

Biyani's Think Tank

*Concept based notes*

# Principal of Programming Language

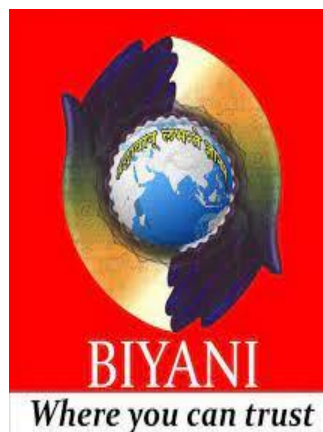
*BCA Part-I*

**Mr. Sachin Bagoria, Mr. Kamlesh Kumar**

Asst. Professor

Dept. of IT

Biyani Girls College, Jaipur



*Published by :*

**Think Tanks**

**Biyani Group of Colleges**

*Concept & Copyright :*

**©Biyani Shikshan Samiti**

Sector-3, Vidhyadhar Nagar,

Jaipur-302 023 (Rajasthan)

Ph : 0141-2338371, 2338591-95 Fax : 0141-2338007

E-mail : [acad@biyanicolleges.org](mailto:acad@biyanicolleges.org)

Website : [www.gurukpo.com](http://www.gurukpo.com); [www.biyanicolleges.org](http://www.biyanicolleges.org)

**ISBN : 978-93-83462-22-3**

**Edition: 2022**

While every effort is taken to avoid errors or omissions in this Publication, any mistake or omission that may have crept in is not intentional. It may be taken note of that neither the publisher nor the author will be responsible for any damage or loss of any kind arising to anyone in any manner on account of such errors and omissions.

*Leaser Type Setted by :*

**Biyani College Printing Department**

## **Preface**

I am glad to present this book, especially designed to serve the needs of the students. The book has been written keeping in mind the general weakness in understanding the fundamental concepts of the topics. The book is self-explanatory and adopts the “Teach Yourself” style. It is based on question-answer pattern. The language of book is quite easy and understandable based on scientific approach.

Any further improvement in the contents of the book by making corrections, omission and inclusion is keen to be achieved based on suggestions from the readers for which the author shall be obliged.

I acknowledge special thanks to Mr. Rajeev Biyani, *Chairman* & Dr. Sanjay Biyani, *Director (Acad.)* Biyani Group of Colleges, who are the backbones and main concept provider and also have been constant source of motivation throughout this Endeavour. They played an active role in coordinating the various stages of this Endeavour and spearheaded the publishing work.

I look forward to receiving valuable suggestions from professors of various educational institutions, other faculty members and students for improvement of the quality of the book. The reader may feel free to send in their comments and suggestions to the under mentioned address.

**Author**

Biyani's Think Tank

**Concept based notes**

# **Principle of Programming Language**

(BCA-I Year)

**Sandhya Saju  
Dhanesh Gupta**

**Updated By**

**Kamlesh Kumar**

Deptt. of Information Technology  
Biyani Girls College, Jaipur



**Biyani's**  
Group of Girls' Colleges

*Published by :*

**Think Tanks**

**Biyani Group of Colleges**

*Concept & Copyright :*

©**Biyani Shikshan Samiti**

Sector-3, Vidhyadhar Nagar,

Jaipur-302 023 (Rajasthan)

Ph : 0141-2338371, 2338591-95 • Fax : 0141-2338007

E-mail : [acad@biyanicolleges.org](mailto:acad@biyanicolleges.org)

Website : [www.gurukpo.com](http://www.gurukpo.com); [www.biyanicolleges.org](http://www.biyanicolleges.org)

**Edition : 2020**

**Price :**

While every effort is taken to avoid errors or omissions in this Publication, any mistake or omission that may have crept in is not intentional. It may be taken note of that neither the publisher nor the author will be responsible for any damage or loss of any kind arising to anyone in any manner on account of such errors and omissions.

*Leaser Type Setted by :*

**Biyani College Printing Department**

## **Preface**

I am glad to present this book, especially designed to serve the needs of the students. The book has been written keeping in mind the general weakness in understanding the fundamental concepts of the topics. The book is self-explanatory and adopts the “Teach Yourself” style. It is based on question-answer pattern. The language of book is quite easy and understandable based on scientific approach.

Any further improvement in the contents of the book by making corrections, omission and inclusion is keen to be achieved based on suggestions from the readers for which the author shall be obliged.

I acknowledge special thanks to Mr. Rajeev Biyani, *Chairman* & Dr. Sanjay Biyani, *Director (Acad.)* Biyani Group of Colleges, who are the backbones and main concept provider and also have been constant source of motivation throughout this Endeavour. They played an active role in coordinating the various stages of this Endeavour and spearheaded the publishing work.

I look forward to receiving valuable suggestions from professors of various educational institutions, other faculty members and students for improvement of the quality of the book. The reader may feel free to send in their comments and suggestions to the under mentioned address.

**Author**



# Syllabus

Basic concepts of programming languages: Programming do mains, language evaluation criterion and language categories, evolution of the major programming languages (FORTRAN, ALGOL 60 COBOL, BASIC, PL/I, ALGOL 68, ADA, C, C++, JAVA) Describing Syntax and Semantics, formal methods of describing syntax, recursive descent parsing, attribute grammars, dynamic semantics.

Names, Variables, Binding, Type cheecking, Scope and lifetime data types, array types, record types, union types, set types and pointer types, arithmetic expressions, type conversions, relational and Boolean expressions, assignment statements, mixed mode assignment.

Statement level control structures, compound statements, selection statement, iterative statements, unconditional branching, guarded commands. Subprogram, fundamentals of subprogram, design issues, parameter passing methods, overloaded subprograms, generic subprograms, separate and independent compilation, design issues for functions, accessing nonlocal environment, user defined overloaded operators,m coroutines, implementing subprograms, blocks, implementing dynamic scoping.

**Programming in C and C++:** Character set, variables and constants, keywords, Instructions, assignment statements, arithmetic expression, comment statements, simple input and output, Boolean expressions, Relational operators, logical operators, control structures, decision control structure, loop control structure, case control structure, functions, subroutines, scope and lifetime of identifiers, parameter passing mechanism, arrays and strings, structures, array of structures, Console Input and Output functions, Disk I/O functions, Interaction with hardware, Interrupts and Interrupt Vectir table, Unions of structures, operations on bits, usage of enumerated data types. Bitfields, Pointers to Function, Function returning Pointers, Graphics in C.

**Object oriented programming in C++:**Basic Concepts of Object Oriented Programming. Characteristics of Object-Oriented Languages, Object, Classes in C++. Constructors, Destructors, Complex Class, Matrix class; Classes, Object and Memory; Structures and Classes; C++ Free Store, Static Class Data, Overload Assignment Operator, Copy Constructor, Data Conversion between Objects of different classes. Data structure through C++, Handling Data files (sequential and random), opening and closing files, stacks and queues, linked lists, trees, Inheritance Multiple, Private and Protected Inheritance, Virtual Functions, Objects Slicing, Input/Output in C++, user defined manipulators, Predefined Stream Objects, File I/O with Streams, Strstreams, Classes within classes, Smart Pointers, Tem plates, Exception Handling.

# Content

S.No.	Name of Topic
1.	Introduction to Programming tit-bits
2.	The Decision , Loop , Case Control Structure
3.	Functions
4.	Array
5.	Structure, Arrays and Union
6.	Pointers
7.	File Handling
8.	C++ programming
9.	Oops Concepts
10.	Case Study
10.	Keywords
11.	Papers 2011-2006
12.	Bibliography



## Chapter 1

# Introduction to Programming tit-bits

**Q 1. How do we define a character set?**

**Ans** Any alphabet ,digit or symbols to represent information is called Character .  
The characters are grouped into following categories:

- 1 Letters
- 2 Digits
- 3 Special characters
- 4 White spaces

The following are the valid alphabets, numbers and special symbols permitted in C.

Digits: From 0 to 9

Letters: From a to z, A to Z.

Special characters : , . ? " ' / \

White space: Blank Spaces , Tab , New Line.

**Q2 What are Identifiers?**

**Ans** Identifiers" are the names that we supply for variables, types, functions, and labels in our program. Identifier names must differ in spelling and case from any keywords. We cannot use keywords (either C or Microsoft) as identifiers; they are reserved for special use. We create an identifier by specifying it in the declaration of a variable, type, or function.

**Q3 Define Variables.**

**Ans** A variable is a name given to the memory location for holding data. The name of the memory location i.e. the variable name, remain fixed during execution of the program but the data stored in that location may change from time to time.

Eg. Marks1 , Marks2, abc , a , ab\_1 ,

**Rules for writing variable names**

1. The first character of variable name must be an alphabetic.
2. Blank spaces are not allowed in a variable name.
3. Special characters such as arithmetic operators, #, ^ can not be used in a variable.
4. Reserved words(Keywords) cannot be used as variable names.

