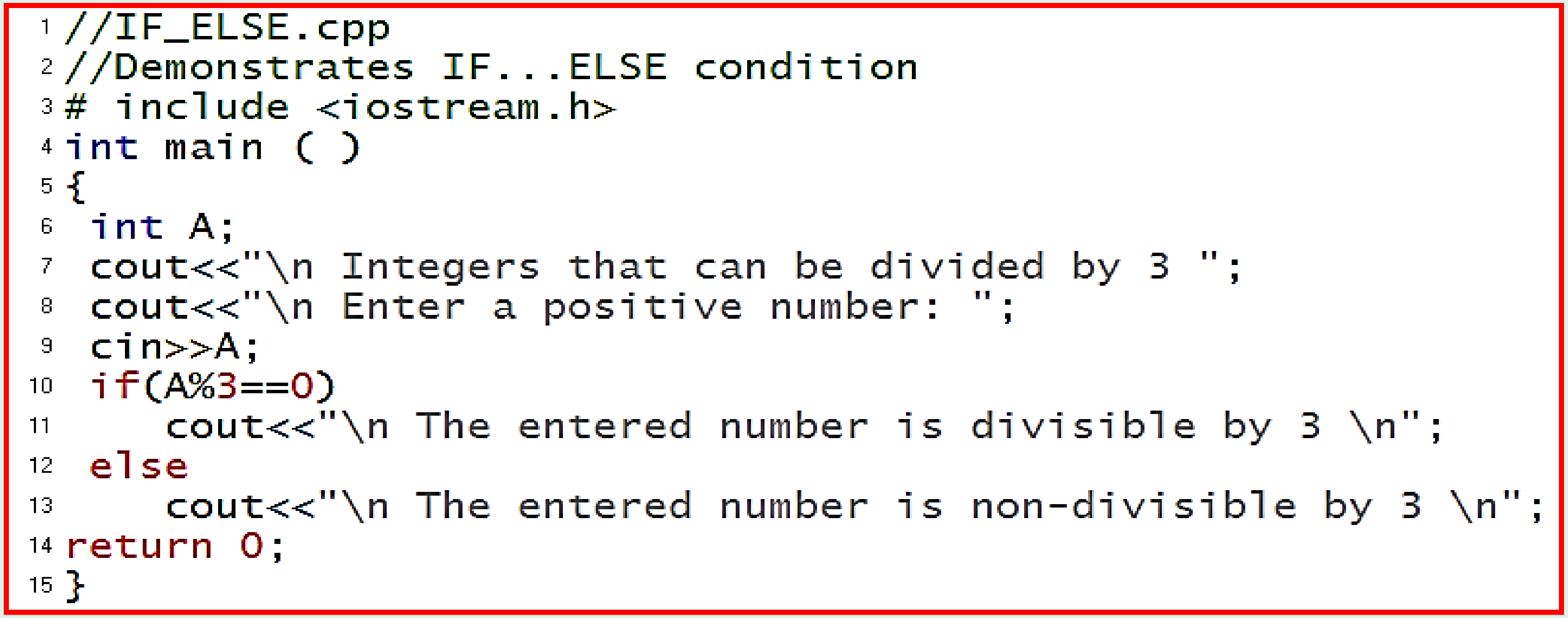
- if condition is logical **TRUE**,

  statement1, statement2, … are executed.
- if condition is logical **FALSE**,

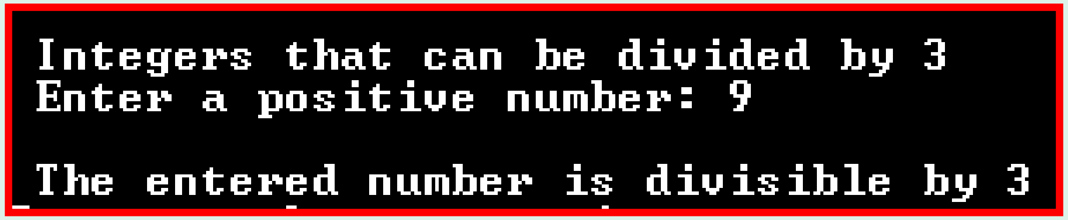
  statement_A, statement_B, … are executed.

