Control Structures

Week 4

Conditional Operators

Syntax:

exp1 ? exp2 : exp3

Where exp1,exp2 and exp3 are expressions

Working of the ? Operator:

Exp1 is evaluated first, if it is nonzero(1/true) then the expression2 is evaluated and this becomes the value of the expression,

If exp1 is false(0/zero) exp3 is evaluated and its value becomes the value of the expression

Ex: m=2;

n=3

r=(m>n) ? m : n;

Increment & Decrement Operators

C++ supports 2 useful operators namely

1. Increment ++
2. Decrement- -operators

The ++ operator adds a value 1 to the operand

The -- operator subtracts 1 from the operand

++a or a++

--a or a--

Rules for ++ & -- Operators

- These require variables as their operands.
- When postfix either ++ or -- is used with the variable in a given expression, the expression is evaluated first and then it is incremented or decremented by one.
- When prefix either ++ or – is used with the variable in a given expression, it is incremented or decremented by .one first and then the expression is evaluated with the new value

Examples for ++ and -- Operators

Let the value of a =5 and b=++a then

a = b =6

Let the value of a = 5 and b=a++ then

a =6 but b=5

i.e.:

1. a prefix operator first adds 1 to the operand and then the result is assigned to the variable on the left
2. a postfix operator first assigns the value to the variable on left and then increments the operand.

Shorthand Assignment Operators

Simple Assignment operator	Shorthand Operator
a = a+1	a + =1
a = a-1	a - =1
a = a* (m+n)	a * = m + n
a = a / (m+n)	a / = m + n
a = a %b	a %=b

Objectives

- To use the selection structure to choose among the alternative actions
  - **if** selection statement
  - Double Selection (**if else**) statements
  - Nested Selection statements

Control structures

- Control structure
  - A control statement and the statements whose execution it controls.
  - a control statement is any statement that alters the linear flow of control of a program. C++ examples include: if-else, while, for, break, continue and return statement
- Sequential execution
  - Statements executed in order
- Transfer of control
  - Next statement to be executed may be other than the next one in sequence.

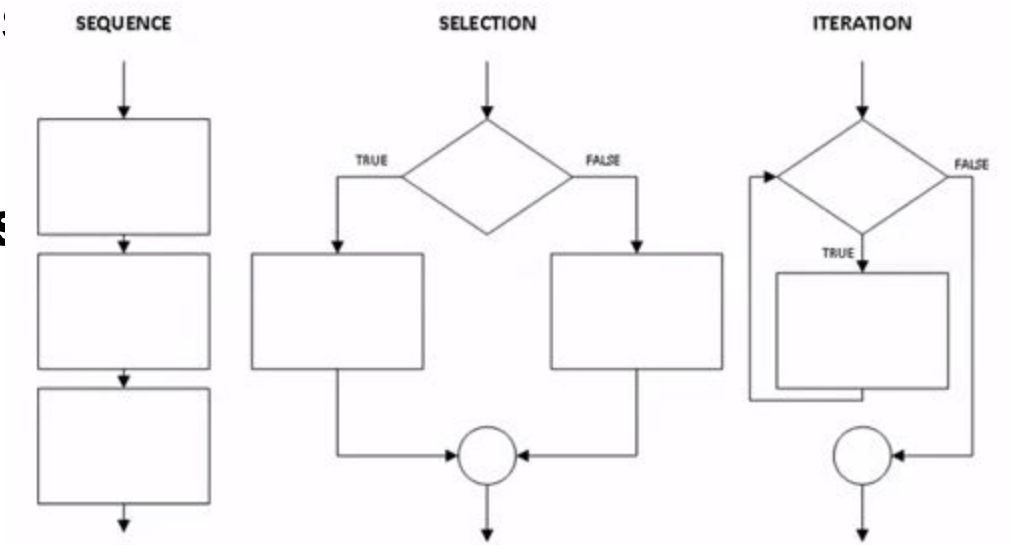
Control structures

Bohm and Jacopini's control structures

Bohm and Jacopino's work demonstrate that all programs could be written in terms of only three control structures.

Sequence structures, selection structures and repetition structures.

- **Sequence**
  - Built into C++.
- **Selection s**
  - C++ has three
- **Repetition**
  - C++ has three



Control structures

if Selection Structure

1. Perform an action if condition is true.
2. .Skip the action if condition is false.

If/else Selection Structure

1. Perform an action if condition is true.
2. Performs a different action if the condition is false.

We are going to discuss if and if/else selection structures and switch first and will explain the repetition structures later. It was just an introduction to these structures.

Decision making structure/ selection structure

- **Selection structure/decision structure**

allows the program to make a decision or comparison and then select one of two paths, depending on the result of the comparison.

- **Condition**

condition is a logical expression that evaluates to true or false. It could be a relational or Boolean expression. In other words

- Specifies the decision you are making
- Must result in either a true or false answer

IF Structure

- One statement or a block of statement enclosed in braces {} **i.e.** (**compound statement**), to be processed If condition met otherwise it is skipped.
- Uses equality and relational operators

Equality and Relational operators

- Conditions in **if** structures can be formed by using the *equality* and *relational* operators

- **Equality operators**
  - Same level of precedence
  - Associated left to right.
- **Relational operators**
  - Same level of precedence.
  - Associated left to right.

Equality operators precedence is lower then precedence of relational operators.

Equality and relational operators

Standard algebraic equality operator or relational operator	C++ equality or relational operator	Example of C++ condition	Meaning of C++ condition
Relational operators			
>	>	`x > y`	`x` is greater than `y`
<	<	`x < y`	`x` is less than `y`
≥	>=	`x >= y`	`x` is greater than or equal to `y`
≤	<=	`x <= y`	`x` is less than or equal to `y`
Equality operators			
=	==	`x == y`	`x` is equal to `y`
≠	!=	`x != y`	`x` is not equal to `y`

Boolean Expressions

- Boolean expressions are expressions that are either true or false
- **comparison operators** such as '>' (greater than) are used to compare variables and/or numbers
  - **(grade >= 60)** Including the parentheses, is the boolean expression from the grade example
  - A few of the comparison operators that use two symbols (No spaces allowed between the symbols!)
    - **>** greater than
    - **!=** not equal or inequality
    - **==** equal or equivalent
    - <= less than or equal to

    etc

If Selection Structure

- The primary C++ selection structure statement used to perform a single-alternative selection.
- Choose among alternative courses of action

- **If the condition is true**
  - statement Print statement executed, program continues to next
- **If the condition is false**
  - Print statement ignored, program continues

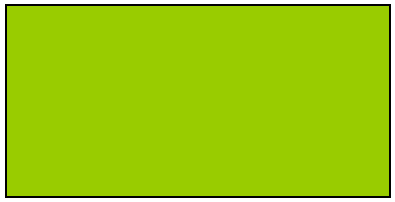
Flow chart

- Graphical representation of an algorithm or a portion of algorithm.
- Drawn using certain special-purpose symbols connected by arrows called flow lines.

- Special purpose symbols.

Following are some symbols to draw a flow chart and there purpose.

