

**Lab Manual**

*for*

# **Object Oriented Programming Lab**

**3137**

**Diploma In Computer Engineering**

**3<sup>rd</sup> Semester**

*by*

**SITTTR**

**Kalamassery**

## STATE INSTITUTE OF TECHNICAL TEACHERS TRAINING AND RESEARCH

### GENERAL INSTRUCTIONS

Rough record and Fair record are needed to record the experiments conducted in the laboratory. Rough records are needed to be certified immediately on completion of the experiment. Fair records are due at the beginning of the next lab period. Fair records must be submitted as neat, legible, and complete.

#### INSTRUCTIONS TO STUDENTS FOR WRITING THE FAIR RECORD

In the fair record, the index page should be filled properly by writing the corresponding experiment number, experiment name, date on which it was done and the page number.

On the **right side** page of the record following has to be written:

1. **Title:** The title of the experiment should be written in the page in capital letters.
2. In the left top margin, experiment number and date should be written.
3. **Aim:** The purpose of the experiment should be written clearly.
4. **Apparatus/Tools/Equipments/Components used:** A list of the Apparatus/Tools /Equipments /Components used for doing the experiment should be entered.
5. **Principle:** Simple working of the circuit/experimental set up/algorithm should be written.
6. **Procedure:** steps for doing the experiment and recording the readings should be briefly described(flow chart/programs in the case of computer/processor related experiments)
7. **Results:** The results of the experiment must be summarized in writing and should be fulfilling the aim.
8. **Inference :** Inference from the results is to be mentioned.

On the **Left side** page of the record following has to be recorded:

1. **Circuit/Program:** Neatly drawn circuit diagrams/experimental set up.
2. **Design:** The design of the circuit/experimental set up for selecting the components should be clearly shown if necessary.

3. **Observations:** i) Data should be clearly recorded using Tabular Columns.

ii) Unit of the observed data should be clearly mentioned

iii) Relevant calculations should be shown. If repetitive calculations are needed, only show a sample calculation and summarize the others in a table.

4. **Graphs :** Graphs can be used to present data in a form that shows the results obtained, as one or more of the parameters are varied. A graph has the advantage of presenting large

amounts of data in a concise visual form. Graph should be in a square format.

### **GENERAL RULES FOR PERSONAL SAFETY**

1. Always wear tight shirt/lab coat , pants and shoes inside workshops.

2. REMOVE ALL METAL JEWELLERY since rings, wrist watches or bands, necklaces, etc. make excellent electrodes in the event of accidental contact with electric power sources.

3. DO NOT MAKE CIRCUIT CHANGES without turning off the power.

4. Make sure that equipment working on electrical power are grounded properly.

5. Avoid standing on metal surfaces or wet concrete. Keep your shoes dry.

6. Never handle electrical equipment with wet skin.

7. Hot soldering irons should be rested in its holder. Never leave a hot iron unattended.

8. Avoid use of loose clothing and hair near machines and avoid running around inside lab .

### **TO PROTECT EQUIPMENT AND MINIMIZE MAINTENANCE:**

**DO:** 1. SET MULTIRANGE METERS to highest range before connecting to an unknown source.

2. INFORM YOUR INSTRUCTOR about faulty equipment so that it can be sent for repair.

**DO NOT:** 1. Do not MOVE EQUIPMENT around the room except under the supervision of an instructor.

## **LIST OF EXPERIMENTS**

1. Simple C++ programs to implement various control structures.
  - a. if statement
  - b. switch case statement and do while loop
  - c. for loop
  - d. while loop
2. Programs to understand structure & unions.
  - a. structure
  - b. union
3. Programs to understand pointer arithmetic.
4. Functions & Recursion.
  - a. recursion
  - b. function
5. Inline functions.
6. Programs to understand different function call mechanism.
  - a. call by reference
  - b. call by value
7. Programs to understand storage specifiers.
8. Constructors & destructors.
9. Use of “this” pointer using class
10. Programs to implement inheritance and function overriding.
  - a. multiple inheritance –access Specifiers
  - b. hierarchical inheritance – function overriding /virtual Function
11. Programs to overload unary & binary operators as member function & non member function.
  - a. unary operator as member function
  - b. binary operator as non member function
12. Programs to understand friend function & friend Class.
  - a. friend Function
  - b. friend class
13. Programs on class templates

## EXP NO-1 a

### Use of if statement

**AIM-**Develop a C++ program to find all roots of a quadratic equation  $ax^2+bx+c=0$ .

**OBJECTIVES-** Study the use of selection structure for branching

Understand the usage of if else statement

**TOOLS-** Hardware Requirement: Desk Top Computer

Software Requirement: Linux Operating System with g++ or eclipse

### THEORY/ALGORITHM

An if statement can be followed by an optional else statement, which executes when the boolean expression is false.

Syntax:

The syntax of an if...else statement in C++ is:

```
if(boolean_expression)
{
    // statement(s) will execute if the boolean expression is true
}
else
{
    // statement(s) will execute if the boolean expression is false
}
```

If the boolean expression evaluates to true, then the if block of code will be executed, otherwise else block of code will be executed.

### Algorithm

Step 1: Start

Step 2: Declare variables a, b, c, D, x1, x2, rp and ip;

Step 3: Calculate discriminant

```

D←b*b-4*a*c
Step 4: if D≥0
    r1←(-b+√D)/2a
    r2←(-b-√D)/2a
    Display r1 and r2 as roots.
else if D=0
    r1←-b/2a
    r2←-b/2a
    Display r1 and r2 as roots.
else
    Display roots are imaginary

```

Step 5: Stop

## PROCEDURE/PROGRAM

```

-----
d=(b*b)-(4*a*c);
if(d>0)
{
cout<<"Two real and distinct roots";
    root1=(-b+sqrt(d))/(2*a);
    root2=(-b-sqrt(d))/(2*a);
cout<<"\nRoots are "<<root1<<" and "<<root2;
}
else
if(d==0)
{
cout<<"\nTwo real and equal roots";
    root1=root2=-b/(2*a);
cout<<"\nRoots are "<<root1<<" and "<<root2;
}
else
cout<<"\nRoots are COMPLEX and IMAGINARY....!!!";
-----

```

## RESULT- Sample Input and Output

### Input

Enter coefficients a, b and c:

4

5

1

### Output

Two real and distinct roots

Roots are -0.25 and -1

### **Similar Programs**

1. An electricity board charges the following rates to domestic users to discourage large consumption of energy:

For the first 100 units - 60P per unit

For next 200 units - 80P per unit

Beyond 300 units - 90P per unit

All users are charged a minimum of Rs.50.00. if the total amount is more than Rs.300.00 then an additional surcharge of 15% is added

Write a C++ program to read the names of users and number of units consumed and print out the charges with names

2. Develop a C++ program find the largest among three different numbers entered by user.

3. Using a C++ program check whether a student passed the exam or not based on total mark which shall be above 40%

## EXP NO -1b

### Use of switch statement and do while loop

**AIM-**Develop a C++ program to implement simple calculator

**OBJECTIVES-** Understand multiple branching in programming  
Use of exit controlled loop with switch

**TOOLS –** Hardware Requirement: Desk Top Computer

Software Requirement: Linux Operating System with gcc and g++, or eclipse

### THEORY/ALGORITHM

A switch statement allows a variable to be tested for equality against a list of values. Each value is called a case, and the variable being switched on is checked for each case.

Syntax:

The syntax for a switch statement in C++ is as follows:

```
switch(expression){  
    case constant-expression :  
        statement(s);  
        break; //optional  
    case constant-expression :  
        statement(s);  
        break; //optional  
    // you can have any number of case statements.  
    default : //Optional  
        statement(s);
```

The following rules apply to a switch statement:



The expression used in a switch statement must have an integral or enumerated type, or be of a class type in which the class has a single conversion function to an integral or enumerated type.

Any number of case statements can be there within a switch. Each case is followed by the value to be compared to and a colon.

The constant-expression for a case must be the same data type as the variable in the switch, and it must be a constant or a literal.

When the variable being switched on is equal to a case, the statements following that case will execute until a break statement is reached.

When a break statement is reached, the switch terminates, and the flow of control jumps to the next line following the switch statement.

Not every case needs to contain a break. If no break appears, the flow of control will fall through to subsequent cases until a break is reached.

A switch statement can have an optional default case, which must appear at the end of the switch. The default case can be used for performing a task when none of the cases is true. No break is needed in the default case.

A do...while loop is similar to a while loop, except that a do...while loop is guaranteed to execute at least one time.

Syntax:

The syntax of a do...while loop in C++ is:

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

Conditional expression appears at the end of the loop, so the statement(s) in the loop execute once before the condition is tested.

If the condition is true, the flow of control jumps back up to do, and the statement(s) in the loop execute again. This process repeats until the given condition becomes false.