5. The maximum length of a variable name depends upon the compiler (8).
6. A variable name declared for one data type cannot be used to declare another data type.

**Q4 What do you mean by constants?**

Ans Constant is fixed value which can not be changed by the program during the execution.

Eg. A=5 in this 5 is constant.

**Q 5. How many types of constants ?**

A2. There are mainly three types of constants namely: integer, real and character constants.

1. Integer Constants:

(i) Decimal Integer Constant:

0 to 9

E.g: 49, 58, -62, ... (40000 cannot come bcoz it is > 32767)

(ii) Octal Integer Constant:

0 to 7

Add "0" before the value.

Eg.: 045, 056, 067

(iii) Hexadecimal Integer: 0 to 9 and A to

F Add 0x before the value

E.g: 0x42, 0x56, 0x67

2. Real Constants:

The real or floating point constants are in two forms namely fractional form and the exponential form.

A real constant in fractional form must have a digit with a decimal part. Ex 456.78

In exponential form, the real constant is represented as two parts. The part lying before the „e“ is the „mantissa“, and the one following „e“ is the „exponent“.

Ex: +3.2e-4, 4.1e8, -0.2e+4, -3.2e-4

3. Character Constants

A character constant is an alphabet, a single digit or a single special symbol enclosed within inverted commas. Ex: "B", "I", "#"

**Q 6. What are key words?**

A.ns They are the reserved words that cannot be used for naming a variable. They perform fix tasks.

Eg. int , char , for , if .

**Q7 Explain Instructions.**

Ans C instruction are of basically three types :

- 1 Type declaration instruction
- 2 Arithmetic Instruction
- 3 Control Instruction

**Type Declaration Instructions**

This instruction is used to declare the type of variables being used in the program. Any variable used in the program must be declared before using it in any statement. The type declaration statements is written at the beginning of the main() function.

The main purpose of type declaration instruction is to declare the type of variable C program.

For Example :

```
int num;
char c; // Type Declaration
float f;
main()
{
Some Statements
}
```

**The Arithmetic Instruction**

A C arithmetic instruction consists of a variable name on the left hand side of = and constants appearing on the right hand side of = are connected by arithmetic operators like +, -, \* and /.

A C arithmetic statement could be of 3 types :

- (a) Integer mode arithmetic statement : This is an arithmetic statement which all operands are either integer or integer constants.

For Example :

```
int i, j, l, m;
i=i+1;
m=i*j+l;
```

- (b) Real Mode Arithmetic Statement : These are arithmetic statement in which all operands are either real constant or real variable.

For Example :

```
float si, roi, p, q ;
```

```
si = roi*p*q/100.0;
```

- c) Mixed mode arithmetic statements : this is an arithmetic statement in which some of the operands are integer and some of the operands are real.

For Example :

```
int a, b, c, num ;
```

```
avg = ( a + b+ c + num)/4;
```

### Control instruction:

To control the sequence of execution of various statements in a C program.

### Q8. What is Expression?

Ans Expression is any valid combination of operators, constants, functions and variables.

Statements like  $a = b + 3$ ,  $++z$  and  $300 > (8 * k)$  are all expressions.

### Q9 Discuss various operator in C and C++.

Ans An operator is a symbol that operates on a certain data type and produces the output as the result of the operation.

Eg. expression  $4 + 5$  is equal to 9. Here 4 and 5 are called operands and + is called operator.

### Category of operators

Unary Operators:-A unary operator is an operator, which operates on one operand.

Binary:-A binary operator is an operator, which operates on two operands

Ternary:-A ternary operator is an operator, which operates on three operands.

### C contains the following operator groups

#### 1 Arithmetic Operator

The arithmetic operator is a binary operator, which requires two operands to perform its operation of arithmetic. Following are the arithmetic operators that are available.

Operator	Description	Eg.
+	Addition	$a+b$
-	Subtraction	$a-b$
/	Division	$a/b$
*	Multiplication	$a*b$
%	Modulo or remainder	$a\%b$

#### 2 Relational Operators

Relational operators compare between two operands and return in terms of true or false i.e. 1 or 0. In C and many other languages a true value is denoted by the integer 1 and a false value is denoted by the integer 0. Relational operators are used in conjunction with logical operators and conditional & looping statements.

< Less than  
> Greater than  
<= Less than or equal to  
>= Greater than or equal to  
!= Not equal to  
== Equal to

### 3 Logical Operators

A logical operator is used to compare or evaluate logical and relational expressions. There are three logical operators available in the C language.

&& Logical AND  
|| Logical OR  
! Logical NOT

### 4 Assignment operator

An assignment operator (=) is used to assign a constant or a value of one variable to another.

Example:

```
a = 5;  
b = a;  
rate = 10.5  
net = (a/b) * 100;
```

- \* There is always difference between the equality operator (==) and the assignment operator (=).

### 5 Conditional or Ternary Operator

A conditional operator checks for an expression, which returns either a true or a false value. If the condition evaluated is true, it returns the value of the true section of the operator, otherwise it returns the value of the false section of the operator.

Its general structure is as follows:

Expression1 ? expression 2 (True Section): expression3 (False Section) Example:

```
a=3,b=5,c;
```

`c = (a>b) ? a+b : b-a;`

The variable `c` will have the value 2, because when the expression `(a>b)` is checked, it is evaluated as false. Now because the evaluation is false, the expression `b-a` is executed and the result is returned to `c` using the assignment operator.

## 6 Bitwise Operators:

These are used to perform bitwise operations such as testing the bits, shifting the bits to left or right, one's complement of bits. This operator can be applied on only `int` and `char` data type.

& AND

| Inclusive OR

^ Exclusive OR

<< Shift Left >>

Shift Right

~ One's complement

`~A = 1100 0011`

## 7 Increment and Decrement Operators

These operators are unary operators.

The increment and decrement operators are very useful in C language. They are extensively used in `for` and `while` loops. The syntax of these operators is given below.

`++`

`--`

## 8 Comma operator ( , ):-

The comma operator ( , ) is used to separate two or more expressions that are included where only one expression is expected. When the set of expressions has to be evaluated for a value, only the rightmost expression is considered.

## Q10 What are the ways to comment statement in C?

Ans Comments are ~~non~~ executable statements

Most of C/C++ will support two types of comments:

// Comment text goes here ( in line)

/\* Comment goes here \*/ (block)

## Q11 Input and output statements

Ans



**Input :**

In any programming language input means to feed some data into program. This can be given in the form of file or from command line. C programming language provides a set of built-in functions to read given input and feed it to the program as per requirement.

printf() function

This is one of the most frequently used functions in C for output

**Output :**

In any programming language output means to display some data on screen, printer or in any file. C programming language provides a set of built-in functions to output required data.

scanf() function

This is the function which can be used to read an input from the command line.



## Chapter 2

# The Decision , Loop , Case Control Structure



**Q.1 Explain control structures available in C and C++.**

A C provides two styles of flow

control: Branching

Looping

**Branching or Decision :**

Branching or Decision is so called because the program chooses to follow one branch or another.

**if statement**

This is the most simple form of the branching statements. It takes an expression in parenthesis and a statement or block of statements. If the expression is true then the statement or block of statements gets executed otherwise these statements are skipped.

Syntax:- if with single statement

```
if (expression)
    statement;
```

Syntax:- if with block statement

```
if (expression)
{
    Block of statements;
}
```

or

Syntax:- if with else statement

```
if (expression)
{
    Block of statements;
}
else
```

```
{  
    Block of statements;  
}
```

Or

Syntex:- if with else if statement

```
if (expression)  
{  
    Block of statements;  
}  
else if(expression)  
{  
    Block of statements;  
}  
else  
{  
    Block of statements;  
}
```

Eg.

```
#include <stdio.h>  
  
main()  
{  
    int cows = 6;  
  
    if (cows > 1)  
        printf("We have cows\n");  
  
    if (cows > 10)  
        printf("loads of them!\n");  
    else  
        printf("Executing else part...\n");  
  
    if (cows == 5 )  
    {  
        printf("We have 5 cows\n");  
    }  
    else if( (cows == 6 )
```

```
{  
    printf("We have 6 cows\n");  
}  
}
```

#### Output

```
We have cows  
Executing else part...!  
We have 6 cows
```

#### ? : Operator

The ? : operator is just like an if ... else statement except that because it is an operator you can use it within expressions.

? : is a ternary operator in that it takes three values, this is the only ternary operator C has.

? : takes the following form:

if condition is true ? then X return value : otherwise Y  
value; switch statement:

The switch statement is much like a nested if .. else statement. Its mostly a matter of preference which you use, switch statement can be slightly more efficient and easier to read.

```
switch( expression )  
{  
    case expression1:  
        statements1;  
    case expression2:  
        statements2;  
    case c-expression3:  
        statements3;  
    default : statements4;  
}
```

#### Use of break

Use If a condition is met in switch case then execution continues on into the next case clause also if it is not explicitly specified that the execution should exit the switch statement. This is achieved by using **break keyword**.