3-2-1 *If…else* statement …

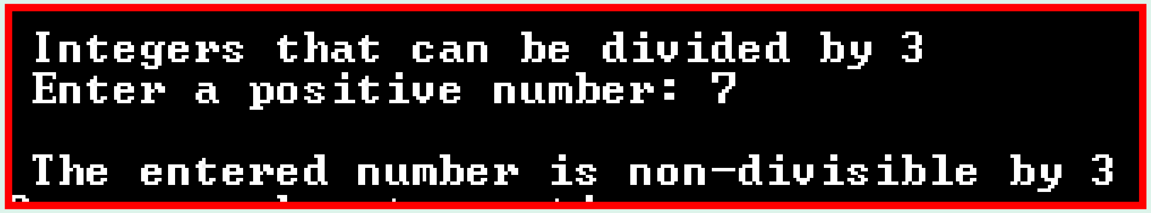
- IF_ELSE.cpp



```cpp
//IF_ELSE.cpp
//Demonstrates IF...ELSE condition
# include <iostream.h>
int main ( )
{
 int A;
 cout<<"\n Integers that can be divided by 3 ";
 cout<<"\n Enter a positive number: ";
 cin>>A;
 if(A%3==0)
      cout<<"\n The entered number is divisible by 3 \n";
 else
      cout<<"\n The entered number is non-divisible by 3 \n";
return 0;
}
```



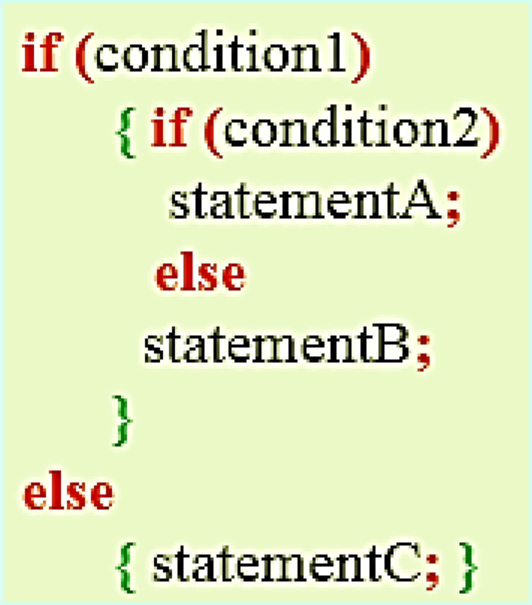
```
 Integers that can be divided by 3
 Enter a positive number: 9

 The entered number is divisible by 3
```



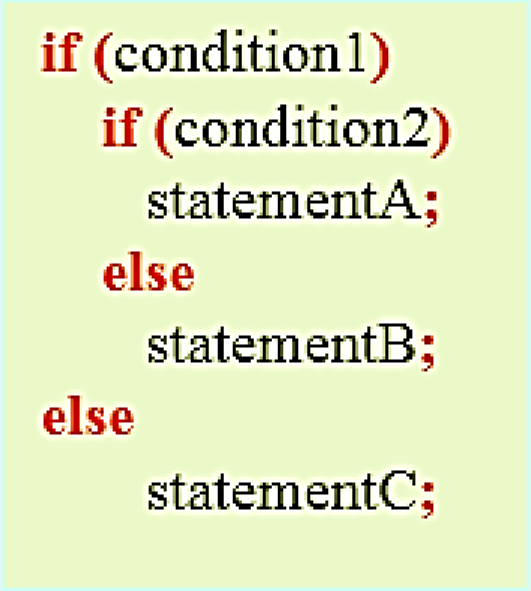
```
 Integers that can be divided by 3
 Enter a positive number: 7

 The entered number is non-divisible by 3
```