### Algorithm

Step 1: Start  
Step 2: Read x and y values  
Step 3: Read option + or - or \* or / or %  
Step 4: If option is '+' res = x + y  
Step 5: If option is '-' res = x - y  
Step 6: If option is '\*' res = x \* y  
Step 7: If option is '/' res = x / y  
Step 8: If option is '%' res = x % y  
Step 9: If option does not match with + or - or \* or / or %  
    Print select option +, -, \*, /, %, % only  
Step 10: Print x, option, y, res values  
Step 11: Stop

### PROCEDURE/PROGRAM

```
-----  
do  
{  
cout<<"Please enter the operand of the problem you would like to solve:"<<endl;  
    cout<<" + for addition"<<endl;  
    cout<<" - for subtraction"<<endl;  
  
    cout<<" * for multiplication"<<endl;  
  
    cout<<" / for division"<<endl;  
  
    cout<<"Enter Q to quit"<<endl;  
  
    cout<<"Enter your choice ==> ";  
  
    cin>> operand;  
  
    cout<<"Please enter the two numbers ==> ";  
  
    cin>> x >> z;  
  
    switch (operand)  
    {  
        case '+':  
            result = x+z;  
            cout<<"The answer is: " << result <<endl;  
  
            break;  
        case '-':  
            result = x-z;  
            cout<<"The answer is: " << result <<endl;  
  
            break;  
        case '*':  
            result = x*z;
```

```

        cout<<"The answer is: " << result <<endl;
    break;
    case '/':
        if (z ==0 )
        {
            cout<<"That is an invalid operation" <<endl;
        }
        else
        {
            result = x/z;
            cout<<"The answer is: " << result <<endl;
        }
    break;
    default :
        cout<<"That is an invalid operation" <<endl;
    break;
}

} while (operand != 'Q');
-----

```

## RESULT- Sample Input and Output

### Input

Please enter the operand of the problem you would like to solve:

+ for addition

- for subtraction

\* for multiplication

/ for division

Enter Q to quit

Enter your choice ==> +

Please enter the two numbers 3      4

### Output

The answer is: 7

### Similar Programs

1. An election is contested by five candidates. The candidates are numbered 1 to 5 and a voting is done by marking the candidate number in a ballot paper. Write a C++ program to read the ballot and count the votes cast for each candidate using an array

variable count. In case, a number read is outside the range 1 to 5 the ballot should be considered as a 'spoilt ballot', and the program should also count the number of spoilt ballots.

2. Develop a C++ Program that reads marks obtained by a student in a test of 100 marks and computes his grade according to the following criteria.

Marks $\geq$ 80 grade=A

Marks $\geq$ 70 &  $<$ 80 grade=B

Marks $\geq$ 60 &  $<$ 70 grade=C

Marks $\geq$ 50 &  $<$ 60 grade=D

Otherwise grade=F

3. Write a C++ Program To Find Electricity Bill Of A Person with the following unit tariff

Unit tariff

>100 RS.1.20 per unit

>200 RS.2 per unit

>300 RS.3 per unit

## EXP NO-1 c

### Use of for statement

**AIM-**Develop a C++ program to find the Fibonacci series till the limit entered by the user

**OBJECTIVES-** Study the use of counter controlled loop

Understand the usage of for loop

**TOOLS –** Hardware Requirement: Desk Top Computer

Software Requirement: Linux Operating System with gcc and g++ or eclipse

### THEORY/ALGORITHM

A for loop is a repetition control structure that allows you to efficiently write a loop that needs to execute a specific number of times.

The syntax of a for loop in C++ is:

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

Here is the flow of control in a 'for' loop:

The init step is executed first, and only once. This step allows you to declare and initialize any loop control variables. It is not required to put a statement here, as long as a semicolon appears.

Next, the condition is evaluated. If it is true, the body of the loop is executed. If it is false, the body of the loop does not execute and flow of control jumps to the next statement just after the 'for' loop.

After the body of the 'for' loop executes, the flow of control jumps back up to the increment statement. This statement allows you to update any loop control variables. This statement can be left blank, as long as a semicolon appears after the condition.

The condition is now evaluated again. If it is true, the loop executes and the process repeats itself (body of loop, then increment step, and then again condition). After the condition becomes false, the 'for' loop terminates.

### Algorithm

Step 1: Start

Step 2: Declare variables a,b , c and limit

Step 3: Initialize variables  $a \leftarrow 0$ ,  $b \leftarrow 1$  ,  $c \leftarrow 0$

Step 4: Display a and b

Step 5: Repeat the steps until  $c \leq \text{limit}$

5.1:  $c \leftarrow a+b$

5.2:  $a \leftarrow b$

5.3:  $b \leftarrow c$

5.4: Display c

Step 6: Stop

### PROCEDURE/PROGRAM

```
-----  
int a=0,b=1,c=0  
-----  
for(int i=1;i<=n;i++)  
{  
    c=a+b;  
    a=b;  
    b=c;  
    cout<<c<<" ";  
}  
-----
```

### RESULT- Sample Input and Output

#### Input

Enter the limit

25

#### Output

0      1      1      2      3      5      8      13      21

## Similar Programs

1. Write a C++ program to print the following by reading number of rows to be printed from the user

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

2. Program to find nth term and sum of an A.P and G.P.
3. Develop a C++ program to find the sum of digits of a number

## EXP NO-1 d

### Use of while statement

**AIM-**Develop a C++ program to find reverse of a number

**OBJECTIVES-**      Understand entry controlled loop in oops

Study the usage of while statement

**TOOLS –**      Hardware Requirement: Desk Top Computer

Software Requirement: Linux Operating System with gcc and g++ or eclipse

### THEORY/ALGORITHM

A while loop statement repeatedly executes a target statement as long as a given condition is true.

#### Syntax of while Loop

```
while (test expression)
{
    statement/s to be executed.
}
```

Here, statement(s) may be a single statement or a block of statements. The test expression may be any expression, and true is any non-zero value. The loop iterates while the condition is true.

When the condition becomes false, program control passes to the line immediately following the loop.

#### Algorithm

step 1: Start  
step 2: Intilize reverse=0.  
step 3: Read digit  
step 4: Check whether digit>0 then go to step 5



else go to step 9  
step 5: reverse = reverse \* 10  
step 6: reverse = reverse + digit % 10  
step 7: digit = digit / 10  
step 8: Go to step 4  
step 9: Print reverse  
step 10: Stop

## PROCEDURE/PROGRAM

```
-----  
while(n != 0) {  
    int remainder = n%10;  
    reverse = reverse*10 + remainder;  
    n/=10;  
}  
  
cout<< "Reversed number = " << reverse;  
-----
```

## RESULT- Sample Input and Output

Input

Enter an integer:

124

Output

Reversed number = 421

## Similar Programs

1. Write a C++ program to check whether a number is perfect or not
2. Compute the LCM and GCD of two numbers using a C++ program
3. Using a C++ program display n multiples of a number

## EXP NO -2a

### Use of structure in C++

**AIM-** Write a C++ program to find average marks of three subjects of N students in a class

**OBJECTIVES-**

- Study the use of structures
- Understand array processing in C++
- Understand heterogeneous data types

**TOOLS –** Hardware Requirement: Desk Top Computer

Software Requirement: Linux Operating System with gcc and g++ or eclipse

#### **THEORY/ALGORITHM**

Structure is the collection of variables of different types under a single name for better visualization of problem.

```
struct [struct_name]
{
    type attribute;
    // ...
    [struct_name *struct_attribute;]
} [instance1, [instance2, ...]];
```

A struct declaration requires the keyword struct, optionally the name of the struct, and a body consisting of one or more attributes. It is possible to optionally include a self-referential pointer, but not possible to include a struct of type struct\_name (struct\_name struct\_attribute). If one or more structs is immediately desired, the entire declaration can be treated as a type, and the name of one or more instances may follow the declaration.

#### **Example**

```
struct person {
    char name[50];
    int age;
    float salary;};
```

### Algorithm

Step 1: Start

Step 2: Declare variables mark1, mark2, and mark3 as members of structure student

Step 3: Read value of n

Step 4: Repeat the steps until i=n

Step 5: Input mark1, mark2 and mark3

Step 6: calculate average= mark1+mark2 + mark3/3

Step 7: Display average

Step 8: Stop

### PROCEDURE/PROGRAM

```
-----  
struct student  
{  
int subject1 ;  
int subject2 ;  
int subject3 ;  
};  
inti , n, total;  
float av ;  
struct student st[20];  
cout<<" \n Enter the Number of Students : " ;  
cin>> n ;  
for (i =0; i<n; i++)  
{  
cout<<" \nEnter Marks of three Subjects of "<<i+1<<" Student : " ;  
total = 0 ;  
cin>>st[i].subject1 >>st[i].subject2>>st[i].subject3;  
total = st[i].subject1+st[i].subject2+st[i].subject3;  
av = (float) total /3 ;  
cout<<" \nAVERAGE Marks of " <<i+1<<" Student is = "<<av ;  
}  
-----
```

### RESULT- Sample Input and Output

#### Input

Enter the Number of Students 2

Enter Marks of three Subjects of 1 student

40 50 60

Enter Marks of three Subjects of 2 student

80 60 40

**Output**

AVERAGE Marks of 1 student  
50

AVERAGE Marks of 1 student  
60

**Similar programs**

1. Create a Structure called employee with the following details as variables within it.
  1. Name of the employee
  2. Age
  3. Designation
  4. SalaryWrite a C++ program to create array of objects for the structure to access these and print the name, age, designation and salary
2. Create a C++ program which takes two distances in inch-feet system and stores in data members of two structure variables. Then, this program calculates the sum of two distances and displays it.
3. Develop a C++ program in which user is asked to enter two time periods and these two periods are stored in structure variables. The program calculates the difference between these two time periods.