#### Looping

Loops provide a way to repeat commands and control how many times they are repeated. C provides a number of looping way.

**while loop**

The most basic loop in C is the while loop. Like an If statement, if the test condition is true, the statements get executed. The difference is that after the statements have been executed, the test condition is checked again. If it is still true the statements get executed again. This cycle repeats until the test condition evaluates to false.

**syntax**

```
while ( expression )
{
    Single statement
    or
    Block of statements;
}
```

**for loop**

for loop is similar to while, it's just written differently. for statements are often used to process lists such a range of numbers:

**syntax:**

```
for( expression1; expression2; expression3)
{
    Single statement
    or
    Block of statements;
}
```

In the above syntax:

expression1 - Initialises variables.

expression2 - Conditional expression, as long as this condition is true, loop will keep executing.

expression3 - expression3 is the modifier which may be simple increment of a variable.

**do...while loop**

do ... while is just like a while loop except that the test condition is checked at the end of the loop rather than the start. This has the effect that the content of the loop are always executed at least once.

**syntax**

```
do
{
    Single statement
    or
    Block of statements;
```

```
}while(expression);
```

**break and continue statements**

C provides two commands to control the loop:

break -- exit from loop or switch.

continue -- skip 1 iteration of loop.

```
#include
main()
{
    int i;
    int j = 10;

    for( i = 0; i <= j; i ++ )
    {
        if( i == 5 )
        {
            continue;
        }
        printf("Hello %d\n", i);
    }
}
Hello 0
Hello 1
Hello 2
Hello 3
Hello 4
Hello 6
Hello 7
Hello 8
Hello 9
Hello 10
```

**The goto statement (unconditional branching)**

goto allows to make an absolute jump to another point in the program. We should use this feature with caution since its execution causes an unconditional jump ignoring any type of nesting limitations.

The destination point is identified by a label, which is then used as an argument for the goto statement. A label is made of a valid identifier followed by a colon (:).

goto loop example



```

#include <stdio.h>
int main ()
{
    int n=10;
loop:
    printf("%d", n);
    n--;
    if (n>0)
        goto loop;
    printf( "FIRE");
}
10, 9, 8, 7, 6, 5, 4, 3, 2, 1, FIRE!

```

### exit function

exit is a function defined in the cstdlib library.

The purpose of exit is to terminate the current program with a specific exit code. Its prototype is:

```
exit()
```

## Chapter 3

# Functions

### Q1 What is a Function?

**Ans** The function is a self contained block of statements which performs a task of a same kind. C program does not execute the functions directly. It is required to invoke or call that functions. When a function is called in a program then program control goes to the function body. Then, it executes the statements. We call function whenever we want to process that functions statements i.e. more than 1 times. Any c program contains at least one function. Function is used to avoids rewriting the same code over and over.

**The following is its format:**

```
type name ( parameter1, parameter2, ...) { statements
} where:
```

- type is the data type specifier of the data returned by the function.
- **name** is the identifier by which it will be possible to call the function.
- parameters (as many as needed): Each parameter consists of a data type specifier followed by an identifier.

- statements is the function's body. It is a block of statements surrounded by braces { }.

Eg.

```
void add()
{
    int a, b, c;
    clrscr();
    printf("\n Enter Any 2 Numbers : ");
    scanf("%d %d",&a,&b);
    c = a + b;
    printf("\n Addition is : %d",c);
}
void main()
{
    void add();
    add();
}
```

```
        getch();  
    }
```

**Q2 What are the properties of functions in C? Ans**

function in a C program has some properties.

- 1 Every function has a unique name. This name is used to call function from "main()" function. A function can be called from within another function.
- 2 A function is independent and it can perform its task without intervention from or interfering with other parts of the program.
- 3 A function performs a specific task. A task is a distinct job that our program must perform as a part of its overall operation, such as adding two or more integer.
- 4 A function returns a value to the calling program. This is optional and depends upon the task your function is going to accomplish..

**Q3 What are the types of functions ?**

**Ans** There are 2(two) types of functions as:

1. Built in Functions
2. User Defined Functions

**1. Built in Functions :**

These functions are also called as 'library functions'. These functions are provided by system. These functions are stored in library files. e.g.

```
scanf()  
printf()  
strcpy
```

**2 User Defined Functions :**

The functions which are created by user for program are known as 'User defined functions'.

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

```
void add()  
{  
    int a, b, c;  
    clrscr();  
    printf("\n Enter Any 2 Numbers :  
"); scanf("%d %d",&a,&b);  
    c = a + b;
```

```

        printf("\n Addition is : %d",c);
    }
    void main()
    {
        void add();
        add();
        getch();
    }

```



#### Q4 Write Parameter passing mechanisms in C?

**Ans** There are two ways to pass parameters to a function:

**Pass by Value:** mechanism is used when you don't want to change the value of passed parameters. When parameters are passed by value then functions in C create copies of the passed in variables and do required processing on these copied variables.

```

int main()
{
    int a = 10;
    int b = 20;

    printf("Before: Value of a = %d and value of b = %d\n", a, b
); swap( a, b );
    printf("After: Value of a = %d and value of b = %d\n", a, b );
}

void swap( int p1, int p2 )
{
    int t;

    t = p2;
    p2 = p1;
    p1 = t;
    printf("Value of a (p1) = %d and value of b(p2) = %d\n", p1, p2 );
}

```

Before: Value of a = 10 and value of b = 20

Value of a (p1) = 20 and value of b(p2) = 10

After: Value of a = 10 and value of b = 20

**Pass by Reference** : This mechanism is used when you want a function to do the changes in passed parameters and reflect those changes back to the calling function. In this case only addresses of the variables are passed to a function so that function can work directly over the addresses.

```
void swap( int *p1, int *p2 );
```

```
int main()
{
    int a = 10;
    int b = 20;

    printf("Before: Value of a = %d and value of b = %d\n", a, b
); swap( &a, &b );
    printf("After: Value of a = %d and value of b = %d\n", a, b );
}
```

```
void swap( int *p1, int *p2 )
{
    int t;

    t = *p2;
    *p2 = *p1;
    *p1 = t;
    printf("Value of a (p1) = %d and value of b(p2) = %d\n", *p1, *p2 );
}
before: Value of a = 10 and value of b = 20
Value of a (p1) = 20 and value of b(p2) = 10
After: Value of a = 20 and value of b = 10
```

#### Q5 Scope and lifetime of variables.

Ans

Storage class	Storage	Default initial value	Scope	Life
Automatic	Memory	Garbage	Local to the block in which variable is defined	Till the control remains within the block in which the variable is

				defined
Register	CPU registers	Garbage	Local to the block in which variable is defined	Till the control remains within the block in which the variable is defined
Static	Memory	Zero	Local to the block in which variable is defined	Value of variable persists between different function calls.
External	Memory	Zero	Global	As long as program execution doesn't come to end.

## Chapter 4

### Array

**Q1. Define arrays.**

**Ans.** A collection of variables which are all of the same type. It is a data structure, which provides the facility to store a collection of data of same type under single variable name. Just like the ordinary variable, the array should also be declared properly. The declaration of array includes the type of array that is the type of value we are going to store in it, the array name and maximum number of elements.

Examples:

```
short val[ 200 ]; //declaration
```

```
val[ 12 ] = 5; //assignment
```

**Q2. How is Array declared?**

**Ans Declaration & Data Types**

Arrays have the same data types as variables, i.e., short, long, float etc. They are similar to variables: they can either be declared global or local. They are declared by the given syntax:



Datatype array\_name[dimensions]={element1,element2,.....,element}

**Q3. Write a program for one dimensional array and two dimensional array.**

**Ans** The declaration form of one-dimensional array is

Data\_type array\_name [size];

The following declares an array called „numbers“ to hold 5 integers and sets the first and last elements. C arrays are always indexed from 0. So the first integer in „numbers“ array is numbers[0] and the last is numbers[4].

```
int numbers [5];
```

```
numbers [0] = 1;    // set first element
```

```
numbers [4] = 5;    // set last element
```

This array contains 5 elements. Any one of these elements may be referred to by giving the name of the array followed by the position number of the particular element in square brackets ([ ]). The first element in every array is the zeroth element. Thus, the first element of array „numbers“ is referred to as numbers[ 0 ], the second element of array „numbers“ is referred to as numbers[ 1 ], the fifth element of array „numbers“ is referred to as numbers[ 4 ]

], and, in general, the n-th element of array „numbers“ is referred to as numbers[ n - 1 ].

