Object Oriented Programming

A Review To Programming Using C++

Lab 01, 02, 03

Basic Terms

- Programming Concepts and their Importance
- Computer Program
- Critical Skills
- Programming Languages

Programming Concepts

- What we are going to learn in Programming?
- The basic concepts of writing computer programs commonly known as software

Imp. of Programming Concepts

- Core of Compute/Software Engineering
- Many courses depend on this course
  - Problem Solving Using C++
  - Object Oriented Programming
  - Data Structures
  - Advance Programming Techniques
  - Web Application Engineering
  - Computer Graphics
  - Compiler Construction
  - Micro-Processor/Controller Programming

Computer Program (Software)

- "**Program** is – a precise sequence of steps to solve a particular problem."
- A sequence of instructions is called a **computer program** or **software**
  - Operating systems, Application software, games etc.
- These instructions are written in a **programming language**
- All Software development is done in some programming language

Critical Skills

- **Analysis**
- **Critical Thinking**
- **Pay Attention to Details**

Basic Programming Concepts

- Basic Programming constructs and building blocks
- Structured programming
- Variables
- Expressions
- Control structures
- Iterations (Loops)
- Functions
- Pointers

Programming Languages

- Machine Language
- Assembly Language
- High level languages
  - Procedural Languages
  - Object Oriented Languages

Programming Languages

Machine Language

- Written in the form of 0 & 1
- Computer directly understands

Assembly Languages

- Use Naturally understandable symbols called "Mnemonics"
- Load 5, Load 1, Add
- Assemblers are used to write code.

Programming Languages

High Level Languages

- Use naturally understandable language
- Compilers are used to write code.
- IDE's are use for rapid development like .NET environment, Visual Basic, Visual C++ and Visual C#

C++ as a Programming Language

- Extended version of **C** language
- C is evolved by Dennis Ritchie

- **C++** is developed by Bjarne Stroustrup

Why C++ ???

Mother of almost all programming languages

Basic Terms

- Variables
- Keywords in C++
- Data Types
- Operators
- Precedence of Operators

Variables

- A variable is a location in computer memory where a value can be stored
- Variables must be declared before they are used in a program
- In a program a variable has:
  - Data Type
  - Name
  - Size
  - Value

Variable Names

- Following rules should follow for naming the variables:
  - Upper case letters
  - Lower case letters
  - Digits (0 to 9)
  - Underscore (Not Recommended)
- Keywords can’t be used as variable name i.e. return, int, void etc.

Keywords in C++

asm	else	new	this
auto	enum	operator	throw
bool	explicit	private	true
break	export	protected	try
case	extern	public	typedef
catch	false	register	typeid
char	float	reinterpret_cast	typename
class	for	return	union
const	friend	short	unsigned
const_cast	goto	signed	using
continue	if	sizeof	virtual
default	inline	static	void
delete	int	static_cast	volatile
do	long	struct	wchar_t
double	mutable	switch	while
dynamic_cast	namespace	template	

Variable Names

- C++ is a case sensitive language: number is different from Number or nUmber

- Valid identifiers:
  - `int abc, char aBc, int first_var, float _first`

- Invalid identifiers:
  - `int 3bc, int a*b, int #a, int void`

Variable Naming Conventions

- **camelCase variable names**:
  - `number, firstName, dateOfBirth`

- **PascalCase variable names**
  - `Number, FirstName, DateOfBirth`

Rule of Thumb

- Use camelCase for variable names
- Use PascalCase for advanced naming e.g. Classes, functions

Variables Names – Examples

```
int firstNumber;

char gender = 'm';

float piValue = 3.14;

int myAge, int countryName;

int a, b, c;
```

Data Types

- **int**
- **short**
- **long**
- **float**
- **double**
- **char**

Data Types

Keyword	Range		Bytes of Memory
	Low	High	
`char`	**-128**	**127**	1
`short`	**-32,768**	**32768**	2
`int`	**-2,147,483,648**	**2,147,483,647**	4
`long`	**-2,147,483,648**	**2,147,483,647**	4
`float`	**$3.4*10^{-38}$**	**$3.4*10^{38}$**	4
`double`	**$1.7*10^{-308}$**	**$1.7*10^{308}$**	8
`long double`	**$3.4*10^{-4932}$**	**$3.4*10^{4932}$**	10

Arithmetic Operators

C++ Operation	Arithmetic Operator
Addition	+
Subtraction	-
Multiplication	*
Division	/
Modulus	%

Arithmetic Operators – Examples

Arithmetic Operators

```
firtNo + secondNo ;
firstValue * secondValue ;
dividend / divisor ;
dividend % divisor ;
```

% = Remainder

```
15 % 2 = 1
20 % 5 = 0
38 % 6 = ?
```

Precedence of Operators

- Highest: `( )`
- Next: `* , / , %`
- Lowest: `+ , –`

Same Precedence

- For addition, subtraction, division and multiplication → Left-to-right rule applies

- A+B+C = (A+B)+C

- For exponentiation → Right-to-left rule applies

- A ↑ B ↑ C = A ↑ (B ↑ C)

Quadratic Equation

In algebra

- Quadratic = $ax^2 + bx + c$

In C++

- Quadratic = a*x*x + b*x + c

Activity

10+2*3+5*4+12/6+5*20/2

88

Activity

5 *4+16/16+5*2

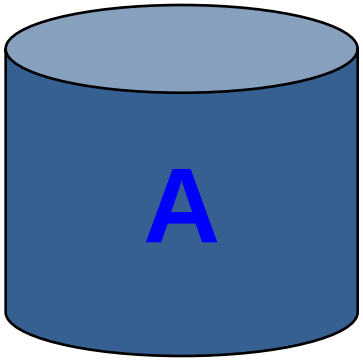
5 *4+16/(16+5*2)

Equal???