## EXP NO -2b

### Use of unions in C++

**AIM-**Write a C++ Program to find total salary of n employees in a department where da=50% basicpay and hra=10% of basicpay

**OBJECTIVES-** Study the use of union in C++  
Compare the usage of structure with union

**TOOLS –** Hardware Requirement: Desk Top Computer

Software Requirement: Linux Operating System with gcc and g++ or eclipse

### THEORY/ALGORITHM

Unions allow one portion of memory to be accessed as different data types. Its declaration and use is similar to the one of structures, but its functionality is totally different:

Syntax of union is

```
uniontype_name {  
    member_type1 member_name1;  
    member_type2 member_name2;  
    member_type3 member_name3;  
    .  
    .  
} object_names;
```

This creates a new union type, identified by type\_name, in which all its member elements occupy the same physical space in memory. The size of this type is the one of the largest member element

### Algorithm

Step 1: Start

Step 2: Declare variables deptid, name, basic pay as members of union

Step 3: Read value of n  
 Step 4: Repeat the steps until i=n  
 Step 5: Input name, deptid and basic pay  
 Step 6: calculate total salary=basicpay+da+hra  
 Step 7: Display total salary  
 Step 8:Stop

## PROCEDURE/PROGRAM

```

-----
union Department
{
intdeptid;
char name[20];

intbasic_pay
float total_salary
};

inti , n, total, da,hra;
union Department d[20];
cout<<" \n Enter the Number of employees : " ;
cin>> n ;
for (i =0; i<n; i++)
{
cout<<"\nEnter name of "<<i+1<<" employee : " ;
cin>>d[i].name;
cout<<"\nEnterdeptid "<<i+1<<" employee : " ;
cin>>d[i].deptid;
cout<<"\nEnter basic pay of "<<i+1<<" employee : " ;
cin>>d[i].basic_pay;
da=basic_pay*(50/100)
hra=basic_pay*(10/100)
total = basic_pay+da+hra;
cout<<"\ntotal salary " <<i+1<<" employee is = "<< total ;
}
-----
  
```

## RESULT- Sample Input and Output

### Input

Enter the Number of employees 2

Enter Enter name of 1 employee

Arun

Enter department idof 1 employee

101

Enter basic pay of 1 employee  
10000

Enter Enter name of 2 employee  
Anil  
Enter department id of 1 employee  
102  
Enter basic pay of 1 employee  
8000

### **Output**

Total salary of 1 employee  
16000

Total salary of 2 employee  
12800

### **Similar Programs**

1. Create a Union called student with the following details as variables within it.

1. Name of the student
2. Age
3. Year of study
4. Semester
5. 5 different subject marks in array

Write a C++ program to create object for the union to access these and print the Name, age, year, semester and grade according to their percentage of marks scored.

90 % and above – S grade

80% to 89% -- A grade

70% to 79% -- B grade

60% to 69% -- C grade

50% to 59% -- D grade

<50% -- F grade

2. Write a program for Library Management in C++. It contains all the basic transaction that occurred in Library day to day life. Create a union to represent library book details.

3. Define a union called Rectangle which keeps track of the length and breadth of a rectangle.

Write functions namely input, displayDimensions, displayArea and edit to input the dimensions of a rectangle, to display the dimensions, to calculate and display the area of a triangle respectively. Write a main function which defines a variable of type Rectangle.

Invoke these functions in main and observe the result.

## EXP NO-3

### Pointer Arithmetic

**AIM-** To write a program in C++ to implement classes with pointers as data member

**OBJECTIVES-**      Study the pointer arithmetic

Understand the manipulation on pointers

**TOOLS –**              Hardware Requirement: Desk Top Computer

Software Requirement: Linux Operating System with gcc and g++

#### **THEORY/ALGORITHM**

A pointer is a derived data type that refers to another data variable by storing its address. It defines where to get value of a specific data variable instead of defining actual data

Pointer arithmetic

1. A pointer can be incremented or decremented
2. Any integer can be added to or subtracted from a pointer
3. One pointer can be subtracted from another

#### **Algorithm**

Step 1: Start

Step 2: create an array with sample values

Step 3: Initialize base address of array to a pointer ptr

Step 4: Perform addition, increment, subtraction and decrement operations on ptr

Step 5: Display the values

Step 6: Stop

#### **PROCEDURE/PROGRAM**

```
-----  
intnum[]={ 56,75,22,18,90};  
int *ptr;  
-----  
cout<< "value of ptr"<<*ptr;  
ptr++;
```



```
cout<< "value of ptr++"<<*ptr;
ptr--;
cout<< "value of ptr--"<<*ptr;
ptr=ptr+2;
cout<< "value of ptr+2"<<*ptr;
ptr=ptr-1;
cout<< "value of ptr-1"<<*ptr;
-----
```

### **RESULT- Sample Input and Output**

```
value of ptr:66
value of ptr++:75
value of ptr--:56
value of ptr+2:22
value of ptr-1:75
```

### **Similar Programs**

1. Write a C++ program to find the number of vowels present in the given character array using pointer arithmetic.
2. Implement a C++ Program to Find Length of the String using Pointer without using library functions
3. Develop C++ program to find the sum of elements of an array using pointers

## EXP NO-4a

### Use of recursive functions in C++

**AIM-** To write a program in C++ to implement recursion in finding the factorial

**OBJECTIVES-** Study the use of recursion in C++ programming  
Understand the usage of local variable in function

**TOOLS –** Hardware Requirement: Desk Top Computer

Software Requirement: Linux Operating System with gcc and g++ or eclipse

### THEORY/ALGORITHM

It is possible to call a function from a same function. This function is known as recursive function and this programming technique is known as recursion.

In recursion, a function calls itself but these two functions are not the same function. They are different functions although they have same name.

Local variables are variables defined inside a function and has scope only inside that function. In recursion, a function call itself but these two functions are different functions (Imagine these functions are function1 and function 2.) The local variables inside function1 and function2 are also different and can only be accessed within that function

#### Algorithm

Step 1: Start

Step 2: Read n value as integer

Step 3: Call function factorial (int n)

Step 4: End

Call function factorial(int n)

begin

if (n = 0)

return 1;

else

```
        return (n * factorial(n - 1));  
    end
```

## PROCEDURE/PROGRAM

```
-----  
fact(int x)  
{  
    int f;  
    if(x==1)  
        return(x);  
    else  
    {  
        f=x*fact(x-1);  
        return(f);  
    }  
}  
-----
```

## RESULT- Sample Input and Output

### Input

Enter a number to find factorial: 4

### Output

Factorial of 4 = 24

## Similar Programs

1. With the help of a C++ program find the sum of first n natural numbers using recursion
2. Develop a C++ program to produce the Fibonacci number for a given index in the series using recursion
3. Implement a C++ recursive function to determine if an input is prime:

## EXP NO-4b

### Use of Functions in C++

**AIM-**Write a program in C++ to prepare a student Record using class and object with functions getdata, compute and display for getting marks, computing total and displaying results

**OBJECTIVES-** Study the use of functions

Understand the concept of class and object

**S –** Hardware Requirement: Desk Top Computer

Software Requirement: Linux Operating System with gcc and g++ or eclipse

#### THEORY/ALGORITHM

C++ allows programmer to define their own function. A user-defined function groups code to perform a specific task and that group of code is given a name. When that function is invoked from any part of program, it all executes the codes defined in the body of function. It is represented as

```
#include <iostream>

void function_name() {
    ... ..
    ... ..
}

int main() {
    ... ..
    function_name();
    ... ..
}
```

#### Algorithm

1. Start
2. Create a class record.
3. Read the name, Regno ,mark1,mark2,mark3 of the student.
4. Calculate the average of mark as  $Avg = \frac{mark1 + mark2 + mark3}{3}$

5. Display the student record.
6. Stop the program.

### PROCEDURE/PROGRAM

```
-----  
void getdata()  
{  
    cout<<"\nenter the name: " ;  
    cin>>name;  
    cout<<"enter the regno: ";  
    cin>>regno;  
    cout<<"enter the m1,m2,m3: \n";  
    cin>>m1>>m2>>m3;  
}  
void calculate()  
{  
    avg=(m1+m2+m3)/3;  
}  
void display()  
{  
    cout<<"*****\n";  
    cout<<"\nName: "<<name;  
    cout<<"\nRegno: "<<regno;  
    cout<<"\nMark1: "<<m1;  
    cout<<"\nMark2: "<<m2;  
    cout<<"\nMark3: "<<m3;  
    cout<<"\nAvg: "<<avg;  
    cout<<"*****\n";  
}  
-----  
void main()  
{  
    -----  
    record r;  
    r.getdata();  
    r.calculate();  
    r.display();  
    -----  
}
```

### RESULT- Sample Input and Output

#### Input

Enter the name: Avanthika  
Enter the reg no: 1  
Enter the m1,m2,m3: 90,90,90

#### Output

Name: Avantika

Regno: 1 Mark1: 90 Mark2: 90 Mark3: 90  
Average:90

### **Similar Programs**

1. Write a C++ program to print the given number in reverse order. Use functions with return type and without return type for reversing the number  
Ex: given number is 2345 , output should be 5432
2. Write a C++ Program to implement a sphere class with appropriate members and member function to find the surface area and the volume. (Surface =  $4 \pi r^2$  and Volume =  $\frac{4}{3} \pi r^3$  )
3. Write a C++ program to implement Bank-SB-Account Class with member functions to deposit, withdraw and show the balance. assume appropriate data members

## EXP NO-5

### Use of inline function

**AIM-**Develop a C++ program write a program to find the multiplication values and the cubic values using inline function.

**OBJECTIVES-** Study the use of inline functions  
Compare inline and non inline functions

**TOOLS –** Hardware Requirement: Desk Top Computer

Software Requirement: Linux Operating System with gcc and g++ or eclipse

### THEORY/ALGORITHM

When inline functions are used, the overhead of function call is eliminated. Instead, the executable statements of the function are copied at the place of each function call. This is done by the compiler. At the time of declaration or definition, function name is preceded by word inline

Example

```
inline int sqr(int x)
{
    int y;
    y = x * x;
    return y;
}
```

#### Algorithm

- Step 1: Start the program.
- Step 2: Declare the class.
- Step 3: Declare and define the inline function for multiplication and cube.
- Step 4: Declare the class object and variables.
- Step 5: Read two values.
- Step 6: Call the multiplication and cubic functions using class objects.
- Step 7: Return the values.
- Step 8: Display.
- Step 9: Stop the program.