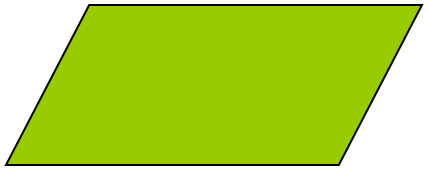
Ovals or rounded rectangles, indicate start or end of the program usually containing the word "Start, begin" or "End".



Rectangles are action symbols to indicate any type of ACTION, including a calculation or an input output operation.

Diamonds are conditional or decision symbols. It indicates a decision to be made.

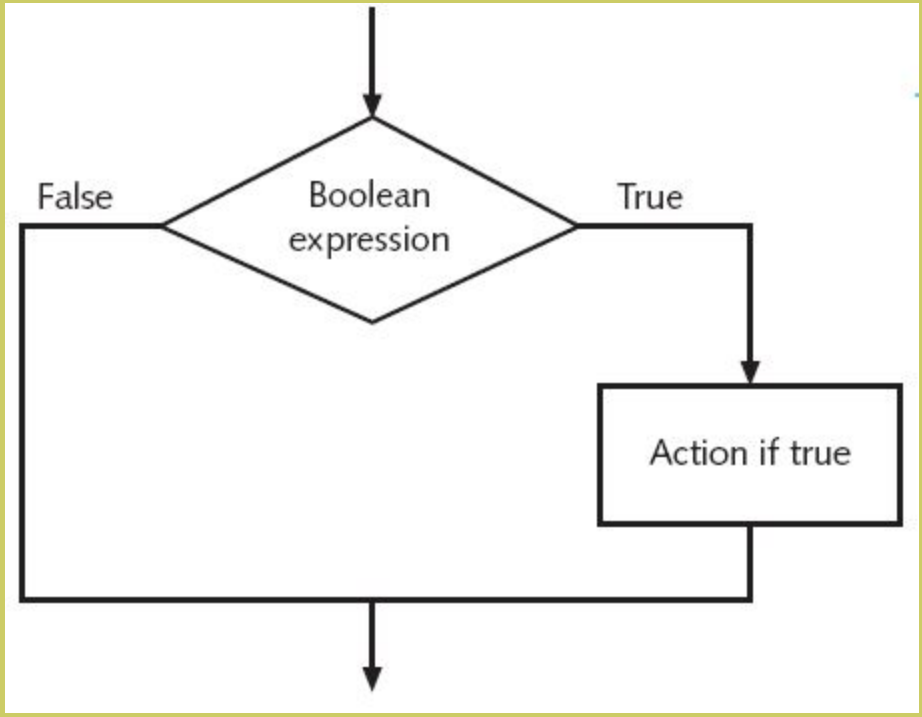
Flow chart



Parallelogram are the input/output symbols

Arrows, showing what's called "flow of control". An arrow coming from one symbol and ending at another symbol represents that control passes to the symbol the arrow points to. Arrows are also called flow lines.

If selection structure flow chart



If selection structure

- example:

Pseudocode

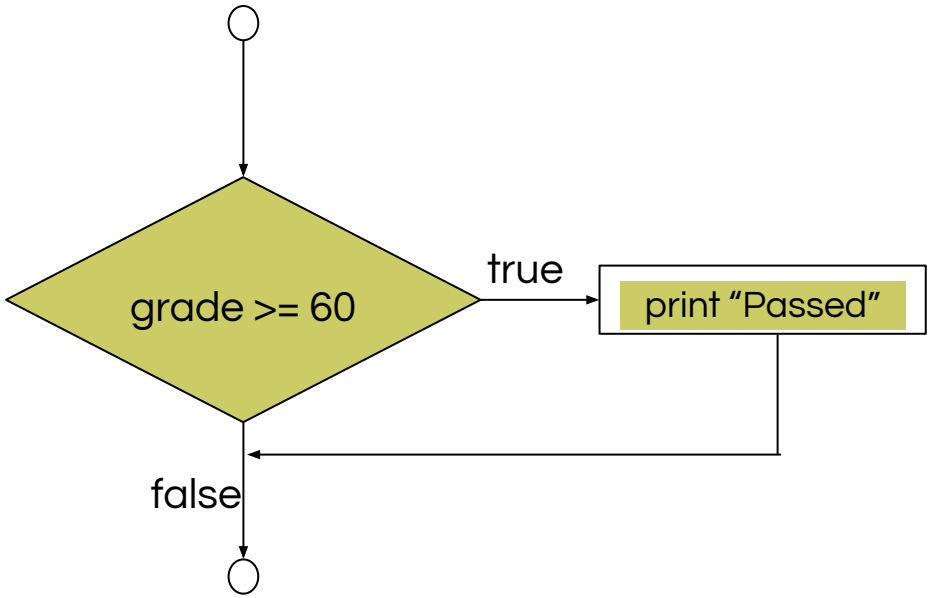
If student's grade is greater than or equal to 60
Print "Passed"

C++ code

```
if ( grade >= 60 )
   cout << "Passed";
```

If selection structure

- Flow chart for the pseudocode statement.



A decision can be made on any expression.

Example

1. Example is using the relational and equality operators.
2. Following example used six if statements to compare two numbers input by the user.
3. If the condition in any of these if statements is satisfied, the output statement associated with if is executed.
4. If the condition with the if statement is false the output statement associated with that if statement is skipped.
5. Observe the different outputs for various inputs at the end of the program.

```
1    // example
2    // Using if statements, relational
3    // operators, and equality operators
4    #include <iostream.h>
5
6
7
8
9
10   int main()
11   {
12      int num1, num2;  // declare variables
13
14      cout << "Enter two inte
15           << "the relationsh
16      cin >> num1 >> num2;
17
18      if ( num1 == num2 )
19         cout << num1 << " is equ
20
21      if ( num1 != num2 )
22         cout << num1 << " is not e
23
24      if ( num1 < num2 )
25         cout << num1 << " is less
26
27      if ( num1 > num2 )
28         cout << num1 << " is
29
30      if ( num1 <= num2 )
31         cout << num1 << " is less than or equal to "
32              << num2 << endl;    //continued on next slide
33
```

if structure compares values of **num1** and **num2** to test for equ

If condition is true (i.e., values are equal), execute this

if structure compar of **num1** and **num2** inequality.