3-2-1 nested *If* statement

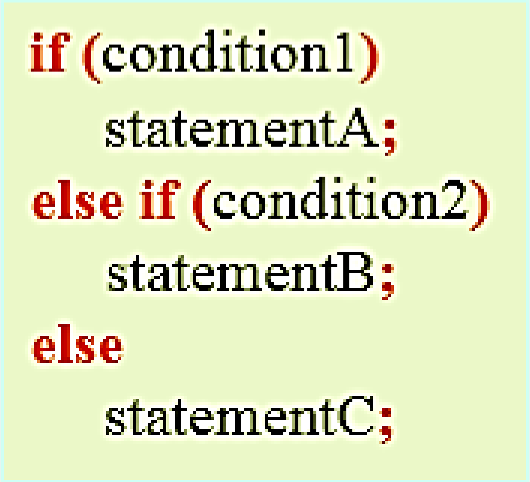
- When an *if* statement is used within *another* *if* statement, this is called nested statement.
- The general form of the nested ***if*** statement is:



```
if (condition1)
     { if (condition2)
        statementA;
       else
      statementB;
     }
else
     { statementC; }
```



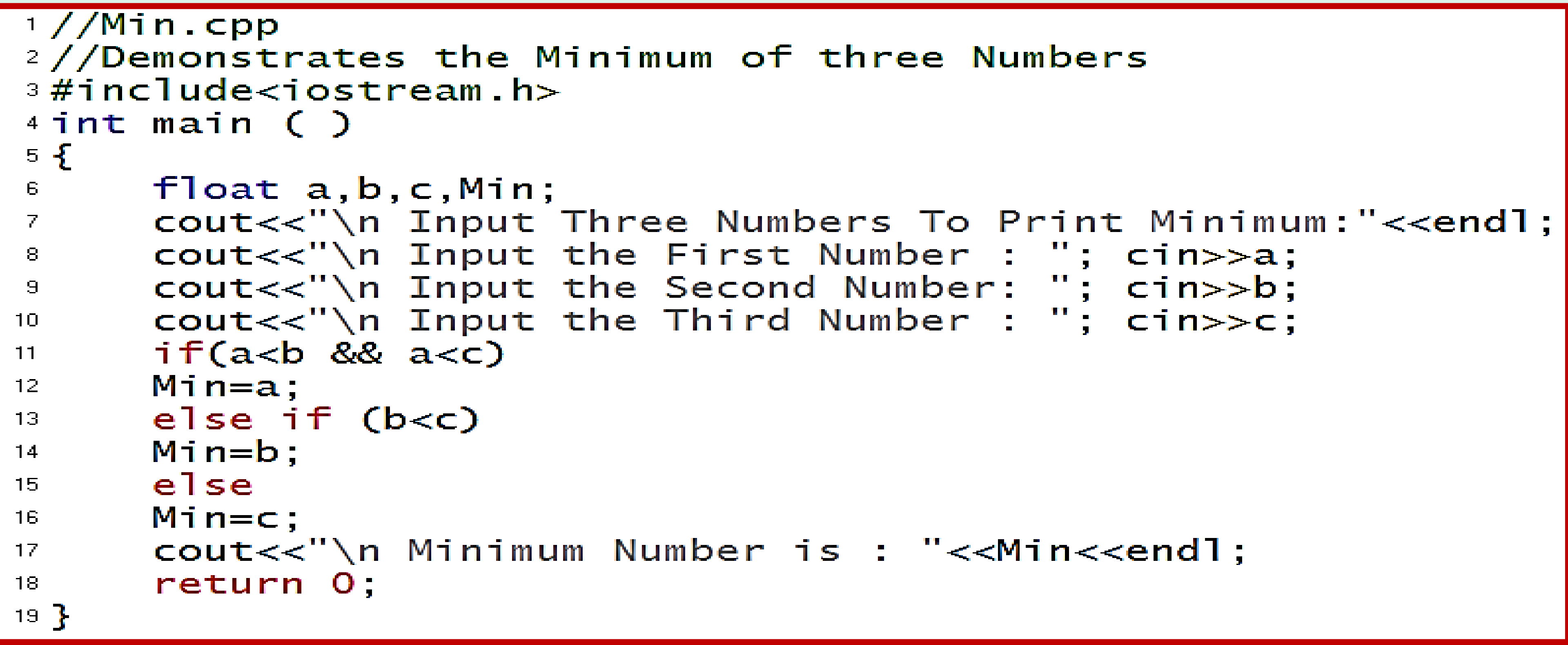
```
if (condition1)
   if (condition2)
     statementA;
   else
     statementB;
else
     statementC;
```