**Example:**

```
#include <stdio.h>
#include <conio.h>
int main( )
{
    char name[7]; /* define a string of characters */
    name[0] = 'A';
    name[1] = 's';
    name[2] = 'h';
    name[3] = 'r';
    name[4] = 'a';
    name[5] = 'f';
    name[6] = '\0'; /* Null character - end of text
    */ name[7] = „X“;
    clrscr();
    printf("My name is %s\n",name);
    printf("First letter is %c\n",name[0]);
    printf("Fifth letter is %c\n",name[4]);
    printf("Sixth letter is %c\n",name[5]);
    printf("Seventh letter is %c\n",name[6]);
    printf("Eight letter is %c\n",name[7]);
    getch();
    return 0;
}
```

**Output**

```
My name is Ashraf
First letter is A
Fifth letter is a
Sixth letter is f
Seventh letter is Null
Eight letter is X
```

**Two dimensional array**

Two-dimensional array are those type of array, which has finite number of rows and finite number of columns. The declaration form of 2-dimensional array is

Data\_type    Array\_name [row size][column size];

Example:

```
#include<stdio.h>
#include<conio.h>
int main()
{
int matrix[3][3],l,j,r,c;
clrscr();
printf("Enter the order of matrix\n");
scanf("%d%d",&r,&c);
printf("Enter the elements of 3x3 matrix\n",r,c);
for(i=0;i<r;i++)
    for(j=0;j<c;j++)
        scanf("%d",&matrix[i][j]);
printf("Given matrix:\n");
for(i=0;i<r;i++)
    for(j=0;j<c;j++)
        printf("%d\t",matrix[i][j]);
printf("\n");
}
getch();
return 0;
}
```

Output

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

#### Q.4 Explain String.

**Ans** A group of characters stored in a character array. A string in C is a sequence of zero or more characters followed by a NULL '\0' character:

String constants have double quote marks around them,. Alternatively, we assign a string constant to a char array - either with no size specified, or you can specify a size, but don't forget to leave a space for the null character!. Eg.

```
char string_2[] = "Hello";
char string_3[6] = "Hello";
```

## Chapter 5

# Structure, Arrays and Union

**Q1. Define a Structure with suitable program.**

**Ans .** A structure is a user defined data type. We know that arrays can be used to represent a group of data items that belong to the same type, such as int or float. However we cannot use an array if we want to represent a collection of data items of different types using a single name. A structure is a convenient tool for handling a group of logically related data items.

The syntax of structure declaration is

```
struct structure_name
{
    type element 1;
    type element 2;
    .....
    type element n;
};
```

In structure declaration the keyword struct appears first, this followed by structure name. The member of structure should be enclosed between a pair of braces and it defines one by one each ending with a semicolon. It can also be array of structure. There is an enclosing brace at the end of declaration and it end with a semicolon.

We can declare structure variables as follows

```
struct structure_name var1,var2,.....,var n;
```

For Example:

To store the names, roll number and total mark of a student you can declare 3 variables. To store this data for more than one student 3 separate arrays may be declared. Another choice is to make a structure. No memory is allocated when a structure is declared. It just defines the “form” of the structure. When a variable is made then memory is allocated. This is

equivalent to saying that there's no memory for "int" , but when we declare an integer that is. `int var;` only then memory is allocated. The structure for the above-mentioned case will look like

```
struct student
{
    int rollno;
    char name[25];
    float totalmark;
};
```

We can now declare structure variables `stud1`, `stud2` as follows `struct student stud1,stud2;`

Thus, the `stud1` and `stud2` are structure variables of type `student`. The above structure can hold information of 2 students.

It is possible to combine the declaration of structure combination with that of the structure variables, as shown below.

```
struct structure_name
{
    type element 1;
    type element 2;
    .....
    type element n;
}var1,var2,...,varn;
```

The following single declaration is equivalent to the two declaration presented in the previous example. `struct student`

```
{
    int rollno;
    char name[25];
    float totalmark;
} stud1, stud2;
```

The different variable types stored in a structure are called its members. The structure member can be accessed by using a dot (.) operator, so the dot operator is known as structure member operator.

Example:

In the above example stud1 is a structure variable of type student. To access the member name, we would write

stud1.name

Similarly, stud1's rollno and stud1's totalmark can be accessed by writing

stud1.rollno                      And

stud1.totalmark

## Q 2. How is Initialization of structure members carried out?

**Ans**      Structure members can be initialized at declaration. This much the same manner as the element of an array; the initial value must appear in the order in which they will be assigned to their corresponding structure members, enclosed in braces and separated by commas. The general form is

```
struct stucture_name var={val1,val2,val3....};
```

**Example:**

```
#include <stdio.h>
#include <conio.h>
int main()
{
    struct student
    {
        char *name;
        int rollno;
        float totalmark;
    };
    struct student stud1={"Ashraf",1,98};
    struct student stud3= {"Rahul",3,97};
    struct student stud2={"Vineeth",2,99};
    clrscr();
    printf("STUDENTS DETAILS:\nRoll
```



```

        number:%d\n\nName:%s\n\nTotal mark:%.2f\n",
        stud1.rollno,stud1.name,stud1.totalmark);

printf("\nRoll number:%d\n\nName:%s\n\nTotal

        mark:%.2f\n",stud2.rollno,stud2.name,stud2.totalmark);

printf("\nRoll number:%d\n\nName:%s\n\nTotal

        mark:%.2f\n",stud3.rollno,stud3.name,stud3.totalmark);

getch();
return 0;
}

```

**Output**

```

Roll Number  1
Name         Ashraf
Total Marks  98
Roll Number  3
Name         Rahul,
Total Marks  97
Roll Number  2
Name         Vineet
Total Marks  9

```

**Q 3. Is it possible to creat Array of structures:?**

**Ans** It is possible to store a structure has an array element. i.e., an array in which each element is a structure. Just as arrays of any basic type of variable are allowed, so are arrays of a given type of structure. Although a structure

contains many different types, the compiler never gets to know this information because it is hidden away inside a sealed structure capsule, so it can believe that all the elements in the array have the same type, even though that type is itself made up of lots of different types.

The declaration statement is given below.

```
struct struct_name
{
    type element 1;
    type element 2;
    .....
    type element n;
} array name[size];
```

Example:

```
struct student
{
    int rollno;
    char name[25];
    float totalmark;
} stud[100];
```

In this declaration stud is a 100-element array of structures. Hence, each element of stud is a separate structure of type student. An array of structure can be assigned initial values just as any other array. So the above structure can hold information of 100 students.

**Program:**

```
#include <stdio.h>
#include <conio.h>
int main()
{
    struct student
    {
        int rollno;
        char name[25];
        int totalmark;
    } stud[100];
```

```

int n,i;
clrscr();
printf("Enter total number of students\n\n");
scanf("%d",&n);
for(i=0;i<n;i++)
{
    printf("Enter details of %d-th student\n",i+1);
    printf("Name:\n");
    scanf("%s",&stud[i].name);
    printf("Roll number:\n");
    scanf("%d",&stud[i].rollno);
    printf("Total mark:\n");
    scanf("%d",&stud[i].totalmark);
}
printf("STUDENTS DETAILS:\n");
for(i=0;i<n;i++)
{
    printf("\nRoll number:%d\n",stud[i].rollno);
    printf("Name:%s\n",stud[i].name);
    printf("Total mark:%d\n",stud[i].totalmark); }

getch();
return 0;
}

```

Output will be the dynamic data entered by the user of all the students.

**Q 4. Explain the functionality of union.**

**Ans** Union is a data type with two or more member similar to structure but in this case all the members share a common memory location. The size of the union corresponds to the length of the largest member. Since the member share a common location they have the same starting address.

The real purpose of unions is to prevent memory fragmentation by arranging for a standard size for data in the memory. By having a standard data size we can guarantee that any hole left when dynamically allocated

memory is freed will always be reusable by another instance of the same type of union. This is a natural strategy in system programming where many instances of different kinds of variables with a related purpose and stored dynamically.

A union is declared in the same way as a structure. The syntax of union declaration is

```
union union_name
{
    type element 1;
    type element 2;
    .....
    type element n;
};
```

This declares a type template. Variables are then declared as:

```
union union_name x,y,z;
```

For example, the following code declares a union data type called Student and a union variable called stud: union student

```
{
    int rollno;
    float totalmark;
};
union student stud;
```

It is possible to combine the declaration of union combination with that of the union variables, as shown below. union union\_name

```
{
    type element 1;
    type element 2;
    .....
    type element n;
}var1,var2,...,varn;
```

The following single declaration is equivalent to the two declaration presented in the previous example. union student

```
{  
    int rollno;  
    float totalmark;  
}x,y,z;
```

**Q 5. Compare structure and Union**

**Ans** Union allocates the memory equal to the maximum memory required by the member of the union but structure allocates the memory equal to sum of the memory allocated to its each individual members.

In Union, one block is used by all the member of union but in case of structure, each member have their own memory space.