## PROCEDURE/PROGRAM

```
-----  
inline float mul(float x,float y)  
    {  
        return(x*y);  
    }  
-----  
    inline float cube(float x)  
    {  
        return(x*x*x);  
    }  
-----
```

## RESULT- Sample Input and Output

### Input

Enter two values: 5 7

### Output

Multiplication Value is: 35

Cube Value is: 25 and 343

### Similar Programs

1. Write a C++ program to perform different arithmetic operation such as addition, subtraction, division, modulus and multiplication using inline function
2. Develop a C++ program to find greatest among two numbers using inline function
3. Implement a C++ program to find the area of a rectangle using an inline function defined outside the class. Area=length\*breadth



## EXP NO-6a

### Implementation of Call by value

**AIM-** write a C++ program to find the value of a number raised to its power that demonstrates a function using call by value

**OBJECTIVES-** Study the use of function call mechanism  
Understand call by value method of parameter passing

**TOOLS –** Hardware Requirement: Desk Top Computer

Software Requirement: Linux Operating System with gcc and g++ or eclipse

### THEORY/ALGORITHM

The call by value method of passing arguments to a function copies the actual value of an argument into the formal parameter of the function. In this case, changes made to the parameter inside the function have no effect on the argument.

#### Algorithm

- 1) Start the program.
- 2) Declare the variables.
- 3) Get two numbers as input
- 4) Call the function power to which a copy of the two variables is passed.
- 5) Inside the function, calculate the value of x raised to power y and store it in p.
- 6) Return the value of p to the main function.
- 7) Display the result.
- 8) Stop the program

### PROCEDURE/PROGRAM

```
-----  
double power(intx,int y)  
{  
double p; p=1.0;  
if(y>=0)  
while(y--)  
p*=x;
```

```
else
while(y++)
p/=x;
return(p);
}
```

-----

### **RESULT- Sample Input and Output**

Input

ENTER X , Y:

2

3

**Output**

2 TO THE POWER 3 IS 8

### **Similar Programs**

1. Write a C++ program to swap two number using call by value mechanism
2. Develop a C++ program to increment each element of a matrix using call by value mechanism
3. Implement a function reverse to print the elements in the reverse order of the given elements in the array by call by value

## EXP NO-6b

### Implementation of Call by Reference

**AIM-** To write a C++ program to swap two values by implementing the concept of call by address

**OBJECTIVES-** Study the use of function call mechanism  
Understand Pass by reference method in C++ programming

**TOOLS –** Hardware Requirement: Desk Top Computer

Software Requirement: Linux Operating System with gcc and g++ or eclipse

### THEORY/ALGORITHM

The call by reference method of passing arguments to a function copies the reference of an argument into the formal parameter. Inside the function, the reference is used to access the actual argument used in the call. This means that changes made to the parameter affect the passed argument.

#### Algorithm

1. Start the program
2. Include suitable header file
3. Declare a function swap with two pointer variables arguments
4. Declare and initialize the value as two variable in main()
5. Print the value of two variable before swapping
6. Call the swap function by passing address of the two variable as arguments
7. Print the value of two variable after swapping
8. Stop the program

### PROCEDURE/PROGRAM

```
-----  
void swap(int *x,int*y)  
{  
    int temp=*x;  
    *x=*y;
```

```
        *y=temp;  
    }  
    -----
```

### **RESULT-** Sample Input and Output

The value of i before swapping is: 20  
The value of j before swapping is: 10  
The value of i after swapping is: 10  
The value of j after swapping is: 20

### **Similar Programs**

1. Develop a C++ program to find the largest of three numbers using call by reference mechanism
2. Implement a C++ program with function which given an integer returns 3 times that integer. Use call by reference
3. Perform basic string manipulation operations like concatenation, reverse, and palindrome checking using a C++ program. Use call by reference mechanism

## EXP NO-7

### Use of storage specifiers in C++

**AIM-** To write a program in C++ to implement storage specifier to increment the value of a variable to specific amounts which is declared as global and static

**OBJECTIVES-** Study the use of storage specifiers  
Familiarise with global and static variables

**TOOLS –** Hardware Requirement: Desk Top Computer

Software Requirement: Linux Operating System with gcc and g++ or eclipse

### THEORY/ALGORITHM

There are 4 types of storage class:

- automatic
- external
- static
- register

Variables declared inside the function body are automatic by default. These variables are also known as local variables as they are local to the function and don't have meaning outside that function.

External variables can be accessed by any function. They are also known as global variables. Variables declared outside every function are external variables.

Register variables are similar to automatic variables and exist inside that particular function only.

The value of a static variable persists until the end of the program. A variable can be declared static using the keyword: static.

## PROCEDURE/PROGRAM

### Global variable

```
void Check();
int a=5;

int main(){
    a+=4;
    Check();
    return 0;
}

void Check(){
    ++a;
    printf("a=%d\n",a);
}
```

-----

## RESULT- Sample Input and Output

### Output

a=10

### Static variable

```
-----
void Check();

int main(){
    Check();
    Check();
    Check();
}

void Check(){
    static int c=0;
    printf("%d\t",c);
    c+=5;
}
```

-----

## RESULT- Sample Input and Output

### Output

0    5    10

### **Similar Programs**

1. Write a C++ program to demonstrate the static and non static variable usage defining them within a function to count the number of odd numbers in an array
2. Write a C++ program to demonstrate the global and non global variable usage for searching an item in an array
3. Write a C++ program to demonstrate the use of local variable by defining a function to calculate sum of factors of a number

## EXP NO-8

### Use of Constructor and Destructor in C++

**AIM-** Develop a C++ program to check whether a number is prime or not

**OBJECTIVES-**      Study the use of constructor  
                         Understand the working of destructor

**TOOLS –**              Hardware Requirement: Desk Top Computer

Software Requirement: Linux Operating System with gcc and g++ or eclipse

#### **THEORY/ALGORITHM**

Constructors are special class functions which performs initialization of every object. The Compiler calls the Constructor whenever an object is created. Constructors initialize values to object members after storage is allocated to the object.

```
class A
{
int x;
public:
A(); //Constructor
};
```

Constructors can be defined either inside the class definition or outside class definition using class name and scope resolution :: operator.

```
class A
{
inti;
public:
A(); //Constructor declared
};
```

```
A::A() // Constructor definition
{
```



```
i=1;  
}
```

Destructor is a special class function which destroys the object as soon as the scope of object ends. The destructor is called automatically by the compiler when the object goes out of scope.

The syntax for destructor is same as that for the constructor, the class name is used for the name of destructor, with a tilde ~ sign as prefix to it.

```
class A  
{  
public:  
~A();  
};
```

### **Algorithm**

STEP 1: Start the program.

STEP 2: Declare the class as Prime with data members, Member functions.

STEP 3: Consider the argument constructor Prime() with integer Argument.

STEP 4: To call the function calculate() and do the following steps.

STEP 5: For i=2 to a/2 do

STEP 6: Check if  $a \% i == 0$  then set  $k=0$  and break.

STEP 7: Else set k value as 1.

STEP 8: Increment the value i as 1.

STEP 9: Check whether the k value is 1 or 0.

STEP 10: If it is 1 then display the value is a prime number.

STEP 11: Else display the value is not prime.

STEP 12: Stop the program.

## 6. PROCEDURE/PROGRAM

```
-----  
class prime  
{  
public:  
  
prime(int x)  
{  
a=x;  
  
k=1;  
{  
for(i=2;i<=a/2;i++)  
if(a%i==0)  
{
```

```

k=0;
break;
}
else
{
k=1;
}
}
}
}
~prime();
-----

```

## RESULT-

### Sample Input and Output

#### Input

Enter the number: 7

#### Output

Given number is Prime Number

#### Input

Enter the number: 6

#### Output

Given number is not Prime

### Similar Programs

1. Create a class for counting the number of objects created and destroyed within various block using constructor and destructors
2. Write a program to calculate gross and net pay of employee from basic salary. Create employee class which consists of employee name, emp\_id, and basic salary as its data members. Use parameterized constructor in the derived class to initialize data members of the base class and calculate gross and net pay of the employee in the derived class.
3. Define a class to represent a Bank account. Include the following members.  
 Data members:-  
 Name of the depositor  
 Account number.  
 Type of account.  
 Balance amount in the account.  
 Rate of interest (static data)  
 Provide a default constructor, a parameterized constructor and a copy constructor to this class.

Also provide Member Functions:-

1. To deposit amount.
2. To withdraw amount after checking for minimum balance.
3. To display all the details of an account holder.
4. Display rate of interest (a static function)

Illustrate all the constructors as well as all the methods by defining objects.

## EXP NO -9

### Use of this pointer

**AIM-** To write a program in C++ to implement classes to compare the volumes of two boxes with this pointer

**OBJECTIVES-** Study the concept and use of this pointer  
Understand reference to an object concept

**TOOLS –** Hardware Requirement: Desk Top Computer

Software Requirement: Linux Operating System with gcc and g++ or eclipse

### THEORY/ALGORITHM

this pointer is a constant pointer that holds the memory address of the current object.  
this pointer is an implicit parameter to all member functions. Therefore, inside a member function, this may be used to refer to the invoking object.

Friend functions do not have this pointer, because friends are not members of a class. Only member functions have this pointer.