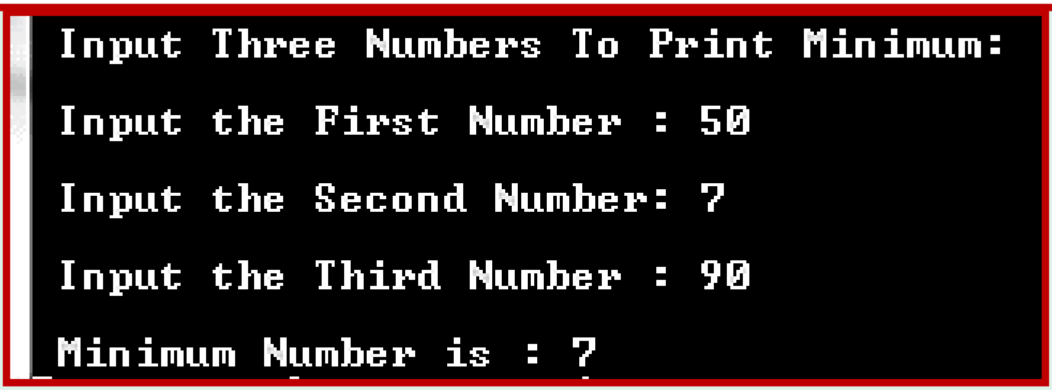
```
if (condition1)
     statementA;
else if (condition2)
     statementB;
else
     statementC;
```