**Example:**

**Structure:**

```
#include <stdio.h>  
  
#include <conio.h>  
  
int main()  
{  
    struct testing  
{  
        int a;  
        char b;  
        float c;  
    }var;  
  
    clrscr();  
  
    printf("Size of var=%d\n",sizeof(var));  
  
    printf("Size of a=%d\n",sizeof(var.a));
```



```
printf("Size of b=%d\n",sizeof(var.b));

printf("Size of c=%d\n",sizeof(var.c));

var.a=10;

var.b="w";

var.c=3.1422;

printf("value of a=%d\n",var.a);

printf("value of b=%c\n",var.b);

printf("value of c=%f\n",var.c);

getch();

return 0;

}
```

**Union:**

```
#include <stdio.h>
#include <conio.h>
int main()
{
union testing
{

int a;

char b;
```

```
float c;                                ()

}var;

clrscr();

printf("Size of
var=%d\n",sizeof(var));

printf("Size of
a=%d\n",sizeof(var.a));

printf("Size of
b=%d\n",sizeof(var.b));

printf("Size of
c=%d\n",sizeof(var.c));

var.a=10;

var.b="w";

var.c=3.1422;

printf("value of a=%d\n",var.a);
printf("value of b=%c\n",var.b);
printf("value of c=%f\n",var.c);

getch();

return 0;

}
```

Output of both these programs will be same. The difference lies in the storage.

## Chapter 6

# Pointers

---

**Q 1. Explain Pointer.**

**Ans .** A pointer refers to a memory location that contains an address.

### Pointers: Operators (1)

Address Operator: &

Note: it looks identical to the bitwise AND operator but it is used in a completely different way! Returns the address of a variable

Example: `prt_v = &x;`

### Pointers: Operators (2)

Indirection Operator: \*

Note: it looks identical to the multiplication operator but it is used in a completely different way!

Retrieves a value from the memory location the pointer points to.

Example: `*ptr_v = 77;`

### Pointer Declaration

A pointer must be declared and the variable type it points to must be specified:

`short *aptr; //pointer declaration`

`double *bptr;`

### Assigning an Address to a Pointer

`short x = 33;`

`short *aptr; //pointer declaration`

`aptr = &x;`

Each time we declare an array, we also declare implicitly a pointer to the "zeroth" element!



**Q 2. Write a program to pass arguments by value and by reference.**

**Ans .** #include <ansi\_c.h>

```
float KE_by_val( float a, float b);
float KE_by_ref( float *a, float *b);
main()
{
float q = 3, v = 5;
float *qptr, *vptr;
qptr = &q;
vptr = &v;
printf("%f\n", KE_by_val(q,v) );
printf("%f\n", KE_by_ref(&q,&v) );
printf("%f\n", KE_by_ref(qptr,vptr) );
}
float KE_by_val( float a, float b)
{
return( a * b);
}
float KE_by_ref( float *a, float *b)
{
return( (*a) * (*b));
}
```

**Output**

15.00

15.00

15.00

()

## Chapter 7

# File Handling

---

**Q 1. Explain File handling.**

**Ans.** Three steps for handling data files:

1. Data File is Opened
2. Data is Read / Written to the File
3. Data File is Closed

Example:

The file is opened using `fopen()` function.

### Opening a Binary File

```
FILE *fptr; // declare a file ptr  
fptr = fopen("Data1.txt", "wt");
```

### Opening a Data File:

```
fopen()  
fopen( arg1, arg2)
```

arg1: a pointer to a string containing the filename or a literal, for example:

"c:\\myfolder\\data.txt"

arg2: activity and filetype constants:

wt	Open an ASCII file for writing
rt	Open an ASCII file for reading
at	Open an ASCII file for appending
wb	Open a Binary file for writing
rb	Open a Binary file for reading

ab      Open a Binary file for appending

### **Closing a Data File:**

File is closed using `fclose()` function.

`fclose()`

Syntax:

`fclose( fptr );`

where `fptr` refers to the file stream pointer returned by the `fopen()` function.

### **Writing to and Reading from an ASCII Data File**

Writing: `fprintf( )`

`fprintf( )` Syntax:

`fprintf( arg1, arg2...)`

where `arg1` is the file stream pointer and `arg2...` are identical to arguments of the

`printf()` function

Example:

`fprintf(fptr, "%d\n", x);`

Reading: `fscanf( )`

`fscanf( )` Syntax:

`fscanf( arg1, arg2...)`

where `arg1` is the file stream pointer and `arg2...` are identical to arguments of the

`scanf()` function

Example:

`fscanf(fptr, "%d", &x);`

### **Writing to and Reading**

from a Binary Data File

Writing: `fwrite()` Reading: `fread()`

Syntax: (identical for `fwrite` and `fread`):

`fwrite( arg1, arg2, arg3, arg4 );`  
arg1: a pointer to the data buffer  
arg2: the size of each data element (in bytes)  
arg3: the number of data elements  
arg4: the file stream pointer.

```
// basic file operations
#include <iostream.h>
#include <fstream.h>
int main () { ofstream myfile;
myfile.open
("example.txt");
myfile << "Writing this to a
file.\n"; myfile.close();
return 0;
}
```

```
// writing on a text file
#include <iostream.h>
#include <fstream.h>
int main () {
ofstream myfile ("example.txt");
if (myfile.is_open())
{
myfile << "This is a line.\n";
myfile << "This is another line.\n";
myfile.close();
}
else cout << "Unable to open file";
return 0;
}
```

### Output

[file example.txt]

This is a line.

This is another line.

**Gurukpo.com**  
No. 1 Educational Web Portal in India

```
// reading a text file
#include <iostream.h>
#include <fstream.h>
#include <string.h> int
main () {
string line;
ifstream myfile ("example.txt"); if
(myfile.is_open())
{
while (! myfile.eof() )
{
getline (myfile,line);
cout << line << endl;
}
myfile.close();
}
else cout << "Unable to open file";
return 0;
}
```

**Output:**

This is a line.

This is another line.

## Chapter 8

# C++ programming

---

**Q 1. What are global and local variables?**

**Ans .** The variables that can be accessed by any portion of the program are known as global variables and the variables that are confined to one particular function or procedure and cannot be accessed outside the function are called Local variables.

**Q 2. What is meant by Dynamic Allocation?**

**Ans .** Dynamic memory allocation is the practice of assigning memory locations to variables during execution of the program by explicit request of the programmer. Dynamic allocation is a unique feature to C (amongst high level languages). It enables us to create data types and structures of any size and length to suit our programs need within the program.

The following functions are used in c for purpose of memory management.

1. malloc()
2. calloc()
3. realloc()
4. free()

In c++ we use new for memory management

**Q3. What is the use of scope resolution operator?**

**Ans .** A *scope resolution* mechanism is for referencing hidden variables



**Q 4. Why do we use sizeof operator in c++?**

**Ans .** We can know the memory usage of a given type using the sizeof operator. It takes either a variable name or a type name as parameter.

```
#include <iostream.h>
int main(int argc, char **argv) {
    int n;
    cout << sizeof(n) << " " << sizeof(double) <<
    "\n"; }
```

**Q 5. What is the use of abort()?**

**An 5.** The abort( ) function is used to interrupt the execution of our program as if there was a serious error. Use it to handle non-expected behavior like out-of bounds exceptions .

**Q6. What are built-in arrays?**

**Ans.** The " " operator defines arrays of char. Similarly, we can define an array of any type with the [ ] operator:

**Q 7. Give an example to show how arrays can be accessed through pointers.**

**Ans .** An array type can be implicitly transformed into a pointer by the compiler, and

the [ ] operator can be used to access to an element of a pointed array.

For example, we can define a function to sum the terms of an array of int :

```
#include <iostream.h>
int sum(int *x, int sz) {
    int s = 0;
    for(int k = 0; k < sz; k++)
        s = s + x[k];
    return s;
}
int main() {
    int cool[10] = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };
```



```
cout << sum(cool, 10) << "\n";  
}
```

Displays 55.

**Q 8. Does the pointer gives the information about the size of the array it points to.**

**Ans .** No Pointers do not give information about pointed array sizes

**Q 9. What are Static variables?**

**Ans .** The variables declared in the source code, without the usage of new and delete are called static variables.

**Q10. What is the difference between declaration and definition?**

**Ans .** The declaration of a function specifies the return and parameter types. The definition specifies the part of the program associated to the function, i.e. the statement between { }.