#### Algorithm

Step 1: Start

Step 2: Define the class box with length, breadth, height as member and functions to input, calculate volume and compare volume

Step 3: Input the two set of values

Step 4: Compare the values and return this pointer if member variable is the largest

Step 5: Stop

### PROCEDURE/PROGRAM

```
-----  
class Box  
{  
int length, breadth, height;
```

```

public:
    -----
    Box(int l, int b, int h)
    {
        length = l;
        breadth = b;
        height = h;
    }
    int Volume()
    {
        return length * breadth * height;
    }

    int compare(Box box)
    {
        return this->Volume() > box.Volume();
    }

};

void main(void)
{
    -----
    cout<<" Enter the length, breadth, height of box1 " ;
    cin>> l1>>b1>>h1;
    Box Box1(l1,b1, h1);
    cout<<" Enter the length, breadth, height of box2" ;
    cin>> l2>>b2>>h2;

    Box Box2(l2,b2,h2);
    if(Box1.compare(Box2))
    {
        cout<< "Box2 is smaller than Box1" <<endl;
    }
    else
    {
        cout<< "Box2 is equal to or larger than Box1" <<endl;
    }
    -----
}

```

## RESULT- Sample Input and Output

### Input

Enter the length, breadth, height of box1  
10,24,36

Enter the length, breadth, height of box1  
20,8,6

### **Output**

Box2 is smaller than Box1

### **Similar Programs**

1. Write a C++ program to create three objects for a class named `pntr_obj` with data members such as `roll_no` & `name`. Create a member function `set_data()` for setting the data values and `print()` member function to print which object has invoked it using 'this' pointer
2. Develop a C++ program to find the greatest of two numbers using this pointer which returns the member variable
3. Write a C++ program to implement flight class with data member as flight no., source, destination and fare. Write a member function to display the flight information using this pointer

## EXP NO-10 a

### Use of Multiple Inheritance

**AIM-** Develop a C++ program write a program to find out the student details including total marks using multiple inheritance

**OBJECTIVES-** Study the use of multiple inheritance.  
Understand the different access specifiers in inheritance

**TOOLS –** Hardware Requirement: Desk Top Computer  
Software Requirement: Linux Operating System with gcc and g++ or eclipse

### THEORY/ALGORITHM

Inheritance allows to define a class in terms of another class, which makes it easier to create and maintain an application. This also provides an opportunity to reuse the code functionality and fast implementation time.

When creating a class, instead of writing completely new data members and member functions, the programmer can designate that the new class should inherit the members of an existing class. This existing class is called the base class, and the new class is referred to as the derived class

In C++, we have 5 different types of Inheritance. Namely,

- Single Inheritance
- Multiple Inheritance
- Hierarchical Inheritance
- Multilevel Inheritance
- Hybrid Inheritance (also known as Virtual Inheritance)

#### **Multiple Inheritance in C++**

Multiple Inheritance is a feature of C++ where a class can inherit from more than one classes.

#### **Algorithm**

- Step 1: Start the program.
- Step 2: Declare the base class student.
- Step 3: Declare and define the function get() to get the student details.
- Step 4: Declare the other class sports.
- Step 5: Declare and define the function getsm() to read the sports mark.

Step 6: Create the class statement derived from student and sports.

Step 7: Declare and define the function display() to find out the total and average.

Step 8: Declare the derived class object, call the functions get(), getsm() and display().

Step 9: Stop the program.

## PROCEDURE/PROGRAM

```
-----
class student
{
    protected:
int rno,m1,m2;
public:
    void get()

-----

};

class sports
{
    protected:
intsm;
public:
    void getsm()

-----

};
class statement:publicstudent,public sports
{
inttot,avg;
void display()
    {
        tot=(m1+m2+sm);
-----
    };
void main()
{
    -----
    statement obj;
    obj.get();
    obj.getsm();
    obj.display();
-----
}
```

## RESULT- Sample Input and Output

### Input



Enter the Roll no: 100

Enter two marks

90

80

Enter the Sports Mark: 90

### **Output**

Roll No: 100

Total : 260

Average: 86.66

### **Similar Experiments**

1. Write a program to enter salary and output income tax and net salary using multiple inheritance concept
2. Write a C++ program to demonstrate multiple inheritance by creating a class cuboid which extends class rectangle, class shape. It calculates area and volume. Use appropriate constructors and member variables.
3. Implement a program of maintaining banking account information system using multiple inheritance in C++ Programming. Here class savings derived from class account and class user. Use appropriate functions and variables

## EXP NO-10 b

### Virtual function implementation by function overriding

**AIM-**Write a C++ program to explain virtual function (polymorphism) by creating a base class c\_polygon which has virtual function area(). Two classes c\_rectangle and c\_traingle derived from c\_polygon and they have area() to calculate and return the area of rectangle and triangle respectively.

<b>OBJECTIVES-</b>	Study the use of Virtua functions Understand the concept of function overriding
--------------------	--

**TOOLS –** Hardware Requirement: Desk Top Computer

Software Requirement: Linux Operating System with gcc and g++ or eclipse

### THEORY/ALGORITHM

If there are member functions with same name in base class and derived class, virtual functions gives programmer capability to call member function of different class by a same function call depending upon different context. This feature in C++ programming is known as polymorphism which is one of the important feature of OOP.

A virtual function is a member function that is declared within a base class and redefined by a derived class. To create virtual function, precede the function's declaration in the base class with the keyword virtual. When a class containing virtual function is inherited, the derived class redefines the virtual function to suit its own needs.

If a base class and derived class has same function if code written to access that function using pointer of base class then, the function in the base class is executed even if, the object of derived class is referenced with that pointer variable

#### Algorithm

- Step 1: Start the program.
- Step 2: Declare the base class c\_polygon
- Step 3: Declare and define the virtual function area()
- Step 4: Declare and define the function getdata().
- Step 5: Create the derived class from the base class.

Step 6: Declare and define the function area()  
Step 7: Create the base class object and pointer variable.  
Step 8: Call the function area using the base class object and pointer.  
Step 9: Create the derived class object and call the function area using the derived class object and pointer.  
Step 10: Stop the program.

## PROCEDURE/PROGRAM

```
-----  
class c_polygon  
{  
-----  
float a,b;  
public:  
void get_data()  
{  
cout<<"\nEnter any two floating values:\n";  
cin>>a>>b;  
}  
virtual float area()  
{  
return 0;  
}  
};  
class c_rectangle:public c_polygon  
{  
public:  
float area()  
{  
return (a*b);  
}  
};  
class c_triangle:public c_polygon  
{  
public:  
float area()  
{  
return (b*a/2);  
}  
};  
void main()  
{  
-----  
c_rectangle r;  
c_triangle t;
```

```

c_polygon *p;
p=&r;
p->get_data();
cout<<"\nArea of rectangle is "<<p->area();
p=&t;
p->get_data();
cout<<"\nArea of triangle is "<<p->area();
-----
}

```

## RESULT- Sample Input and Output

### Input

Enter any two floating values

10

12

### Output

Area of rectangle is

120

Area of triangle is

60

## Similar Programs

1. Write a program to calculate bonus of the employees. The class master derives the information from both admin and account classes which derives information from class person. Create base and all derived classes having same member functions called getdata, display data and bonus. Create a base class pointer that capable of accessing data of any class and calculates bonus of the specified employee. Use virtual functions
2. Write a program to count the words and characters in given text using virtual function
3. To implement a C++ program to add and subtract two numbers using pure virtual function operation() with two derived classes add and sub from base class

## EXP NO-11 a

### Unary operator overloading

**AIM-**Develop a C++ program to increment and decrement complex numbers using unary operator overloading.

**OBJECTIVES-**

- Study the use of operator overloading
- Understand unary operator used member function in overloading

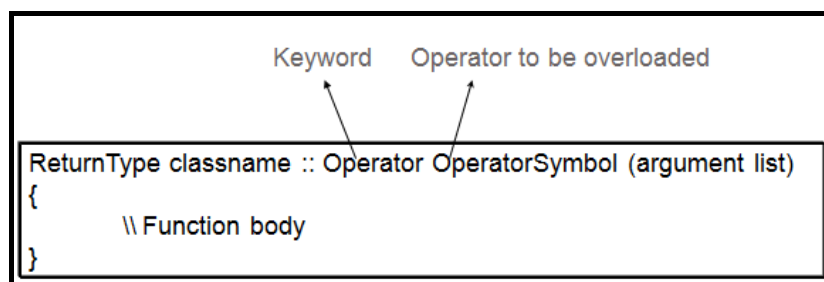
**TOOLS –** Hardware Requirement: Desk Top Computer

Software Requirement: Linux Operating System with gcc and g++ or eclipse

### THEORY/ALGORITHM

It is a type of polymorphism in which an operator is overloaded to give user defined meaning to it. Overloaded operator is used to perform operation on user-defined data type. For example '+' operator can be overloaded to perform addition on various data types, like for Integer, String (concatenation) etc.

Operator overloading syntax



#### Algorithm

- Step 1: Start the program.
- Step 2: Declare the class.
- Step 3: Declare the variables and its member function.
- Step 4: Using the function `getvalue()` to get the two numbers.

Step 5: Define the function operator ++ to increment the values  
Step 6: Define the function operator - -to decrement the values.  
Step 7: Define the display function.  
Step 8: Declare the class object.  
Step 9: Call the function getvalue  
Step 10: Call the function operator ++() by incrementing the class object and call the function display.  
Step 11: Call the function operator - -() by decrementing the class object and call the function display.  
Step 12: Stop the program.

## PROCEDURE/PROGRAM

```
-----  
class complex  
{  
inta,b,c;  
-----  
void operator++()  
{  
    a=++a;  
    b=++b;  
}  
  
void operator--()  
{  
    a=--a;  
    b=--b;  
}  
-----  
void main()  
{  
    -----  
    complex obj;  
    obj.getvalue();  
    obj++;  
    -----  
    obj.display();  
    obj--;  
    -----  
}
```

## RESULT- Sample Input and Output

### Input

Enter the two numbers: 3 6

### Output

Increment Complex Number

$4 + 7i$

Decrement Complex Number

$3 + 6i$

### Similar Programs

1. Write a C++ program to count the number of persons inside a bank, by increasing count whenever a person enters a bank, using an increment(++) operator overloading function, and decrease the count whenever a person leaves the bank using a decrement(--) operator overloading function inside a class
2. Write a C++ program to implement matrix class. Overload operator ~ to find the transpose of the matrix.
3. Implement a C++ program to overload unary minus operator to find the negation of a vector.

