VIII Resources required :

Sr. No	Name of Resource	Specification	Quantity	Remarks
1.	Hardware : Computer System	Computer (i3-i5 preferable), RAM minimum 2 GB and onwards, HDD 40GB and above	As per batch size	For all Experiments
2.	Operating system	Windows /LINUX		
3.	Software	Turbo C++ Version 3.0 or any other		

IX Precautions

1. Handle computer system and peripherals with care.
2. Follow safety practices.

X Resources used

S. No.	Name of Resource	Specificati
1	Computer System with broad specifications	Computer (i3-i5 preferable), RAM mini. 2GB & onwards, HDD 40GB
2	Software	Turbo C++ version 3.0 or any other
3	Any other resource used	Hardware, Operating system (Windows /LINUX)

XI Result (Output of the Program)

Program get output 20.

XII Practical Related Questions

Note: Below given are few sample questions for reference. Teacher must design more such questions so as to ensure the achievement of identified CO.

(Note: Use Point VIII to X and XIII to XV for all relevant programming exercise use blank pages provided or attach more pages if needed.)

1. Constant variables can be created in C++ by using _____ .
   a. const b. #define d. Both a& b d. None of these

2. State output of the following code:

```
#include <iostream.h>
void main()
{
    typedef int num;
    num a = 10, b = 15;
    num c = a + b + a - b;
    cout << c;
}
```

Output 20;

3. Regarding following statement which of the statements is true? const int a = 100;
   a) Declares a variable a with 100 as its initial value
   b) Declares a construction a with 100 as its initial value c) Declares a constant a whose value will be 100
   d) Constructs an integer type variable with a as identifier and 100 as value

4. Which of the following statement is not true about preprocessor directives?
   a) These are lines read and processed by the preprocessor b) They do not produce any code by themselves
   c) These must be written on their own line d) They end with a semicolon

XIII Exercise

Attempt Q1 or Q2 and Q.3 a or b from the following:
(Note: Use Point VIII to X and XIII to XV for all relevant programming exercise use blank pages provided or attach more pages if needed.)

1. Write a C++ program to evaluate the following expressions:

$$x_1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a}, \quad x_2 = \frac{-b - \sqrt{b^2 - 4ac}}{2a}$$

2. Write a C++ program to demonstrate the use of operator precedence.

3. Complete the given table:

Program Code	Write & justify Output

a) `#include<iostream.h>`
`#define PI 3.14159`
`int main ()`
`{`
`float r = 2;`
`float circle;`
`circle = 2 * PI * r;`
`cout << circle;`
`return 0;`
`}`
	Output 12.56656

a)
```
#include<iostream.h>
#include<conio.h>
void main()
{
   clrscr();
   float res;
   float f1=15.5, f2=2;
   res = (int)f1/(int)f2;
   cout<<res<<endl;
   res = (int)(f1/f2);
   cout<<res<<endl;
   res = f1/f2;
   cout<<res;
   getch();
}
```

Output:

7
7
7.5

(Space for Answers)

Q1: Write a C++ program to evaluate the foll. expressions.

$$x_1 = \frac{-b + \sqrt{b^2-4ac}}{2a} \quad , \quad x_2 = \frac{-b - \sqrt{b^2-4ac}}{2a}$$

```
#include <iostream.h>
#include <conio.h>
  void main ()
{
int a,b,c, x1, x2, r;
clrscr ();
cout << "Enter a, b & c";
cin >> a >> b >> c;
 r = (b*b) - (4*a*c);
x1 = (-b * sqrt (r)) / 2 * a;
x2 = (-b - sqrt (r)) / 2 * a;
cout << "\n x1 =" << x1;
cout << "\n x2 =" << x2;
 getch ();
}
```


2. Write a C++ program to demonstrate the use of operator precedence

```
#include <iostream.h>
#include <conio.h>
void main ()
{
int a=20;
int b=10;
int c=15;
int d=5;
int e;
clrscr ();
e = (a+b) * c /d;
cout << "Value of (a+b) * c/d is;" <<e <<endl;
e = (a+b * c/d);
cout <<" Value of (a+b) * (c/d) is;" <<e<<endl;
e = (a+b) * (c/d);
cout <<" Value of a+(b*c)\d is;"<<e<<endl;
getch ();
}
```

Output :-

Value of (a+b) * (c/d) is; 90

Value of (a+b) * (c/d) is; 50

Value of (a+b) * (c/d) is; 90

Value of a+ (b*c)/d is; 50

XIV References / Suggestions for further Reading

1. https://www.sanfoundry.com/c-plus-plus-interview-questions-experienced-references/
2. http://people.scs.carleton.ca/~dehne/projects/cpp-doc/tutorial/tut1-2.html
3. http://www.cplusplus.com/doc/tutorial/constants/
4. http://home.pacifier.com/~mmead/cs161/lesson03_07.html

XV Assessment Scheme

Performance indicators		Weightage
Process related(35 Marks)		**70%**
1	Logic formation	20%
2.	Appropriate use of Variables, constants, arithmetic expressions and data type conversions.	20%
3.	Debugging ability	20%
4.	Follow ethical practices.	10%
Product related (15 Marks)		**30%**
5	Expected Output	10%
6	Timely Submission of report	10%
7	Answer to sample questions	10%
Total (50 Marks)		**100%**

List of Students /Team Members

1. .....
2. .....
3. .....
4. .....

Marks Obtained			Dated signature of Teacher
Process Related(35)	Product Related(15)	Total(50)	
34	15	49	

Practical No. 2: Program to implement decision making statements (If-Else, Switch)

I Practical Significance:

The decision making statements like if-else and switch-case helps to decide the order of execution of statements based on certain conditions or repeat a group of statements until certain specified conditions are met.

II Relevant Program Outcomes (POs)

- **Basic knowledge:** Apply knowledge of basic mathematics, sciences and basic engineering to solve the broad-based Computer engineering problem.
- **Discipline knowledge:** Apply Computer engineering discipline - specific knowledge to solve core computer engineering related problems.
- **Experiments and practice:** Plan to perform experiments and practices to use the results to solve broad-based Computer engineering problems.
- **Engineering tools:** Apply relevant Computer technologies and tools with an understanding of the limitations.
- **Communication:** Communicate effectively in oral and written form.

III Competency and Practical skills

This practical is expected to develop the following skills in you:

Develop C++ programs to solve broad-based problems

1. Define decision making statements applicable to the particular problem.
2. Compile the program.
3. Debug and execute the program.

IV Relevant Course Outcome(s)

Develop C++ programs using decision making statements.

V Practical Outcome (POs)

a)Write/ Compile/ debug / Execute simple C++ program using decision making statements.

VI Relevant Affective domain related Outcome(s)

1. Select proper programming environment in C++.
2. Follow ethical practices.

VII Minimum Theoretical Background

Decision Making Statement

1. Decision making statement is depending on the condition block need to be executed or not which is decided by condition.
2. If the condition is "true" statement block will be executed, if condition is "false" then statement block will not be executed.
3. In this section we are discuss about if-then (if), if-then-else (if else), and switch statement. In C++ language there are three types of decision making statement.

if
if-else
switch

if-else Statement :- In general it can be used to execute one block of statement among two blocks, if and else are the keyword in C++.

Syntax

```
if(condition)
{
.....
statements
.....
}
else
{
.....
statements
.....
}
```

Switch Statement:-

A switch statement work with byte, short, char and int primitive data type, it also works with enumerated types and string.

Syntax

```
switch(expression/variable)
{
case value:
//statements
// any number of case statements
break; //optional
default: //optional
//statements
}
```

VIII Resources required

Sr. No.	Name of Resource	Specification	Quantity	Remarks
1	Hardware: Computer System	Computer (i3-i5 preferable), RAM minimum 2 GB and onwards, HDD 40GB and above	As per batch size	For all Experiments
2	Operating system	Windows /LINUX	Windows	
3	Software	Turbo C++ Version 3.0 or any other	Turbo C++	

IX Precautions

1. Handle computer system and peripherals with care.
2. Follow safety practices.

X Resources used

S. No.	Name of Resource	Specification
1	Computer System with broad specifications	Computer (i3-i5 preferable), RAM min. 2GB and onwards
2	Software	Turbo C++ version 3.0 or any other
3	Any other resource used	

XI Result (Output of the Program)

We learned the program to implement decision making statements (If-Else, switch)

XII Practical Related Questions

Note: Below given are few sample questions for reference. Teacher must design more such questions so as to ensure the achievement of identified CO.

(Note: Use Point VIII to X and XIII to XV for all relevant programming exercise use blank pages provided or attach more pages if needed.)

1. Observe the following block of code and determine what happens when x=2?

```
switch (x){
case 1:
case 2:
case 3:cout<< "x is 3, so jumping to third
branch"; goto thirdBranch;
default:
cout<<"x is not within the range, so need to say
Thank You!";
        }
```

2. Conditional operator in C++ is a
   a) Unary operator b) Binary operator c)Ternary operator d) None of them

3. Which of the following is selection statement in C++?
   a) break b) goto c) exit d) switch

[Space for Answers]

1) Observe the Foll^n back of code & determine what happens when x = 2 ?

```
#include <iostream.h>
#include <conio.h>
```

```
void main()
{
int x;
clrscr();
cout << "Enter value of x";
cin >> x
switch (x)
{
case 1:
case 2:
cout << "x is 2, so jumping to third branch";
case 3:
cout << "x is 3, so jumping to third branch";
default:
cout << "x is not within the range, so need
to say thank you!";
}
getch();
}
```

Output: Enter value of x 3
x is 3, so jumping to third branch
x is not within the range, so need to
say ~~too~~ thank you!

Enter the value of x 2
x is 2 so jumping to third branch
x is not within the range, so need to
say thank you!

2) c] Ternary operator

3) d] switch

XIII Exercise:
Attempt Q1 or Q2 or Q3 and Q.4 a ,b or a, c from the following:

(Note: Use Point VIII to X and XIII to XV for all relevant programming exercise use blank pages provided or attach more pages if needed.)

1. Write a C++ program check leap year.
2. Write a C++ program armstrong numbers between two integers.
3. Write a C++ program to check whether a number is palindrome or not.
4. Complete the given table:

Program Code	Write & justify Output

a) `#include <iostream.h>`
`void main()`
`{`
`int a = 10;`
`if (a < 15)`
`{`
`time:`
`cout << a;`
`goto time;`
`}`
`break;`
`}`
	a = 10
	a < 15
	10 < 15
	10

b)
`#include <iostream.h>`
`long factorial (long a)`
`{`
`if (a > 1)`
`return (a * factorial (a + 1));`
`else`
`return (1);`
`}`
`int main ()`
`{`
`long num = 3;`
`cout << num << "! = " << factorial ( num );`
`}`
	output;
	segmention Fault core dumped

c) `#include <iostream.h>`
`void main()`
`{`
`int percentage;`
`cout << "Enter the percentage : ";`
`cin >> percentage;`
`switch (percentage / 10)`
`{`
`case 10: case 9:`
`cout << "You have got grade A+" << endl;`
`break;`
`case 8:`
`cout << "You have got grade A" << endl;`
`break;`
`case 7:`
	error :-
	'cout' was not declared in this scope.
	cout << "Enter the percentage"

```
        cout << "You have got grade B+" << endl;
        break;
    case 6:
        cout << "You have got grade B" << endl;
        break;
    case 5:
        cout << "You have got grade C" << endl;
        break;
    default:
        cout << "You have got grade D" << endl;
        break;
    }
}
```

(Space for Answers)

3) Write a C++ program to check leap year Palindrom

→

```
#include <iostream.h>
#include <conio.h>
  void main ()
{
int n, rem, sum = 0, temp = 0;
clrscr ();
cout <<" enter a number";
cin >> n;
temp = n;
while (n > 0)
{
rem = n % 10;
sum = (sum * 10) + rem;
n = n / 10;
}
if (temp == sum)
{
cout << " Number is Palindrome";
}
else
{
cout << "Number is not a Palindrome";
}
getch();
```


Write a C++ program to check leap year.

```
#include <iostream.h>
#include <conio.h>
void main ()
{
int year;
clrscr ();
cout <<" Enter the year";
 cin >> year;
if (year % 4 == 0)
{
cout << "\n it is a leap year";
}
else
{
cout <<" \n it is not leap year";
}
getch ();
}
```

2) Write a C++ program to display armstr

```
#include <iostream.h>
#include <conio.h>
 void main()
{
int n, sum=0, rem, temp
 clrscr();
cout <<" Enter any number";
 cin >> n;
temp = n;
 while (n>0)
{
 rem = n % 10;
sum = sum + (rem * rem * rem);
   n = n /10;
}
if (temp == sum)
{
 cout <<" Number is armstrong";
}
else
{
cout <<" Number is not armstrong";
}
 getch();
}
```


XIII **References / Suggestions for further Reading**

1. https://www.programiz.com/c-programming/examples/leap-year
2. https://www.cpp.thiyagaraaj.com/c-programs/c-basic-example- programs/if-else-example-program-in-c
3. https://www.sanfoundry.com/cpp-program-illustrate-switch-statement/

XV **Assessment Scheme**

Performance indicators		Weightage
Process related(35 Marks)		**70%**
1	Logic formation	20%
2	Appropriate use of Decision making statements (if-else , switch)	20%
3	Debugging ability	20%
4	Follow ethical practices.	10%
Product related (15 Marks)		**30%**
5	Expected Output	10%
6	Timely Submission of report	10%
7	Answer to sample questions	10%
Total (50 Marks)		**100%**

List of Students /Team Members

1. Vaishnavi Chaudhari
2. .....
3. .....
4. .....

Marks Obtained			Dated signature of Teacher
Process Related(35)	Product Related(15)	Total(50)	
34	13	47	Sudarshan

Practical No. 3: Program to Implement Control Structure (For, While, Do-While)

I **Practical Significance:**
Loops are used to execute a set of statements repeatedly until a particular condition is satisfied.

II **Relevant Program Outcomes (POs)**
- **Basic knowledge:** Apply knowledge of basic mathematics, sciences and basic engineering to solve the broad-based Computer engineering problem.
- **Discipline knowledge:** Apply Computer engineering discipline - specific knowledge to solve core computer engineering related problems.
- **Experiments and practice:** Plan to perform experiments and practices to use the results to solve broad-based Computer engineering problems.
- **Engineering tools:** Apply relevant Computer technologies and tools with an understanding of the limitations.
- **Communication:** Communicate effectively in oral and written form.

III **Competency and Practical skills**
This practical is expected to develop the following skills in you :

Develop C++ programs to solve broad-based problems
1. Use appropriate looping construct(for, while, do-while) in the given problem.
2. Compile the program.
3. Debug and execute the program.

IV **Relevant Course Outcome(s)**
Develop C++ programs to solve problems using Procedure Oriented Approach.

V **Practical Outcome (POs)**
Write/ Compile/ debug / Execute simple C++ program using for loop, while loop and do-while loop .

VI **Relevant Affective domain related Outcome(s)**
1. Select proper programming environment in C++.
2. Follow ethical practices.

VII **Minimum Theoretical Background**
Control Structure
Loops are used in programming to repeat a specific block until some end condition is met. There are three types of loops in C++ programming:
for loop
while loop
do...while loop

for loop:-
- **Syntax**

```
for(initializationStatement; testExpression; updateStatement)
  {
    // codes
  }
```

How for loop works?
- The initialization statement is executed only once at the beginning. Then, the test expression is evaluated.
- If the test expression is false, for loop is terminated.
- But if the test expression is true, codes inside body of for loop is executed and update expression is updated.
- Again, the test expression is evaluated and this process repeats until the test expression is false.

while Loop:-

The syntax of a while loop is:

```
while (testExpression)
{
  // codes
}
```

where, test Expression is checked on each entry of the while loop.

How while loop works?
The while loop evaluates the test expression.
If the test expression is true, codes inside the body of while loop is evaluated.
Then, the test expression is evaluated again. This process goes on until the test expression is false.
When the test expression is false, while loop is terminated.

do-while Loop:-

The do...while loop is a variant of the while loop with one important difference. The body of do...while loop is executed once before the test expression is checked.

The syntax of do..while loop is:

```
do {
  // codes;
}
while (testExpression);
```

How do...while loop works?
- The codes inside the body of loop is executed at least once. Then, only the test expression is checked.
- If the test expression is true, the body of loop is executed. This process continues until the test expression becomes false.
- When the test expression is false, do...while loop is terminated.

VIII **Resources required**

Sr.	Name of Resource	Specification	Quantity	Remarks
1	Hardware: Computer System	Computer (i3-i5 preferable), RAM minimum 2 GB and onwards, HDD 40GB and above	As per batch size	For all Experiments
2	Operating system	Windows /LINUX		
3	Software	Turbo C++ Version 3.0 or any other		

IX **Precautions**

1. Handle computer system and peripherals with care.
2. Follow safety practices.

X **Resources used**

S. No.	Name of Resource	Specification
1	Computer System with broad specifications	computer (i3-i5 preferable), RAM min. 2GB & onwards, HDD 40GB & above
2	Software	Turbo C++ version 3.0 or any other
3	Any other resource used	Operating system windows.