Assignment Operator

=

A = 20



20

Assignment Operator

- **L.H.S = R.H.S**

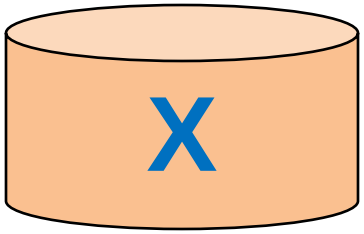
- `a + 2 = b + 10` Wrong
- `z = x + 4` OK
- `a + 14 = m` Wrong

Basic Concepts

- Comments
- Adding Two Integers
- Using Character
- Variables Type & Range
- Integer Overflow

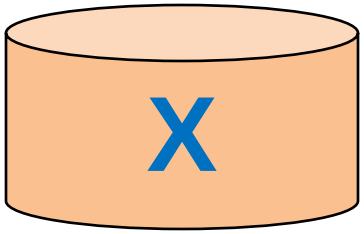
Assignment Operator

```
x = 5 ;
```



5

```
x = 10 ;
```



10

Comments

- We can **comment** something if we do not want it to be compiled

- It is mostly used to define or explain something in the code OR to identify errors

- **Single line comments →** ***//***
- **Multi line comment →** ***/*…….*/***

Comments

```
/*
This is our First Program.
We will display Hello World on screen
*/

#include<iostream>
int main()
{     //start of the function
      cout << "Hello World!!!";
}
```

Suggestion

Comment your code properly!!!

Adding Two Integers

```
#include<iostream>
using namespace std;
int main()
{
	int firstNo, secondNo, sum;
	cout << "Enter first integer\t";
	cin >> firstNo;
	cout << "Enter second integer\t";
	cin >> secondNo;
	sum = firstNo + secondNo;
	cout << "The sum is\t:" << sum << endl;
}
```

Adding Two Integers

```
Enter first integer          100

Enter second integer         200

The sum is:                  300
```

Adding Two Integers

```
#include<iostream>
using namespace std;
int main()
{
	int firstNo = 0;
	int secondNo = 0;
	int sum = 0;
	cout << "Enter first integer\t";
	cin >> firstNo;
	cout << "Enter second integer\t";
	cin >> secondNo;
	sum = firstNo + secondNo;
	cout << "The sum is\t:" << sum << endl;
}
```

Task 1

- Ask user to enter a three digit number and then display the number in reverse order

Task 1 – Solution

```
#include<iostream>
using namespace std;
int main()
{
    int number;
    cout << "Enter a three digit number";
    cin >> number;
    cout << number % 10;
    number = number / 10;
    cout << number % 10;
    number = number / 10;
    cout << number;
}
```

```
Enter a three digit number          123
321
```

Using Character

```
#include<iostream>
using namespace std;
int main()
{
      char x;
      x = 'a';
      cout << "The character is: << x << endl;
}
```

```
The character is:    a
```

Using Character

```
#include<iostream>
using namespace std;
int main()
{
      char x;
      cout << "Enter Character:";
      cin >> x ;
      cout << "You entered " << x << endl;
}
```

```
Enter Character:          a
You entered               a
```

Using Character

```
#include<iostream>
using namespace std;
int main()
{
      char x;
      cout << "Enter Character:";
      cin >> x ;
      cout << "ASCII Value:" << int(x);
}
```

```
Enter Character:          a
ASCII Value:              97
```

Variables Type & Range

Keyword	Range		Bytes of Memory
	Low	High	
`unsigned char`	0	256	1
`unsigned short`	0	65535	2
`unsigned int`	0	4,294,967,295	4
`unsigned long`	0	2,147,483,647	4

Integer Overflow

```
int main()
{
      int n = 1000;
      cout << "n = " << n <<endl;
      n = n * 1000;
      cout << "n = "<< n <<endl;
      n = n * 1000;
      cout << "n = "<< n <<endl;
      n = n * 1000;
      cout << "n = " << n <<endl;
}
```

```
n = 1000;

n = 1000000;

n = 1000000000

n = -727379968
```

Code Flow

- In any programming language the flow of the code can be of four types
  1) Sequential
  2) Selection
  3) Repetition
  4) Go to (Obsolete Style)

Code Flow

- **Sequential**
- **Selection**
  - If-Else, Switch
- **Repetition**
  - while Loop
  - do-while Loop
  - for Loop

Selection Statements

Consider the following statements:

- If x is divisible by 2 then x is even number
- If marks are equal or greater than 85 then grade is 'A‘
- If 8%4 is 0 then 8 is divisible by 4
- If number is greater than 0 then number is positive number
- If marks are greater than 80% and age is equal to or greater than 16 then grant admission in university

Relational Operators

Algebraic	In C++	Example	Meaning
>	>	`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
=	==	`x==y`	x is equal to y
≠	!=	`x!=y`	x is not equal to y

Relational Operators

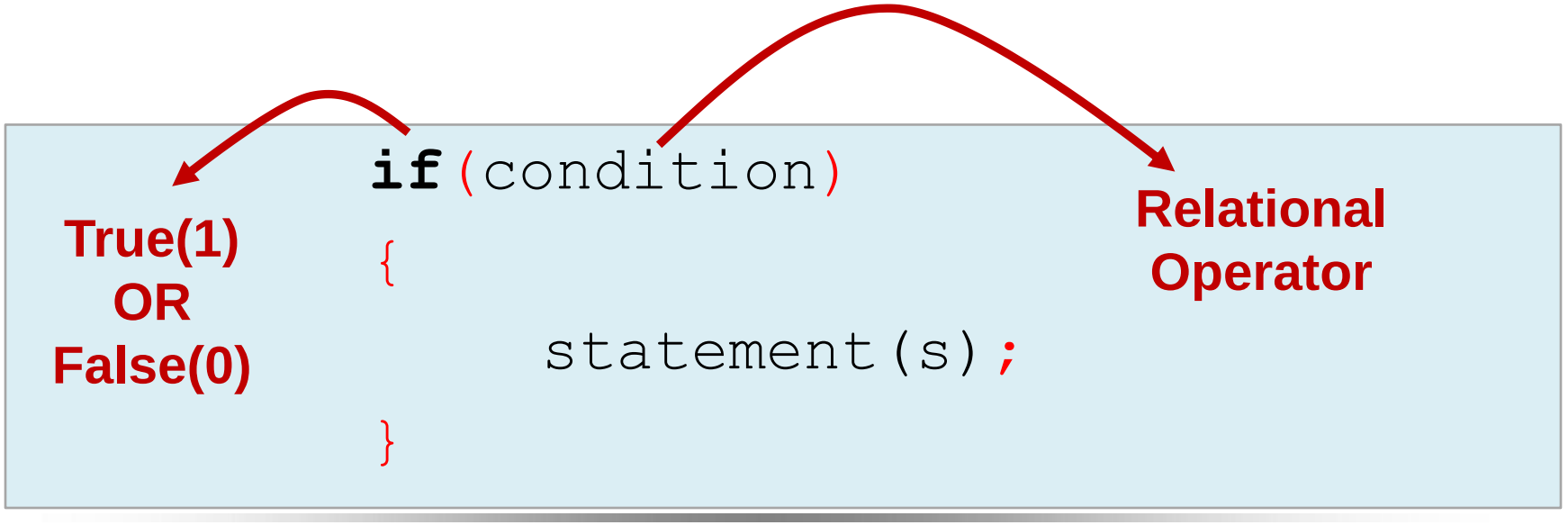
```
#include<iostream>
using namespace std;
int main()
{
	int x = 3;
	int y = 5;
	cout << (x > y) << endl;
	cout << (x < y) << endl;
}
```

```
0
1
```