If condition is true (i.e., values are not equal), execute this

If structure c values of nu

If condition is true (i.e., num1 is less than num2), execute

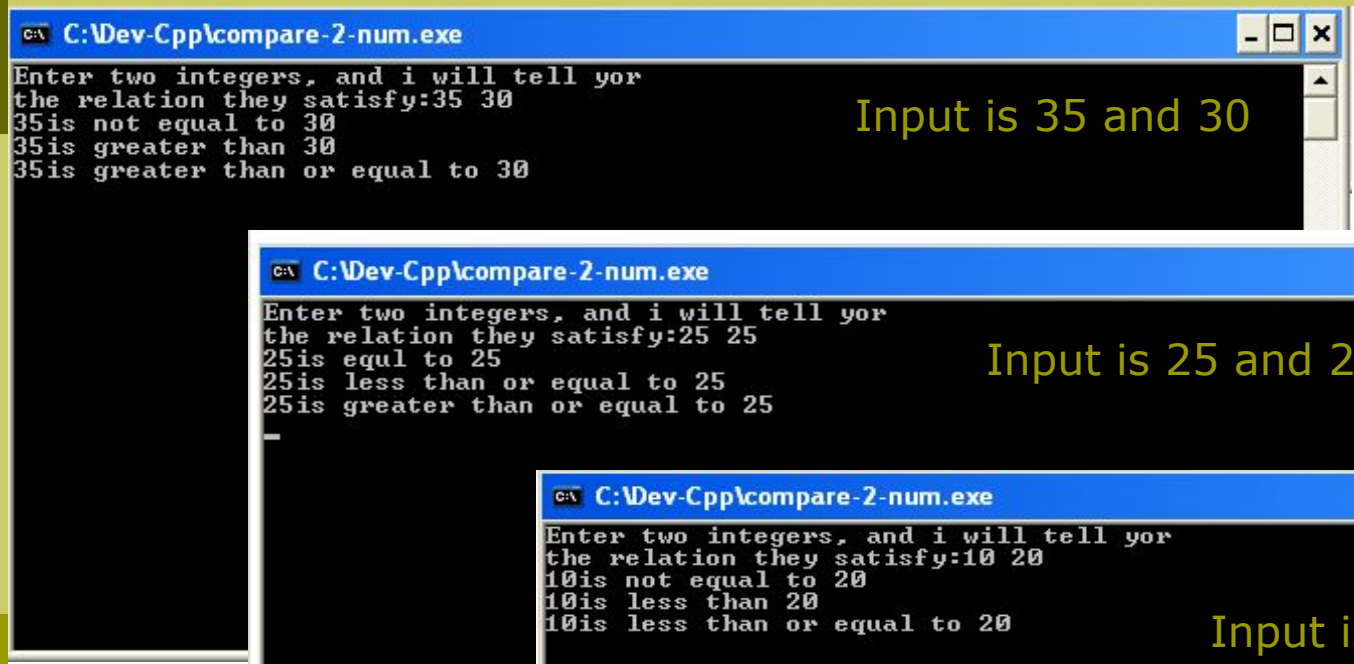
if structure of **num1** and

If condition is true (i.e., num1 is greater than num2),

if structure compares values of **num1** and **num2** to test num1 is less than or equal num2

If condition is true (i.e., num1 is less than or equal to num2), execute this statement.

```
34      if ( num1 >= num2 )
35         cout << num1 << " is greater than or equal to "
36              << num2 << endl;
37
38      return 0;   // indicate that program ended successfully
39   }
```



The if-else statement

- The if statement by itself will execute a single statement, or a group of statements , when conditions following if is true.
- It does nothing when the conditions is false.
- Can we execute one group of statements if the condition is true and another group of statements if the condition is false?
- Of course this is what is the purpose of else statement, which is demonstrated in the following example:

Ques: **In a company an employee is paid as under:**

If his basic salary is less than $1500, then HRA = 10% of basic salary and DA=90% of basic. If his salary is either equal to or above $1500, then HRA = $500 and DA=98% of basic salary. If the employee's salary is input through keyboard write a program to find his gross salary.

If/else selection structure

- *Different actions if conditions true or false*

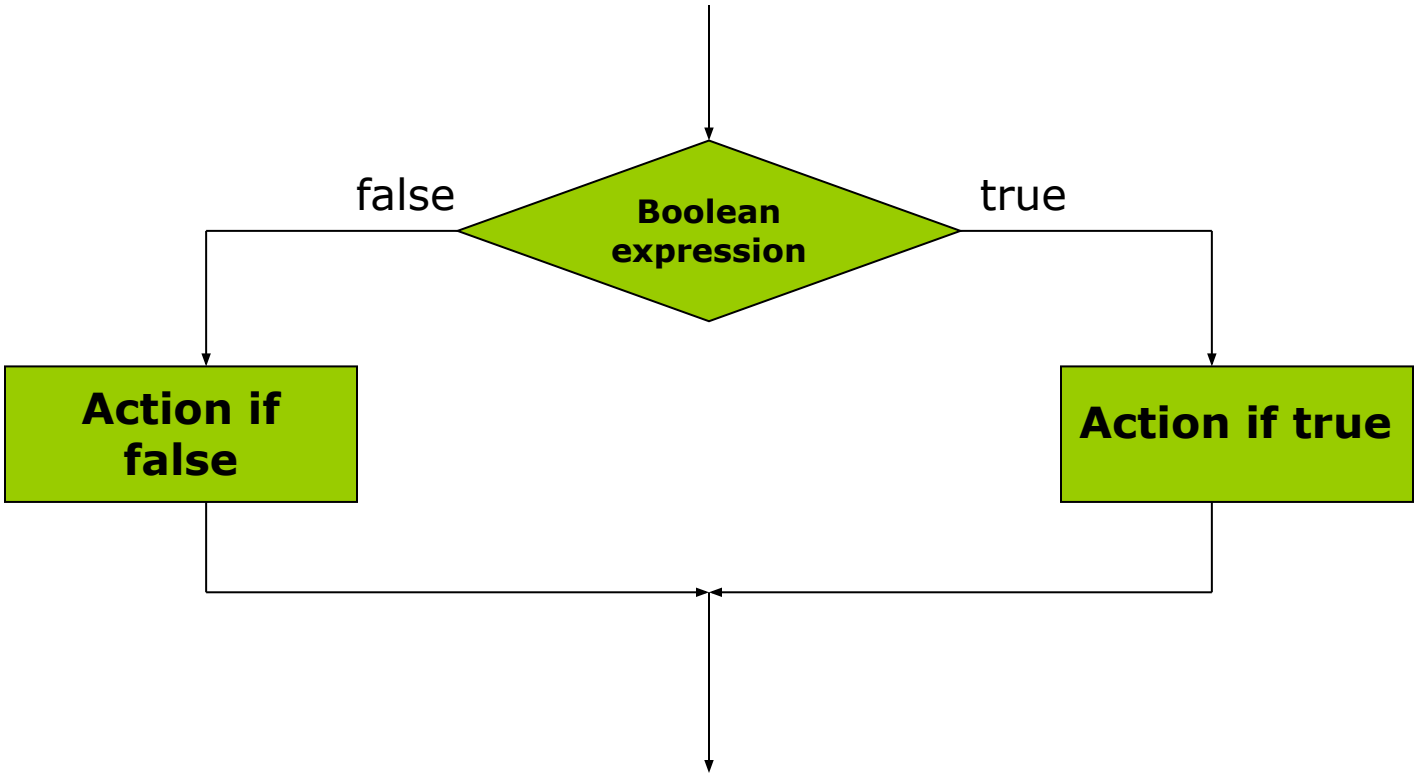
- Syntax
- **A single statement for each alternative**

```
if  (Boolean_expression)
    yes_statement;
else
    no_statement;
```

- **A sequence statement for each alternative**

```
if  (Boolean_expression)
    {   yes_statement1;
        yes_statement2;
        yes_statement last; }
else
    {   no_statement1;
        no_statement2;
        no_statement last; }
```