## EXP NO-11 b

### Binary operator overloading

**AIM-**Develop a C++ program write a program to add two complex numbers using binary operator overloading.

#### OBJECTIVES-

Study the use of binary operator overloading

Understand the use of non-member function in overloading

#### TOOLS –

Hardware Requirement: Desk Top Computer

Software Requirement: Linux Operating System with gcc and g++ or eclipse

#### THEORY/ALGORITHM

There are two types of operator overloading:

Unary operator overloading

Binary operator overloading

Whenever a unary operator is used, it works with one operand, therefore with the user defined data types, the operand becomes the caller and hence no arguments are required.

Whenever a binary operator is used - it works with two operands, therefore with the user defined data types - the first operand becomes the operator overloaded function caller and the second is passed as an argument

#### Algorithm

Step 1: Start the program.

Step 2: Declare the class.

Step 3: Declare the variables and its member function.

Step 4: Using the function getvalue() to get the two numbers.

Step 5: Define the non-member function operator + () to add two complex numbers.

Step 6: Define the non member function operator – () to subtract two complex numbers.

Step 7: Define the display function.

Step 8: Declare the class objects obj1, obj2 and result.



Step 9: Call the function getvalue using obj1 and obj2  
 Step 10: Calculate the value for the object result by calling the function operator + and operator -.  
 Step 11: Call the display function using obj1 and obj2 and result.  
 Step 12: Return the values.  
 Step 13: Stop the program.

## PROCEDURE/PROGRAM

```

-----
class complex
{
inta,b;
    public:
        void getvalue()
{
-----
void display()
    {
-----
void main()
{
complex obj1,obj2,result,result1;
-----
complex operator+(complex ob1, complex ob2)
    {
        complex t;
t.a=obj1.a+ob2.a;
t.b=obj1.b+ob2.b;
        return(t);
    }
complex operator-( complex ob1, complex ob2)
    {
        complex t;
t.a=obj1.a-ob2.a;
t.b=obj1.b-ob2.b;
        return(t);
    }

    result = obj1+obj2;
    result1=obj1-obj2;
-----
    obj1.display();
    obj2.display();
-----

```

}

### RESULT- Sample Input and Output

#### Input

Enter the value of Complex Numbers a, b

4            5

Enter the value of Complex Numbers a, b

2            2

#### Output

6 + 7i

2 + 3i

### Similar Programs

1. Write a C++ program to create two objects of a class called company and add their data members using an operator overloaded function for '+' operator and '-' operator
2. Write a program to concatenate two strings and comparing it by overloading '+' & '<=' operators respectively
3. Write a C++ program to implement time class that has separate data members for hours, minutes and seconds. Overload + Operator to add two times (object) and ++ operator to increment the time by one second.

## EXP NO - 12a

### Use of friend function

**AIM-**Develop a C++ program to find the mean value of given numbers

**OBJECTIVES-** Study the use of friend function  
Understand parameter passing mechanism of friend function

**TOOLS –** Hardware Requirement: Desk Top Computer

Software Requirement: Linux Operating System with gcc and g++ or eclipse

### THEORY/ALGORITHM

A friend function of a class is defined outside that class' scope but it has the right to access all private and protected members of the class. Even though the prototypes for friend functions appear in the class definition, friends are not member functions.

To declare a function as a friend of a class, precede the function prototype in the class definition with keyword friend

#### Algorithm

- STEP 1: Start the program.
- STEP 2: Declare the class name as Base with data members and member functions.
- STEP 3: The function get() is used to read the 2 inputs from the user.
- STEP 4: Declare the friend function mean(base ob) inside the class.
- STEP 5: Outside the class to define the friend function and do the following.
- STEP 6: Return the mean value (ob.val1+ob.val2)/2 as a float.
- STEP 7: Stop the program.

### PROCEDURE/PROGRAM

```
-----  
class base  
{  
int val1,val2;  
public:
```

```

    void get()
    {
    -----
    friend float mean(base ob);
    -----
    float mean(base ob)
    {
        return float(ob.val1+ob.val2)/2;
    }

    void main()
    {
    -----
        base obj;
        obj.get();
        cout<<"\n Mean value is : "<<mean(obj);
    -----

```

### RESULT- Sample Input and Output

#### Input

Enter two values: 10, 20

#### Output

Mean Value is: 15

### Similar Programs

1. Write a program to accept five different numbers by creating a class called friendfunc1 and friendfunc2 taking 2 and 3 arg respectively and calculate the average of these numbers by passing object of the class to friend function.
2. Create a class 'COMPLEX' to hold a complex number. Write a friend function to add two complex numbers. Write a main function to add two COMPLEX objects.
3. Create a C++ program to overload '!' operator using friend function

## EXP NO-12 b

### Use friend class

**AIM-** Develop a C++ program to find the area of a rectangle by converting the member of a class square which is a friend class of rectangle. Declare Rectangle as a friend of Square so that Rectangle member functions could have access to the private member of square

<b>OBJECTIVES-</b>	Study the use of friend class in C++ Understand the advantages of friend class
--------------------	---

**TOOLS –** Hardware Requirement: Desk Top Computer

Software Requirement: Linux Operating System with gcc and g++ or eclipse

### THEORY/ALGORITHM

A friend class is a class whose members have access to the private or protected members of another class

A class can be made a friend of another class using keyword friend. For example:

```
.....  
class A{  
    friend class B;    // class B is a friend class  
    .....  
}  
class B{  
    .....  
}
```

#### Algorithm

STEP 1: Start the program.

STEP 2: Declare the classes Rectangle and Square with necessary data members

STEP 3: Declare Rectangle as a friend of Square so that Rectangle member functions could have access to the private member of square

STEP 4: Convert the side value of square to height and width of rectangle using convert function.

STEP 5: Input the side value from the user

STEP 6: Display the area of rectangle

STEP 7: Stop the program.

## PROCEDURE/PROGRAM

```
class Square;
class Rectangle {
int width, height;
public:
int area ()
    {return (width * height);}
void convert (Square a);
};

class Square {
    friend class Rectangle;
private:
int side;
public:
    Square (int a) : side(a) {}
};

void Rectangle::convert (Square a) {
    width = a.side;
    height = a.side;
}

int main () {
    -----
int n;
    Rectangle rect;
cout<<" Enter the value of the side : " ;
cin>> n ;
    Square sqr (n);
rect.convert(sqr);
cout<<rect.area();
    -----
}
```

## RESULT- Sample Input and Output

### Input

Enter the value of the side

6

### Output

**Similar Programs**

1. Write a program to accept the student detail such as name and 3 different marks by `get_data()` method and display the name and average of marks using `display()` method. Define a friend class for calculating the average of marks using the method `mark_avg()`.
2. Develop a C++ Program to Implement Friend Class for Adding two Numbers where class `sum` is declared as friend of other class used to read values
3. Create two classes `Employee` and `Department`. Make `Department` class, a friend class of `Employee` class. In order to access the private and protected members of `Employee` class into `Department` class explicitly pass an object of `Department` class to the member functions of `Employee` class. Display the net salary of employee.

## EXP NO-13

### Class Templates

**AIM-** Write a program create a template T for a class named pair having two data members of type T which are inputted by a constructor and a member function get-max() return the greatest of two numbers to main.

**OBJECTIVES-** Study the concept of class templates  
Understand the advantages of templates

**TOOLS –** Hardware Requirement: Desk Top Computer

Software Requirement: Linux Operating System with gcc and g++ or eclipse

#### THEORY/ALGORITHM

Templates in C++ programming allows function or class to work on more than one data type at once without writing different codes for different data types. Templates are often used in larger programs for the purpose of code reusability and flexibility of program.

Class templates are associated with the generic types and they can use some specific types as well. But all objects must be of some specific size, so before creating an object of the template class, the data-type must be specified. This is done by specifying the data-type as parameter when the object is created for the template class.

Template classes are a great help for creating applications with generic types, which are common applications such as linked list, stack, and queues etc.

#### Algorithm

STEP 1: Start the program.

STEP 2: Declare the template class T

STEP 3: Using the member functions initialise value pair and find the largest

STEP 4: Create object corresponding to template

STEP 5: Display result

STEP 6: Stop the program.



## PROCEDURE/PROGRAM

```
template <class T>
class mypair {
    T a, b;
public:
    mypair (T first, T second)
        {a=first; b=second;}
    T getmax ();
};
```

```
template <class T>
T mypair<T>::getmax ()
{
    T retval;
    retval = a>b? a : b;
    return retval;
}
```

```
int main () {
    -----
    int n1, n2;
    cout<<" Enter the values " ;
    cin>> n1>>n2 ;
    -----
    mypair<int>myobject (n1,n2);
    cout<<myobject.getmax();
    -----
}
```

## **RESULT- Sample Input and Output**

### **Input**

Enter the values

**12**

**4**

### **Output**

**12**

## **Similar Programs**

1. Write a program to search a key element in a given set of elements using class template.
2. Write a class template to represent a generic vector. Include member functions to perform the following tasks:
  - 1) To create the vector.
  - 2) To modify the value of a given element.
  - 3) To multiply the vector by a scalar value.
  - 4) To display the vector in the form (10, 20, 30,.....)
3. Write a C++ program to find the read two data items and find the sum using class template

## APPENDIX

### C++ Development using eclipse IDE

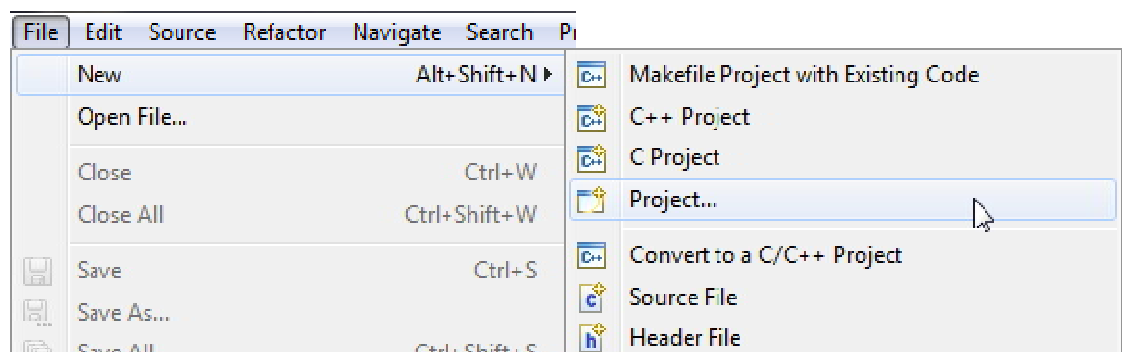
The C++ Development Toolkit (CDT) is a collection of Eclipse-based features that provides the capability to create, edit, navigate, build, and debug projects that use C++ as a programming language.