XI **Result (Output of the Program)**

We learnt to implement programs for looping statement like For, while & do-while

XII **Practical Related Questions**

Note: Below given are few sample questions for reference. Teacher must design more such questions so as to ensure the achievement of identified CO.

(**Note:** Use Point VIII to X and XIII to XV for all relevant programming exercise use blank pages provided or attach more pages if needed.)

1. Observe the following block of code and determine what happens when x=2?

```
#include<iostream.h>
void main ()
{
int a,d,n,sum,term=0;

cout<<"Enter the first term, common difference,"
```



```
<<"and the number of terms to be summed"
<<"respectively:\n";
Cin>>a>>d>>n;
sum=0;

int i=1;
cout<<"\nThe terms are ";
do
{
  term= a+ (i-1)*d;
  sum+=term;
  cout<<term<<" ";
  ++i;
}
while (i<=n) ;
cout<<"\nThe sum of A.P. is "<<sum;
}
}
```

2. Which loops needs a semi colon after?
   a) for b) do c) while

3. In the while statement, while(x<100)..., when does the statement controlled by the condition execute?
   a. When x is less than one hundred
   b. When x is greater than one hundred
   c. When x is equal to one hundred
   d. While it wishes

(Space for Answers)

2: The do loop needs a semicolon.

3: a] when x is less than one hundred

XIII Exercise

Attempt any 2 from 1-4 and Q.5 a or b from the following:

(**Note:** Use Point VIII to X and XIII to XV for all relevant programming exercise use blank pages provided or attach more pages if needed.)

1. Write a C++ program to print a multiplication table of 7
2. Write a C++ program to print 100 to 1.
3. Write a C++ program which will print (half pyramid) pattern of natural numbers.

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

4. Write a C++ program; ask the user to enter the number of rows to print the pyramid of stars (*)

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


5. Complete the given table:

Program Code	Write & justify Output

a) `#include <iostream.h>`
`int main()`
`{`
`  int a, d, n;`
 	
`  std::cout << "Enter a, d, n of AP: ";`
`  std::cin >> a >> d >> n;`
`  std::cout << std::endl << "Terms of AP : a = " << a`
`  << ", d = " << d << ", n = " << n << std::endl;`
`  do {`
`  std::cout << a << "\t";`
`  a = a + d;`
`  }`
`  while (n-- > 1);`
`  std::cout << std::endl;`
`}`
	1) Type, qualifier 'std' must be a struct or class name.
	2) Statement missing ;

b)
`#include <iostream.h>`
`#include <stack.h>`
`int main()`
`{`
`  std::stack< char > s;`
 	
`  // Pushing elements into stack`
`  for (char c = 'a'; c <= 'f'; c++)`
`  {`
`  s.push(c);`
`  }`
`  while (!s.empty())`
`  {`
`  std::cout << "Top element of stack : '" << s.top()`
`  << "'" << std::endl;`
`  s.pop();`
`  }`
`  std::cout << "Stack is empty!" << std::endl;`
`}`
	1) Statement missing ;
	2) Unable to open include file 'struct.h'
	3) Undefined symbol 's'

(Space for Answers)

```
1) #include <iostream.h>
   #include <conio.h>
   void main ()
   {
   int i , t = 7 ;
   clrscr ( ) ;
   for (i=1 ; i<=10; ++i)
```

```
{
cout <<t<<" * "<<i<<" = "<<t*i<<endl;
}
getch();
}
```

XIV References / Suggestions for further Reading

1. https://www.programiz.com/cpp-programming/for-loop
2. http://ecomputernotes.com/cpp/control-structures/iteration-statements
3. https://www.sanfoundry.com/cpp-program-demonstrate-do-while-loop/
4. https://codescracker.com/cpp/program/cpp-program-print-star-pyramid-patterns.htm

XV Assessment Scheme

	Performance indicators	Weightage
	Process related(35 Marks)	**70%**
1	Logic formation	20%
2	Appropriate use of control structure (for, while,do-while)	20%
3	Debugging ability	20%
4	Follow ethical practices.	10%
	Product related (15 Marks)	**30%**
5	Expected Output	10%
6	Timely Submission of report	10%
7	Answer to sample questions	10%
	Total (50 Marks)	**100%**

List of Students /Team Members

1 Vaishnavi Chaudhari
2 .....
3 .....
4 .....

Marks Obtained			Dated signature of Teacher
Process Related(35)	Product Related(15)	Total(50)	
34	14	48	Sabnavis

)WAP to print 100 to 1

```
#include <iostream.h>
#include <conio.h>
void main()
{
int i;
clrscr();
for (i=100; i>=1; i--)
{
cout<<i<<"\t";
}
getch();
}
```

WAP to print (half pyramid) pattern of natural nos

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

```
#include <iostream.h>
#include <conio.h>
void main()
{
int i,j,k=1;
clrscr();
for (i=1; i<=5; i++)
{
for (j=1; j<=i; j++)
{
```

```
cout<<k;
k++;
}
cout<<"\n";
}
getch();
}
```

4) Pyramid of stars (*)

→
```
#include<iostream.h>
#include<conio.h>
void main()
{
  int i,j,k;
  clrscr();
 for(i=4;i>=i;j--)
 {
  for(j=4;j>=i,j--)
 {
  cout<<" ";
 }
  for(k=1;k<=(2*i-1);k++)
 {
  cout<<"*";
 }
  cout<<\n";
 }
 getch();
}
```


Practical No. 4: Program to Implement One Dimension Array

I Practical Significance:

The arrays helps to represent collection of similar type of data under common name.

II Relevant Program Outcomes (POs)

- **Basic knowledge:** Apply knowledge of basic mathematics, sciences and basic engineering to solve the broad-based Computer engineering problem.
- **Discipline knowledge:** Apply Computer engineering discipline - specific knowledge to solve core computer engineering related problems.
- **Experiments and practice:** Plan to perform experiments and practices to use the results to solve broad-based Computer engineering problems.
- **Engineering tools:** Apply relevant Computer technologies and tools with an understanding of the limitations.
- **Communication:** Communicate effectively in oral and written form.

III Competency and Practical skills

This practical is expected to develop the following skills in you :

Develop C++ programs to solve broad-based problems

1. Define one dimensional array.
2. Compile the program.
3. Debug and execute the program.

IV Relevant Course Outcome(s)

Develop C++ programs to solve problems using Procedure Oriented Approach..

V Practical Outcome (POs)

Write/ Compile/ debug / Execute simple C++ program using one dimensional arrays.

VI Relevant Affective domain related Outcome(s)

1. Select proper programming environment in C++.
2. Follow ethical practices.

VII Minimum Theoretical Background

Arrays in C++

Array is a collection of data of same types stored in sequential memory location. It is a linear data structure, where data is stored sequentially one after the other. The elements in an array is accessed using an index.

For example, In an array of n elements, the first element has index *zero* and the last element has index *(n-1)*.

Elements with consecutive index (i.e. I and i+1) are stored in consecutive memory location in the system.

Array can be divided into following types:

One Dimensional Array

Multi-Dimensional Array

One Dimensional Array Syntax:

type arrayName [arraySize];

Examples:

```
double salary[15000];
int age[5]={22,25,30,32,35};
```

VIII Resources required

Sr. No.	Name of Resource	Specification	Quantity	Remarks
1	Hardware: Computer System	Computer (i3-i5 preferable), RAM minimum 2 GB and onwards, HDD 40GB and above	As per batch size	For all Experiments
2	Operating system	Windows /LINUX		
3	Software	Turbo C++ Version 3.0 or any other		

IX Precautions

1. Handle computer system and peripherals with care.
2. Follow safety practices.

X Resources used

S. No.	Name of Resource	Specification
1	Computer System with broad specifications	RAM minimum 2GB & onwards
2	Software	Turbo c++
3	Any other resource used	

XI Result (Output of the Program)

Implemented one dimension array programs & got correct output

XII Practical Related Questions

Note: Below given are few sample questions for reference. Teacher must design more such questions so as to ensure the achievement of identified CO.

(Note: Use Point VIII to X and XIII to XV for all relevant programming exercise use blank pages provided or attach more pages if needed.)

1. State output of the following code:

```
#include <iostream.h>
int main()
{
    int n, i;
```



```
    float num[100], sum=0.0, average;
    cout << "Enter the number of data: ";
    cin >> n;

    while (n > 100 || n <= 0)
    {
    cout << "Error! number should in range of (1 to 100)." <<
        endl;
        cout << "Enter the number again: ";
        cin >> n;
    }
    for(i = 0; i < n; ++i)
    {
        cout << i + 1 << ". Enter number: ";
        cin >> num[i];
        sum += num[i];
    }
    average = sum / n;
    cout << "Average = " << average;
    return 0;
}
```

2. Which of the following correctly declares an array?
   a) int array[10]; b) int array;
   c) array{10}; d) array array[10];
3. Can we change the size of an array at run time?
4. What is the default value of Array?
5. Can we declare array size as a negative number?
6. What is the meaning of anonymous array? Explain with an example?
7. Is there any difference between int[] a and int a[]?
8. How to copy an array into another array?
9. Can we change the size of an array at run time?
10. Can you declare an array without assigning the size of an array?

(Space for Answers)

1) Output:
Enter the number of data: 5
1. Enter number: 1
2. Enter number: 2
3. Enter number: 3
4. Enter number: 4
5. Enter number: 5
Average: 3

2) int array [10] is correct

3) No we cant change the size of array at run time

4) The default value of array is Null

5) No we cant declare array size as a negative no

6) Anonymous array is array having name but can be declared in a function
Eg: New int [ ] = {'x', 'y', 'z'};

7) There is no diff^{n} in Funtionality bet^{n} both styles of declaration. Both declare array of int but int [ ] a keeps type information together and is more verbose.

9) No we cant change size of an array at run time

XIII Exercise

Attempt any 2 programs from 1-4 and Q.5 attempt any 2 from the following:

(**Note:** Use Point VIII to X and XIII to XV for all relevant programming exercise use blank pages provided or attach more pages if needed.)

1. Write a C++ program to find median of two sorted arrays of same size.
2. Write a C++ program to find the two repeating elements in a given array.
3. Find the smallest and second smallest elements in an array.
4. Write a C++ program to find the Missing Number.
5. Complete the given table:

2)
```
#include<iostream.h>
#include<conio.h>
void main()
{
 int arr[5],n,i,j;
 clrscr();
 cout<<"\n Enter size of array=";
 cin>>n;
 cout<<"Enter any "<<n<<" no in array=";
 for(i=0;i<n;i++)
 {
  cin>>arr[i];
 }
 cout<<"Nos in array are";
 for(i=0;i<n;i++)
 {
 cout<<"\t"<<arr[i];
 }
 cout<<"\n"<<"\t"<<"Repeating nos. are";
 for(i=0;i<n;i++)
  {
   if(arr[i]==arr[j])
   {
    cout<<"\n"<<arr[i];
   }
  }
 }
 getch();
```

Program Code	Write & Justify Output

a) #include <iostream.h>
int array1[] = {1200, 200, 2300, 1230, 1543};
int array2[] = {12, 14, 16, 18, 20};
int temp, result = 0;
void main()
{
for (temp = 0; temp < 5; temp++)
{
result += array1[temp];
}
for (temp = 0; temp < 4; temp++)
{
result += array2[temp];
}
cout << result;
}
	Output
	6538

b) #include <iostream.h>
void main ()
{
int array[] = {0, 2, 4, 6, 7, 5, 3};
int n, result = 0;
for (n = 0; n < 8; n++) {
result += array[n];
}
cout << result;
}
	Output
	22098

c) #include <iostream.h>
void main()
{
char str[5] = "ABC";
cout << str[3];
cout << str;
}
	Error: 'cout was not declared in this scope
	cout<<str[3];

(Space for answers)

```
a) #include <iostream.h>
   #include <conio.h>
   void main ()
   {
   int arr[100], temp, i, size;
   clrscr ();
   cout<<" Enter the size of array";
   cin>>size;
   cout<<"\n Enter elements in array";
   for (i=0; i<size; i++)
    {
    cin >> arr[i];
    }
    temp = (size+1) * (size+2) /2;
   for (i=0; i<size; i++)
    {
    temp = temp - arr[i];
    }
   cout <<"\n missing elements in
         array is" << temp;
   getch ();
   }
```

XIV References / Suggestions for further Reading

1. https://www.w3schools.in/cplusplus-tutorial/arrays/
2. https://www.geeksforgeeks.org/c-programs-gq/array-programs-gq/
3. https://www.sanfoundry.com/c-programming-examples-arrays/
4. https://www.programtopia.net/cplusplus/docs/arrays

XV Assessment Scheme

Performance indicators		Weightage
Process related(35 Marks)		**70%**
1	Logic formation	20%
2	Appropriate one dimensional array declaration	20%
3	Debugging ability	20%
4	Follow ethical practices.	10%
Product related (15 Marks)		**30%**
5	Expected Output	10%
6	Timely Submission of report	10%
7	Answer to sample questions	10%
Total (50 Marks)		**100%**

List of Students /Team Members

1. Vaishnavi Chaudhari
2. .....
3. .....
4. .....

Marks Obtained			Dated signature of Teacher
Process Related(35)	Product Related(15)	Total(50)	
34	14	48	

Practical No. 5: Program to Perform Matrix Operation Using Multi-Dimension Array.

I Practical Significance:

The Two Dimensional Array helps to represent the real life data in tabular format with different operations on it.

II Relevant Program Outcomes (POs)

- **Basic knowledge:** Apply knowledge of basic mathematics, sciences and basic engineering to solve the broad-based Computer engineering problem.
- **Discipline knowledge:** Apply Computer engineering discipline - specific knowledge to solve core computer engineering related problems.
- **Experiments and practice:** Plan to perform experiments and practices to use the results to solve broad-based Computer engineering problems.
- **Engineering tools:** Apply relevant Computer technologies and tools with an understanding of the limitations.
- **Communication:** Communicate effectively in oral and written form.

III Competency and Practical skills

This practical is expected to develop the following skills in you :

Develop C++ programs to solve broad-based problems

1 Define real life problem and solve it using multidimensional array if applicable.

2. Compile the program.

3. Debug and execute the program.

IV Relevant Course Outcome(s)

Develop C++ programs to solve problems using Procedure Oriented Approach.

V Practical Outcome (POs)

a)Write/ Compile/ debug / Execute simple C++ program using multidimensional array.

VI Relevant Affective domain related Outcome(s)

1. Select proper programming environment in C++.
2. Follow safety measures
3. Follow ethical practices.

VII Minimum Theoretical Background

Multi-dimensional array :-

A multi-dimensional **array** is an array of arrays. 2-dimensional arrays are the most commonly used. They are used to store data in a tabular manner.

- **Syntax:** type arr[row_size][column_size];
- **Example:** int arr[3][5];
- **2D array initialization**: An array can either be initialized during or after declaration. The format of initializing an array during declaration is as follows:
- **Syntax** :type arr[row_size][column_size]= {{elements}, {elements} ... };
- **Example** :

int arr[3][5] = {{5, 12, 17, 9, 3}, {13, 4, 8, 14, 1}, {9, 6, 3, 7, 21}};

VIII Resources required

Sr. No.	Name of Resource	Specification	Quantity	Remark
1	Hardware: Computer System	Computer (i3-i5 preferable), RAM minimum 2 GB and onwards, HDD 40GB and above	As per batch size	For all Experiments
2	Operating system	Windows /LINUX		
3	Software	Turbo C++ Version 3.0 or any other		

IX Precautions

1. Handle computer system and peripherals with care.
2. Follow safety practices.

X Resources used

S. No.	Name of Resource	Specification
1	Computer System with broad specifications	Ram mini. 2GB & onwards, HDD 40 GB & above.
2	Software	Turbo C++
3	Any other resource used	

XI Result (Output of the Program)

We learn how to perform matrix operation using multi-dimension array.

XII Practical Related Questions

Note: Below given are few sample questions for reference. Teacher must design more such questions so as to ensure the achievement of identified CO.

(**Note:** Use Point VIII to X and XIII to XV for all relevant programming exercise use blank pages provided or attach more pages if needed.)

1. What is the error in this C++ code?

```
#include<iostream.>
void main()
{
    int ary[2][3];
    ary[][] = {{1, 2, 3}, {4, 5, 6}};
        cout<<ary[1][0];
```



```
}
```

2. State output of the following code:

```
#include<iostream.h>
void main()
{
    // initializing the 3-dimensional
    array int x[2][3][2] =
    {
        { {0,1}, {2,3}, {4,5} },
        { {6,7}, {8,9}, {10,11} }
    };

    // output each element's value for
    (int i = 0; i < 2;++i)
    {
        for (int j = 0; j < 3; ++j)
        {
            for (int k = 0; k < 2; ++k)
            {
                cout << "Element at x[" << i << "][" << j
                    << "][" << k << "] = " << x[i][j][k]
                     <<
                     endl;
            }

        }
    }
}
```

3. How many kinds of elements an array can have?
   a) Char and int type b) Only char type
   c) Only int type d) All of them have same type

(Space for answers)

Q.1 The error is that the array size for first block is 2 but there are 3 array elements.