If/else selection structure flow chart



If/else selection structure

Example

- Pseudocode

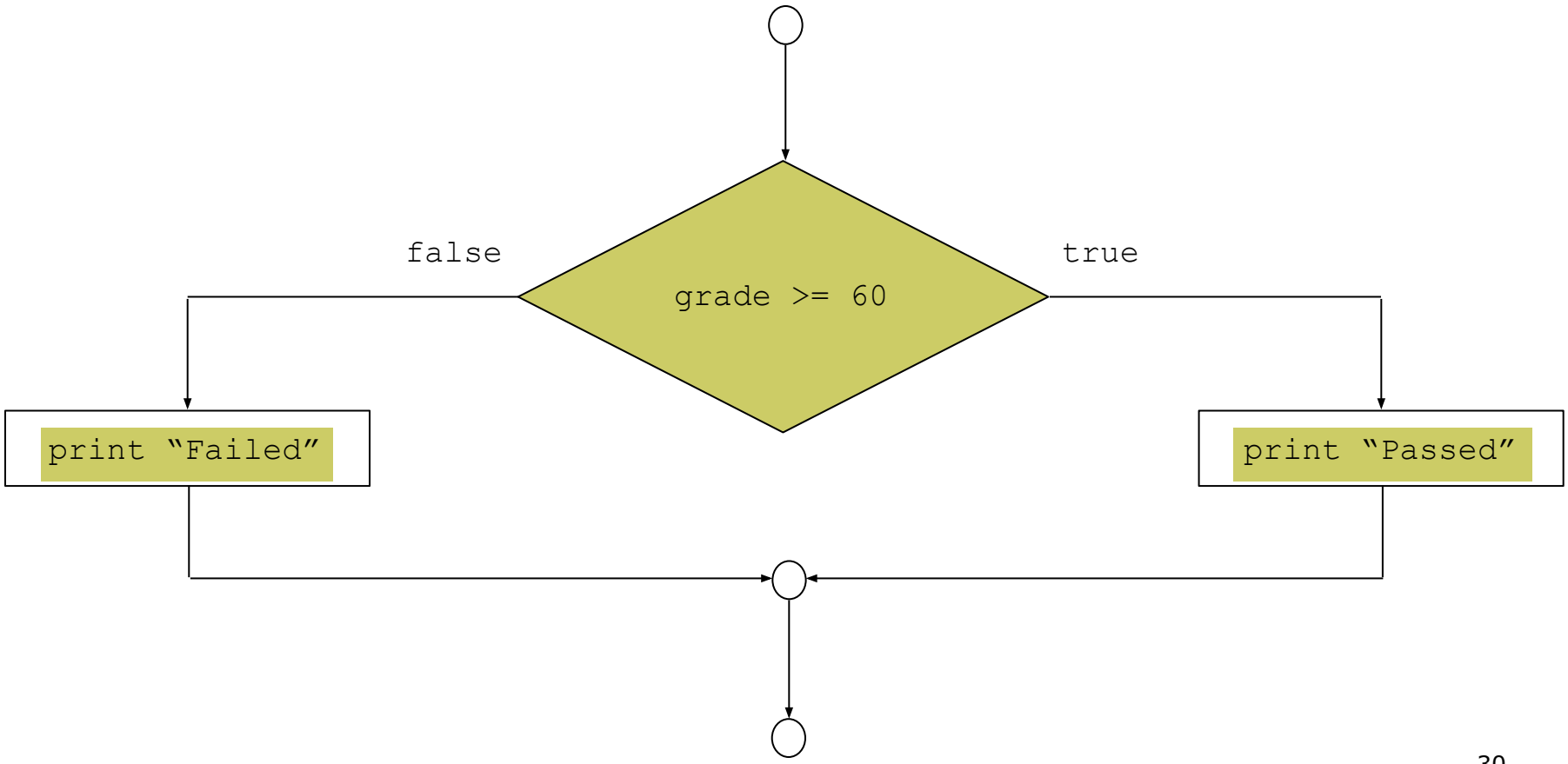
  if student's grade is greater than or equal to 60
  print "Passed"
  else
  print "Failed"

- C++ code

```
if ( grade >= 60 )
   cout << "Passed";
else
   cout << "Failed";
```

If/else selection structure

Flow chart for the pseudocode



Compound Statement

- Compound statement
  - Set of statements within a pair of braces also known as **BLOCK**

```
if ( grade >= 60 )
   cout << "Passed.\n";
else {
   cout << "Failed.\n";
   cout << "You must take this course again.\n";
}
```

  - Without braces,

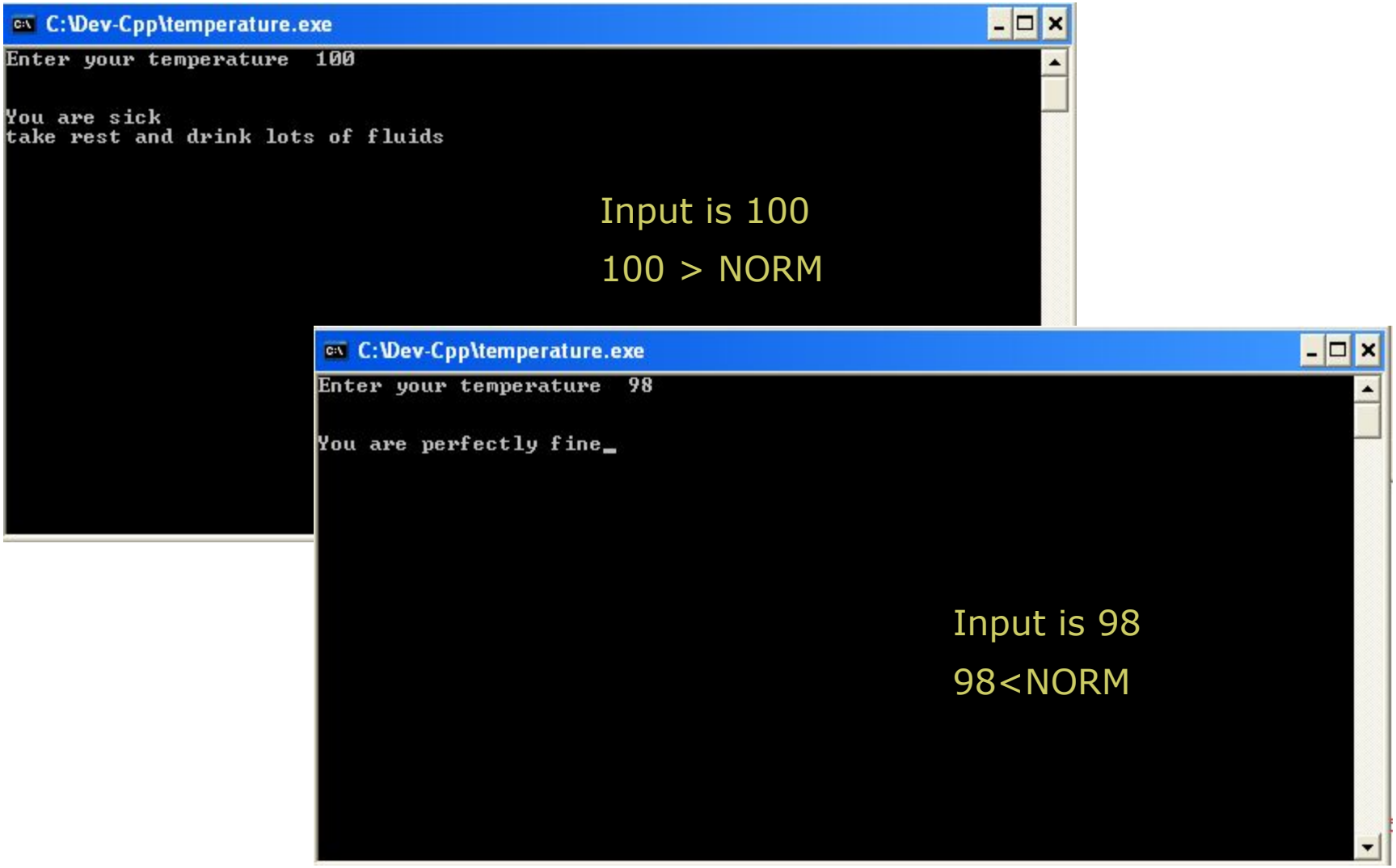
```
cout << "You must take this course again.\n";
```