**Q11. What is the use of break statement?**

**Ans.** It is used to bypass the natural ending of statements by using the break. It terminates the current for, while or switch statement (roughly, jump to the part of the code after the next closing }

**Q12. Given a sorted array of integers.write a program to find the rank of a given value in that array.**

**Ans.** #include <iostream.h>

```
// Returns the rank of x in a and -1 if not found  
int rank2(int x, int *a, int n) {  
    // Let's optimize a bit  
    if(a[0] > x) return -1;  
    if(a[n-1] < x) return -1;  
    int i, j;  
    i = 0; j = n-1;  
    while(i+1 < j) {
```

```
int k = (i+j)/2;
if(a[k] <= x) i = k; else j = k;
}
if(a[i] == x) return i; else if(a[j] == x) return j;
else return -1;
}
int main() {
int a[] = {1, 5, 6, 7, 9, 12, 14, 23, 24, 24, 123};
cout << rank2(14, a, sizeof(a)/sizeof(int)) << "\n";
}
```

**Q13. What is the difference between the = operator and the copy constructor.**

**Ans .** The copy constructor is called either when we initialize a variable (with the = operator!), or when we pass parameters by value. The = is called for all other = assignments.

**Q 14. What is the usage of #include<iostream.h>?**

**Ans** #include is a preprocessor directive to include the file iostream.h. The <> denotes that the file is in the standered directory. Iostream.h contains the c++ i/o class declaration and function prototypes.

**Q 15. What are Special member functions?**

**Ans .** The default constructor, destructor, copy constructor, and copy assignment operator are *special member functions*. These functions create, destroy, convert, initialize, and copy class objects.

**Q 16. Explain the term Preprocessor.**

**Ans** The preprocessor is a program that is invoked by the compiler to process code before compilation. Commands for that program, known as *directives*, are lines of the source file beginning with the character #, which distinguishes them from lines of source program text. The effect of each preprocessor directive is a change to the text of the source code, and the result is a new

source code file, which does not contain the directives. The preprocessed source code, an intermediate file, must be a valid C or C++ program, because it becomes the input to the compiler

Preprocessor directives consist of the following:

Macro definition directives, which replace tokens in the current file with specified replacement tokens

File inclusion directives, which imbed files within the current file

Conditional compilation directives, which conditionally compile sections of the current file

Message generation directives, which control the generation of diagnostic messages

Assertion directives, which specify attributes of the system the program is to run on

The null directive (#), which performs no action

Pragma directives, which apply compiler-specific rules to specified sections of code

**Q17. What is the use of templates in c++.**

**Ans .** Templates are a feature of the C++ programming language that allow functions and classes to operate with generic types. This allows a function or class to work on many different data types without being rewritten for each one.

There are two kinds of templates: *function templates* and *class templates*.

### **Function templates**

A *function template* behaves like a function except that the template can have arguments of many different types. In other words, a function template represents a family of functions. For example, the C++ Standard Library contains the function template `max(x, y)` which returns either  $x$  or  $y$ , whichever is larger. `max()` could be defined like this, using the following template:

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

```
template <typename T>
```

```
const T& max(const T& x, const T& y)
{
    if(y < x)
        return x;
    return y;
}
```

A function template does not occupy space in memory. The actual definition of a function template is generated when the function is called with a specific data type. The function template does not save memory.

### Class templates

A class template provides a specification for generating classes based on parameters. Class templates are commonly used to implement containers. A class template is instantiated by passing a given set of types to it as template arguments. The C++ Standard Library contains many class templates, in particular the containers adapted from the Standard Template Library, such as vector.

#### Q18. What is the use of Virtual function?

**Ans** By default, C++ matches a function call with the correct function definition at compile time. This is called *static binding*. We can specify that the compiler match a function call with the correct function definition at run time; this is called *dynamic binding*. We declare a function with the keyword `virtual` if we want the compiler to use dynamic binding for that specific function.

The following examples demonstrate the differences between static and dynamic binding. The first example demonstrates static binding:

```
#include <iostream.h>
struct A {
    void f() { cout << "Class A" << endl; }
};

struct B: A {
    void f() { cout << "Class B" << endl; }
```

```
};

void g(A& arg) {
    arg.f();
}

int main() {
    B x;
    g(x);
}
```

The following is the output of the above example:

Class A

When function `g()` is called, function `A::f()` is called, although the argument refers to an object of type `B`. At compile time, the compiler knows only that the argument of function `g()` will be a reference to an object derived from `A`; it cannot determine whether the argument will be a reference to an object of type `A` or type `B`. However, this can be determined at run time. The following example is the same as the previous example, except that `A::f()` is declared with the `virtual` keyword:

The following is the output of the above example:

```
#include <iostream.h>

struct A {
    virtual void f() { cout << "Class A" << endl; }
};

struct B: A {
    void f() { cout << "Class B" << endl; }
};

void g(A& arg) {
    arg.f();
}
```



```
int main() {  
    B x;  
    g(x);  
}
```

#### Class B

The `virtual` keyword indicates to the compiler that it should choose the appropriate definition of `f()` not by the type of reference, but by the type of object that the reference refers to.

Therefore, a *virtual function* is a member function we may redefine for other derived classes, and can ensure that the compiler will call the redefined virtual function for an object of the corresponding derived class, even if we call that function with a pointer or reference to a base class of the object.

A class that declares or inherits a virtual function is called a *polymorphic class*.

#### Q19. What is meant by Inline function?

**Ans .** The inline function takes the format as a normal function but when it is compiled it is compiled as inline code. The function is placed separately as inline function, thus adding readability to the source program. When the program is compiled, the code present in function body is replaced in the place of function call.

General Format of inline Function:

The general format of inline function is as follows:

```
inline datatype function_name(arguments)
```

The keyword `inline` specified in the above example, designates the function as inline function. For example, if a programmer wishes to have a function named `inlinexmple` that return value as integer and with no arguments as inline it is written as follows:

```
inline int inlinexmple ( )
```

**Example:**

The concept of inline functions:

```
#include <iostream.h>
int inlinexmple (int);
void main()
{
    int x;
    cout << "\n Enter the Input Value: ";
    cin >> x;
    cout << "\n The Output is: " << inlinexmple
    (x); }
```

```
inline int inlinexmple (int x1)
{
    return 5*x1;
}
```

The output of the above program is:

```
Enter the Input Value: 10
The Output is: 50
```

The output would be the same even when the inline function is written solely as a function. The concept, however, is different. When the program is compiled, the code present in the inline function `inlinexmple ( )` is replaced in the place of function call in the calling program. The concept of inline function is used in this example because the function is a small line of code.

The above example, when compiled, would have the structure as follows:

```
#include <iostream.h>
int inlinexmple (int);
void main()
{
    int x;
    cout << "\n Enter the Input Value: ";
    cin >> x;
```

```
//The inlinexmple (x) gets replaced with code return  
5*x1; cout<<"\n The Output is: " << inlinexmple (x);  
}
```

When the above program is written as normal function the compiled code would look like below:

```
#include <iostream.h>  
int inlinexmple (int);  
void main( )  
{  
int x;  
cout << "\n Enter the Input Value: ";  
cin>>x;  
//Call is made to the function inlinexmple  
  
cout<<"\n The Output is: " << inlinexmple  
(x); }  
  
int inlinexmple (int x1)  
{  
return 5*x1;  
}
```

( )



---

## Chapter 8

# Oops Concepts

---

**Q 1. What is the advantage of using Object Oriented programming?**

**Ans .** The advantage of using Object Oriented programming are :

1. Modularity : each object has a clear semantic (Employer or DrawingDevice), a clear set of methods (getSalary(), getAge(), or drawLine(), drawCircle()).
2. Less bugs : the data are accessed through the methods and we can use them only the way to object"s creator wants us to.
3. Reutilisability : we can extend an existing object, or we can build a new one which could be used in place of the first one, as long as it has all the methods required for example the Employer could be either the CEO or a worker, both of them having the required methods but different data associated to them, DrawingDevice could either be a window, a printer, or anything else.

**Q 2. Define class.**

**Ans .** A class is a user defined data type that consists of data variables and functions binded together. These variables and functions are called member functions and member variables of the class. The member functions are also called methods. The data members are called properties of the class. An object is the instance of the class. An object is like a compound variable of the user defined type. It links both code and data.. The declaration of a class is syntactically same as structure. The class is declared using keyword class. The general form of the declaration of the class is:-

```
class class_name
{
    access_specifier:
```

```
        data functions
    access_specifier:
        data functions
```

```
    } object_list;
```

The object\_list is optional. The object\_list is used to declare objects of the class. The class\_name is the name of the class. The access\_specifier can either public, private or protected. The members of the class by default are private to the class. If the access\_specifier is private then members of the class are not accessible outside the class. If the access\_specifier is public then members of the class can be accessed from outside the class. The protected access\_specifier is needed at the time of inheritance. The members can be accessed using an object's name, a dot operator and name of the member.