3-2-1 nested *If* statement …

- Input three numbers and find the minimum of them?
- Min.cpp



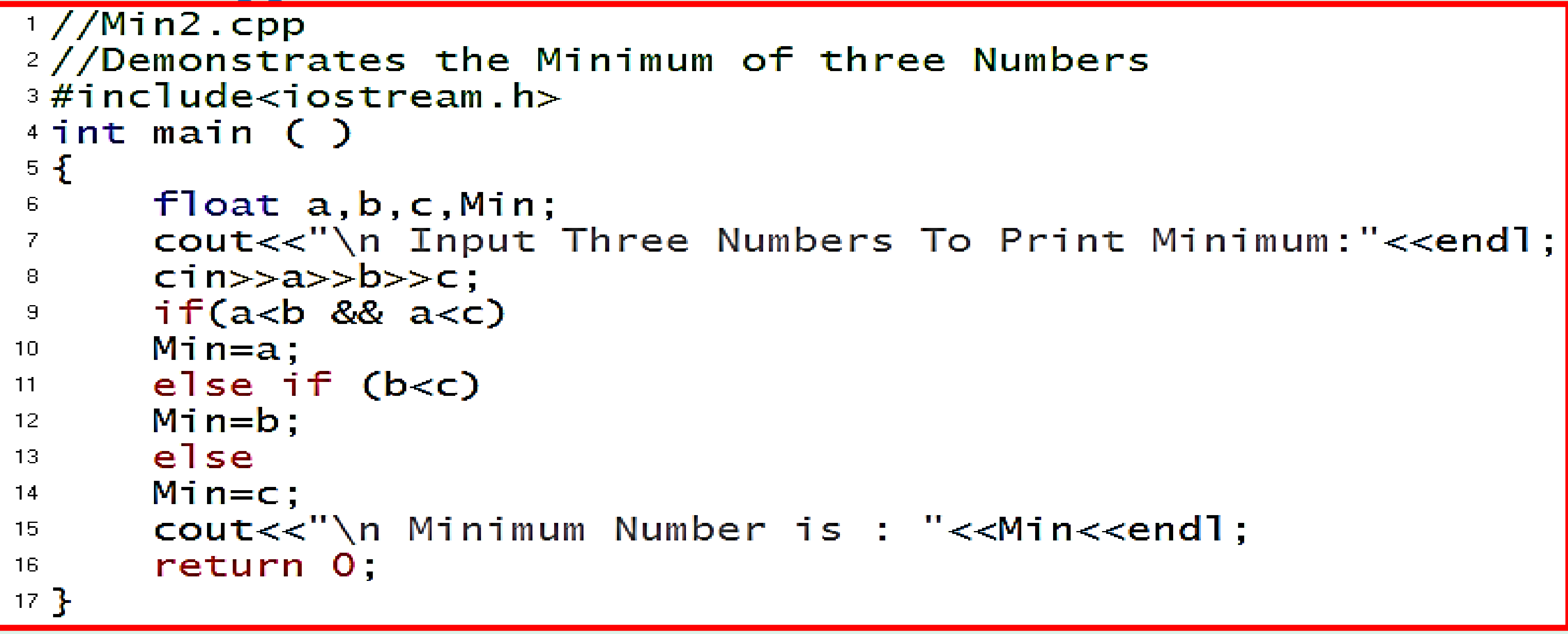
```cpp
//Min.cpp
//Demonstrates the Minimum of three Numbers
#include<iostream.h>
int main ( )
{
      float a,b,c,Min;
      cout<<"\n Input Three Numbers To Print Minimum:"<<endl;
      cout<<"\n Input the First Number : "; cin>>a;
      cout<<"\n Input the Second Number: "; cin>>b;
      cout<<"\n Input the Third Number : "; cin>>c;
      if(a<b && a<c)
      Min=a;
      else if (b<c)
      Min=b;
      else
      Min=c;
      cout<<"\n Minimum Number is : "<<Min<<endl;
      return 0;
}
```



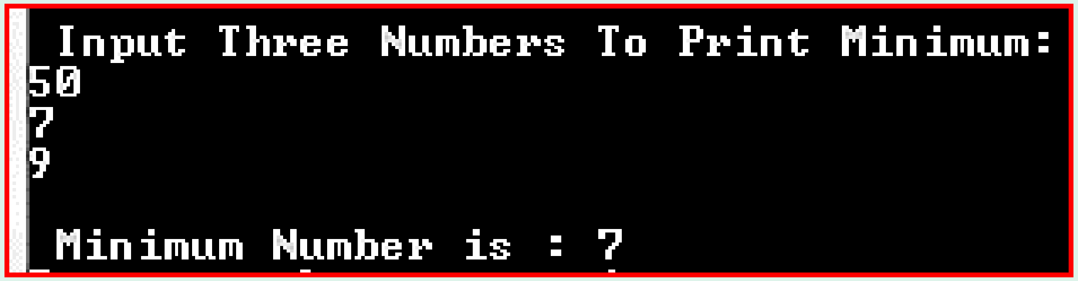
```
Input Three Numbers To Print Minimum:

Input the First Number : 50

Input the Second Number: 7

Input the Third Number : 90

Minimum Number is : 7
```

3-2-1 nested *If* statement …

- The previous example in another form.
- Min2.cpp



```cpp
//Min2.cpp
//Demonstrates the Minimum of three Numbers
#include<iostream.h>
int main ( )
{
	float a,b,c,Min;
	cout<<"\n Input Three Numbers To Print Minimum:"<<endl;
	cin>>a>>b>>c;
	if(a<b && a<c)
	Min=a;
	else if (b<c)
	Min=b;
	else
	Min=c;
	cout<<"\n Minimum Number is : "<<Min<<endl;
	return 0;
}
```