if Statement

- **If** statement is used in C++ for selection purposes

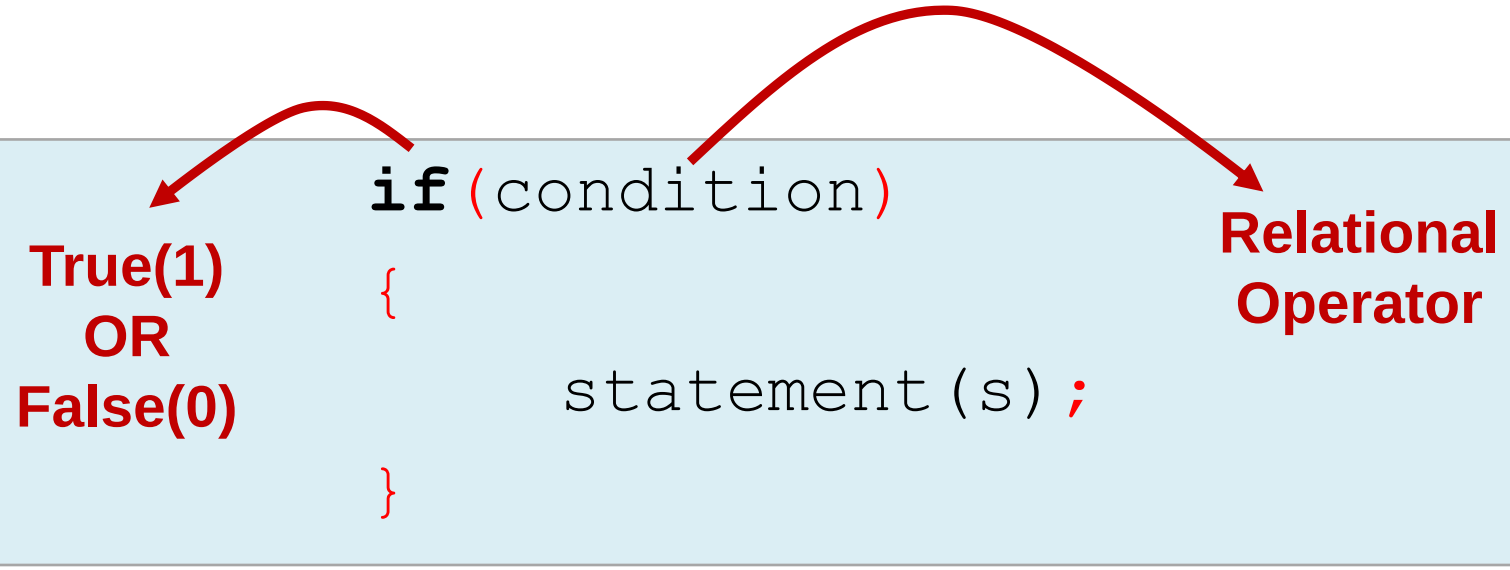
- Structure of **if** statement is:



if Statement

```
if grade is greater than or equal to 60
     Print "Passed"
```

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



if Statement – Example

```
#include<iostream>
using namespace std;
int main()
{
	int x, y;
	cout << "Enter two integers: ";
	cin >> x >> y;

	if(x == y)
	{
		cout << x << " is equal to " << y;
	}
```

if Statement – Example

```
        if(x != y)
        {
                cout << x << "is not equal to" << y;
        }
        if(x < y)
        {
                cout << x << " is less than " << y;
        }
        if(x > y)
        {
                cout << x << "is greater than" << y;
        }

} // End of Program
```

if Statement – Example

```
Enter two integers:   10    5

10 is greater than 5
```

```
Enter two integers:   15    25

15 is less than 25
```

```
Enter two integers:   50    50

50 is equal to 50
```

Problem – 1 (Statement)

- Prompt user to enter a value. Check if the number is even then display a message that number is even. If number if odd then display message number is odd

Problem – 1 (Solution)

```
int main()
{
	int num;
	cout << "Enter a number:\t";
	cin >> num;
	if(num%2 == 0)
	{
		cout << num <<": is an even number";
	}
	if(num%2 != 0)
	{
		cout << num << ": is an odd number";
	}
}
```

Problem – 1 (Output)

```
Enter a number:399
399 is an odd number
```

```
Enter a number:500
500 is an even number
```

if-else Statement – Example

```
#include<iostream>
using namespace std;
int main()
{
	int num;
	cout << "Enter your grade: ";
	cin >> num;
	if(num >= 50)
		cout<<"Congraulations you are Passed";
	else
		cout<<"You are failed";
}
```

if-else Statement – Example

```
Enter your grade:   75
Congratulations you are Passed
```

```
Enter your grade:   45
You are failed
```

Problem – 2 (Statement)

- Ask user to enter two positive numbers. Display which number is greater between two.

Problem – 2 (Solution)

```
int main()
{
      int num1, num2;
      cout << "Enter first num:\t ";
      cin >> num1;
      cout << "Enter second num:\t ";
      cin >> num2;
      if(num1 > num2)
            cout << num1 << " is greater";
      else
            cout << num2 << " is greater";
}
```

```
Enter first number:          100
Enter second number:         300
300 is greater
```

Problem – 3 (Statement)

- Ask user to enter two positive numbers. Display which number is greater between two. If the user enters negative number(s) then display message that negative number(s) is/are not allowed.