**Q 3. Write a program to show how Classes and objects are created.**

**Ans** #include<iostream.h>

```
class cube
{
    public:
        double side;
        double volume()
        {
            return(side*side*side);
        }
};

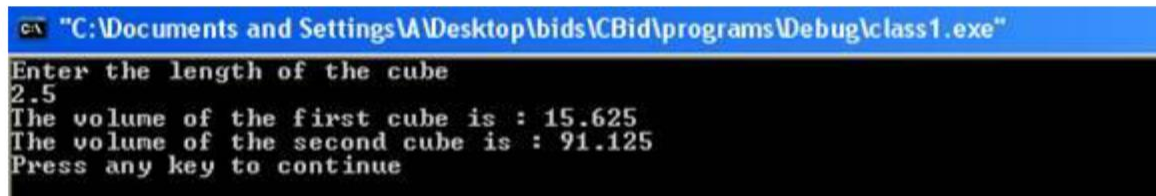
int main()
{
    double volume1=0;
    cube c1,c2;
    cout << "Enter the lenght of the cube" <<
    endl; cin >> c1.side;
    cout << "The volume of the cube is : " << c1.volume() << endl;
```

```

c2.side=c1.side +2;
cout << "The volume of the second cube is : " << c2.volume() <<
endl; return(0);
}

```

The result of the program is:-



```

C:\Documents and Settings\A\Desktop\Bids\CBid\programs\Debug\class1.exe
Enter the length of the cube
2.5
The volume of the first cube is : 15.625
The volume of the second cube is : 91.125
Press any key to continue

```

The program consists of a class cube which has data member side of type double and member function which calculates the volume of the cube.

**Q 4. What is encapsulation?**

**Ans .** Encapsulation is the process of combining data and functions into a single unit called class. Using the method of encapsulation, the programmer cannot directly access the data. Data is only accessible through the functions present inside the class. Data encapsulation led to the important concept of data hiding. Data hiding is the implementation details of a class that are hidden from the user. The concept of restricted access led programmers to write specialized functions or methods for performing the operations on hidden members of the class.

**Q5. Define Constructor and Destructor.**

**Ans** A constructor is a special kind of a function which is the member of the class. The name of the constructor is same as name of the class. A constructor is automatically called when object is created. A constructor does not have a return type. There is a special type of constructor called Default constructor which has no parameters. If no constructor is defined by the user then compiler supplies the default constructor. Once the constructor is defined by the user then compiler does not supply default constructor and then user is responsible for defining default constructor.

A destructor works opposite to that of the constructor. It is used to destroy the objects. The objects are destroyed in order to deallocate the memory occupied by them. The name of the destructor is same as the name of the

constructor as is preceded by a tilt operator „~“. A destructor for objects is executed in the reverse order of the constructor functions.

Example:

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

```
class cube
```

```
{
```

```
    public:
```

```
    double side;
```

```
    double volume()
```

```
    {
```

```
        return(side*side*side);
```

```
    }
```

```
    cube(double side1)
```

```
    {
```

```
        cout << "A constructor is called" << endl;
```

```
        side=side1;
```

```
    }
```

```
    cube()
```

```
    {
```

```
        cout << "A default constructor is called " << endl;
```

```
    }
```

```
    ~cube()
```

```
    {
```

```
        cout << "Destructing " << side << endl;
```

```
    }
```

```
};
```

```
int main()
```

```
{
```

```
    cube c1(2.34);
```

```
    cube c2;
```

```
    cout << "The side of the cube is: " << c1.side << endl;
```

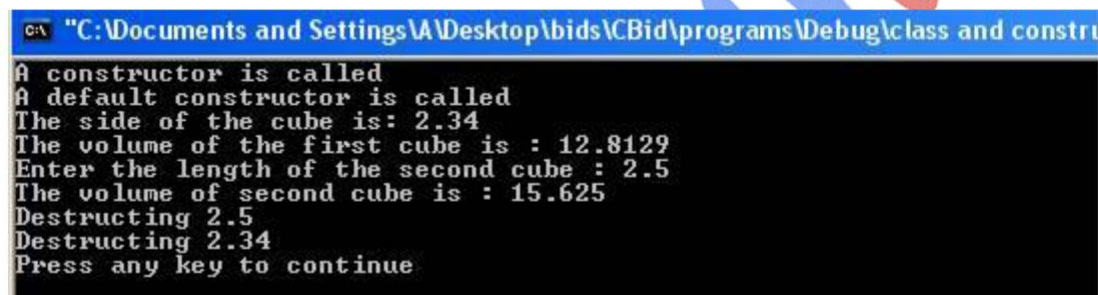


```

    cout << "The volume of the first cube is : " << c1.volume() << endl;
    cout << "Enter the length of the second cube : "
    ; cin >> c2.side;
    cout << "The volume of second cube is : " << c2.volume() <<
    endl; return(0);
}

```

The result of the program is:-



```

C:\Documents and Settings\A\Desktop\abids\CBid\programs\Debug\class and constr
A constructor is called
A default constructor is called
The side of the cube is: 2.34
The volume of the first cube is : 12.8129
Enter the length of the second cube : 2.5
The volume of second cube is : 15.625
Destructing 2.5
Destructing 2.34
Press any key to continue

```

**Q5. Explain the term polymorphism.**

**Ans** In object-oriented programming polymorphism is a generic term that means 'many shapes'. (from the Greek meaning having multiple forms ). Polymorphism is briefly described as one interface many implementations. Polymorphism is a characteristic of being able to assign a different meaning or usage to something in different contexts - specifically to allow an entity such as a variable a function or an object to have more than one form.

There are two types of polymorphism one is compile time polymorphism and the other is run time polymorphism. Compile time polymorphism is functions and operators overloading. Runtime time polymorphism is done using inheritance and virtual functions. Here are some ways how we implement polymorphism in Object Oriented programming languages. Compile time polymorphism -> Operator Overloading Function Overloading Run time polymorphism -> Interface and abstract methods Virtual member functions.

Example:

```
#include <iostream.h>

class figure {
protected:
    double x, y;
public:
    void set_dim(double i, double j=0) {
        x = i;
        y = j;
    }
    virtual void show_area() {
        cout << "No area computation defined ";
        cout << "for this class.\n";
    }
};

class triangle : public figure {
public:
    void show_area() {
        cout << "Triangle with height ";
        cout << x << " and base " << y;
        cout << " has an area of ";
        cout << x * 0.5 * y << ".\n";
    }
};

class square : public figure {
public:
    void show_area() {
        cout << "Square with dimensions ";
        cout << x << "x" << y;
        cout << " has an area of ";
    }
};
```

```
        cout << x * y << ".\n";
    }
};

class csircle : public figure {
public:
    void show_area() {
        cout << "Circle with radius ";
        cout << x;
        cout << " has an area of ";
        cout << 3.14 * x * x << ".\n";
    }
};
```

```
int main()
{
    figure *p; // create a pointer to base type

    triangle t; // create objects of derived types
    square s;
    circle c;

    p = &t;
    p->set_dim(10.0, 5.0);
    p->show_area();

    p = &s;
    p->set_dim(10.0, 5.0);
    p->show_area();

    p = &c;
    p->set_dim(9.0);
    p->show_area();
}
```

```
    return 0;
}
```

**Q 6. What is Inheritance. Explain with example.**

**Ans .** *Inheritance* is a mechanism of reusing and extending existing classes without modifying them, thus producing hierarchical relationships between them.

. Suppose that we declare an object x of class A in the class definition of B. As a result, class B will have access to all the public data members and member functions of class A. However, in class B, we have to access the data members and member functions of class A through object x. The following example demonstrates this:

```
#include <iostream.h>
class A {
    int data;
public:
    void f(int arg) { data = arg; }
    int g() { return data; }
};

class B {
public:
    A x;
};

int main() {
    B obj;
    obj.x.f(20);
    cout << obj.x.g() << endl;
    // cout << obj.g() << endl;
}
```



In the main function, object obj accesses function A::f() through its data member B::x with the statement obj.x.f(20). Object obj accesses A::g() in a similar manner with the statement obj.x.g(). The compiler would not allow the statement obj.g() because g() is a member function of class A, not class B.

The inheritance mechanism lets us use a statement like obj.g() in the above example. In order for that statement to be legal, g() must be a member function of class B.

Inheritance lets us include the names and definitions of another class's members as part of a new class. The class whose members we want to include in our new class is called a *base class*. Our new class is *derived* from the base class. The new class contains a *subobject* of the type of the base class. The following example is the same as the previous example except it uses the inheritance mechanism to give class B access to the members of class A:

```
#include <iostream.h>

class A {
    int data;
public:
    void f(int arg) { data = arg; }
    int g() { return data; }
};

class B : public A { };

int main() {
    B obj;
    obj.f(20);
    cout << obj.g() << endl;
}
```

Class A is a base class of class B. The names and definitions of the members of class A are included in the definition of class B; class B inherits the members of class A. Class B is derived from class A. Class B contains a subobject of type A.

We can also add new data members and member functions to the derived class. We can modify the implementation of existing member functions or

data by overriding base class member functions or data in the newly derived class.