The CDT does not include the necessary compilers and debuggers to convert C++ code into executable programs and to debug those programs, but it does provide the frameworks that allow such tools to be integrated in a consistent fashion. This allows you to mix and match such tools depending on project requirements

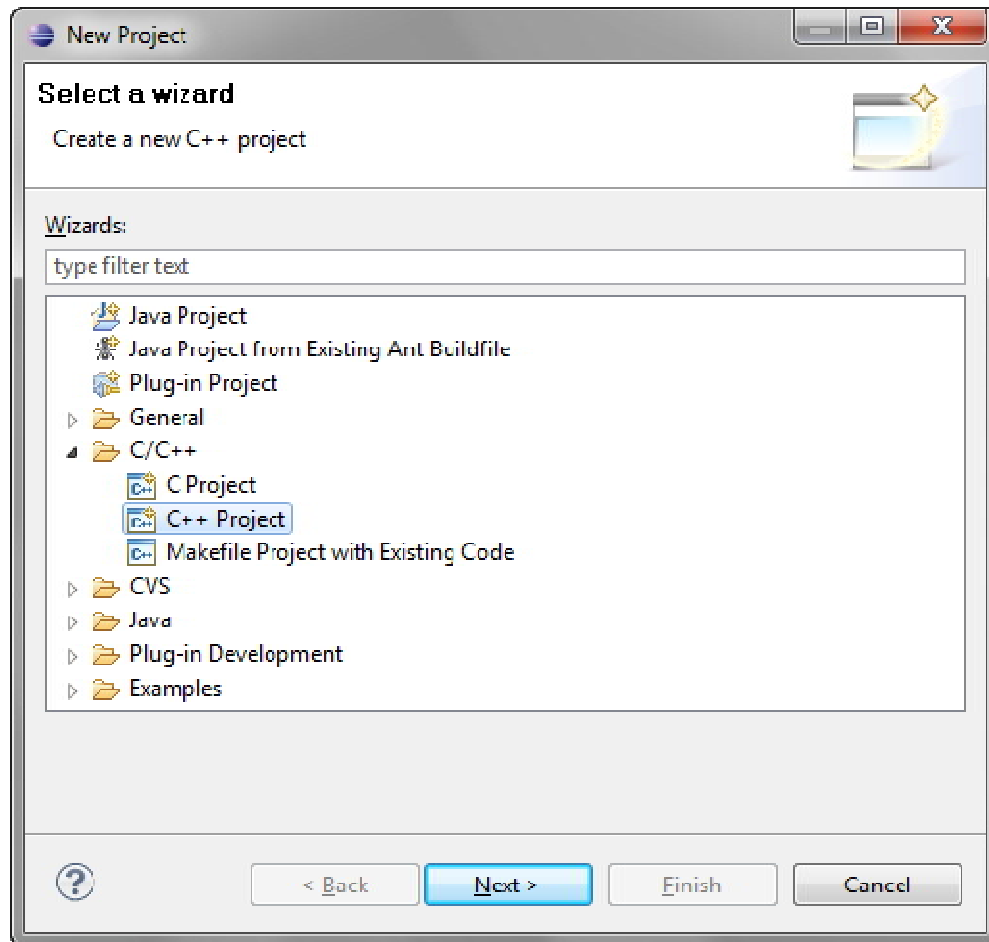
All Linux distributions include the GNU toolchain. They may not, however, be installed by default.

#### Step 1: Creating a project

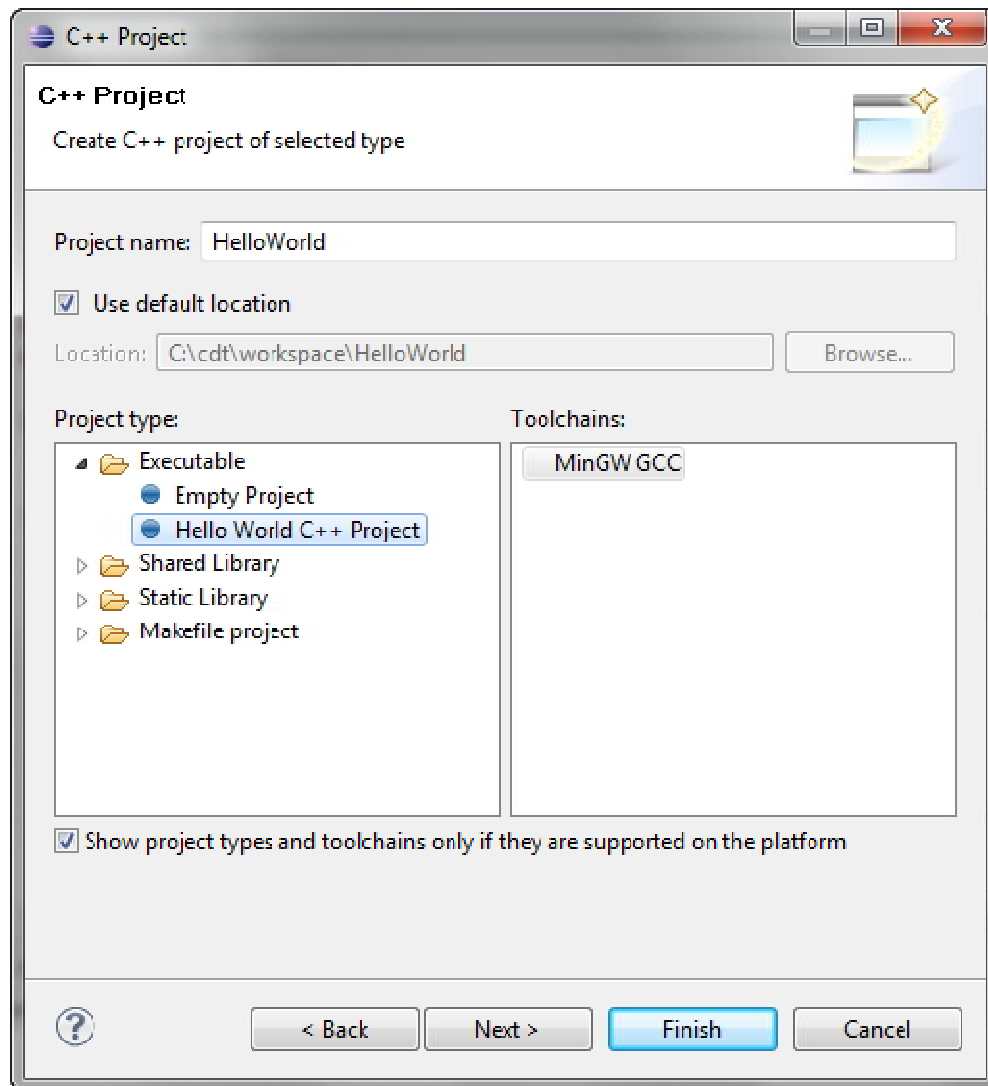
1. Select **File > New > Project**.