Q.2 Error: 'array' was not declared in this scope array n[2][3][2]=
error: 'n' was not declared in this scope
cout<<"Element at ["<<i<<"]["<<j<<"]["<<i<<"] =" <<n[i][j][k]

I Exercise

Attempt Q1 or Q2 and Q.3 a or b from the following:

(Note: Use Point VIII to X and XIII to XV for all relevant programming exercise use blank pages provided or attach more pages if needed.)

1. Write a C++ program to Multiply Two 3*3 Matrix Using Multi-dimensional Arrays.
2. Write a C++ program to find Transpose of a 2*2 Matrix
3. Complete the given table:

Program Code	Write & justify Output

a)
#include <iostream.h>
void main()
{
int a[2][3] = {1, 2, 3, , 4, 5};
int i = 0, j = 0;
for (i = 0; i < 2; i++)
for (j = 0; j < 3; j++)
cout<<a[i][j];
}
	Output
	123450

b) #include <iostream.h>
void main()
{
int a[2][3] = {1, 2, 3, 4, 5};
int i = 0, j = 0;
for (i = 0; i < 2; i++)
for (j = 0; j < 3; j++)
cout<<a[i][j];
}
	The output is
	123450

(Space for answers)


```
#include <iostream.h>
using namespace std;
int main()
{
int a[10][10], trans[10][10], r, c, i, j;
cout << " Enter rows & columns of matrix : ";
cin >> r >> r;
cout << endl << " Ender elements of matrix " << endl;
for (i=0; i=r; i++)
for (j=0; j<c; j++)
{
cout << " Enter elements a " << i+1 << j+1 << " ";
cin >> a[i][j];
}
}
cout << endl << " Enter elements of matrix : " << endl;
for (i=0; i<r; i++)
for (j=0; j<c; j++)
{
cout << " " << a[i][j];
if (j == (-1)
cout << endl << endl;
}
for (i=0; i<c; i++)
for (j=0; j<r; j++) }
cout << " " << trans[i][j];
if (j == r-1)
cout << endl << endl;
}
```

XIII References / Suggestions for further Reading

1 https://www.sanfoundry.com/multiple-choice-questions-c-multidimensional-arrays/
2 https://www.hackerearth.com/practice/data-structures/arrays/multi- dimensional/tutorial/
3 https://www.programiz.com/cpp-programming/examples/matrix-transpose

XIV Assessment Scheme

	Performance indicators	Weightage
	Process related(35 Marks)	**70%**
1	Logic formation	20%
2	Appropriate multidimensional array definition	20%
3	Debugging ability	20%
4	Follow ethical practices.	10%
	Product related (15 Marks)	**30%**
5	Expected Output	10%
6	Timely Submission of report	10%
7	Answer to sample questions	10%
	Total (50 Marks)	**100%**

List of Students /Team Members

1. Vaishnavi Chaudhari
2. .....
3. .....
4. .....

Marks Obtained			Dated signature of Teacher
Process Related(35)	Product Related(15)	Total(50)	
34	14	48	

Practical No. 6: Program to Implement Classes and Objects

I **Practical Significance:**
The classes and objects help to represent real life entity with different attributes and related member functions.

II **Relevant Program Outcomes (POs)**
- **Basic knowledge:** Apply knowledge of basic mathematics, sciences and basic engineering to solve the broad-based Computer engineering problem.
- **Discipline knowledge:** Apply Computer engineering discipline - specific knowledge to solve core computer engineering related problems.
- **Experiments and practice:** Plan to perform experiments and practices to use the results to solve broad-based Computer engineering problems.
- **Engineering tools:** Apply relevant Computer technologies and tools with an understanding of the limitations.
- **Communication:** Communicate effectively in oral and written form.

III **Competency and Practical skills**
This practical is expected to develop the following skills in you :

Develop C++ programs to solve broad-based problems
1. Define real life entity into classes and objects.
2. Define and use classes and objects.
3. Compile the program.
4. Debug and execute the program.

IV **Relevant Course Outcome(s)**
Develop C++ programs using classes and objects.

V **Practical Outcome (POs)**
a)Write/ Compile/ debug / Execute simple C++ program using classes and objects.

VI **Relevant Affective domain related Outcome(s)**
1. Select proper programming environment in C++.
2. Follow ethical practices.

VII **Minimum Theoretical Background**
- **Class:**. Class is a user defined data type, which holds its own data members and associated member functions, which can be accessed and used by creating an instance of that class.

- **Object:** An **Object** is an instance of a Class. When a class is defined, no memory is allocated but when it is instantiated (i.e. an object is created) memory is allocated.

```
class ClassName
        {
        Access specifier:     //can be private, public or protected
        Data members;         // Variables to be used
        Member functions()    //Methods to access data members
        {
                //body of the function
        }
        };                    //Class ends with semicolon
```

Declaring Objects: When a class is defined, only the specification for the objec is defined; no memory or storage is allocated. To use the data and access functio defined in the class, you need to create objects.

Syntax:

className ObjectName;

eg.

Student std;

Accessing data members and member functions: The data members an member functions of class can be accessed using the dot('.') operator with th object.

For example if the name of object is std and you want to access the membe function with the name getData() then you will have to write std.getData() .

Member Functions in Classes: There are 2 ways to define a member function:

Inside class definition:

Syntax: class className

```
{
    //other members declaration
    return type functionName(list of parameters1)
    {
    // body of function
    }
    //other members declaration
};
```

Outside class definition:

Syntax: return type className :: functionName(list of parameters)

```
{
    //body of function

}
```

VIII Resources required

Sr. No.	Name of Resource	Specification	Quantity	Remarks
1	Hardware: Computer System	Computer (i3-i5 preferable), RAM minimum 2 GB and onwards, HDD 40GB and above	As per batch size	For all Experiments
2	Operating system	Windows /LINUX		
3	Software	Turbo C++ Version 3.0 or any other		

IX Precautions

1. Handle computer system and peripherals with care.
2. Follow safety practices.

X Resources used

S. No.	Name of Resource	Specification
1	Computer System with broad specifications	RAM of minimum 2GB & onword HDD of 40G & onward
2	Software	Turbo C++ version 3.0
3	Any other resource used	

XI Result (Output of the Program)

we learned to implement classes & objects

XII Practical Related Questions

Note: Below given are few sample questions for reference. Teacher must design more such questions so as to ensure the achievement of identified CO.

(Note: Use Point VIII to X and XIII to XV for all relevant programming exercise use blank pages provided or attach more pages if needed.)

1. What is the difference between struct and class in C++?
2. State output of the following code:

```
#include<iostream.h>
class Empty {};
void main()
{
  cout << sizeof(Empty);
}
```

a) A non-zero value b) 0 c) Compile time error d) Runtime error

3. State True or False: Data items in C++ class are by default public.

(Space for answers)

struct	class
i) struct is a keyward used to declare structure	i) class is a keyward used to declare class
ii) It is collection of data members	ii) It is collection of data members & members fu'ns
iii) Syntax: struct structname { data members; };	iii) Syntax: class class name { access specifies data members member-fu'ns };

2) Output :- 1

3) Ans - false

XIII Exercise

Attempt Q1 or Q2 and Q.3 a or b from the following:
(Note: Use Point VIII to X and XIII to XV for all relevant programming exercise use blank pages provided or attach more pages if needed.)

1. Write a C++ program to declare a class "Book" having data members book_name, author and price. Accept and display data for book having maximum price.

2. Write a C++ program to declare a class "staff" having data members name, basic salary, DA, HRA and calculate gross salary. Accept and display data for one staff.
   a. Where DA=74.5% of basic
      i. HRA= 30% of basic.
      ii. Gross_salary=basic+HRA+DA

3. Complete the given table:

Program Code	Write & justify Output

a) #include<iostream.h>
class Empty { };
 	
void main()
{
Empty a, b;
 	
if (a == b)
cout << "impossible " <<
endl;
else
cout << "Fine " << endl;
}
	Error: Illegal structure operation

Program Code	Write & justify Output

b) #include <iostream.h>
class Rectangle {
int width, height;
public:
void set_values (int,int);
int area();
{return width*height;}
};
void Rectangle::set_values (int
x, int y) {
width = x;
height = y;
}
void main () {
rectangle rect;
rect.set_values (3,4);
cout << "area: " <<
rect.area();
 	
}
	Area : 12

(Space for answers)

2) # include <iostream.h>
class staff
{
char name [20];
long int b_sal;
float DA, HRA, grass_sal;
public:
void get ()
{
DA = 74.5 ; HRA = 30;
cout << "\n Enter name:";
cin >> name;
cout << "\n Enter basic salary:";
cin >> b_sal;
grass_sal = b_sal + HRA + DA;
cout << "Gross sal =" << grass_sal;
}
void main()
{
staff S;
S.get ();
}

XIV References / Suggestions for further Reading

1. https://www.geeksforgeeks.org/c-classes-and-objects/
2. https://www.tutorialspoint.com/cplusplus/cpp_classes_objects.htm

XV Assessment Scheme

Performance indicators		Weightage
Process related(35 Marks)		**70%**
1	Logic formation	20%
2	Appropriate class and object definition	20%
3	Debugging ability	20%
4	Follow ethical practices.	10%
Product related (15 Marks)		**30%**
5	Expected Output	10%
6	Timely Submission of report	10%
7	Answer to sample questions	10%
Total (50 Marks)		**100%**

List of Students /Team Members

1. .....
2. .....
3. .....
4. .....

Marks Obtained			Dated signature of Teacher
Process Related(35)	Product Related(15)	Total(50)	
35	14	49/50	

Practical No. 7: Program to Implement Array of Objects

I Practical Significance:

The classes and objects help to represent real life entity with different attributes and related member functions.
Array of objects are used to represent the data of similar type.

II Relevant Program Outcomes (POs)

- **Basic knowledge:** Apply knowledge of basic mathematics, sciences and basic engineering to solve the broad-based Computer engineering problem.
- **Discipline knowledge:** Apply Computer engineering discipline - specific knowledge to solve core computer engineering related problems.
- **Experiments and practice:** Plan to perform experiments and practices to use the results to solve broad-based Computer engineering problems.
- **Engineering tools:** Apply relevant Computer technologies and tools with an understanding of the limitations.
- **Communication:** Communicate effectively in oral and written form.

III Competency and Practical skills

This practical is expected to develop the following skills in you :

Develop C++ programs to solve broad-based problems

1.Define real life entity into classes and objects.
2..Define and use classes and array of objects.
3..Compile the program.
4..Debug and execute the program.

IV Relevant Course Outcome(s)

Develop C++ programs using classes and objects.

V Practical Outcome (POs)

a)Write/ Compile/ debug / Execute simple C++ program using classes and array of objects.

VI Relevant Affective domain related Outcome(s)

1. Select proper programming environment in C++.
2. Follow safety measures
3. Follow ethical practices.

VII Minimum Theoretical Background

Array of objects: Arrays of variables of type "class" is known as "Array of objects".

Declaring Array of Objects:

Syntax:

```
className ObjectArrayName[size];
```

eg.

```
Student std[5];
```

Accessing data members and member functions: The data members and member functions of class can be accessed using the dot('.') operator with the object.
For example if the name of object is *std[2]* and you want to access the member function with the name *getData()* then you will have to write *std[2].getData()* .

VIII Resources required

Sr. No.	Name of Resource	Specification	Quantity	Remarks
1	Hardware: Computer System	Computer (i3-i5 preferable), RAM minimum 2 GB and onwards, HDD 40GB and above	As per batch size	For all Experiments
2	Operating system	Windows /LINUX		
3	Software	Turbo C++ Version 3.0 or any other		

IX Precautions

1. Handle computer system and peripherals with care.
2. Follow safety practices.

X Resources used

S. No.	Name of Resource	Specification
1	Computer System with broad specifications	RAM of 2 GB
2	Software	Turbo C version 3
3	Any other resource used	–

XI Result (Output of the Program)

Output of 9th program was the array of 3 object

XII Practical Related Questions

Note: Below given are few sample questions for reference. Teacher must design more such questions so as to ensure the achievement of identified CO.

(**Note:** Use Point VIII to X and XIII to XV for all relevant programming exercise use blank pages provided or attach more pages if needed.)

1. State the output of the following program:

```
#include<iostream.h>
class books
{
  char title [30];
  float price ;
  public:
  void getdata ();
  void putdata ();
} ;
void books :: getdata ()
{
  cout<<"Title:";
  Cin>>title;
  cout<<"Price:";
  cin>>price;
}

void books :: putdata ()
{
  cout<<"Title:"<<title<< "\n";
  cout<<"Price:"<<price<< "\n";
  const int size=3 ;
}
void main ()
{
  books book[size] ;
  for(int i=0;i<size;i++)
  {
  cout<<"Enter details of book "<<(i+1)<<"\n";
  book[i].getdata();
  }
  for(int i=0;i<size;i++)
  {
  cout<<"\nBook "<<(i+1)<<"\n";
  book[i].putdata() ;
}
```

(Space for answers)

Practical Related Questions

1) OUTPUT :-

Enter details of book 1
Title : OOP
Price : 250
Enter details of book 2
Title : CGR
Price : 300
Enter details of book 3
Title : DMS
Price : 270

Book 1	Book 2	Book 3
Title : OOP	Title : CGR	Title : DMS
Price : 250	Price : 300	price : 270

XIII. Exercise

Attempt Q1 or Q2 and Q3 a or b from the following:

(Note: Use Point VIII to X and XIII to XV for all relevant programming exercise use blank pages provided or attach more pages if needed.)

1. Write a C++ program to define a class "Employee" having data members emp_id, emp_name and emp_salary. Accept and display data for employees having salary greater than 25,000/-.

2. Write a C++ program to define a class "City" having data members name, population. Accept and display data for 10 cities.

3. Complete the given table:

Program Code	Write & justify Output

```
a) #include <iostream.h>

class Student
{
char name[20]; int marks; public:
void getName()
{
cin>>name;
}
void getMarks()
{
cin >> marks;
}
void displayInfo()
{
cout << "Name : " << name <<
endl;
cout << "Marks : " << marks <<
endl;
}
};

void main()
{
Student st[5];
for( int i=0; i<5; i++ )
{
cout << "Student " << i + 1 <<
```

Student 1
Enter Name
manasi
Enter marks
85
Student 2
Enter Name
Jana
Enter marks
79
Student 3
Enter Name
Rava
Enter mark
75
Student 4
Enter name
Neha
Enter mark
70
Student 5
Enter name
sharaddha
Enter marks
72


```
endl;
cout << "Enter name" << endl;
st[i].getName();
cout << "Enter marks" << endl;
st[i].getMarks();
}

for( int i=0; i<5; i++ )
{
cout << "Student " << i + 1 <<
endl;
st[i].displayInfo();
}
}
```

Student 1
Name : manasi
marks : 85
Student 2
Name : Jana
marks : 79
Student 3
Name : Rava
marks : 75
Student 4
Name : Neha
marks : 70
Student 5
Name : shraddha
marks : 72

```
b) #include<iostream.h>
   #include<conio.h>
      class Employee
      {
          int Id;
          char Name[25];
          int Age;
          long Salary;
          public:


      void GetData()
      {
       cout<<"\n\tEnter Employee Id : ";
       cin>>Id;
       cout<<"\n\tEnter Employee Name : ";
       cin>>Name;
       cout<<"\n\tEnter Employee Age : ";
       cin>>Age;
       cout<<"\n\tEnter Employee Salary : ";
       cin>>Salary;
      }
       void PutData()
      {

cout<<"\n"<<Id<<"\t"<<Name<<"\t"<<Age<<"\t"<<Salary;
      }
   };
   void main()
   {
       int i;
       Employee E[3];
       for(i=0;i<3;i++)
       {
        cout<<"\nEnter details of "<<i+1<<"
              Employee";
        E[i].GetData();
       }

       cout<<"\nDetails of Employees";
       for(i=0;i<3;i++)
          E[i].PutData();
   }
```

Enter details of 1 Employee
Enter employee ID: 1
Enter employee name : A
Enter employee Salary: 25000
Enter details of 2 employee
Enter employee ID : 2
Enter employee name : B
Enter employee age: 28
Enter employee Salary : 30,000
Enter details of 3 employee
Enter employee Name : C
Enter employee ID = 3
Enter employee age : 30
Enter salary = 40,000

(Space for answers)

Q1)
```
# include <iostream.h>
# include <conio.h>
class Employee
 {
    int id;
    char name [20];
    long int salary;
    public:
    void get data ()
    {
    cout << "\n Enter emp ID : " ;
      cin >> id;
    cout << " \n Enter name : " ;
      cin >> name;
    cout << \n Enter salary : " ;
      cin >> salary;
    }
    void put data ()
    {
      cout << " \n id : " << id ;
      cout << " \n Name : << name;
      cout << " \n salary : " << salary;
    }
};
void main ()
  {
    int i;
    Employee e [3];
    for (i=0 ; i<3 ; i++)
    {
    cout << " Enter details : " ;
    e[i]. get data ();
    }
    for (i=0 ; i<3 ; i++)
    { e[i] put data ();
    }
}
```

Practical No. 8: Program to Implement Friend Function

I **Practical Significance:**
The classes and objects help to represent real life entity with different attributes a
related member functions.
The use of friend function allows non-member functions to access the private a
protected data of the class.