Nested if…else Statements

- Nested if…else statements test for multiple cases by placing if…else selection statements inside other if…else selection statements

Nested if…else Statements

```
bool job; char martial; int age;

cout << "Enter Martial Status:";
cin >> status;
cout << "Enter age:";
cin >> age;
cout << "Enter job Status:";
cin >> job;

if(status == 'u')
      if(age <= 25)
            if(job == false)
                  cout << "Eligible for Loan";
else
      cout << "Not eligible for Loan";
```

Nested if…else Statements

```
Enter Martial Status:       u
Enter age:                  20
Enter job Status:           0

Eligible for Loan
```

```
Enter Martial Status:       u
Enter age:                  18
Enter job Status:           1

Not eligible for Loan
```

Multiple Selection (else-if)

```
if(condition)
{
      statement(s);
}

else if(condition)
{
      statement(s);
}
……

else
{
      statement(s);
}
```

Multiple Selection (else-if)

```
if(condition)
{
      statement(s);
}

else if(condition)
{
      statement(s);
}
.....

else
{
      statement(s);
}
```

Multiple Selection (else-if)

```
int main()
{
	int number;
	cout << "Enter a  number between 1 to 3:";
	cin >> number;

	if(number == 1)
	{
		cout << "You pressed 1" << endl;
	}
	else if(number == 2)
	{
		cout << "You pressed 2" << endl;
	}
```

Multiple Selection (else-if)

```
        else if(number == 3)
        {
                cout << "You pressed 3" << endl;
        }
        else
        {
                cout << "Invalid input";
        }
}
```

```
Enter a  number between 1 to 3:      3
You pressed 3
```

```
Enter a  number between 1 to 3:      50
Invalid input
```

Problem (Statement)

- Develop a calculator which is able to handle two numbers. The calculator can perform addition, subtraction, multiplication, and division.

Problem (Analysis)

- Ask the user to enter two numbers
- Then ask to enter the operator (**+**, **-**, ***I***, ***)**
- Calculate the result
- Display the result

Problem (Solution)

```
#include<iostream>
using namespace std;
int main()
{
	int firstNo, secondNo;
	char opChoice;

	cout << "Enter first number:\t";
	cin >> firstNo;

	cout << "Enter second number:\t";
	cin >> secondNo;

	cout << "Enter operator (+,-,/,*):\t";
	cin >> opChoice;
```

Problem (Solution)

```
        if(opChoice == '+')
        cout << firstNo << " + " << secondNo << " = "
        << firstNo + secondNo;

        else if(opChoice == '-')
              cout << firstNo << " - " << secondNo <<
              " = " << firstNo - secondNo;

        else if(opChoice == '*')
              cout << firstNo << " * " << secondNo <<
              " = " << firstNo * secondNo;

        else(opChoice == '/')
              cout << firstNo << " / " << secondNo <<
              " = " << firstNo / secondNo;
}
```

Problem (Output)

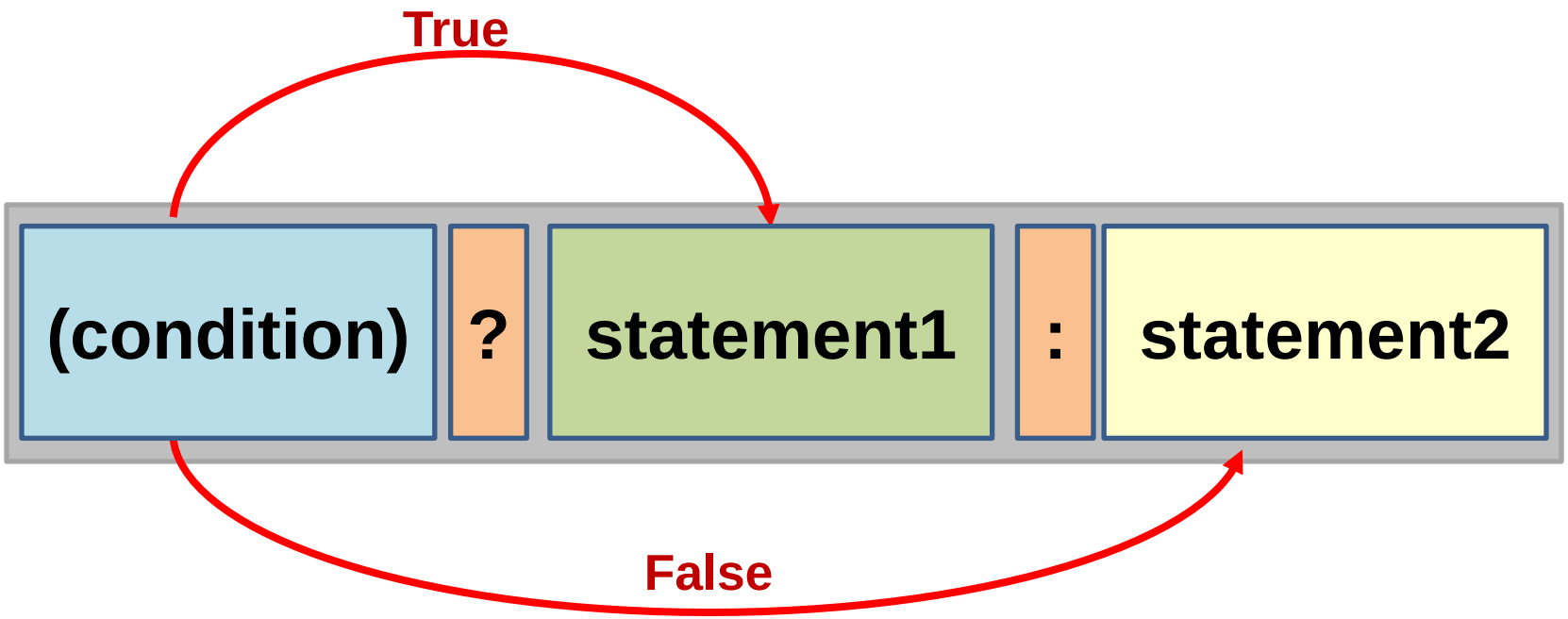
```
Enter first number:                 400
Enter second number:                200
Enter operator (+,-,/,*):           +
400 + 200 = 600
```

```
Enter first number:                 300
Enter second number:                400
Enter operator (+,-,/,*):           *
300 + 400 = 1200
```

```
Enter first number:                 500
Enter second number:                100
Enter operator (+,-,/,*):           /
500 + 100 = 100
```