2. Select the type of project to create. Expand the **C++** folder and select **C++ Project** and click **Next**.



3. The **C++ Project** wizard opens.



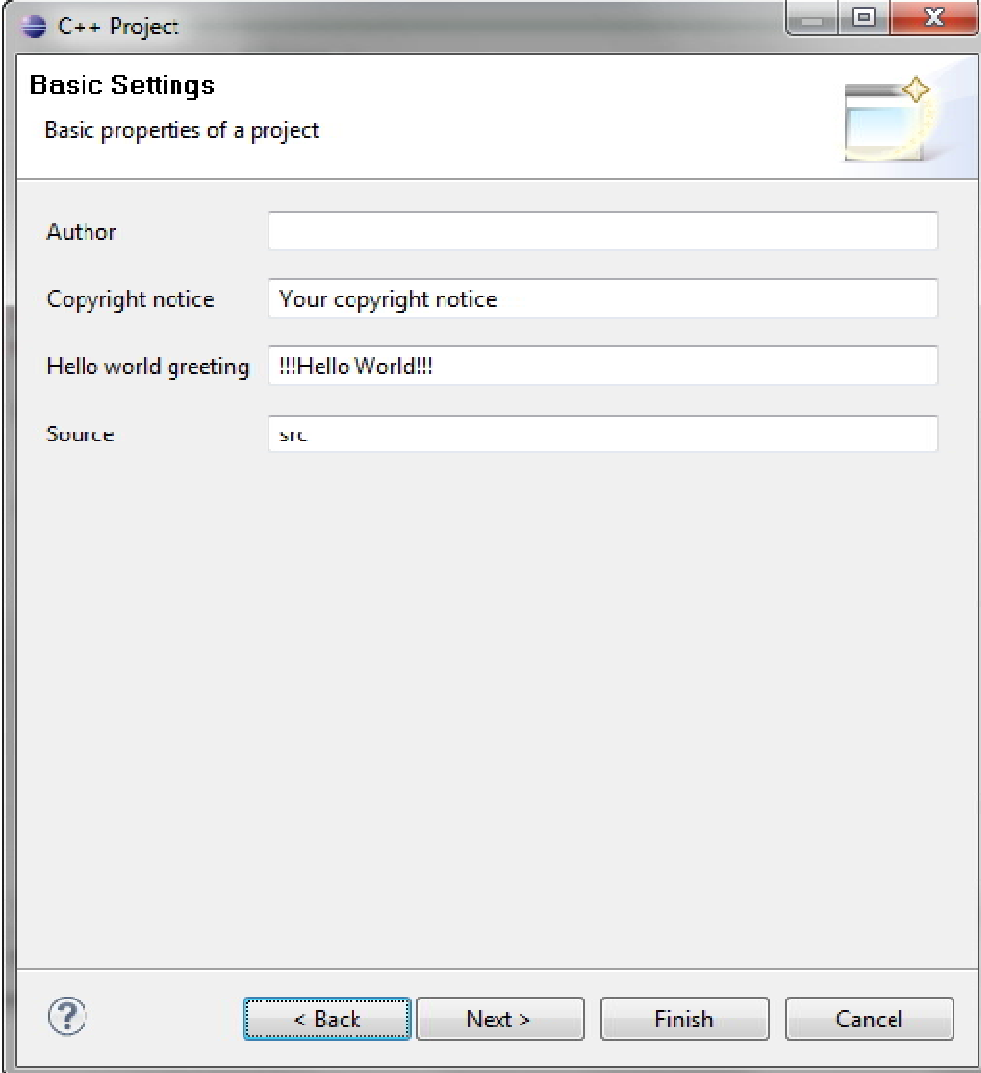
By default, the CDT filters the **Toolchain** and **Project types** that it displays in those lists based on the language support for the C++ Project wizard

- a. In the **Project name** field, type a name for the project, such as **HelloWorld**.
- b. From the **Project type** list, expand **Executable** and select **Hello World C++ Project**. This project type provides a simple Hello World application in C++, and the makefile is automatically created by the CDT.
- c. Select a required toolchain from the **Toolchain** list.

A toolchain is a set of tools (such as a compiler, linker, and assembler) intended to build your project. Additional tools, such as a debugger, can be associated with a toolchain. There can be several toolchains available, depending on the compilers installed on your system.

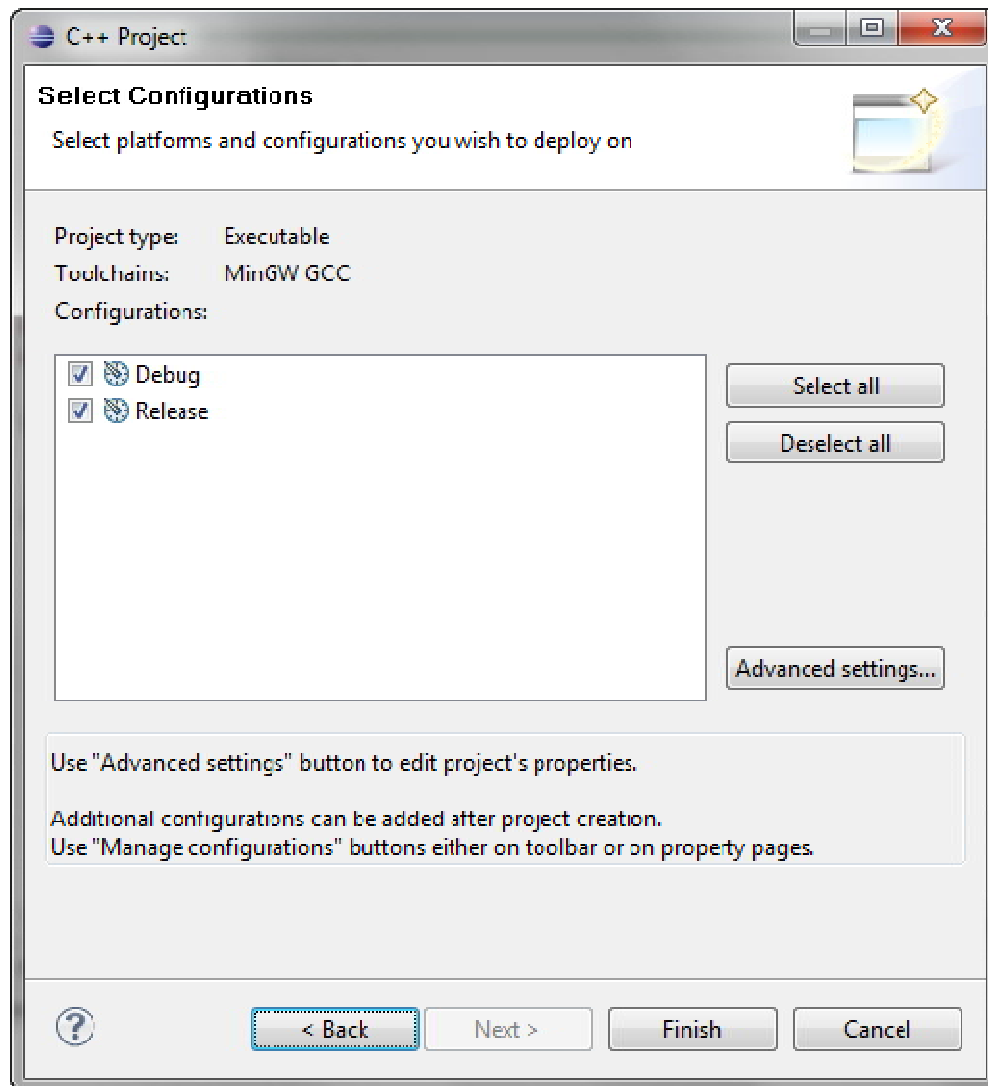
- d. Click **Next**.

4. Specify the **Basic Properties** for the new project, such as your author, copyright, and source information, then click **Next**.



The screenshot shows a Windows-style dialog box titled "C++ Project". The main content area is titled "Basic Settings" with the subtitle "Basic properties of a project". There is a small icon of a folder with a star in the top right corner. The form contains four input fields: "Author" (empty), "Copyright notice" (containing "Your copyright notice"), "Hello world greeting" (containing "!!!Hello World!!!"), and "Source" (containing "src"). At the bottom, there is a help icon (question mark in a circle) on the left, and four buttons: "< Back" (highlighted with a blue dashed border), "Next >", "Finish", and "Cancel".

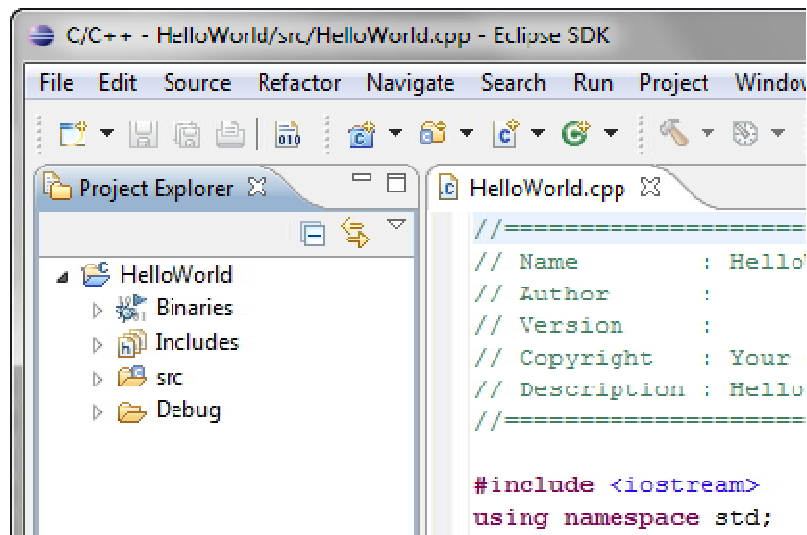
5. The **Select Configurations** page displays a list of configurations based on the project type and toolchain(s) selected earlier.



**OPTIONAL:** To change the default project settings, click **Advanced Setting** to open the Project Properties dialog for your new project allowing you change any of the project specific settings, such as includes paths, compiler options, and libraries.

6. Click **Finish**.

A project is created with the default settings and a full set of configurations based on the project type and toolchain selected. The new project is seen in Project Explorer view.



---

## Step 2: Reviewing the code and building the project

1. From the **Project Explorer** view, double-click the .cpp file created for the project, for example, HelloWorld.cpp. It is found within the project "src" folder.

This file opens in a default editor. It contains C++ template code for the Hello World example project you selected earlier. In addition, the Outline view has also been populated with objects created from code.





**NOTE:** A different editor can be specified, and add or modify existing code templates in **Window > Preferences**.

**OPTIONAL:** Additional code can be typed in this file, and then save the changes by clicking **File > Save**, or pressing **CTRL+S**.

Next, project needs to be built before it can be run.

2. Build the project by pressing **CTRL+B**, or select the project in the **Project Explorer** view and select **Project > Build Project**.

**NOTE:** If a build generates any errors or warnings, that can be seen in the Problemsview.

3. Read through the build messages in the Console view. The project should build successfully.

**Outline** view has also been populated with objects created from your code. If you select an item from the **Outline** view, the corresponding text in the editor is highlighted.

### Step 3: Running the application

To run the application:

1. Within the C++ Perspective, click **Run > Run Configurations...**
2. Select **C++ Application**.
3. Click **New**.  
A new Run Configuration is created. Its name and path to the executable are provided by the project ('Hello World' in this case).
4. Click **Run**.

Now, see the Hello World application running in the **Console** view. The **Console** also shows which application is running in a title bar.

5. Click the other views to see the information that they contain.