II **Relevant Program Outcomes (POs)**

- **Basic knowledge:** Apply knowledge of basic mathematics, sciences and bas
engineering to solve the broad-based Computer engineering problem.
- **Discipline knowledge:** Apply Computer engineering discipline - specifi
knowledge to solve core computer engineering related problems.
- **Experiments and practice:** Plan to perform experiments and practices to use th
results to solve broad-based Computer engineering problems.
- **Engineering tools:** Apply relevant Computer technologies and tools with a
understanding of the limitations.
- **Communication:** Communicate effectively in oral and written form.

III **Competency and Practical skills**
This practical is expected to develop the following skills in you :

Develop C++ programs to solve broad-based problems

1. Define real life entity into classes and objects.
2. Define and use of friend function.
3. Compile the program.
4. Debug and execute the program.

IV **Relevant Course Outcome(s)**
Develop C++ programs using classes and objects.

V **Practical Outcome (POs)**
Write/ Compile/ debug / Execute simple C++ program using classes, objects and friend function.

VI **Relevant Affective domain related Outcome(s)**

1. Select proper programming environment in C++.
2. Follow safety measures
3. Follow ethical practices.

VII **Minimum Theoretical Background**

- **Friend Function:** A friend function of a class is defined outside that cla
scope but it has the right to access all private and protected members of th
class. Even though the prototypes for friend functions appear in the cla
definition, friends are not member functions. **friend** keyword is used to declare th
function as friend of class.
- **Declaring friend function**
Syntax:

```
class className
{
        //other members declaration
public:
        friend return type friendFunctionName(list of parameters);
        //other members declaration
};
return type friendFunctionName(list of parameters)
{
        //body of friend function
}
```


VIII **Resources required**

Sr. No.	Name of Resource	Specification	Quantity	Remarks
1	Hardware: Computer System	Computer (i3-i5 preferable), RAM minimum 2 GB and onwards, HDD 40GB and above	As per batch size	For all Experiments
2	Operating system	Windows /LINUX		
3	Software	Turbo C++ Version 3.0 or any other		

IX **Precautions**

1. Handle computer system and peripherals with care.
2. Follow safety practices.

X **Resources used**

S. No.	Name of Resource	Specification
1	Computer System with broad specifications	RAM minimum 2 GB & HDD 500 GB
2	Software	Turbo C Version 3
3	Any other resource used	-

XI **Result (Output of the Program)**

we executed program to implement friend function

XII **Practical Related Questions**
Note: Below given are few sample questions for reference. Teacher must design more such questions so as to ensure the achievement of identified CO.
(Note: Use Point VIII to X and XIII to XV for all relevant programming exercise use blank pages provided or attach more pages if needed.)

1. Can member functions of one class be friend functions of another class?
   a) Yes b) No
2. A function can be declared as friend maximum only in two classes.
   a) True b) False
3. Which of the following rules will not affect the friend function?
   a) private and protected members of a class cannot be accessed from outside
   b) private and protected member can be accessed anywhere
   c) protected member can be accessed anywhere
   d) none of the mentioned
4. Where does keyword 'friend' should be placed?
   a) function declaration
   b) function definition
   c) main function
   d) none of the mentioned

(Space for answers)

1) Ans - b - No
2) Ans - b - false
3) Ans - a - private & protected members of a class cannot be accessed from outside
4) Ans - function declaration

XIII Exercise

Attempt Q1 or Q2 and Q.3 a or b from the following:
(Note: Use Point VIII to X and XIII to XV for all relevant programming exercise use blank pages provided or attach more pages if needed.)

1. Write a C++ program to swap the values of two variables using friend function.
2. Write a C++ program to define a class "Distance" having data member as meters.
   Display the addition of two Distance objects using friend function.
3. Complete the given table:

Program Code	Write & justify Output

a)
#include <iostream.h>
class A
{
int a;
public:
A() {a = 10;}
friend void showA(A&);
};
void showA(A& x) {
cout << "a=" << x.a;
}
void main()
{
A a;
showA(a);
}
	output :
	a = 10

#include <iostream.h>
class Box
{
double width;
public:
friend void printWidth(Box box);
void setWidth(double wid);
};
void Box::setWidth(double wid)
{
width = wid;
}
void printWidth(Box box)
{
box.width = box.width * 2;
cout << "Width of box : " << box.width << endl;
}
void main( )
{
Box box;
box.setWidth(10.0);
printWidth(box);
}
	width of box : 20

(Space for answers)

```
1) #include <iostream.h>
   class swap
   {
      int temp, a, b;
      public:
      swap (int a, int b)
      {
      a = a;
      b = b;
      }
      friend void swap (swap);
   };
   void swap ( swap s1 )
   {
   cout<< " In Before swapping : "
         << s1.a<< s1.b;
      s1. temp = s1.a
      s1. a    = s1.b
      s1. b    = s1.temp
      cout<<" In After swapping"
            << s1.a << s1.b;
   }
   void main ()
   {
   swap s(4,6);
   swap (s);
   }
```

Practical No. 9: Program to Implement Inline Function

I Practical Significance:

The classes and objects help to represent real life entity with different attributes and related member functions.

The use of inline functions facilitates faster execution of the program.

II Relevant Program Outcomes (POs)

- **Basic knowledge:** Apply knowledge of basic mathematics, sciences and basic engineering to solve the broad-based Computer engineering problem.
- **Discipline knowledge:** Apply Computer engineering discipline - specific knowledge to solve core computer engineering related problems.
- **Experiments and practice:** Plan to perform experiments and practices to use the results to solve broad-based Computer engineering problems.,
- **Engineering tools:** Apply relevant Computer technologies and tools with an understanding of the limitations.
- **Communication:** Communicate effectively in oral and written form.

III Competency and Practical skills

This practical is expected to develop the following skills in you :

Develop C++ programs to solve broad-based problems

1. Define real life entity into classes and objects.
2. Define and use of inline function.
3. Compile the program.
4. Debug and execute the program.

IV Relevant Course Outcome(s)

Develop C++ programs using classes and objects.

V Practical Outcome (POs)

Write/ Compile/ debug / Execute simple C++ program using classes, objects and inline functions.

VI Relevant Affective domain related Outcome(s)

1. Select proper programming environment in C++.
2. Follow safety measures
3. Follow ethical practices.

VII Minimum Theoretical Background

Inline Function: C++ inline function is concept that is commonly used with classes. If a function is inline, the compiler places a copy of the code of that function at each point where the function is called at compile time.

A function definition in a class definition is an inline function definition, even without the use of the **inline** keyword.

To inline a function, place the keyword **inline** before the function name and define the function before any calls are made to the function. The compiler can ignore the inline qualifier in case defined function is more than a line.

Declaring inline function:

Syntax:

```
class className
{
    //other members declaration public:
        inline return type FunctionName(list of parameters)
        {
            //body of inline function
        }
    //other members declaration
};
```

VIII Resources required

Sr. No.	Name of Resource	Specification	Quantity	Remarks
1	Hardware: Computer System	Computer (i3-i5 preferable), RAM minimum 2 GB and onwards, HDD 40GB and above	As per batch size	For all Experiments
2	Operating system	Windows /LINUX		
3	Software	Turbo C++ Version 3.0 or any other		

IX Precautions

1 Handle computer system and peripherals with care.

2 Follow safety practices.

X Resources used

S. No.	Name of Resource	Specification
1	Computer System with broad specifications	RAM 2GB & HDD 500 GB
2	Software	Turbo C

XI Result (Output of the Program)

We executed a program to implement Inline functions.

XII Practical Related Questions

Note: Below given are few sample questions for reference. Teacher must design more such questions so as to ensure the achievement of identified CO.

(**Note:** Use Point VIII to X and XIII to XV for all relevant programming exercise use blank pages provided or attach more pages if needed.)

1. What does the inline keyword do?
   A. Indicates a function declaration
   B. Tells the compiler to use the function only within the same source code file
   C. Causes all function calls to be replaced by the code from the function
   D. Allows one-line function declarations

2. Why would you want to use inline functions?
   A. To decrease the size of the resulting program
   B. To increase the speed of the resulting program
   C. To simplify the source code file
   D. To remove unnecessary functions

3. Which of the following is a limit on inline functions?
   A. Inline functions cannot return a value.
   B. Inline functions must return a value.
   C. Inline functions must be less than ten lines.
   D. The compiler may choose to ignore an inline directive

4. Which of the following is a valid inline for function foo?
   A. inline void foo() {}
   B. void foo() inline {}
   C. inline:void foo() {}
   D. None of the above

(Space for answers)

1) → causes all functions calls to be replaced by the code & to m the function.

2) → To increase the speed of resulting program

3) → The m compiler may choose to ignore on inline directive

4) → inline void foo() {}

XIII Exercise

Attempt Q1 and Q.2 from the following:

(**Note:** Use Point VIII to X and XIII to XV for all relevant programming exercise use blank pages provided or attach more pages if needed.)

1. Write a C++ program to create a class "Number" having data members x and y and perform mathematical operations like addition, subtraction, multiplication and division on two numbers using inline functions.

2. Complete the given table:

Program Code	Write & justify Output

a)#include <iostream.h>
 	
class InlineDemo
{
public:
int square(int s); // declare the function
};
 	
//use inline prefix
 	
inline int InlineDemo::square(int s)
{
return s*s;
}
 	
void main()
{
InlineDemo s;
cout<<"Square of a No is :"<<s.square(10);
}
	square of No is : 200

(Space for answers)

1) #1 include < iostream.h>
class Number
{

```
int x, y;
public:
inline void add( int a, int b );
{
x = a; y = b;
   cout << " In Addition = " << x + y;
}
inline void sub (int a, int b)
{      x = a;   y = b;
       cout << " In subtraction : " << x - y;
}
// inline void multi ( int a, int b )
{ x = a;   y = b;
   cout << " In multiplication = " << int x * y;
}
inline void div (int a, int b)
{ x = a;   y = b;
   cout << " In division = " << x / y;
}
};
void main ()
{
Number n;
n. add (10, 20);
n. sub (10, 20);
n. multi (30, 2);
n   div (30, 5);
}
```

Practical No. 10: Program to Implement Constructors and Destructors

I **Practical Significance:**
The classes and objects help to represent real life entity with different attributes and related member functions.
The use of constructor functions facilitates initialization of instance variables.
The use of destructor functions facilitates deallocation of object memory.

II **Relevant Program Outcomes (POs)**

- **Basic knowledge:** Apply knowledge of basic mathematics, sciences and basic engineering to solve the broad-based Computer engineering problem.
- **Discipline knowledge:** Apply Computer engineering discipline - specific knowledge to solve core computer engineering related problems.
- **Experiments and practice:** Plan to perform experiments and practices to use the results to solve broad-based Computer engineering problems.
- **Engineering tools:** Apply relevant Computer technologies and tools with an understanding of the limitations.
- **Communication:** Communicate effectively in oral and written form.

III **Competency and Practical skills**
This practical is expected to develop the following skills in you :

Develop C++ programs to solve broad-based problems

1. Define real life entity into classes and objects.
2. Define and use of constructor function.
3. Define and use of destructor function.
4. Implement constructor overloading.
5. Compile the program.
6. Debug and execute the program.

IV **Relevant Course Outcome(s)**
Develop C++ programs using classes and objects.

V **Practical Outcome (POs)**
Write/ Compile/ debug / Execute simple C++ program using constructors and destructors.

VI **Relevant Affective domain related Outcome(s)**

1. Select proper programming environment in C++.
2. Follow safety measures
3. Follow ethical practices.

VII **Minimum Theoretical Background**

- **Constructor:** Constructors are special member functions which perform initialization of every object. The Compiler calls the Constructor whenever an object is created. Constructors initialize values to instance variables after storage is allocated to the object.

  Constructors are of three types :
  1. Default Constructor
  2. Parameterized Constructor
  3. Copy Constructor

- **Declaring constructor functions:**
  Syntax:

```
class className
{
    //other members declaration
    public:
        className(list of parameters)
        {
            //body of constructor function
        }
    //other members declaration
};
```

- **Constructor Overloading:**

Just like other member functions, constructors can also be overloaded. In fact when you have both default and parameterized (more than one) constructors defined in your class you are having **Overloaded Constructors**, one with no parameter and other with parameter.

- **Declaring overloaded constructor functions:**
  Syntax:

```
class className
{
    //other members declaration
    public:
        className()  //Default constructor
        {
            //body of default constructor function
        }
        className(list of parameters)  //Parameterized constructor
        {
            //body of parameterized constructor function
        }
        className(className &obj)  //copy constructor
        {
            //body of copy constructor function
        }
    //other members declaration
};
```

Destructor: Destructor is a special member 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.

Declaring destructor function:

Syntax:

```
class className
{
        //other members
        declaration public:
        //other members declaration

            ~className()
            {
                //body of destructor
                    function
            }
};
```

VIII Resourcesrequired

Sr. No.	Name of	Specification	Quantity	Remarks
1	Hardware: Computer System	Computer (i3-i5 preferable), RAM minimum 2 GB and onwards, HDD 40GB and above	As per batch size	For all Experiments
2	Operating system	Windows /LINUX		
3	Software	Turbo C++ Version 3.0 or any other		

IX Precautions

1. Handle computer system and peripherals with care.
2. Follow safety practices.

X Resources used

S. No.	Name of Resource	Specification
1	Computer System with broad specifications	RAM of 2GB & HDD 500 GB
2	Software	Turbo C 3
3	Any other resource used	

XI Result (Output of the Program)

.....We executed a program for constructor and destructor.....

XII Practical Related Questions

Note: Below given are few sample questions for reference. Teacher must design more such questions so as to ensure the achievement of identified CO.

(Note: Use Point VIII to X and XIII to XV for all relevant programming exercise use blank pages provided or attach more pages if needed.)

1. What happens when a class with parameterized constructors and having no default constructor is used in a program and we create an object that needs a zero-argument constructor?
   a. Compile-time error. b. Preprocessing error. c. Runtime error. d. Runtime exception.
2. For automatic objects, constructors and destructors are called each time the objects a. enter and leave scope
   b. inherit parent class c. are constructed
   d. are destroyed
3. Which of the following statement is correct about the program given below?

```
#include<iostream.h>
class abc
{
  public:
  abc()
  {
    cout<< "Welcome";
  }
  ~abc()
  {
    cout<< "India";
  }
};
void main()
{
  abc obj;

}
```

   a. The program will print the output Welcome.
   b. The program will print the output India.
   c. The program will print the output Welcome India.
   d. The program will report compile time error.

(Space for answers)

2) → are constructed

3) → The program will print the output welcome

XIII Exercise

Attempt Q1 and Q.2 from the following:

(**Note:** Use Point VIII to X and XIII to XV for all relevant programming exercise use blank pages provided or attach more pages if needed.)

1. Write a C++ program to define a class "Number" having data members x and y and perform mathematical operations like addition, subtraction, multiplication and division on two numbers using constructor and destructor functions.
2. Write a C++ program to define a class "Product" having data members Prod_id, Prod_name, Prod_price. Accept and display data for 3 products using constructor overloading.

3. Complete the given table:

Program Code	Write & justify Output

a)
`#include<iostream.h>`
`int val = 0;`
`class Test`
`{`
`public:`
`Test()`
`{`
`cout<< ++val;`
`}`
`~Test()`
`{`
`cout<< val--;`
`}`
`};`
`void main()`
`{`
`Test obj1, obj2, obj3;`
`{`
`Test obj4;`
`}`
`}`
	Output :-
	1 2 3 4 4

b)
`#include <iostream.h>`
`class construct`
`{`
`public:`
`float area;`
`// Constructor with no parameters`
`construct()`
`{`
`area = 0;`
`}`
`construct(int a, int b)`
`{`
`area = a * b;`
`}`
`void disp()`
`{`
`cout<< area<< endl;`
`}`
`};`
`void main()`
`{`
`// Constructor Overloading`
`// with two different constructors`
`// of class name`
`construct o;`
`construct o2( 10, 20);`
`o.disp();`
`o2.disp();`
`}`
	Output :-
	200

(Space for answers)

```
1) # include (iostream.h)
   class Number
   {
   int x, y;
   public
   Number (int a, int b)
   {
     x = a;  y = b;
     cout << " inAddition = " << x + y;
   }
   Number (int a)
   {
   x = a;  y = b;
   cout << " insubtraction = " << x - y;
   Number ()
   {
     x = 5;  y = 3;
     cout << " in multiplication : " << x * y;
   }
   Number ()
   {
     x = 36 ;  y = 4;
     cout << " indivision : " << x / y;
   }
   };
   void main ()
   {
   Number n (10, 20);
   Number n1 (30);
   number n2;
   getche ();
   }
```


XIV References / Suggestions for further Reading

1. https://www.geeksforgeeks.org/c-classes-and-objects/
2. https://www.tutorialspoint.com/cplusplus/cpp_classes_objects.htm
3. http://www.studytonight.com/cpp/constructors-and-destructors-in-cpp
4. https://www.geeksforgeeks.org/constructor-overloading-c/

XV Assessment Scheme

	Performance indicators	Weightage
	Process related(35 Marks)	**70%**
1	Logic formation	20%
2	Appropriate constructor, destructor function declaration and definition	20%
3	Debugging ability	20%
4	Follow ethical practices.	10%
	Product related (15 Marks)	**30%**
5	Expected Output	10%
6	Timely Submission of report	10%
7	Answer to sample questions	10%
	Total (50 Marks)	**100%**

List of Students /Team Members

1.....