Ternary Operator

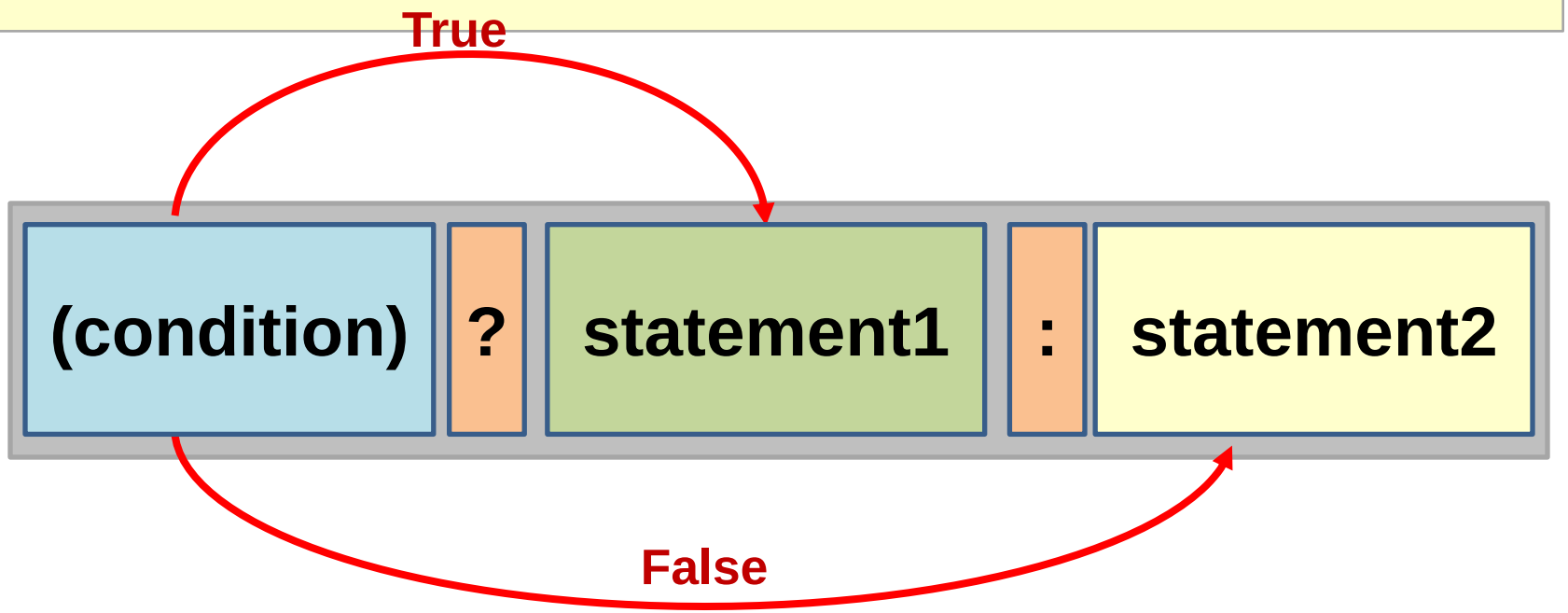
- Ternary operator exactly work as if-else statement do
- Structure of ternary operator is:



Ternary Operator

```
cout << (grade >= 60 ? "Passed" : "Failed");
```

```
grade >= 60 ? cout<<"Passed" : cout<<"Failed";
```



Ternary Operator – Example

```
#include<iostream>
using namespace std;
int main()
{
      int firstNo, secondNo;
      cout << "Enter first number\t";
      cin >> firstNo;
      cout << "Enter second number\t";
      cin >> secondNo;

       cout << (firstNo > secondNo ? "First number
      is greater" : "Second number is greater");
}
```

```
Enter first number          200
Enter second number         600
Second number is greater
```

Multiple Selection (switch)

- C++ provides the **switch** multiple-selection statement to perform many different actions based on the possible values of a variable or expression

- Each action is associated with the value of a constant integral expression (i.e., any combination of character and integer constants that evaluates to a constant integer value)

Multiple Selection (switch)

```
switch(choice)
{
        case 1:
                statement(s);
                break;

        case 2:
                statement(s);
                break;
        ..............

        default:
                statement;
}
```

Multiple Selection (switch)

```
switch(choice)
{
    case 1:
        statement(s);
        break;

    case 2:
        statement(s);
        break;
    ..............

    default:
        statement;
}
```

Problem – 2 (Statement)

- Develop a calculator which is able to handle two numbers. The calculator can perform addition, subtraction, multiplication, and division.

Problem – 2 (Solution)

```
int firstNo, secondNo;
char opChoice;

cout << "Enter first number:\t";
cin >> firstNo;
cout << "Enter second number:\t";
cin >> secondNo;
cout << "Enter operator (+,-,/,*):\t";
cin >> opChoice;

switch(opChoice)
{
      case '+':
            cout << firstNo << " + " << secondNo <<
            " = " << firstNo + secondNo;
            break;
```

Problem – 2 (Solution)

```
        case '-':
                cout << firstNo << " - " << secondNo <<
                " = " firstNo - secondNo;
                break;
        case '*':
                cout << firstNo << " * " << secondNo <<
                " = " << firstNo * secondNo;
                break;
        case '/':
                cout << firstNo << " / " << secondNo <<
                " = " << firstNo / secondNo;
                break;
        default:
                cout << "Invalid input";
    }
} //End of program
```

Problem – 2 (Solution)

```
Enter first number:                 400
Enter second number:                200
Enter operator (+,-,/,*):           +
400 + 200 = 600
```

```
Enter first number:                 600
Enter second number:                300
Enter operator (+,-,/,*):           /
600 / 100 = 6
```

```
Enter first number:                 300
Enter second number:                400
Enter operator (+,-,/,*):           *
300 * 400 = 1200
```

Problem – 3 (Statement)