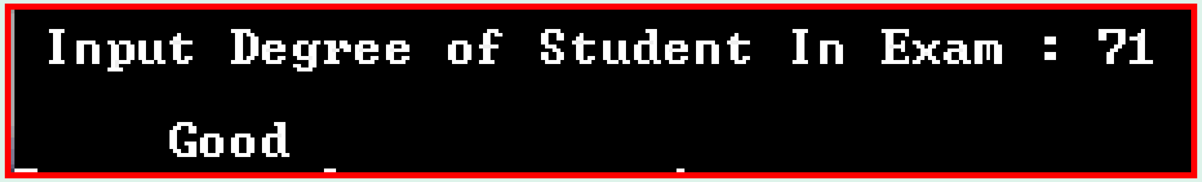


```
 Input Three Numbers To Print Minimum:
50
7
9

 Minimum Number is : 7
```

3-2-1 nested *If* statement …

- Input the degree of a student in one subject and print the grade of that subject (EX, VG, G, P, F)…where
- F:0→50, P:50→65, G:65→75, VG: 75→85, EX: 85→100
- Grade.cpp

```
//Grade.cpp
//Demonstrates the Grade of Student In Exam
#include<iostream.h>
int main ( )
{
    float Deg;
    cout<<"\n Input Degree of Student In Exam : ";
    cin>>Deg;
    if(Deg>=85 && Deg<=100)
    cout<<"\n   Excellent "<<endl;
    else if(Deg>=75 && Deg<85)
    cout<<"\n   Very Good  "<<endl;
    else if(Deg>=65 && Deg<75)
    cout<<"\n     Good   "<<endl;
    else if(Deg>=50 && Deg<65)
    cout<<"\n     Path   "<<endl;
    else
    cout<<"\n Fail! \t You Are Our Eyes Light! "<<endl;
    return 0;
}
```



```
 Input Degree of Student In Exam : 71

     Good
```

3-2-2 *switch…case* statement

- The ***nested if*** statement will become very complex if there are many decisions that need to be made.
- The ***switch…case*** statement, can be used to make unlimited decisions or choices. It allows you to branch on any of a number of different values.
- The general form of the ***switch…case*** statement is:

```
switch (Number)
{
    case value1:  statement1; break;
    case value2:  statement2; break;
    case value3:  statement3; break;
    case value4:  statement4; break;
    …
            default: statement-default; break;
}
```

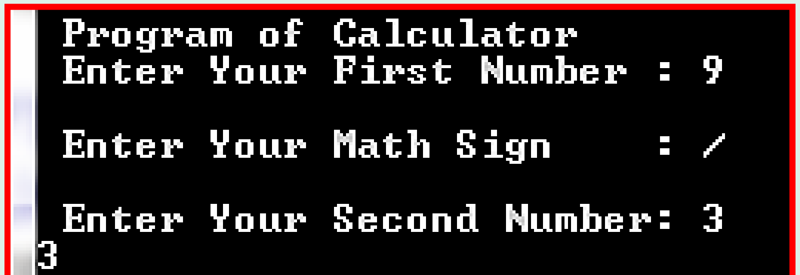
3-2-2 *switch…case* statement …

- If the value of **Number** is the same as the value of **value1**, the statement **statement1** is executed.
- If the value of **Number** is the same as the value of **value2**, the statement **statement2** is executed. and soon…
- However, the value of **Number** is not equal to any values, the statement (**statement-default**) is executed.
- You must use **case** keyword to label each case.
- The **default** keyword is recommended to be used for the default case.
- *Note that:* no constant expressions are identical in the switch statement.
- The **break** statement: if you don't use it, the program will read all **cases**. But if you use it, the program do execute the right **case** and then jump out of the ***switch…case*** body.