always executed

Another simple Example

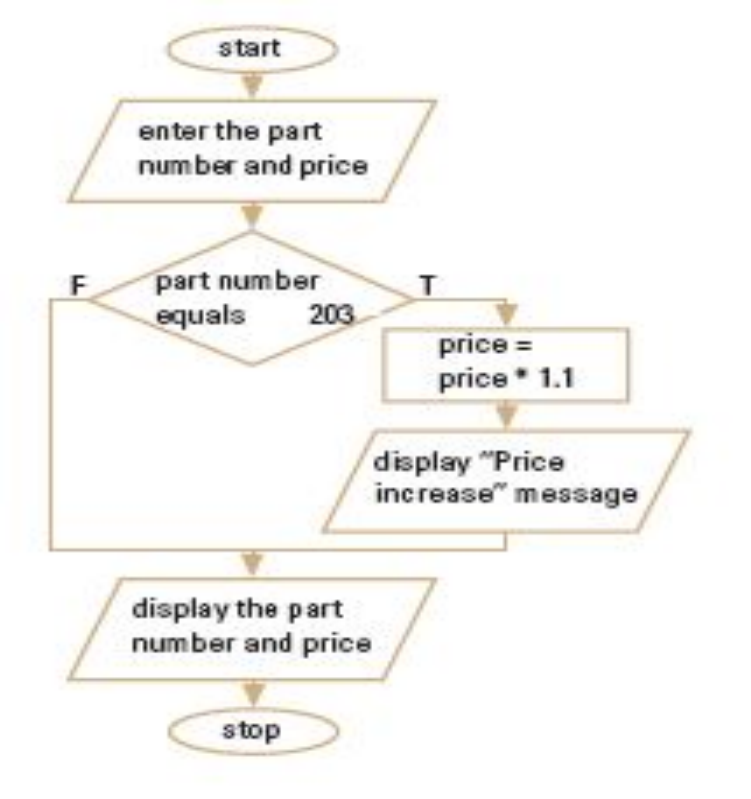
```
1.  //program to determine if user is ill

2.  #include <iostream>
3.  using namespace std;
4.  int main()
5.  {
6.     const double NORM = 98.6; // degree Fahranheit;
7.     double temperature;
8.     cout<<"Enter your temperature\t";
9.     cin>>temperature;

10.    if (temperature>NORM) //if temperature is greater than NORM print following statement
11.       cout<<"\n\nYou are sick\n"<<"take rest and drink lots of fluids";
12.    else              //if temperature is <= NORM print following statement
13.        cout<<"\n\nYou are perfectly fine";

14.    return 0;
15. }
```

output

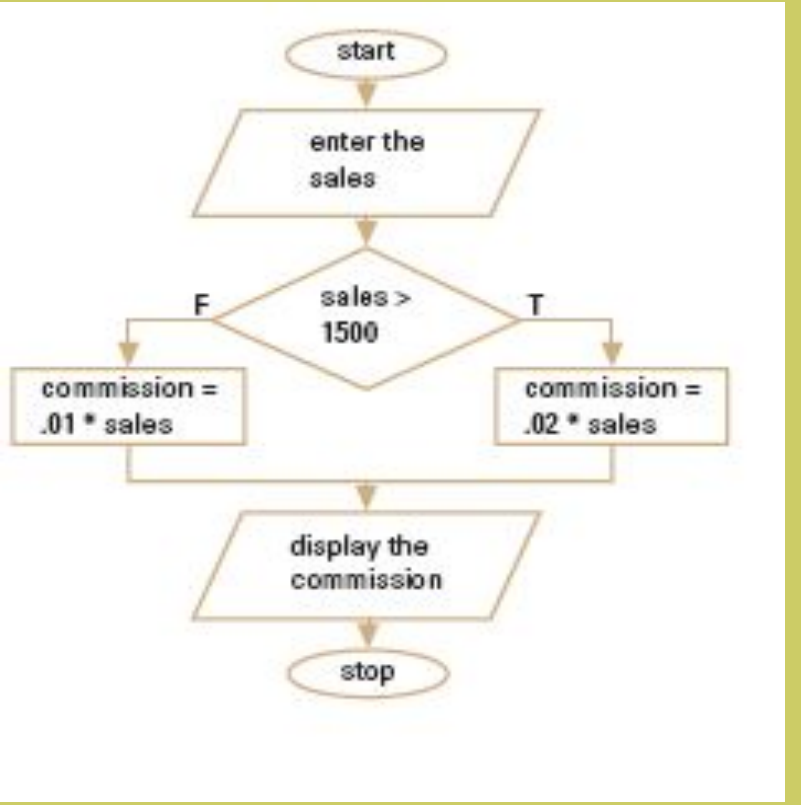

C:\Dev-Cpp\temperature.exe
Enter your temperature 100
You are sick
take rest and drink lots of fluids
Input is 100
100 > NORM
C:\Dev-Cpp\temperature.exe
Enter your temperature 98
You are perfectly fine
Input is 98
98<NORM

Example of if and if/else selection structure drawn in flow chart form

If selection structure(program 1)