- Write a program that takes grade as an input from the user and depending upon the grade display the message. Following is the criteria:
  - If grade is 'A' then display “Excellent”
  - If grade is 'B' then display “Very Good”
  - If grade is 'C' then display “Good”
  - If grade is 'D' then display “Poor”
  - If grade is 'F' then display “Fail”

Problem – 3 (Solution)

```
#include<iostream>
using namespace std;
int main()
{
      char grade ;
      cout << "Please enter the student's grade : ";
      cin >> grade ;

      switch(grade)
      {
            case 'A' : // grade was case A
                  cout << "Excellent" ;
                  break ; //necessary to exit switch

            case 'B' : // grade was case B
                  cout << "Very Good" ;
                  break ; //necessary to exit switch
```

Problem – 3 (Solution)

```
            case 'C' : // grade was case C
                  cout << "Good";
                  break ; //necessary to exit switch

            case 'D' : // grade was case D
                  cout << "Poor" ;
                  break ; //necessary to exit switch

            case 'F' : // grade was case F
                  cout << "Fail" ;
                  break ; //necessary to exit switch

            default :
                  cout << "Enter grade from A to D
                        or F";
}
```

Problem – 3 (Output)

```
Please enter the student's grade :    A
Excellent
```

```
Please enter the student's grade :    F
Fail
```

```
Please enter the student's grade :    H
Enter grade from A to D or F
```

What will happen if user enters lower letter?

Problem – 4 (Statement)

- Write a program that takes grade as an input from the user and depending upon the grade display the message. Following is the criteria:
  - If grade is 'A' or 'a' then display “Excellent”
  - If grade is 'B' or 'b' then display “Very Good”
  - If grade is 'C' or 'c' then display “Good”
  - If grade is 'D' or 'd' then display “Poor”
  - If grade is 'F' or 'f' then display “Fail”

Problem – 4 (Solution)

```
char grade ;
cout << "Please enter the student's grade : ";
cin >> grade ;

switch(grade)
{
      case 'A' : // grade was upper case A
      case 'a' : // grade was lower case a
            cout << "Excellent" ;
            break ; // necessary to exit switch

      case 'B' : // grade was upper case B
      case 'b' : // grade was lower case b
            cout << "Very Good" ;
            break ; // necessary to exit switch
```

Problem – 4 (Solution)

```
        case 'C' : // grade was upper case C
        case 'c' : // grade was lower case c
                   cout << "Good";
                   break ; //necessary to exit switch

        case 'D' : // grade was upper case D
        case 'd' : // grade was lower case d
                   cout << "Poor" ;
                   break ; // necessary to exit switch

        case 'F' : // grade was upper case F
        case 'f' : // grade was lower case f
                   cout << "Fail" ;
                   break ; // necessary to exit switch

        default :
                   cout << "Enter grade from A to D or F" ;
}
```

Problem – 4 (Output)

```
Please enter the student's grade :    A
Excellent
```

```
Please enter the student's grade :    f
Fail
```

```
Please enter the student's grade :    H
Enter grade from A to D or F
```

Problem – 5 (Statement)

- In a company, there are deductions from the salary of the employees for a fund. The deductions rules are as follows:
  - If salary is less than Rs.10,000 then no deduction
  - If salary is equal to or more than Rs.10,000 and less than Rs.20,000 then deduct Rs.1,000 as fund
  - If salary is equal to or more than 20,000 then deduct 7 % of the salary for fund
- Input salary from user and after appropriate deduction show the net payable amount.

Home Work Assignment

Loops

- Repetition
- While Loop
- Increment and Decrement operators

Repetition (Loops)

- Every loop can be defined by three way:
  1. Starting Point
  2. Ending Point
  3. Sequence of Moving

while Repetition Statement

- A **repetition statement** specifies that a program should repeat an action while the given condition remains true

- The pseudo code statement:

```
While there are items on my shopping list
      Purchase next item
```

while Repetition Statement

- The action will be performed repeatedly while the condition remains true
- The statement contained in the `While` repetition statement constitutes the body of the `While`, which can be a single statement or a block
- Eventually, the condition will become false
- At this point, the repetition terminates statement after the repetition statement executes

While Loop

- Structure of while loop can be defined as:

```
while(condition)
{
    statement(s);
}
```