2.....

3.....

4.....

Marks Obtained			Dated signature of Teacher
Process Related(35)	Product Related(15)	Total (50)	
35 + 15 =		50	Sukard

Practical No. 11: Program to Implement Single Inheritance

I Practical Significance:
The use of inheritance shows the reusability of the existing class properties and deriving a new class with additional properties.

II Relevant Program Outcomes (POs)

- **Basic knowledge:** Apply knowledge of basic mathematics, sciences and basic engineering to solve the broad-based Computer engineering problem.
- **Discipline knowledge:** Apply Computer engineering discipline - specific knowledge to solve core computer engineering related problems.
- **Experiments and practice:** Plan to perform experiments and practices to use the results to solve broad-based Computer engineering problems.
- **Engineering tools:** Apply relevant Computer technologies and tools with an understanding of the limitations.
- **Communication:** Communicate effectively in oral and written form.

III Competency and Practical skills
This practical is expected to develop the following skills in you :

Develop C++ programs to solve broad-based problems

1. Define real life example class and its derived class with additional properties.
2. Compile the program.
3. Debug and execute the program.

IV Relevant Course Outcome(s)
Implement Inheritance in C++ program.

V Practical Outcome (POs)
Write/ Compile/ debug / Execute simple C++ program using single inheritance.

VI Relevant Affective domain related Outcome(s)

1. Select proper programming environment in C++.
2. Follow safety measures
3. Follow ethical practices.

VII Minimum Theoretical Background

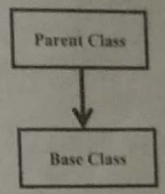
Inheritance: It is the process of inheriting properties of objects of one class by objects of another class. The class which inherits the properties of another class is called Derived or Child or Sub class and the class whose properties are inherited is called Base or Parent or Super class. When a single class is derived from a single parent class, it is called Single inheritance. It is the simplest of all inheritance

Types of Inheritance:

1. Single Inheritance
2. Multiple Inheritance
3. Multilevel Inheritance
4. Hierarchical Inheritance
5. Hybrid Inheritance

Syntax of Single Inheritance:

```
class base_classname
{
  properties;
  member functions;
};
class derived_classname : visibility_mode base_classname
{
  properties;
member functions;
};
```



Single Inheritance

Visibility Modes of Inheritance:

Base Class	Derived Class public Mode	Derived Class private Mode	Derived Class protected Mode
Private	Not inherited	Not inherited	Not inherited
Protected	protected	Private	Protected
Public	Public	Private	Protected

VIII Resources required

Sr. No.	Name of Resource	Specification	Quantity	Remarks
1	Hardware: Computer System	Computer (i3-i5 preferable), RAM minimum 2 GB and onwards, HDD 40GB and above	As per batch size	For all Experiments
2	Operating system	Windows /LINUX		
3	Software	Turbo C++ Version 3.0 or any other		

IX Precautions

1. Handle computer system and peripherals with care.
2. Follow safety practices.

X Resources used

S. No.	Name of Resource	Specification
1	Computer System with broad specifications	RAM of 2 GB & HDD 500 GB
2	Software	Turbo C
3	Any other resource used	

XI Result (Output of the Program)

output of program was single inheritance

XII Practical Related Questions

Note: Below given are few sample questions for reference. Teacher must design more such questions so as to ensure the achievement of identified CO.

(Note: Use Point VIII to X and XIII to XV for all relevant programming exercise use blank pages provided or attach more pages if needed.)

1. A derived class with only one base class is called …… inheritance.
   a) single
   b) multiple
   c) multilevel
   d) hierarchical
2. When a base class is privately inherited by the derived class, then _____ .
   a. protected members of the base class become private members of derived class
   b. public members of the base class become private members of derived class
   c. both a and b
   d. only b
3. The derivation of Child class from Base class is indicated by ___ symbol.
   a. ::
   b. :
   c. ;
   d. |

(Space for answers)

1) A drived class with only one base class is called single inheritance

2) → both a & b

3) → The derivation of child class from base class is indicated by : symbol.

XIII Exercise

Attempt Q1 and Q.2 from the following:

(Note: Use Point VIII to X and XIII to XV for all relevant programming exercise use blank pages provided or attach more pages if needed.)

1. Write a C++ program to define a class "Student" having data members roll_no, name . Derive a class "Marks" from "Student" having data members m1,m2,m3, total and percentage. Accept and display data for one student.

2. Write a C++ program to define a class "Employee" having data members emp_no, emp_name and emp_designation. Derive a class "Salary" from "Employee" having data members basic, hra, da, gross_sal. Accept and display data for one employee.

3. Complete the given table:

Program Code	Write & justify Output

a) `#include <iostream.h>`
`class Base`
`{`
`public:`
`Base(){}`
`~Base(){}`
`protected:`
`private:`
`};`
`class Derived:public Base`
`{`
`public:`
`Derived(){}`
`Derived(){}`
`private:`
`protected:`
`};`
`void main()`
`{`
`cout << "The program exceuted" << endl;`
`}`
	The program executed

b) `#include<iostream.h>`
`class Test`
`{`
`int value;`
`public:`
`Test(int v = 0) { value = v; }`
`int getValue() { return value; }`
`};`
`void main()`
`{`
`const Test t;`
`cout << t.getValue();`
`}`
	o/p value

(Space for answers)

1)
```
# include < iostream.h >
class student
{ int roll-no,
  char name[20];
  public
  void get()
  {
  cout << " In Enter roll no:";
  cin >> roll-no;
  cout << " In Enter name: ";
  cin >> name;
  }
};
class marks : public student
{ int m1, m2, m3;
  float per, total;
  public
  void get:()
  {
  cout << " In Enter marks : ";
  cin >> m1 >> m2 >> m3;
  total = m1 + m2 + m3;
  per = ( total / 300)*100;
  cout << " In Percentage =" << per;
  }: }
void main() { marks m;
              m. get()
              m get(); }
```

XIV. **References / Suggestions for further Reading**

1. https://www.programtopia.net/cplusplus/docs/single-inheritance/
2. https://www.geeksforgeeks.org/inheritance-in-c/
3. http://www.siteforinfotech.com/2017/05/top-20-mcq-questions-inheritance-in- cpp.html
4. https://www.careerride.com/mcq/inheritance-c-mcq-questions-and-answers-114.aspx
5. https://www.geeksforgeeks.org/output-of-c-program-set/
6. https://ide.geeksforgeeks.org/index.php

XV **Assessment Scheme**

Performance indicators		Weightage
Process related (35 Marks)		**70%**
1	Logic formation	20%
2	Appropriate base class and derived class declaration using single inheritance	20%
3	Debugging ability	20%
4	Follow ethical practices.	10%
Product related (15 Marks)		**30%**
5	Expected Output	10%
6	Timely Submission of report	10%
7	Answer to sample questions	10%
Total (50 Marks)		**100%**

List of Students /Team Members

1. .....
2. .....
3. .....
4. .....

Marks Obtained			Dated signature of Teacher
Process Related(35)	Product Related(15)	Total(50)	
34	15	49/50	Saisawadh

Practical No. 12: Program to Implement Multi Level Inheritance

I **Practical Significance:**
The use of inheritance shows the reusability of the existing class properties and deriving a new class with additional properties.

II **Relevant Program Outcomes (POs)**

- **Basic knowledge:** Apply knowledge of basic mathematics, sciences and basic engineering to solve the broad-based Computer engineering problem.
- **Discipline knowledge:** Apply Computer engineering discipline - specific knowledge to solve core computer engineering related problems.
- **Experiments and practice:** Plan to perform experiments and practices to use the results to solve broad-based Computer engineering problems.
- **Engineering tools:** Apply relevant Computer technologies and tools with an understanding of the limitations.
- **Communication:** Communicate effectively in oral and written form.

III **Competency and Practical skills**
This practical is expected to develop the following skills in you :

Develop C++ programs to solve broad-based problems

1. Define real life example class and its multilevel derived class hierarchy with additional properties.
2. Compile the program.
3. Debug and execute the program.

IV **Relevant Course Outcome(s)**
Implement Inheritance in C++ program.

V **Practical Outcome (POs)**
Write/ Compile/ debug / Execute simple C++ program using multilevel inheritance.

VI **Relevant Affective domain related Outcome(s)**

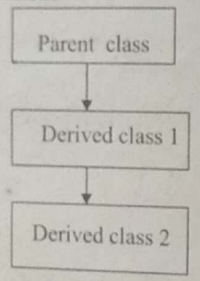
1. Select proper programming environment in C++.
2. Follow safety measures
3. Follow ethical practices.

VII **Minimum Theoretical Background**
Multilevel Inheritance: In C++ programming, not only you can derive a class from the base class but you can also derive a class from the derived class. This form of inheritance is known as multilevel inheritance.

Syntax of multilevel Inheritance:

```
class base_classname
{
  properties;
  member functions;
};
class derived_classname1 : visibility_mode base_classname
{
  properties;
  member functions;
};
class derived_classname2 : visibility_mode derived_classname1
{
  properties;
  member functions;
};
```

Pictorial Representation:



VIII Resources required

Sr. No.	Name of Resource	Specification	Quantity	Remarks
1	Hardware: Computer System	Computer (i3-i5 preferable), RAM minimum 2 GB and onwards, HDD 40GB and above	As per batch size	For all Experiments
2	Operating system	Windows /LINUX		
3	Software	Turbo C++ Version 3.0 or any other		

IX Precautions

1. Handle computer system and peripherals with care.
2. Follow safety practices.

X Resources used

S. No.	Name of Resource	Specification
1	Computer System with broad specifications	RAM of 2 GB
2	Software	Turbo C
3	Any other resource used	

XI Result (Output of the Program)

output of program was
mutilevel inheritance

XII Practical Related Questions

Note: Below given are few sample questions for reference. Teacher must design more such questions so as to ensure the achievement of identified CO.

(Note: Use Point VIII to X and XIII to XV for all relevant programming exercise use blank pages provided or attach more pages if needed.)

1.

```
#include <iostream.h>
class A
{
public:
void print() { cout << "print() in A"; }
};

class B : private A
{
public:
void print() { cout << "print() in B"; }
};

class C : public B
{
public:
void print() {
cout << "print() in
C"; A::print(); }
};
void main()
{


C b;
b.print();
}
```

a) print() in A b) print() in B
c) Compile time error d) None of the above

2. A class can be derived from another derived class which is known as ..... inheritance.
A) single B) multiple C) multilevel D) hierarchical

3. In ..... inheritance, the constructors are executed in the order of inheritance.
A) multipath B) multiple C) multilevel D) hierarchical

4. base class and derived class relationship comes under
A) Inheritance B) Polymorphism C) encapsulation D)None

(Space for answers)

2) -) A class can be derived from another derived classe which is known as multi level inheritance

3) → In multilevel inheritance, the constructors are executed in the order of inheritance.

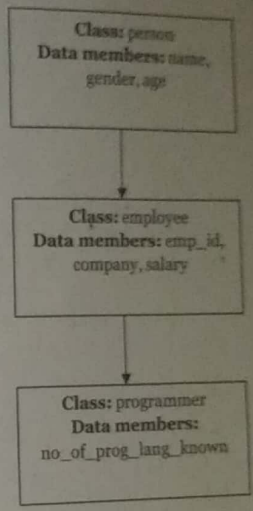
4) → Inheritanc

XIII Exercise

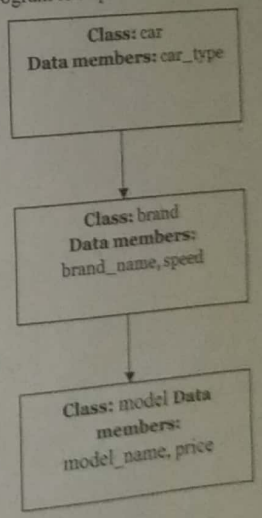
Attempt Q1 and Q.2 from the following:

(Note: Use Point VIII to X and XIII to XV for all relevant programming exercise use blank pages provided or attach more pages if needed.)

1 Write a C++ program to implement following Multilevel Inheritance



2 Write a C++ program to implement following Multilevel Inheritance



(Space for answers)

XIV References / Suggestions for further Reading

1. https://www.programtopia.net/cplusplus/docs/single-inheritance/
2. https://www.geeksforgeeks.org/inheritance-in-c/
3. http://www.siteforinfotech.com/2017/05/top-20-mcq-questions-inheritance- in-cpp.html
4. https://www.careerride.com/mcq/inheritance-c-mcq-questions-and-answers-114.aspx

XV Assessment Scheme

	Performance indicators	Weightage
	Process related(35 Marks)	**70%**
1	Logic formation	20%
2	Appropriate base class and derived classes declaration using multilevel inheritance	20%
3	Debugging ability	20%
4	Follow ethical practices.	10%
	Product related (15 Marks)	**30%**
5	Expected Output	10%
6	Timely Submission of report	10%
7	Answer to sample questions	10%
	Total (50 Marks)	**100%**

List of Students /Team Members

1. .....
2. .....
3. .....
4. .....

Marks Obtained			Dated signature of Teacher
Process Related(35)	**Product Related(15)**	**Total(50)**	
34 +	15 =	49	Sasarde

Practical No. 13: Program to Implement Concept Multiple Inheritance

I **Practical Significance:**
Multiple Inheritances is a feature of C++ where a class can inherit from more than one classes.

II **Relevant Program Outcomes (POs)**

- **Basic knowledge:** Apply knowledge of basic mathematics, sciences and basic engineering to solve the broad-based Computer engineering problem.
- **Discipline knowledge:** Apply Computer engineering discipline - specific knowledge to solve core computer engineering related problems.
- **Experiments and practice:** Plan to perform experiments and practices to use the results to solve broad-based Computer engineering problems.
- **Engineering tools:** Apply relevant Computer technologies and tools with an understanding of the limitations.
- **Communication:** Communicate effectively in oral and written form.

III **Competency and Practical skills**
This practical is expected to develop the following skills in you :

Develop C++ programs to solve broad-based problems
1. Apply multiple inheritances to real life problems.
2. Compile the program.
3. Debug and execute the program.

IV **Relevant Course Outcome(s)**
Implement Inheritance in C++ program.

V **Practical Outcome (POs)**
Write/ Compile/ debug / Execute simple C++ program using multiple inheritance.

VI **Relevant Affective domain related Outcome(s)**
1. Select proper programming environment in C++.
2. Follow safety measures
3. Follow ethical practices.

VII **Minimum Theoretical Background**

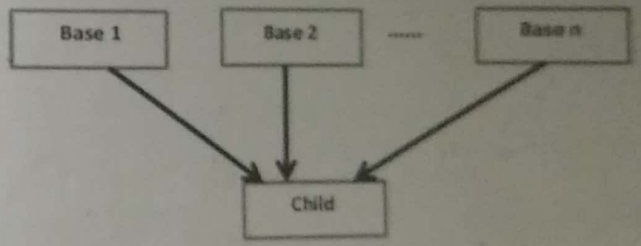
1. **Inheritance** :-It is the process of inheriting properties of objects of one class by objects of another class. The class which inherits the properties of another class is called Derived or Child or Sub class and The class whose properties are inherited is called Base or Parent or Super class.
2. **Multiple Inheritance** :-When a class is derived from two or more base classes such inheritance is called Multiple Inheritance. It allow us to combine the features of several existing classes into a single class.