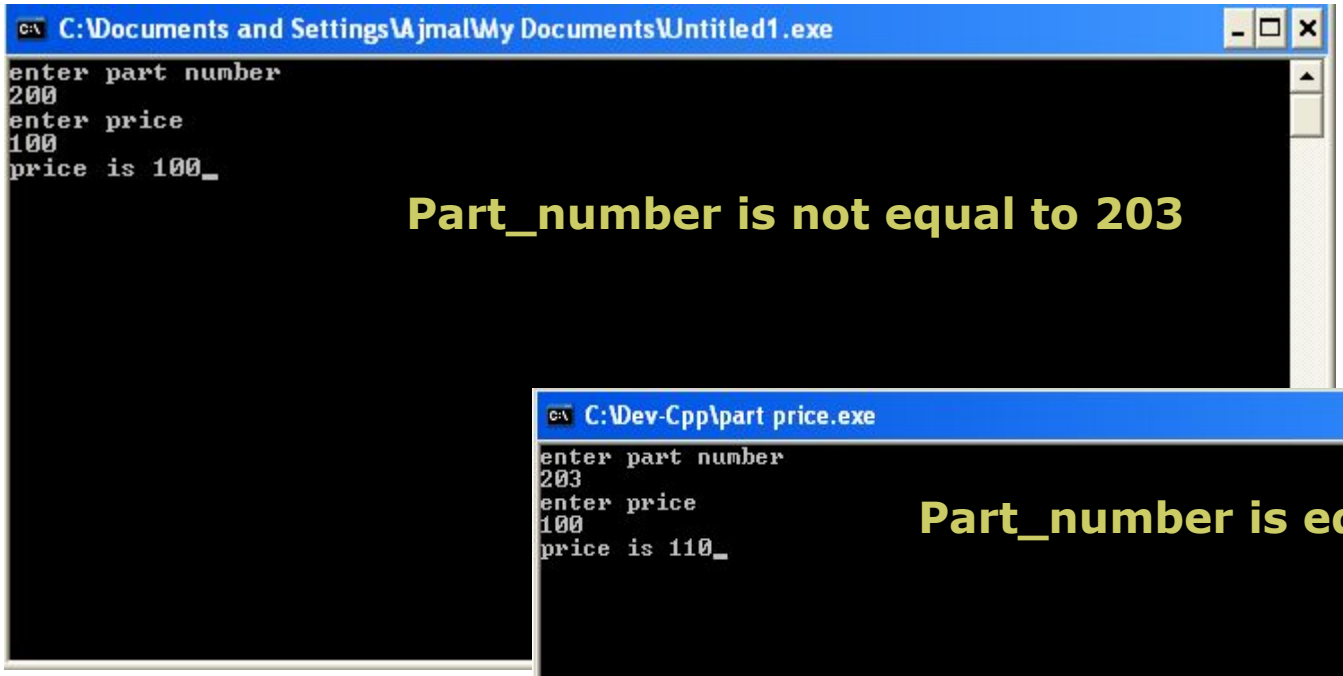
If/else selection structure(program2)

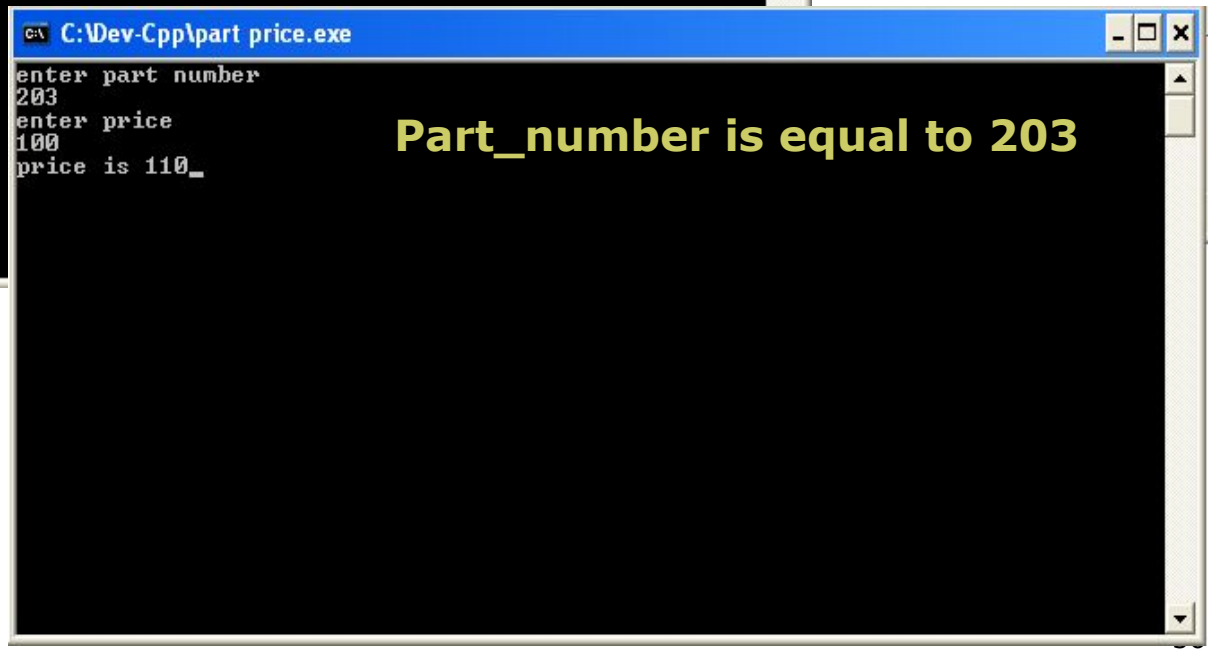


c++ code for the flow chart of if selection structure (program 1).

```
1.  //program 1
2.  //program to find the part prices for the given part number.

3.  #include<iostream>
4.  using namespace std;
5.  int main()
6.  {
7.      int part_number, ;
8.      float price;
9.      cout<<"enter part number"<<endl;
10.     cin>>part_number;
11.     cout<<"enter price"<<endl;
12.     cin>>price;
13.     if(part_number==203)
14.        price = price*1.1;
15.     cout<<"price is "<<price;
16.     return 0;
17. }
```