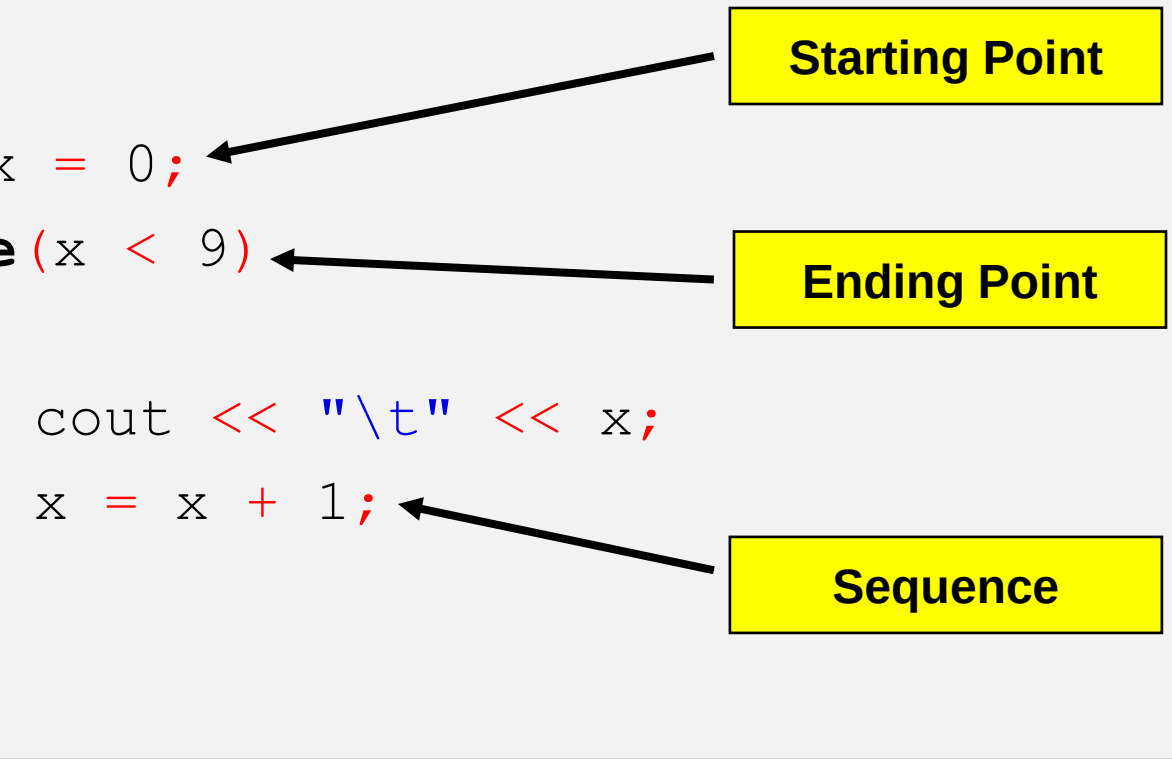
While Loop

- Structure of while loop can be defined as:

```
while(condition)
{
    statement(s);
}
```

While Loop

```
int main()
{
        int x = 0;
        while(x < 9)
        {
                cout << "\t" << x;
                x = x + 1;
        }
}
```




```
0       1       2       3       4       5       6       7       8
```

While Loop

```
int main()
{
	int x = 15;
	while(x > 0)
	{
		cout << x << endl;
		x = x - 1;
	}
}
```

```
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
```

While Loop

```
int main()
{
	int x = 0;
	while(x < 25)
	{
		if(x%2 == 0)
		cout << x << endl;
		x = x + 1;
	}
}
```

```
0
2
4
6
8
10
12
14
16
18
20
22
24
```

While Loop

```
int main()
{
	int x = 0;
	while(x < 25)
	{
		if(x%2 == 0)
		cout << x << endl;
		x++;
	}
}
```


```
0
2
4
6
8
10
12
14
16
18
20
22
24
```

Increment and Decrement operators

- **++** is unary increment operator which replaces `a=a+1` into `a++`
- **--** is unary decrement operator which replaces `a=a-1` into `a--`
- Used as pre or post fashion

Increment and Decrement operators

Operator	Called	Sample
++	Pre-increment	`++a`
++	Post-increment	`a++`
--	Pre-decrement	`--a`
--	Post-decrement	`a--`

Increment and Decrement operators

```
int main()
{
      int c = 5;
      cout << "Behavior of Post-increment" << c;
      cout << c++ << endl;
      cout << c << endl;
      c = 5;
      cout << "Behavior of Pre-increment" << c;
      cout << ++c << endl;
      cout << c << endl;
}
```

Increment and Decrement operators

```
Behavior of Post-increment
5
5
6

Behavior of Pre-increment
5
6
6
```

Problem – 1 (Statement)

- Calculate the sum of numbers from **1** to **100**

Problem – 1 (Solution)

```
int main()
{
      int number = 1;
      int sum = 0;
      while(number <= 100)
      {
            sum = sum + number;
            number++;
      }
      cout << "The sum of first " << number-1
      << " number is:\t " << sum << endl;
}
```

```
The sum of first 100 number is:  5050
```

Problem – 2 (Statement)

- Calculate the sum of even numbers from **1** to **100**

Problem – 2 (Statement)

```
int main()
{
	int number = 1;
	int sum = 0;
	while(number < 100)
	{
		if(number%2 == 0)
		{
			sum = sum + number;
		}
	number++;
	}
	cout << "The sum of first " << number
	<< "even numbers is:\t" << sum << endl;
}
```

```
The sum of first 100 even numbers is:  2550
```

Problem – 2 (Statement)

- Calculate the sum of odd numbers from **1** to **100**

Problem – 2 (Statement)

```
int main()
{
	int number = 1;
	int sum = 0;
	while(number <= 100)
	{
		if(number%2 != 0)
		{
			sum = sum + number;
		}
	number++;
	}
	cout << "The sum of first " << number
	<< "odd numbers is:\t" << sum << endl;
}
```

```
The sum of first 100 odd numbers is:  2500
```

Home Task

- Input the grades of a class of ten students. The grades must be between **0** to **100** inclusively. Calculate and display the total of the grades and the class average.

do-while Statement

- There may be certain situations when the body of while loop does not execute even a single time
- This occurs when the condition in while is false
- In while loop, the condition is tested first and the statements in the body are executed only when this condition is true

do-while Statement

- If the condition is false, then the control goes directly to the statement after the closed brace of the while loop
- In while structure, the loop can execute zero or more times
- There may be situations where it may need that some task must be performed at least once

do-while Statement

- The structure of do-while loop is as follows:

```
do
{
    statement(s) ;
}
while ( condition ) ;
```

do-while Statement

- The structure of do-while loop is as follows:

```
do
{
    statement(s) ;
}
while ( condition ) ;
```

Problem – 1 (Statement)

- Write a program which inputs from user a character between **a** to **z** and guess whether the character matches or not. If it matches then display a message "Congratulations". It gives the user five chances or tries to guess the character.

Problem – 1 (Solution)

```
int main() {
        int tryNo = 0 ;
        char guess ;
        do {
                cout << "Enter character(a-z) to guess:" ;
                cin >> guess ;
                //check the entered character for equality
                if (guess == 's') {
                        cout << "Congratulations..." ;
                        tryNo = 6 ;
                }
                else {
                        tryNo++ ;
                }
        }
        while ( tryNo < 5 ) ;
}
```

Problem – 1 (Solution)

```
int main() {
        int tryNo = 0 ;
        char guess ;
        do {
                cout << "Enter character(a-z) to guess:";
                cin >> guess ;
                //check the entered character for equality
                if (guess == 's') {
                        cout << "Congratulations...";
                        break ;
                }
                else {
                        tryNo = tryNo + 1;
                }
        }
        while (tryNo < 5);
}
```

Problem – 1 (Statement)

```
Enter character(a-z) to guess:   m
Enter character(a-z) to guess:   s
Congratulations...
```

```
Enter character(a-z) to guess:   a
Enter character(a-z) to guess:   b
Enter character(a-z) to guess:   c
Enter character(a-z) to guess:   d
Enter character(a-z) to guess:   s
Congratulations...
```

Problem – 2 (Statement)

- Calculate the sum of numbers from **1** to **100** using do-while loop.