Syntax of Multiple Inheritance

```
class base_class1
{
   properties;
   member functions;
};

class base_class2
{
   properties;
   member functions;
};
... ... ...
... ... ...
class base_classN
{
   properties;
   member functions;
};

class derived_classname : visibility_mode base_class1,
visibility_mode base_class2,... ,visibility_mode base_classN
{
   properties;
   member functions;
};
```

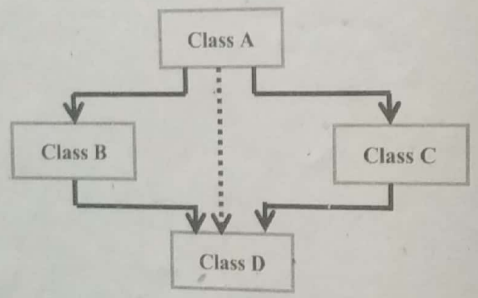


The different access specifier for inheritance concept

Access	public	protected	Private
Same class	yes	Yes	yes
Derived classes	yes	Yes	no
Outside classes	yes	No	no

Virtual Base Class:

1. Virtual Base class: An ambiguity can arise when several paths exist to a class from the same base class. This means that a child class could have duplicate sets of members inherited from a single base class.
C++ solves this issue by introducing a virtual base class. When a class is made virtual, necessary care is taken so that the duplication is avoided regardless of the number of paths that exist to the child class.
When two or more objects are derived from a common base class, we can prevent multiple copies of the base class being present in an object derived from those objects by declaring the base class as virtual when it is being inherited. Such a base class is known as virtual base class. This can be achieved by preceding the base class's name with the word **virtual**.



Syntax:

```
class derived_class_name: virtual visibility_mode base_class
{
  ----------

  ----------//members of derived class
};
```

VIII Resources required

Sr. No.	Name of Resource	Specification	Quantity	Remarks
1	Hardware: Computer System	Computer (i3-i5 preferable), RAM minimum 2 GB and onwards, HDD 40GB and above	As per batch size	For all Experiments
2	Operating system	Windows /LINUX		
3	Software	Turbo C++ Version 3.0 or any other		

IX Precautions

1. Handle computer system and peripherals with care.
2. Follow safety practices.

X Resources used

S. No.	Name of Resource	Specification
1	Computer System with broad specifications	RAM of 2GB & HDD of 500 GB
2	Software	Turbo C
3	Any other resource used	

XI Result (Output of the Program)

output of program was multilevelple inheritance

XII Practical Related Questions

Note: Below given are few sample questions for reference. Teacher must design more such questions so as to ensure the achievement of identified CO.

(Note: Use Point VIII to X and XIII to XV for all relevant programming exercise use blank pages provided or attach more pages if needed.)

1. What is the difference between multiple and multilevel inheritance in C++?
2. State output of the following code:

```
#include <iostream.h>
#include <conio.h>
class liquid
{
    float specific_gravity;
    public:
        void input()
        {
            cout<<"Specific gravity: ";
            cin>>specific_gravity;
        }
        void output()
        {
            cout<<"Specific gravity: "<<specific_gravity<<endl;
        }
};

class fuel
{
    float rate;
    public:
        void input()
        {
            cout<<"Rate(per liter): $";
            cin>>rate;
        }
        void output()
        {
            cout<<"Rate(per liter): $"<<rate<<endl;
```

```
};

class petrol: public liquid, public fuel
{
    public:
        void input()
        {
            liquid::input();
            fuel::input();
        }
        void output()
        {
            liquid::output();
            fuel::output();
        }
};

int main()
{
    petrol p;
    cout<<"Enter data"<<endl;
    p.input();
    cout<<endl<<"Displaying data"<<endl;
    p.output();
    getch();
    return 0;
}
```

3.

```
#include<iostream.h>
#include<conio.h>

class ClassA


{
      public:
      int a;
};

class ClassB : virtual public ClassA
{
      public:
      int b;
};
class ClassC : virtual public ClassA
{
      public:
      int c;
};

class ClassD : public ClassB, public ClassC
{
      public:
      int d;
};

void main()
{

  Class D obj;
  obj.a =10;
  obj.a= 100;

  obj.b = 20;
```



```
obj.d= 40;
cout<< "\n A : "<< obj.a;
cout<< "\n B : "<< obj.b;
cout<< "\n C : "<< obj.c;
cout<< "\n D : "<< obj.d;
}
```

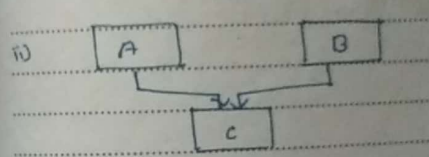
4. Which of the following advantages we lose by using multiple inheritance?
   a) Dynamic binding
   b) Polymorphism
   c) Both Dynamic binding & Polymorphism
   d) None of the mentioned

(Space for answers)

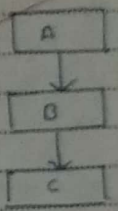
i) Multiple inheritance
i) In this class inherits more than one base class.

ii) multilevel inheritance
i) In this one derived class is derived from another derived class

ii)



i)



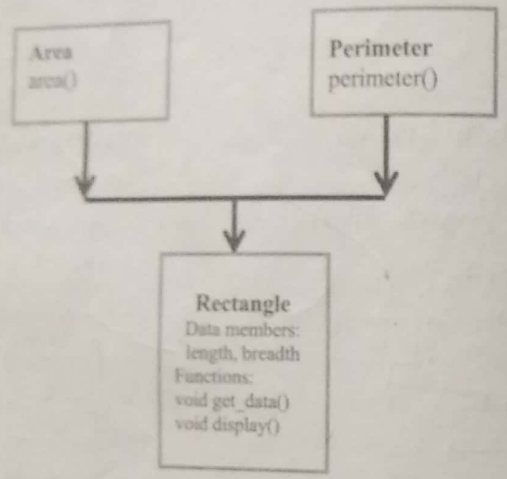
ercise

ttempt Q1 or Q2 and Q.3 a or b from the following:

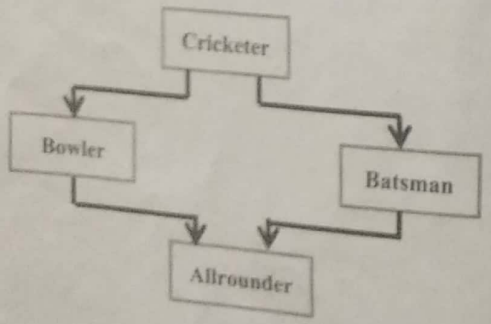
ote: Use Point VIII to X and XIII to XV for all relevant programming exercise use

ank pages provided or attach more pages if needed.)

Write a C++ program to calculate the area and perimeter of rectangles using concept of inheritance.



Write a C++ program for representation of class hierarchy as below .Assume suitable data and function members.



3 Complete the given table:

Program Code	Write & justify Output

a) `#include <iostream.h>`
`class Base`
`{`
`public:`
`virtual void print() const = 0;`
`};`
`class DerivedOne : public Base`
`{`
`public:`
`void print() const`
`{`
`cout << "DerivedOne\n";`
`}`
`};`
`class DerivedTwo : public Base`
`{`
`public:`
`void print() const`
`{`
`cout << "DerivedTwo\n";`
`}`
`};`
`class Multiple : public DerivedOne, public DerivedTwo`
`{`
`public:`
`void print() const`
`{`
`DerivedTwo :: print();`
`}`
`};`
`int main()`
`{`
`int i;`
`Multiple both;`
`DerivedOne one;`
`DerivedTwo two;`
`Base *array[ 3 ];`
`array[ 0 ] = &both;`
`array[ 1 ] = &one;`
`array[ 2 ] = &two;`
`array[ i ] -> print();`
`return 0;`
`}`
	array[0] = & both = cannot convert multiple *to base* !

b)
```
#include <iostream.h>
struct a
{
    int count;
};
struct b {
int*  value;
};
struct  c  :  public  a,  public  b
{
};
int  main()
{
c*  p  =  new  c;
p->value  =  0;
cout  <<  "Inherited";
return  0;
}
```

(Space for answers)

XIV References / Suggestions for further Reading

1. https://www.geeksforgeeks.org/multiple-inheritance-in-c/
2. https://www.programiz.com/cpp-programming/multiple-inheritance

XV Assessment Scheme

	Performance indicators	Weightage
	Process related(35 Marks)	**70%**
1	Logic formation	20%
2	Appropriate definition of multiple inheritance.	20%
3	Debugging ability	20%
4	Follow ethical practices.	10%
	Product related (15 Marks)	**30%**
5	Expected Output	10%
6	Timely Submission of report	10%
7	Answer to sample questions	10%
	Total (50 Marks)	**100%**

List of Students /Team Members

1. .....
2. .....
3. .....
4. .....

Marks Obtained			Dated signature of Teacher
Process Related(35)	**Product Related(15)**	**Total(50)**	
34	15	49	

Practical No. 14: Program to Implement Pointer to Object

I Practical Significance:

The pointer to objects helps to improve the efficiency of the program and access the objects dynamically.

II Relevant Program Outcomes (POs)

- **Basic knowledge:** Apply knowledge of basic mathematics, sciences and basic engineering to solve the broad-based Computer engineering problem.
- **Discipline knowledge:** Apply Computer engineering discipline - specific knowledge to solve core computer engineering related problems.
- **Experiments and practice:** Plan to perform experiments and practices to use the results to solve broad-based Computer engineering problems.
- **Engineering tools:** Apply relevant Computer technologies and tools with an understanding of the limitations.
- **Communication:** Communicate effectively in oral and written form.

III Competency and Practical skills

This practical is expected to develop the following skills in you :

Develop C++ programs to solve broad-based problems

1. Define real life problems using pointer to objects.
2. Compile the program.
3. Debug and execute the program.

IV Relevant Course Outcome(s)

Implement Inheritance in C++ program.
Use Polymorphism in C++ program.

V Practical Outcome (POs)

Write/ Compile/ debug / Execute simple C++ program using pointer to object.

VI Relevant Affective domain related Outcome(s)

1. Select proper programming environment in C++.
2. Follow safety measures
3. Follow ethical practices.

VII Minimum Theoretical Background

Pointers to objects:-

C++ allows you to have pointers to objects.
The pointers pointing to objects are referred to as Object Pointers.
Syntax:- class-name * object-pointer ;

where class-name is the name of an already defined class and object-pointer is the pointer to an object of this class type.
For example, to declare optr as an object pointer of Sample class type, we shall write Sample *optr ;where Sample is already defined class.
When accessing members of a class using an object pointer, the arrow operator (->) is used instead of dot operator.

VIII Resources required

Sr. No.	Name of Resource	Specification	Quantity	Remarks
1	Hardware: Computer System	Computer (i3-i5 preferable), RAM minimum 2 GB and onwards, HDD 40GB and above	As per batch size	For all Experiments
2	Operating system	Windows /LINUX		
3	Software	Turbo C++ Version 3.0 or any other		

IX Precautions

1. Handle computer system and peripherals with care.
2. Follow safety practices.

X Resources used

S. No.	Name of Resource	Specification
1	Computer System with broad specifications	RAM 2GB & HDD 500 GB
2	Software	Turbo C
3	Any other resource used	

XI Result (Output of the Program)

.....
.....We executed program to implement.....
.....pointer to object.....

XII Practical Related Questions

Note: Below given are few sample questions for reference. Teacher must design more such questions so as to ensure the achievement of identified CO.

(Note: Use Point VIII to X and XIII to XV for all relevant programming exercise use blank pages provided or attach more pages if needed.)

1. State output of the following code:

```
#include <iostream.h>
#include<conio.h>
class myclass
{
int i;
public:
        void read(int j)
{
        i= j;
  }
        int getint()
{
                return i;
```

```
};
void main()
{
    clrscr();
    myclass ob, *objectPointer;
    objectPointer = &ob;
    objectPointer->read(10);
    cout<<objectPointer->getint();
    getch();
}
```

2. Which is the pointer which denotes the object calling the member function?
   a) Variable pointer b) This pointer
   c) Null pointer d) Zero pointer

3. A pointer can be initialized with _____.
   a) null b) zero
   c) Address of an object of same type d) All of them

(Space for answers)

2) This pointer

3) Address of an object with type

Complete the given table:

Program Code	Write & justify Output

```
a)
#include<iostream.h>
#include<conio.h>
class Time
{
      short int hh, mm, ss;
      public:
            Time()
            {
                  hh = mm = ss = 0;
            }
            void getdata(int i, int j, int k)
            {
                  hh = i;
                  mm = j;
                  ss = k;
            }
            void prndata(void)
            {
cout<<"\nTime is "<<hh<<":"<<mm<<":"<<ss<<"\n";
      }
};
void main()
{
      clrscr();
      Time T1, *tptr;
      cout<<"Initializing data members using the
object, with values 12, 22, 11\n";
T1.getdata(12,22,11);
      cout<<"Printing members using the object
";
      T1.prndata();
      tptr = &T1;
      cout<<"Printing members using the object
pointer ";
      tptr->prndata();
      cout<<"\nInitializing data members using
the object pointer, with values 15, 10, 16\n";
      tptr->getdata(15, 10, 16);
      cout<<"printing members using the object
";
      T1.prndata();
      cout<<"Printing members using the object
      pointer "; tptr->prndata();
      getch();
      }
```

Time is:
12:22:11

Time is
12:22:11

Time is
15:10:16

Time is
15:10:16


```
b)#include <iostream>
#include <string>
using namespace std;
class student
{
private:
            int rollno;
            string name;
public:
            student():rollno(0),name("")
            {}
            student(int r, string n):
rollno(r),name (n)
            {}
            void get()
            {
                        cout<<"enter roll no";
                        cin>>rollno;
                        cout<<"enter  name";
                        cin>>name;
            }
            void print()
            {
                        cout<<"roll no is
"<<rollno;
                        cout<<"name is "<<name;
            }
};
void main ()
{
            student *ps=new student;
            (*ps).get();
            (*ps).print();
            delete ps;
}
```

o/p :
Rollno is
32 name
is A RAM

(Space for answers)

XIV **References / Suggestions for further Reading**

1. https://gradestack.com/Programming-in-C-/Pointers/Pointer-To-Object/
2. https://stackoverflow.com/questions/653423/pointer-to-objects

XV **Assessment Scheme**

Performance indicators		Weightage
Process related(35 Marks)		**70%**
1	Logic formation	20%
2	Appropriate use of pointer to object	20%
3	Debugging ability	20%
4	Follow ethical practices.	10%
Product related (15 Marks)		**30%**
5	Expected Output	10%
6	Timely Submission of report	10%
7	Answer to sample questions	10%
Total (50 Marks)		**100%**

List of Students /Team Members

1. .....
2. .....
3. .....

Marks Obtained			Dated signature of Teacher
Process Related(35)	Product Related(15)	Total(50)	
35	+ 15	= 50	Satisfied

Practical No. 15: Program to Implement Pointer Derived Class

I **Practical Significance:**

The key feature of class inheritance is that a pointer to a derived class is type-compatible with a pointer to its base class.

II **Relevant Program Outcomes (POs)**

- **Basic knowledge:** Apply knowledge of basic mathematics, sciences and basic engineering to solve the broad-based Computer engineering problem.
- **Discipline knowledge:** Apply Computer engineering discipline – specific knowledge to solve core computer engineering related problems.
- **Experiments and practice:** Plan to perform experiments and practices to use the results to solve broad-based Computer engineering problems.
- **Engineering tools:** Apply relevant Computer technologies and tools with an understanding of the limitations.
- **Communication:** Communicate effectively in oral and written form.

III **Competency and Practical skills**

This practical is expected to develop the following skills in you

Develop C++ programs to solve broad-based problems

1. Define pointer to derived class.
2. Compile the program.
3. Debug and execute the program.

IV **Relevant Course Outcome(s)**

Implement Inheritance in C++ program.
Use Polymorphism in C++ program.

V **Practical Outcome (POs)**

Write/ Compile/ debug / Execute simple C++ program using pointers to derived class.

VI **Relevant Affective domain related Outcome(s)**

1. Select proper programming environment in C++.
2. Follow safety measures
3. Follow ethical practices.

VII **Minimum Theoretical Background**

Pointers to derived class

1. C++ allows base class pointers to point to derived class objects.
2. Let we have –

```
class base { ... };
class derived : public base { ... };
```

Then we can write –

```
base *p1;
derived
```

d_obj; p1 =
&d_obj;
base *p2 = new derived;

3. Using a base class pointer (pointing to a derived class object) we can access only those members of the derived object **that were inherited from the base.**
   a. It is different from the behavior that Java shows.
   b. We can get Java-like behavior using virtual functions.
4. This is because the **base pointer** has knowledge only of the base class.
5. It knows nothing about the members added by the derived class.

VIII Resources required

Sr. No.	Name of Resource	Specification	Quantity	Remarks
1	Hardware: Computer System	Computer (i3-i5 preferable), RAM minimum 2 GB and onwards, HDD 40GB and above	As per batch size	For all Experiments
2	Operating system	Windows /LINUX		
3	Software	Turbo C++ Version 3.0 or any other		

IX Precautions

1. Handle computer system and peripherals with care.
2. Follow safety practices.

X Resources used

S. No.	Name of Resource	Specification
1	Computer System with broad specifications	RAM of 2GB & HDD 500 GB
2	Software	Turbo C
3	Any other resource used	

XI Result (Output of the Program)

output of program was a program to implement pointer derived class

XII Practical Related Questions

Note: Below given are few sample questions for reference. Teacher must design more such questions so as to ensure the achievement of identified CO.
(Note: Use Point VIII to X and XIII to XV for all relevant programming exercise use blank pages provided or attach more pages if needed.)