Output of program 1

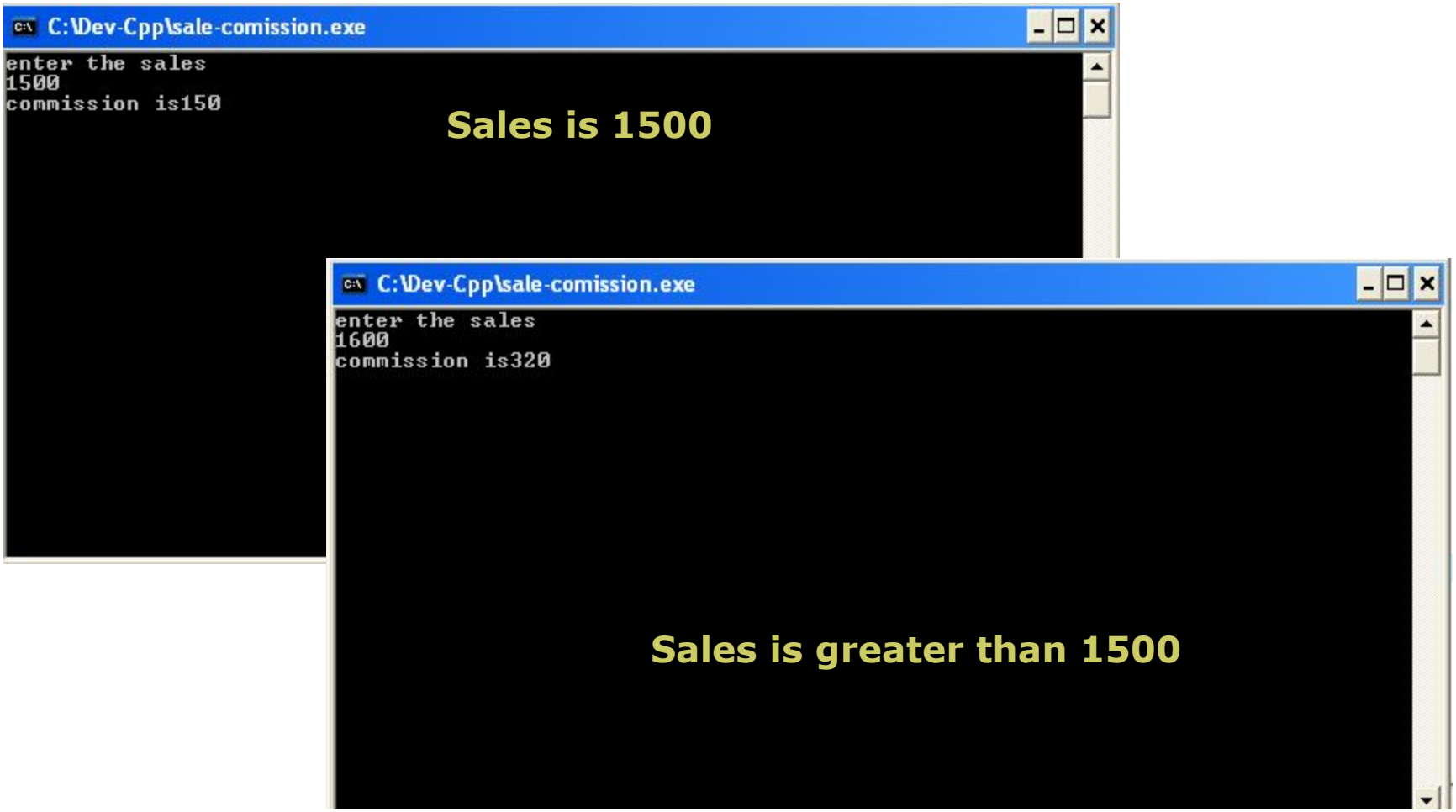




c++ code for the flow chart of if/else selection structure (program 2).

```
//program to calculate the sales commission

#include <iostream>
using namespace std;
int main()
{
    float sales, commission;
    cout<<"enter the sales"<<endl;
    cin>>sales;
    if (sales>1500)
        commission = sales*0.2;
    else
        commission = sales*0.1;
    cout<<"commission is"<<commission;

    return 0;
}
```

Output of program 2



Example of if/else

- Write a program to calculate the gross pay of an employee.
- The employee is paid at his basic hourly rate for the first 40 hours worked during a week. Any hours worked beyond 40 is considered overtime.
- overtime is paid at the 1.5 times the hourly rate.

Pseudocode

If total number of hours >40

Gross pay = rate*40 + rate*1.5(hours – 40)