3-2-2 *switch…case* statement …

- Write a program to make the same job of the calculator?
- Calculator.cpp

```
//Calculator.cpp
#include <iostream.h>
int main ( )
{
    float X,Y;
    char sign;
    cout<<"\n Program of Calculator";
    cout<<"\n Enter Your First Number  : "; cin>>X;
    cout<<"\n Enter Your Math Sign     : "; cin>>sign;
    cout<<"\n Enter Your Second Number: "; cin>>Y;
    switch (sign)
    {
        case '+': cout<<X+Y<<endl; break;
        case '-': cout<<X-Y<<endl; break;
        case '*': cout<<X*Y<<endl; break;
        case '/': cout<<X/Y<<endl; break;
        default : cout<<" \n Invalid Operation"<<endl;
    }
    return 0;
}
```



```
 Program of Calculator
 Enter Your First Number : 9

 Enter Your Math Sign    : /

 Enter Your Second Number: 3
3
```

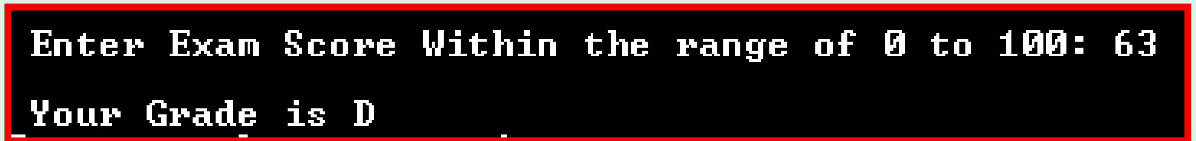
3-2-2 *switch…case* statement …

- Write a program to print out the grade of student exam?

Grade A	Grade B	Grade C	Grade D	Grade P	Grade F
90:100	80:90	70:80	60:70	50:60	0:50

- Exam.cpp

```cpp
//Exam.cpp
//Demonstrates Grade of Student Exam
# include <iostream.h>
int main ( )
{
int score;
cout<<"\n Enter Exam Score Within the range of 0 to 100: ";
cin>>score;
    switch (score/10)
    {
        case 10 : cout<<"\n Your Grade is A "<<endl; break;
        case 9  : cout<<"\n Your Grade is A "<<endl; break;
        case 8  : cout<<"\n Your Grade is B "<<endl; break;
        case 7  : cout<<"\n Your Grade is C "<<endl; break;
        case 6  : cout<<"\n Your Grade is D "<<endl; break;
        case 5  : cout<<"\n Your Grade is P "<<endl; break;
        case 4  : cout<<"\n Your Grade is F "<<endl; break;
        case 3  : cout<<"\n Your Grade is F "<<endl; break;
        case 2  : cout<<"\n Your Grade is F "<<endl; break;
        case 1  : cout<<"\n Your Grade is F "<<endl; break;
        case 0  : cout<<"\n Your Grade is F "<<endl; break;
        default : cout<<" \n Invalid Input Score"<<endl;
    }
    return 0;
}
```



```
Enter Exam Score Within the range of 0 to 100: 63

Your Grade is D
```

3-2-3 *Conditional Operator* statement

- Using of *conditional (Ternary)* operator is similar to usage of *if…else* statement. Look at the two cases in the table.

Conditional (Ternary) Operator لو الاجابة True يبقي مابعد علامة الإستفام يكون الخرج. ولو كانت الإجابة False يكون مابعد العلامة (:) هو الخرج.		*if…else* statement لو الاجابة True يكون مابعد *if* مباشرة يكون الخرج. ولو كانت الإجابة False يكون مابعد *else* مباشرة يكون الخرج.

(Min = (a<b)) ? a : b;	➡	if (a<b)
		    Min = a;
		else
		    Min = b;

(Max = (a<b)) ? b : a;	➡	if (a<b)
		    Max = b;
		else
		    Min = a;

3-3 Assignment (3)