1. State output of the following code:

```
#include<iostream.h>
class base {
public:
   void show() {
      cout << "base\n";
   }
};
class derived : public base {
public:
   void show() {
      cout << "derived\n";
   }
};
void main() {
   base b1;
   b1.show();
   derived d1;
   d1.show();
   base *pb = &b1;
   pb->show();
   pb = &d1;
   pb->show();
}
```

2. A pointer to the base class can hold address of
   A) only base class object B) only derived class object
   C) base class object as well as derived class object D) None of the above

3. Which variable stores the memory address of another variable?
   A) Reference B) Pointer
   C) Array D) None of the above

(Space for answers)

3) - pointer

4) - Only base class object.

XIII Exercise

Attempt Q1 and Q.3 a or b from the following:

(Note: Use Point VIII to X and XIII to XV for all relevant programming exercise use blank pages provided or attach more pages if needed.)

1. Write a C++ program declare a class "polygon" having data members width and height. Derive classes "rectangle" and "triangle" from "polygon" having area() as a member function. Calculate area of triangle and rectangle using pointer to derived class object.

2. Complete the given table:

Program Code	Write & justify Output

a)
#include<iostream.h>
class base
{
public:
int n1;
void show()
{
cout<<"\nn1 = "<<n1;
}
};
class derive : public base
{
public:
int n2;
void show()
{
cout<<"\nn1 = "<<n1;
cout<<"\nn2 = "<<n2;
}
};
int main()
{
base b;
base *bptr;
cout<<"Pointer of base class points to it";
bptr=&b;
bptr->n1=44;
bptr->show();
derive d;
cout<<"\n";
bptr=&d;
bptr->n1=66;
bptr->show();
return 0;
}
	Pointer of base class Pointer to it
	n1 = 44
	n1 = 66

b) #include <iostream.h>
class BaseClass {
int x;
public:
void setx(int i) {
x = i;
}
int getx() {
return x;
}
};
class DerivedClass : public BaseClass {
int y;
public:
void sety(int i) {
y = i;
	Base object
	S1 = : 10
	derived object
	4 : 99
	Derived object. 88

```
  }
  int getY() {
     return y;
  }
};

int main()
{
  BaseClass *p; BaseClass
  baseObject; DerivedClass
  derivedObject;


  p = &baseObject;
  p->setx(10);
  cout << "Base object x: " << p->getx() <<
'\n';
  p = &derivedObject;
  p->setx(99);

  derivedObject.sety(88);
  cout << "Derived object x: " << p->getx() <<
'\n';
  cout << "Derived object y: " <<
derivedObject.getY() << '\n';

  return 0;
}
```

(Space for answers)

XIV References / Suggestions for further Reading

1. http://www.learntoprogramming.com/content/pointer-derived-class
2. http://www.java2s.com/Code/Cpp/Class/Cannotassignpointertoderived class.htm
3. http://www.siteforinfotech.com/2014/08/derived-class-in-cpp-programming-language.html

XV Assessment Scheme

	Performance indicators	Weightage
	Process related (35 Marks)	**70%**
1	Logic formation	20%
2	Appropriate use of pointer to derived class	20%
3	Debugging ability	20%
4	Follow ethical practices.	10%
	Product related (15 Marks)	**30%**
5	Expected Output	10%
6	Timely Submission of report	10%
7	Answer to sample questions	10%
	TOTAL	100%

List of Students /Team Members

1. .....
2. .....
3. .....
4. .....

Marks Obtained			Dated signature of Teacher
Process Related(35)	Product Related(15)	Total(50)	
35+	14 =	49/50	Sharma

Practical No. 16: Program to Implement Operator Overloading Using Unary Operator

I Practical Significance:

The concept of operator overloading helps to assign the new meaning to the existing operator and helps to extend the concept of polymorphism.

II Relevant Program Outcomes (POs)

- **Basic knowledge:** Apply knowledge of basic mathematics, sciences and basic engineering to solve the broad-based Computer engineering problem.
- **Discipline knowledge:** Apply Computer engineering discipline - specific knowledge to solve core computer engineering related problems.
- **Experiments and practice:** Plan to perform experiments and practices to use the results to solve broad-based Computer engineering problems.
- **Engineering tools:** Apply relevant Computer technologies and tools with an understanding of the limitations.
- **Communication:** Communicate effectively in oral and written form.

III Competency and Practical skills

This practical is expected to develop the following skills in you :

Develop C++ programs to solve broad-based problems

1. Define and use the overloaded operator function in the class.
2. Compile the program.
3. Debug and execute the program.

IV Relevant Course Outcome(s)

Use Polymorphism in C++ program.

V Practical Outcome (POs)

Write/ Compile/ debug / Execute simple C++ program using unary operator overloading.

VI Relevant Affective domain related Outcome(s)

1. Select proper programming environment in C++.
2. Follow safety measures
3. Follow ethical practices.

VII Minimum Theoretical Background

Operator overloading

Operator overloading is a type of polymorphism in which an operator is overloaded to give user defined meaning to it. The main purpose of operator overloading is to perform operation on user defined data type. For eg. The '+' operator can be overloaded to perform addition on various data types. Operator overloading is used by the programmer to make a program clearer. It is an important concept in C++.

Syntax:

```
Return_type classname :: operator OperatorSymbol (Argument_List)
{
        //Statements;
}
```

The operator keyword is used for overloading the operators.

There are a few operators which cannot be overloaded are follows,

1. Scope resolution operator (::)
2. sizeof
3. member selector (.)
4. member pointer selector (*)
5. ternary operator (? :)

There are some restrictions considered while implementing the operator overloading,

1. The number of operands cannot be changed. Unary operator remains unary, binary remains binary etc.
2. Only existing operators can be overloaded.
3. The precedence and associativity of an operator cannot be changed.
4. Cannot redefine the meaning of a procedure.

The Unary Operators

I. The unary operators require only one operand.
II. They perform various operations such as incrementing/decrementing a value by one, negating an expression, or inverting the value of a boolean.

Operator	Description
+	Unary plus operator; indicates positive value (numbers are positive without this, however)
-	Unary minus operator; negates an expression
++	Increment operator; increments a value by 1
--	Decrement operator; decrements a value by 1
!	Logical complement operator; inverts the value of a boolean

Unary Operator Overloading Algorithm/Steps:

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.

VIII Resources required

Sr. No.	Name of Resource	Specification	Quantity	Remarks
1	Hardware: Computer System	Computer (i3-i5 preferable), RAM minimum 2 GB and onwards, HDD 40GB and above	As per batch size	For all Experiments
2	Operating system	Windows /LINUX		
3	Software	Turbo C++ Version 3.0 or any other		

IX Precautions

1. Handle computer system and peripherals with care.
2. Follow safety practices.

X Resources used

S. No.	Name of Resource	Specification
1	Computer System with broad specifications	RAM of 2GB & HDD of 500 GB
2	Software	Turbo C
3	Any other resource used	

XI Result (Output of the Program)

we executed program to implement operator overloading using unary operator

XII Practical Related Questions

Note: Below given are few sample questions for reference. Teacher must design more such questions so as to ensure the achievement of identified CO.

(Note: Use Point VIII to X and XIII to XV for all relevant programming exercise use blank pages provided or attach more pages if needed.)

(1) What is function overloading and operator overloading?

(2) State output of the following code:

```
#include <iostream.h>
class Distance {
   private:
      int feet;
      int inches;
   public:
      // required constructors
      Distance() {
         feet = 0;
         inches = 0;
      }
      Distance(int f, int i) {
         feet = f;
         inches = i;
      }

      void displayDistance() {
         cout << "F: " << feet << " I:" << inches <<endl;
      }

      // overloaded minus (-) operator
      Distance operator- () {
         feet = -feet;
         inches = -inches;
         return Distance(feet, inches);
      }
};
int main() {
   Distance D1(11, 10), D2(-5, 11);
    -D1;
   D1.displayDistance();
   -D2;
   D2.displayDistance();
   return 0;
}
```

Output :
F:-11
I:-10
F: 5
I:-11

(3) Write a C++ program using operator overloading for the following:

1. >> : To accept the time.
2. << : To display the time.

(Space for answers)

1) function overloading and operator overloading means writing multiple functions and operators having same name but different parameters list respectively

XIII Exercise

Attempt Q1 and Q.3 a or b from the following:

(Note: Use Point VIII to X and XIII to XV for all relevant programming exercise use blank pages provided or attach more pages if needed.)

1. Write a C++ program to overload unary operators (++) increment and (--)decrement.

2. Complete the given table:

Program Code	Write & justify Output

a)
class 3D
{
int x, y, z;
public:
3D (int a=0, int b=0, int c=0)
{
x = a;
y = b;
z = c;
}
 	
3D operator ++()
{
x = x + 1;
y = y + 1;
z = z + 1;
return *this;
}
3D operator ++(int)
{
3D t = *this;
x = x + 1;
y = y + 1;
z = z + 1;
return t;
}
 	
3D show()
{
cout<<"The elements are:\n"
cout<<"x:"<<this->x<<", y:<<this->y <<",
z:"<<this->z;
}
};
int main()
{
3D pt1(2,4,5), pt2(7,1,3);
cout<<"Point one's dimensions before increment
are:"<< pt1.show();
++pt1;
cout<<"Point one's dimensions after increment
are:"<< pt1.show();
cout<<"Point two's dimensions before increment
are:"<< pt2.show();
pt2++;
cout<<"Point two's dimensions after increment
are:"<< pt2.show();
return 0;
}
	Pointers dimensions Before inerement
	x = 5
	y = 5
	z = 6
	Point of 2 two is before inerement
	x = 7
	y = 1
	z = 3
	after inerement
	x = 8
	y = 2
	z = 4

b)
```
#include <iostream.h>
 class Point
{
private:
    double m_x, m_y, m_z;
 public:
    Point(double x=0.0, double y=0.0, double z=0.0):
        m_x(x), m_y(y), m_z(z)
    {
    }
    Point operator- () const;
    bool operator! () const;
    double getX() { return m_x; }
    double getY() { return m_y; }
    double getZ() { return m_z; }
};
Point Point::operator- () const
{
    return Point(-m_x, -m_y, -m_z);
}
bool Point::operator! () const
{
    return (m_x == 0.0 && m_y == 0.0 && m_z == 0.0);
}
int main()
{
    Point point; // use default constructor to set
to (0.0, 0.0, 0.0)

    if (!point)
        std::cout << "point is set at the
origin.\n";
    else
        std::cout << "point is not set at the
origin.\n";
    return 0;
}
```

Output
Point is set as the origin

(Space for answers)

XIV References / Suggestions for further Reading

1. http://www.careerride.com/C++-what-is-overloading-unary-operator.aspx
2. http://www.learncpp.com/cpp-tutorial/95-overloading-unary-operators/
3. http://web.itu.edu.tr/bkurt/Courses/blg252e/blg252e_mod05.pdf
4. https://www.tutorialride.com/cpp-operator-overloading-programs/unary-operator-overloading-c-program.htm
5. https://www.tutorialride.com/cpp-operator-overloading- programs/accept-display-compare-time-with-operator-overloading.htm
6. https://www.tutorialride.com/cpp-operator-overloading-programs/overload-unary-minus-operator-c-program.htm

XV Assessment Scheme

Performance indicators		Weightage
Process related (35 Marks)		**70%**
1	Logic formation	20%
2	Appropriate definition of unary operator overloading function.	20%
3	Debugging ability	20%
4	Follow ethical practices.	10%
Product related (15 Marks)		**30%**
5	Expected Output	10%
6	Timely Submission of report	10%
7	Answer to sample questions	10%
Total (50 Marks)		**100%**

List of Students /Team Members

1. .....
2. .....
3. .....
4. .....

Marks Obtained			Dated signature of Teacher
Process Related(35)	Product Related(15)	Total(50)	
34	+ 15 =	49/50	Sillarod

Practical No. 17: Program to Implement Operator Overloading Using Binary Operator

I Practical Significance:

The concept of operator overloading helps to assign the new meaning to the existing operator and helps to extend the concept of polymorphism.

II Relevant Program Outcomes (POs)

- **Basic knowledge:** Apply knowledge of basic mathematics, sciences and basic engineering to solve the broad-based Computer engineering problem.
- **Discipline knowledge:** Apply Computer engineering discipline - specific knowledge to solve core computer engineering related problems.
- **Experiments and practice:** Plan to perform experiments and practices to use the results to solve broad-based Computer engineering problems.
- **Engineering tools:** Apply relevant Computer technologies and tools with an understanding of the limitations.
- **Communication:** Communicate effectively in oral and written form.

III Competency and Practical skills

This practical is expected to develop the following skills in you :

Develop C++ programs to solve broad-based problems

1. Define real life entity pointer to objects.
2. Compile the program.
3. Debug and execute the program.

IV Relevant Course Outcome(s)

Use Polymorphism in C++ program.

V Practical Outcome (POs)

Write/ Compile/ debug / Execute simple C++ program using binary operator overloading.

VI Relevant Affective domain related Outcome(s)

1. Select proper programming environment in C++.
2. Follow safety measures
3. Follow ethical practices.

VII Minimum Theoretical Background

The Binary Operators

Binary operator works with two operands.
The first operand becomes the operator overloaded function caller and the second is passed as an argument.

Binary Operator Overloading Algorithm/Steps:

Step 1: Start the program. Step 2:
Declare the class.
Step 3: Declare the variables and its member function.

Step 4: Using the function get value() to get the two numbers.
Step 5: Define the function operator +() to add two complex numbers.
Step 6: Define the 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 get value 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.

VIII Resources required

Sr. No.	Name of Resource	Specification	Quantity	Remarks
1	Hardware: Computer System	Computer (i3-i5 preferable), RAM minimum 2 GB and onwards, HDD 40GB and above	As per batch size	For all Experiments
2	Operating system	Windows /LINUX		
3	Software	Turbo C++ Version 3.0 or any other		

IX Precautions

1. Handle computer system and peripherals with care.
2. Follow safety practices.

X Resources used

S. No.	Name of Resource	Specification
1	Computer System with broad specifications	RAM of 2 GB & HDD 500 GB
2	Software	Turbo C
3	Any other resource used	

XI Result (Output of the Program)

We executed program to implement operator overloading using binary operator

XII Practical Related Questions

Note: Below given are few sample questions for reference. Teacher must design more such questions so as to ensure the achievement of identified CO.

(Note: Use Point VIII to X and XIII to XV for all relevant programming exercise use blank pages provided or attach more pages if needed.)

1. What is the difference between unary operator overloading and binary operator overloading in C++?
2. State output of the following code:

```
#include<iostream.h>
#include<conio.h>

class overloading {
    int value;
public:

    void setValue(int temp) {
        value = temp;
    }

    overloading operator+(overloading ob) {
        overloading t;
        t.value = value + ob.value;
        return (t);
    }

    void display() {
        cout << value << endl;
    }
};

void main() {
    overloading obj1, obj2, result;
    int a, b;

    cout << "Enter the value of Complex Numbers a,b:";
    cin >> a>>b;
    obj1.setValue(a);
    obj2.setValue(b);

    result = obj1 + obj2;

    cout << "Input Values:\n";
    obj1.display();
    obj2.display();

    cout << "Result:";
    result.display();

    getch();

}
```

(Space for answers)

XIII Exercise

Attempt Q1 or Q2 or Q3 and Q.4 a or b from the following:

(Note: Use Point VIII to X and XIII to XV for all relevant programming exercise use blank pages provided or attach more pages if needed.)

1. Write a C++ program to add two complex numbers using operator overloaded by a friend function.
2. Write a C++ program to create a class Binary that contains one float data member. Overload the 4 arithmetic operators'.
3. Write a C++ program to compare two strings using '= =' operator overloading.