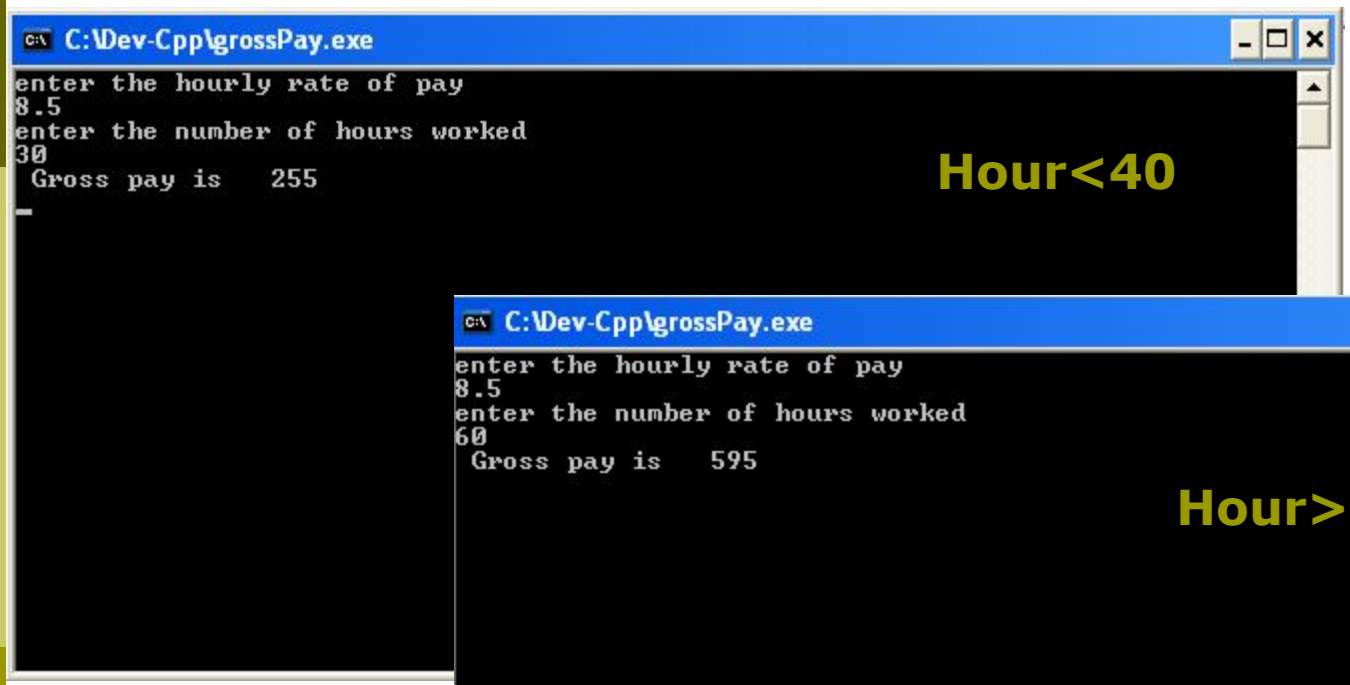
Else

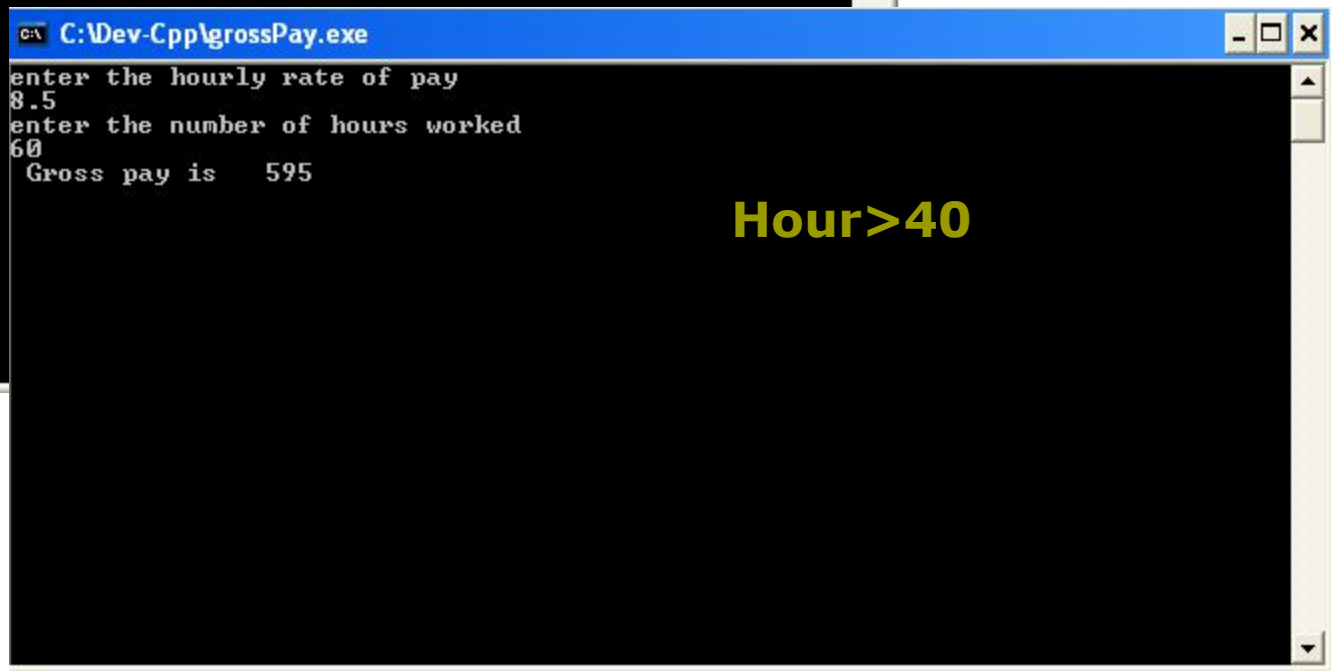
Gross pay = rate * hours

C++ code

```
1.  #include <iostream>
2.  using namespace std;
3.  int main()
4.  {
5.     int hour;          //declaration
6.     double gross_pay, rate;
7.     cout<<"enter the hourly rate of pay\t"<<endl;       //prompt
8.     cin>>rate;                              //reading data
9.     cout<<"enter the number of hours worked \t"<<endl;  //prompt
10.    cin>>hour;                        //readin data

11.    if(hour>40)                                        //condition if working hours is greater than 40
12.       gross_pay = rate*40 + 1.5*rate*(hour - 40);  //gross pay including extra hour
13.
14.    else                //if working hour is <=40 then use this formula to get gross pay
15.       gross_pay = rate*hour;
16.
17.    cout<<" Gross pay is \t"<<gross_pay<<endl; //printing the gross pay on screen.
18. return 0;
19. }
```

output


C:\Dev-Cpp\grossPay.exe
enter the hourly rate of pay
8.5
enter the number of hours worked
30
Gross pay is 255
Hour<40


C:\Dev-Cpp\grossPay.exe
enter the hourly rate of pay
8.5
enter the number of hours worked
60
Gross pay is 595
Hour>40

Nested if/else structure

- if structure that rests entirely within another if structure, within either the if or the else clause
  - One inside another, test for multiple cases
  - Once condition met, other statements skipped

Simple example to understand the nested if/else

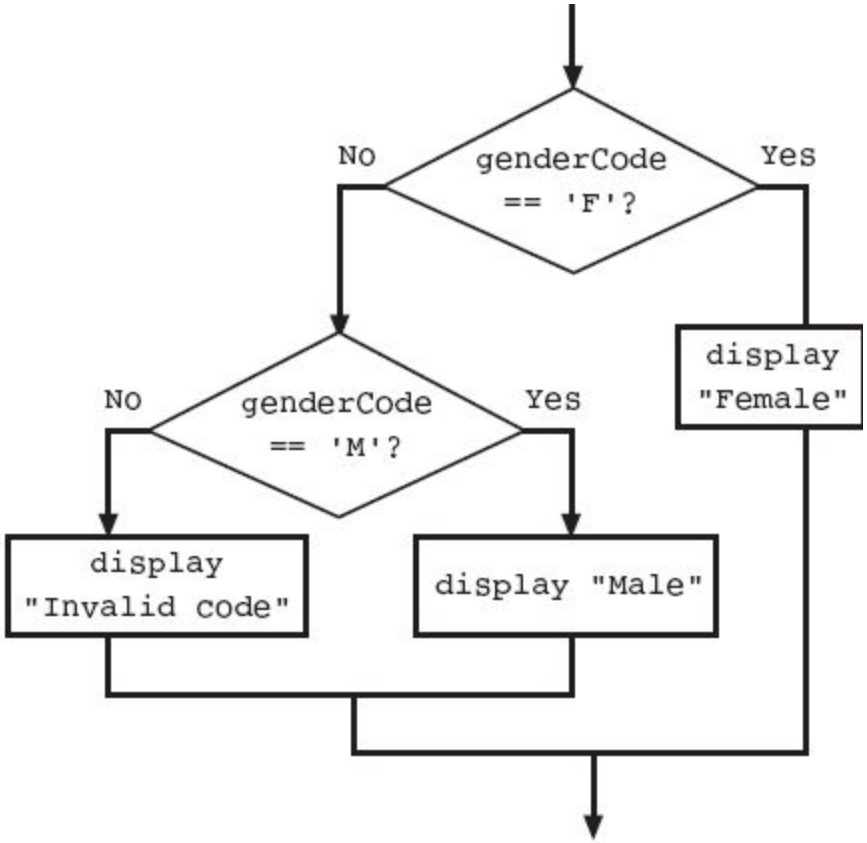
- Consider an example of a program segment that accepts a gender code and print gender according to that code.

- **Pseudocode**

```
if the gender code is F
  print female
  else
      if the gender code is M
  print male
    else
  print invalid code
```

Simple example to understand the nested if/else

- **Flow chart**



Simple example to understand the nested if/else

- **C++ code**

```
#include<iostream>
using namespace std;
int main()
{
    char genderCode;
    cout<<"Enter F for female or M for male ";
    cin>>genderCode;
    if(genderCode == 'F')
       cout<<"Female"<<endl;
    else
       if(genderCode == 'M')
            cout<<"Male"<<endl;
       else
            cout<<"Invalid code"<<endl;
}
```

```
//F and M are correct codes
//f and m are invalid codes
```

Nested if/else structure

- following pseudocode is example of nested if/else
- *if student's grade is greater than or equal to 90*
  Print "A"
- *else*
  if student's grade is greater than or equal to 80
  Print "B"
  else
  if student's grade is greater than or equal to 70
  Print "C"
  else
  if student's grade is greater than or equal to 60
  Print "D"
  else
  Print "F"

Nested if/else structure

C++ code

```
if ( grade >= 90 )        // 90 and above
      cout << "A";
   else

     if ( grade >= 80 )  // 80-89
        cout << "B";
    else

       if ( grade >= 70 )  // 70-79
          cout << "C";
     else

      if ( grade >= 60 )  // 60-69
          cout << "D";
       else                      // less than 60
          cout << "F";
```

- if grade>=90, first four conditions will be true. But only the **cout** statement after the first test will be executed. After that **cout** is executed, the **else** part of the outer **if/else** statement is skipped.

Avoiding Common Pitfalls with if Statements

- Forgetting that C++ is case sensitive
- Assuming that indentation has a logical purpose
- Adding an unwanted semicolon
- Forgetting curly braces
- Using = instead of == (explained with example on next slide)
- Making unnecessary comparisons

Confusing Equality (==) and Assignment (=) Operators

- L-values
  - Expressions that can appear on left side of equation
  - Can be changed (I.e., variables)
    - `x = 4;`
- R-values
  - Only appear on right side of equation
  - Constants, such as numbers (i.e. cannot write `4 = x;`)
- L-values can be used as R-values, but not vice versa