Problem – 2 (Solution)

```
int main()
{
      int number = 1 ;
      int sum = 0 ;
      do
      {
            sum = sum + number ;
            number++ ;
      }
      while ( number <= 100 ) ;

      cout << "The sum of first " << number - 1;
      cout << " number is:\t " << sum << endl;
}
```

```
The sum of first 100 number is:  5050
```

Problem – 3 (Statement)

- Calculate the sum of even numbers from **1** to **100** using do-while loop.

Problem – 3 (Statement)

```
int main()
{
        int number = 0 ;
        int sum = 0 ;
        do
        {
                if ( number%2 == 0 )
                {
                        sum = sum + number ;
                }
        number++ ;
        } while ( number <= 100 ) ;
        cout << "The sum of first " << number ;
        cout << " even numbers is:\t" << sum << endl ;
}
```

```
The sum of first 100 even numbers is:  2450
```

Problem – 4 (Statement)

- Calculate the sum of odd numbers from **1** to **100** using do-while loop.

Problem – 4 (Statement)

```
int main()
{
        int number = 0 ;
        int sum = 0;
        do
        {
                if ( number%2 != 0 )
                {
                        sum = sum + number ;
                }
        number++ ;
        } while ( number <= 100 ) ;
        cout << "The sum of first " << number ;
        cout << "odd numbers is:\t" << sum << endl ;
}
```

```
The sum of first 100 odd numbers is:  2500
```

Problem – 5 (Statement)

- Write the table of **5** using do-while loop.

Problem – 5 (Solution)

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

int main()
{
	int counter = 1 ;

	do
	{
		cout<<"5 X "<<counter<<" = "<<5*counter;
		cout << endl ;

		counter++ ;
	}
	while ( counter <= 10 ) ;
}
```

Problem – 5 (Output)

```
5 X 1 = 5
5 X 2 = 10
5 X 3 = 15
5 X 4 = 20
5 X 5 = 25
5 X 6 = 30
5 X 7 = 35
5 X 8 = 40
5 X 9 = 45
5 X 10 = 50
```

For Loop

- The structure of for loop can be described as:

```
for ( statement;  condition;  statement )
{
      statement(s) ;
}
```

For Loop

```
#include<iostream>
using namespace std;
int main()
{
      for ( int i = 1; i <= 5; i++ )
      {
            cout << "\t" << i ;
      }
}
```

```
      1        2        3        4        5
```

Problem – 1 (Statement)

- Calculate the sum of numbers from **1** to **100** using for loop.

Problem – 1 (Solution)

```
int main()
{
        int number ;
        int sum = 0 ;

        for ( number = 1; number <= 100; number++ )
        {
                sum = sum + number ;
        }

        cout << "The sum of first " << number - 1 ;
        cout << " number is:\t " << sum << endl ;
}
```

```
The sum of first 100 number is:  5050
```

Problem – 2 (Statement)

- Calculate the sum of even numbers from **1** to **100** using for loop.

Problem – 2 (Statement)

```
int main()
{
	int sum = 0 ;

	for ( int number = 0; number < 100; number++ )
	{
		if ( number%2 == 0 ) {
			sum = sum + number ;
		}
	}

	cout << "The sum of first " << number ;
	cout << " even numbers is:\t" << sum << endl ;
}
```

```
The sum of first 100 even numbers is:  2450
```

Problem – 3 (Statement)

- Calculate the sum of odd numbers from **1** to **100** using for loop.

Problem – 3 (Statement)

```
int main()
{
        int sum = 0 ;

        for ( int number = 0; number < 100; number++ )
        {
                if ( number%2 != 0 )
                {
                        sum = sum + number ;
                }
        }

        cout << "The sum of first " << number ;
        cout << "odd numbers is:\t" << sum << endl ;
}
```

```
The sum of first 100 odd numbers is:  2500
```

Problem – 4 (Statement)

- Write the table of **5** using for loop.

Problem – 4 (Solution)

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

int main()
{
	int counter ;

	for ( counter = 1; counter <= 10; counter++ )
	{
		cout<<"5 X "<<counter<<" = "<<5*counter;
		cout << endl;
	}
}
```

Problem – 4 (Output)

```
5 X 1 = 5
5 X 2 = 10
5 X 3 = 15
5 X 4 = 20
5 X 5 = 25
5 X 6 = 30
5 X 7 = 35
5 X 8 = 40
5 X 9 = 45
5 X 10 = 50
```

Problem – 5 (Statement)

- Write a table of a number entered by the user using for loop. Take the table number and the end limit from the user.

Problem – 5 (Solution)

```
int main()
{
	int tableNo, limit ;

	cout << "Enter Table No.\t\t" ;
	cin >> tableNo ;
	cout << "Enter Ending No.\t" ;
	cin >> limit ;

	for ( int i = 1; i <= limit; i++ )
	{
		cout << tableNo << " X " << i << " = " ;
		cout << tableNo * i << endl ;
	}
}
```

Problem – 5 (Output)

```
Enter Table No.          6
Enter Ending No.         10
6 X 1 = 6
6 X 2 = 12
6 X 3 = 18
6 X 4 = 24
6 X 5 = 30
6 X 6 = 36
6 X 7 = 42
6 X 8 = 48
6 X 9 = 54
6 X 10 = 60
```

Nested Loops

- We can use loops inside loop body
- Lot of care is needed in this type of structure
- First inner loop complete its iteration and then control shift to outer one, this process continues till end

Nested Loops

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

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

Controlling Loops

- We can use break and continue statement to control the interactions of a loop
- **break** statement stops the loops and start executing program from the line after loop
- **continue** statement breaks only the current iteration

break statement

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

int main()
{
	int i ;
	for ( i = 1; i <= 10; i++ )
	{
		cout << i << endl;
		if ( i == 5 )
		break ;
	}

	cout << "Its line after break execution" ;
}
```

break statement

```
1
2
3
4
5

Its line after break execution
```

continue statement

```
#include<iostream>
using namespace std ;
int main()
{
      int i ;
      for ( i = 1; i <= 10; i++ )
      {
            if( i == 5 )
            continue ;
            cout << i << endl ;
      }
}
```