4. Complete the given table:

Program Code	Write & justify Output

a)
#include<iostream>
#include<cstdlib>
using namespace std;
 	
class Fraction
{
public:
int num, deno;
public:
Fraction()
{
num = 1;
deno = 1;
}
Fraction(int n, int d)
{
num = n;
if (d==0)
{
cout << "Error: Attempting to Divide by Zero" << endl;
exit(0); // it will terminate the program if division by 0 is attempted
}
else
deno = d;
}
Fraction operator +(Fraction f)
{
int n = num*f.deno+f.num*deno;
int d = deno*f.deno;
return Fraction(n/gcd(n,d),d/gcd(n,d));
}
Fraction operator -(Fraction f)
{
int n = num*f.deno-f.num*deno;
int d = deno*f.deno;
return Fraction(n/gcd(n,d),d/gcd(n,d));
}
Fraction operator *(Fraction f)
{
int n = num*f.num; int d = deno*f.deno; return Fraction(n/gcd(n,d),d/gcd(n,d));
}
Fraction operator /(Fraction f)	

```
{
int n = num*f.deno; int d = deno*f.num; return
Fraction(n/gcd(n,d),d/gcd(n,d));
}
bool operator == (Fraction &f)
{
{
int n = num*f.deno; int d = deno*f.num; return
Fraction(n/gcd(n,d),d/gcd(n,d));
}
bool operator == (Fraction &f)
{

(deno==f.deno);
}

return (num==f.num) &&
int gcd(int n, int d)
{
int rem;
while (d != 0)
{
rem = n % d;
n = d;
d = rem;
}
return n;
}
void accept()
{
cout<<"\n Enter Numerator        :  ";
cin>>num;
cout<<"\n Enter Denominator     :  ";
cin>>deno;
}
};
void main()
{
Fraction f1; Fraction f2; Fraction  f3;

cout<<"\n Enter 1st Fraction Value "; cout<<"\n -------
---------- \n"; f1.accept();

cout<<"\n Enter 2nd Fraction Value "; cout<<"\n ------
---------- \n"; f2.accept();

f3=f1+f2;
cout<<"\n ------------------------------
----";
cout<<"\n Sum of Two Numbers                :
"<<f3.num<<"/"<<f3.deno<<endl;
```



```
f3=f1-f2;
cout<<"\n Difference of Two Numbers :
"<<f3.num<<"/"<<f3.deno<<endl;

f3=f1*f2;
cout<<"\n Product of Two Numbers    :
"<<f3.num<<"/"<<f3.deno<<endl;
f3=f1/f2;
cout<<"\n Division of Two Numbers   :
"<<f3.num<<"/"<<f3.deno;
cout<<"\n ------------------------------
----";

if(f1 == f2)
cout<<"\n Fraction 1 is Equal to
Fraction 2"<<endl;
else
cout<<"\n Fraction 1 is Not Equal to
Fraction 2"<<endl;

}
```

b)

```
#include <iostream.h>
class Box {
double length;
double breadth;
double height;

public:

double getVolume(void) {
return length * breadth * height;
}

void setLength( double len ) {
length = len;
}

void setBreadth( double bre ) {
breadth = bre;
}

void setHeight( double hei ) {
height = hei;
}

objects.
Box operator+(const Box& b) {
Box box;
box.length = this->length + b.length;
```

```
box.breadth = this->breadth + b.breadth; box.height =
this->height + b.height; return box;
}
};

int main() { Box Box1; Box Box2; Box  Box3;
double volume = 0.0;

Box1.setLength(6.0); Box1.setBreadth(7.0);
Box1.setHeight(5.0);

Box2.setLength(12.0); Box2.setBreadth(13.0);
Box2.setHeight(10.0);


volume = Box1.getVolume();
cout << "Volume of Box1 : " << volume <<endl;

volume = Box2.getVolume();
cout << "Volume of Box2 : " << volume <<endl;

Box3 = Box1 + Box2;

volume = Box3.getVolume();
cout << "Volume of Box3 : " << volume <<endl;
return 0;
}
```

(Space for answers)

XIV References / Suggestions for further Reading

1. https://www.tutorialride.com/cpp-operator-overloading-programs/demonstrating-operator-overloading-by-using-friend-function.htm
2. https://www.tutorialride.com/cpp-operator-overloading-programs/overload-arithmetic-insertion-and-extraction-operators.htm
3. https://www.tutorialride.com/cpp-operator-overloading-programs/compare-perform-arithmetic-operations-on-two-fractions.htm
4. http://www.includehelp.com/cpp-programs/cpp-program-to-add-two-objects-using-binary-plus-operator-overloading.aspx

XV Assessment Scheme

Performance indicators		Weightage
Process related(35 Marks)		**70%**
1	Logic formation	20%
2	Appropriate definition of binary operator overloading function.	20%
3	Debugging ability	20%
4	Follow ethical practices.	10%
Product related (15 Marks)		**30%**
5	Expected Output	10%
6	Timely Submission of report	10%
7	Answer to sample questions	10%
Total (50 Marks)		**100%**

List of Students /Team Members

1. .....
2. .....
3. .....
4 . .....

Marks Obtained			Dated signature of Teacher
Process Related(35)	Product Related(15)	Total(50)	
34	+ 15 =	49/50	Gaikwad

Practical No. 18: Program to Implement Functional Overloading

I **Practical Significance:**

The concept of function overloading helps to use more than one definitions for a function name in the same scope and helps to extend the concept of polymorphism.

II **Relevant Program Outcomes (POs)**

- **Basic knowledge:** Apply knowledge of basic mathematics, sciences and basic engineering to solve the broad-based Computer engineering problem.
- **Discipline knowledge:** Apply Computer engineering discipline - specific knowledge to solve core computer engineering related problems.
- **Experiments and practice:** Plan to perform experiments and practices to use the results to solve broad-based Computer engineering problems.
- **Engineering tools:** Apply relevant Computer technologies and tools with an understanding of the limitations.
- **Communication:** Communicate effectively in oral and written form.

III **Competency and Practical skills**

This practical is expected to develop the following skills in you :

Develop C++ programs to solve broad-based problems

1. Define more than one definitions for a function name in the same scope to implement function overloading.
2. Compile the program.
3. Debug and execute the program.

IV **Relevant Course Outcome(s)**

Use Polymorphism in C++ program

V **Practical Outcome (POs)**

Write/ Compile/ debug / Execute simple C++ program using function overloading.

VI **Relevant Affective domain related Outcome(s)**

1 Select proper programming environment in C++.
2 Follow safety measures
3 Follow ethical practices.

VII **Minimum Theoretical Background**

Function overloading.

1) Function overloading is a feature in C++ where two or more functions can have the same name but different parameters.
2) The C++ compiler selects the proper function by examining the number, types and order of the arguments in the call. Function overloading is commonly used to create several functions of the same name that perform similar tasks but on different data types.
3) Function overloading can be considered as an example of polymorphism feature in C++ .
4) There are two ways to overload the method in C++
   a) By changing number of arguments or parameters
   b) By changing the data type

Syntax

```
class class_Name
{
returntype method()
{
.....
.....
}
returntype method(datatype1 variable1)
{
.....
.....
}
returntype method(datatype1 variable1, datatype2 variable2)
{
.....
.....
}
};
```

Examples

```
void display();          //function with no arguments
void display( int );     //function with one integer type arguments
void display( float );   //function with one floating point arguments
void display( int, float ); //function with one floating and one integer type argument
```

VIII Resources required

Sr. No.	Name of Resource	Specification	Quantity	Remarks
1	Hardware: Computer System	Computer (i3-i5 preferable), RAM minimum 2 GB and onwards, HDD 40GB and above	As per batch size	For all Experiments
2	Operating system	Windows /LINUX		
3	Software	Turbo C++ Version 3.0 or any other		

IX Precautions

1 Handle computer system and peripherals with care.
2 Follow safety practices.

X Resources used

S. No.	Name of Resource	Specification
1	Computer System with broad specifications	RAM of 2GB & HDD of 500 GB
2	Software	Turbo C
3	Any other resource used	

XI Result (Output of the Program)

We executed program to implement functional overloading

XII Practical Related Questions

Note: Below given are few sample questions for reference. Teacher must design more such questions so as to ensure the achievement of identified CO.

(Note: Use Point VIII to X and XIII to XV for all relevant programming exercise use blank pages provided or attach more pages if needed.)

1. Does constructor overloading is implementation of function overloading?
2. State output of the following code:

```
#include <iostream.h>
#include<conio.h>

int operate (int a, int b)
{
    return (a * b);
}
float operate (float a, float b)
{
    return (a / b);
}
int main()
{
    int x = 5, y = 2;
    float n = 5.0, m = 2.0;
    cout << operate(x, y) <<"\t";
    cout << operate (n, m);
    return 0;
}
```

3. Which of the following in Object Oriented Programming is supported by Function overloading and default arguments features of C++?
   a) Inheritance b) Polymorphism
   c) Encapsulation d) None of these

4. Overloaded functions are
   a) Very long functions that can hardly run
   b) One function containing another one or more functions inside it.
   c) Two or more functions with the same name but different number parameters or type.
   d) None of these

(Space for answers)

1) - Yes

3) - Polymorphism

4) - Two or more fu'ns with the same name but different number parameters

XIII Exercise

Attempt Q1 or Q2 or Q3 and Q.4 a or b from the following:

(**Note:** Use Point VIII to X and XIII to XV for all relevant programming exercise use blank pages provided or attach more pages if needed.)

1. Write a C++ Program to interchange the values of two int , float and char using function overloading .

2. Write a C++ Program that find the distance between two points in 2D and 3D space using function overloading.

3. Write C++ program to find the area of various geometrical shapes by function overloading.

4. Complete the given table:

Program Code	Write & Justify Output

a)
#include<iostream.h>
int absolute(int);
float absolute(float);
 	
int main() {
int a = -5;
float b = 5.5;
cout << "Absolute value of " << a << " = " <<
absolute(a) << endl;
cout << "Absolute value of " << b << " = " <<
absolute(b);
return 0;
}
 	
int absolute(int var) {
if (var < 0)
var = -var;
return var;
}
 	
float absolute(float var){
if (var < 0.0)
var = -var;
return var;
 	
}
	output
	The absolute value of -5 = 5
	Absolute value 5.5

b)	Output

`#include <iostream.h>`
`class Test`
`{`
`public:`
`    int main(int s)`
`    {`
`        cout << s << "\n";`
`        return 0;`
`    }`
`    int main(char *s)`
`    {`
`        cout << s << endl;`
`        return 0;`
`    }`
`    int main(int s ,int m)`
`    {`
`        cout << s << " " << m;`
`        return 0;`
`    }`
`};`
`int main()`
`{`
`    Test obj;`
`    obj.main(3);`
`    obj.main("I like C++");`
`    obj.main(9, 6);`
`    return 0;`
`}`
	3
	I like C++
	9 6

(Space for answers)

XIV References / Suggestions for further Reading

1. https://www.geeksforgeeks.org/can-main-overloaded-
2. http://www.trytoprogram.com/cplusplus-programming/function- overloading
3. http://www.includehelp.com/cpp-tutorial/function-overloading-cpp-programming-tutorial.aspx
4. https://resume.mcalglobal.com/socialresume/net/take-quiz.htm?catid=ccpp&nameid=dc180db6-4064-4663-b4ae-5dbcfed6b04b

XV Assessment Scheme

	Performance indicators	Weightage
	Process related(35 Marks)	70%
1	Logic formation	20%
2	Appropriate definition of overloaded functions	20%
3	Debugging ability	20%
4	Follow ethical practices.	10%
	Product related (15 Marks)	30%
5	Expected Output	10%
6	Timely Submission of report	10%
7	Answer to sample questions	10%
	Total (50 Marks)	100%

List of Students /Team Members

1. ....
2. ....
3. ....
4. ....

Marks Obtained			Dated signature of Teacher
Process Related(35)	**Product Related(15)**	**Total(50)**	
35	13	48/50	[signature]

Practical No. 19: Program to Read and Write Data From a File.

I Practical Significance:

The File I/O streams helps to store data permanently on storage and handle it using different file operations.

II Relevant Program Outcomes (POs)

- **Basic knowledge:** Apply knowledge of basic mathematics, sciences and basic engineering to solve the broad-based Computer engineering problem.
- **Discipline knowledge:** Apply Computer engineering discipline - specific knowledge to solve core computer engineering related problems.
- **Experiments and practice:** Plan to perform experiments and practices to use the results to solve broad-based Computer engineering problems.
- **Engineering tools:** Apply relevant Computer technologies and tools with an understanding of the limitations.
- **Communication:** Communicate effectively in oral and written form.

III Competency and Practical skills

This practical is expected to develop the following skills in you :

Develop C++ programs to solve broad-based problems

1. Use Read & Write File Operation.
2. Compile the program.
3. Debug and execute the program.

IV Relevant Course Outcome(s)

Develop C++ programs to perform file operations.

V Practical Outcome (POs)

a) Write/ Compile/ debug / Execute simple C++ program using read and write data from a file.

VI Relevant Affective domain related Outcome(s)

1. Select proper programming environment in C++.
2. Follow safety measures
3. Follow ethical practices.

VII Minimum Theoretical Background

File Handling

1. This concept in C++ language is used for store a data permanently in computer.
2. Using file handling we can store our data in Secondary memory (Hard disk).

Standard File handling Classes

1. **Ofstream:** This file handling class in C++ signifies the output file stream and is applied to create files for writing information to files.
2. **Ifstream:** This file handling class in C++ signifies the input file stream and is applied for reading information from files.
3. **Fstream:** This file handling class in C++ signifies the file stream generally, and has the capabilities for representing both ofstream and ifstream.

Opening a File

1. The first operation generally performed on an object of one of these classes to use a file is the procedure known as to opening a file.
2. An open file is represented within a program by a stream and any input or output task performed on this stream will be applied to the physical file associated with it.
3. The syntax of opening a file in C++ is:

   open (filename, mode);
4. There are some mode flags used for file opening.

   ios::app: append mode
   ios::ate: open a file in this mode for output and read/write controlling to the end of the file
   ios::in: open file in this mode for reading
   ios::out: open file in this mode for writing
   ios::trunk: when any file already exists, its contents will be truncated before file opening

Closing a File

1. When any C++ program terminates, it automatically flushes out all the streams releases all the allocated memory and closes all the opened files.
2. But it is good to use the close() function to close the file related streams and it is a member of ifsream, ofstream and fstream objects.
3. The structure of using this function is:

   void close();

General functions used for file handling

1. open(): To create a file
2. close(): To close an existing file
3. get(): to read a single character from the file
4. put(): to write a single character in the file
5. read(): to read data from a file
6. write(): to write data into a file

VIII Resources required

Sr. No.	Name of Resource	Specification	Quantity	Remarks
1	Hardware: Computer System	Computer (i3-i5 preferable), RAM minimum 2 GB and onwards, HDD 40GB and above	As per batch size	For all Experiments
2	Operating system	Windows /LINUX		
3	Software	Turbo C++ Version 3.0 or any other		

IX Precautions

1. Handle computer system and peripherals with care.
2. Follow safety practices.

X Resources used

S. No.	Name of Resource	Specification
1	Computer System with broad specifications	RAM of 2GB & HDD 500 GB
2	Software	Turbo C
3	Any other resource used	

XI Result (Output of the Program)

We executed program to read & write data from a file

XII Practical Related Questions

Note: Below given are few sample questions for reference. Teacher must design more such questions so as to ensure the achievement of identified CO.

(Note: Use Point VIII to X and XIII to XV for all relevant programming exercise use blank pages provided or attach more pages if needed.)

1. What are file handling classes in C++?
2. State output of the following code:

```
#include<iostream.h>
#include<fstream.h>
#include<string.h>
#include<cctype.h>

int main()
{

    ifstream ifile;
    ifile.open ("text.txt");
    cout << "Reading data from a file :-" << endl ;
    int c = ifile.peek();
    if ( c == EOF ) return 1;
    if ( isdigit(c) )
    {
    }
    int n;
    ifile >> n;
    cout << "Data in the file: " << n << '\n';
    }
    else
    {
    string str;
    ifile >> str;
    cout << "Data in the file: " << str << '\n';
```



```
    ifile.close();
    return 0;
}
```

3. State output of the following code:

```
#include<iostream.h>
#include<fstream.h>
#include<cctype.h>
int main ()
{
    ifstream ifile;
    ifile.open ("text.txt");
      char last;
    ifile.ignore (256, ' ');
    last = ifile.get();

    cout << "Your initial is " << last << '\n';
    ifile.close();
    return 0;
}
```

4. Which operator is used to insert the data into file?
   a) >> b) <<
   c) < d) none of the mentioned

5. Which function is used to position back from the end of file object?
   a) seekg () b) seekp()
   c) both seekg() & seekp() d) none of the mentioned

(Space for answers)

1) - file handling classes in c++ are :-
i) ifstream
ii) of stream
iii) fstream

2) Error - undefined symbol 'string'
Error - statement missing
Error - undefined symbol 'str'
Error - compound statement missing ?

3) → Your initial is p

4) → >>

5)

XIII Exercise

Attempt any 2 from (1-4) and Q.5 a or b from the following:

(Note: Use Point VIII to X and XIII to XV for all relevant programming exercise use blank pages provided or attach more pages if needed.)

1. Write a C++ program ask to the user to enter file name to encrypt its content.
2. Write a C++ program to merge two files.
3. Write a C++ program to Read and Display File's Content.
4. Write a C++ program to List Files in Current Directory.
5. Complete the given table:

Program Code	Write & justify Output

a) #include<iostream.h>
#include<fstream.h>
int main()
{
ofstream ofile;
ofile.open ("text.txt");
ofile << "geeksforgeeks" << endl;
cout << "Data written to file" << endl;
ofile.close();
return 0;
}	

b)#include <iostream.h>
#include<fstream.h>
int main()
{
char data[100];
ifstream ifile;
 	
//create a text file before executing.
ifile.open ("text.txt");
while (!ifile.eof())
{
ifile.getline (data, 100);
cout << data << endl;
}
ifile.close();
return 0;
}	

(Space for answers)