We may derive classes from other derived classes, thereby creating another level of inheritance. The following example demonstrates this:

```
struct A { };  
struct B : A { };  
struct C : B { };
```

Class B is a derived class of A, but is also a base class of C. The number of levels of inheritance is only limited by resources.

**Q 7. What is the use of Multiple Inheritance?**

**Ans .** *Multiple inheritance* allows us to create a derived class that inherits properties from more than one base class. Because a derived class inherits members from all its base classes, ambiguities can result. For example, if two base classes have a member with the same name, the derived class cannot implicitly differentiate between the two members. Note that, when we are using multiple inheritance, the access to names of base classes may be ambiguous.

**Q8. How many types of base classes are there?**

**Ans .** There are two types of Base Class.

1) A *direct base class*

A *direct base class* is a base class that appears directly as a base specifier in the declaration of its derived class.

2) An *indirect base class*

An *indirect base class* is a base class that does not appear directly in the declaration of the derived class but is available to the derived class through one of its base classes. For a given class, all base classes that are not direct base classes are indirect base classes. The following example demonstrates direct and indirect base classes:

```
class A {  
    public:  
    int x;  
};  
class B : public A {
```

```
public:
    int y;
};
class C : public B { };
```

Class B is a direct base class of C. Class A is a direct base class of B. Class A is an indirect base class of C. (Class C has x and y as its data members.)

**Q 9. Define function Overloading and Operator overloading with example.**

**Ans Function Overloading**

We can Overload a function name *f* by declaring more than one function with the name *f* in the same scope. The declarations of *f* must differ from each other by the types and/or the number of arguments in the argument list. When we call an overloaded function named *f*, the correct function is selected by comparing the argument list of the function call with the parameter list of each of the overloaded candidate functions with the name *f*. A *candidate function* is a function that can be called based on the context of the call of the overloaded function name.

Consider a function `print`, which displays an `int`. As shown in the following example, we can overload the function `print` to display other types, for example, `double` and `char*`. We can have three functions with the same name, each performing a similar operation on a different data type:

```
#include <iostream.h>
void print(int i) {
    cout << " Here is int " << i << endl;
}
void print(double f) {
    cout << " Here is float " << f << endl;
}

void print(char* c) {
    cout << " Here is char* " << c << endl;
```

```
}  
  
int main() {  
    print(10);  
    print(10.10);  
    print("ten");  
}
```

The following is the output of the above example:

```
Here is int 10  
Here is float 10.1  
Here is char* ten
```

### Operator Overloading

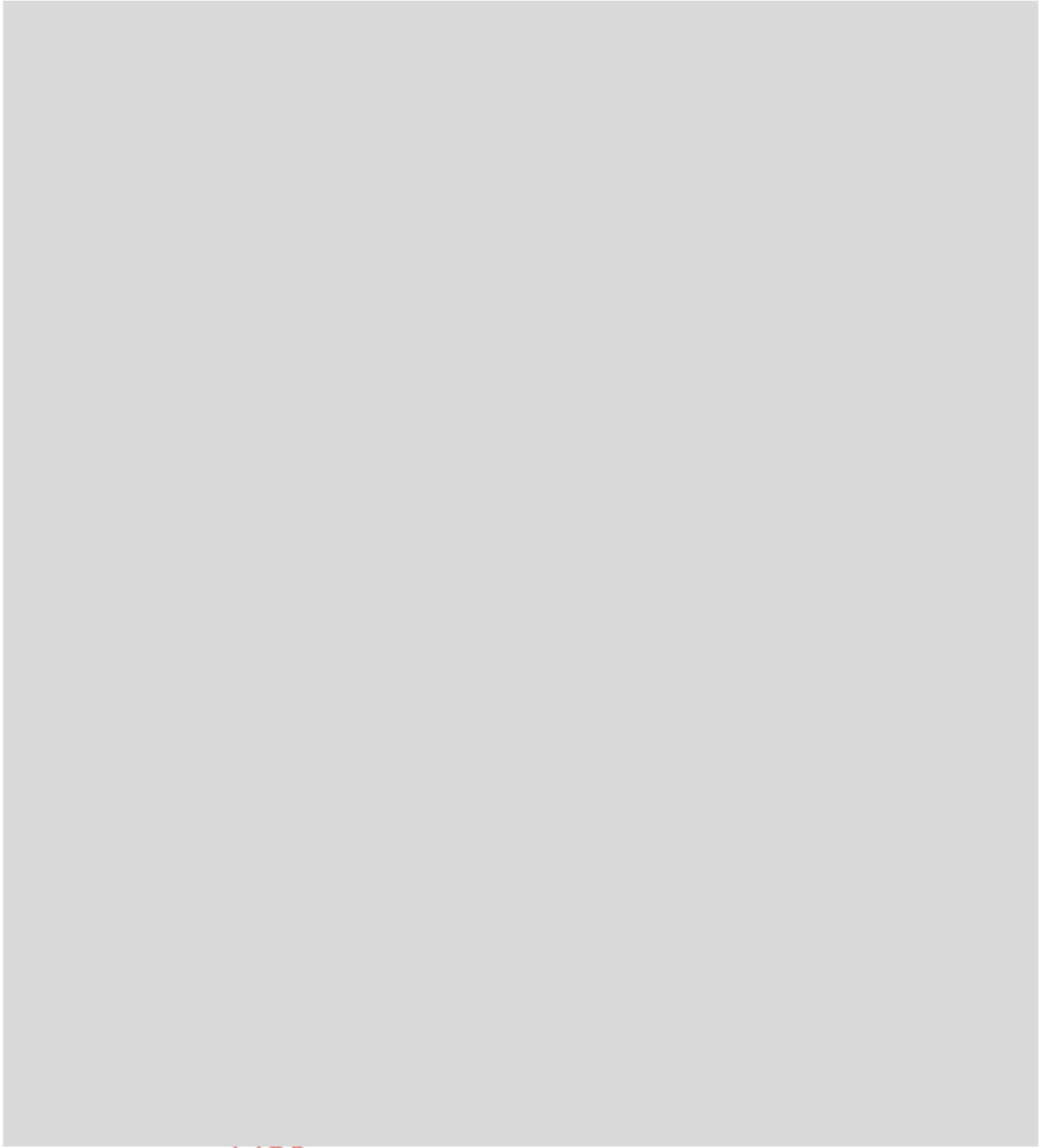
An overloaded operator is called an *operator function*. We declare an operator function with the keyword `operator` preceding the operator. Overloaded operators are distinct from overloaded functions, but like overloaded functions, they are distinguished by the number and types of operands used with the operator.

Consider the standard `+` (plus) operator. When this operator is used with operands of different standard types, the operators have slightly different meanings. For example, the addition of two integers is not implemented in the same way as the addition of two floating-point numbers. C++ allows you to define we own meanings for the standard C++ operators when they are applied to class types. In the following example, a class called `complex` is defined to model complex numbers, and the `+` (plus) operator is redefined in this class to add two complex numbers.

```
// This example illustrates overloading the plus (+)
```

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

```
class complex  
{  
    double real,  
        imag;
```



No.

```

public:
    complx( double real = 0., double imag = 0.); // constructor
    complx operator+(const complx&) const;    // operator+()
};

// define constructor
complx::complx( double r, double i )
{
    real = r; imag = i;
}

// define overloaded + (plus) operator
complx complx::operator+ (const complx& c)
const {
    complx result;
    result.real = (this->real + c.real);
    result.imag = (this->imag + c.imag);
    return result;
}

int main()
{
    complx x(4,4);
    complx y(6,6);
    complx z = x + y; // calls complx::operator+()
}

```

## Chapter 9

# CASE STUDY

---

## 1. //Reversing each word and whole

```
sentence. #include<iostream.h>
#include<conio.h>
#include<string.h>
#include<ctype.h>
#include<stdio.h>
void main()
{
clrscr();
int ch,i,j,k=0,p=0,l;
char str1[50],str2[50];
cout<<"Enter Your Choice : "<<endl<<"1.Each Word
Reversing"<<endl<<"2.Whole Sentence Reversing"<<endl;
cin>>ch;
switch(ch)
{
case 1:
cout<<"Enter the String"<<endl;
gets(str1);
l=strlen(str1);
for(i=0;i<=l;i++)
{
if((str1[i]==' ') || (str1[i]=='\0'))
{
for(j=i-1;j>=p;j--)
{
St2r[k]=str1[j];
```

```
    k++;  
    }  
    Str2[k]=' '  
    k++;  
    p=i+1;  
    }  
    }  
    for(i=0;i<p;i++)  
        cout<<str2[i];  
    break;
```

case 2:

```
    cout<<"Enter the String"<<endl;  
    gets(str1);  
    l=strlen(str1);  
    p=l-1;  
    for(i=l-1;i>=-1;i--)  
    {  
        if((str1[i]==' ') || (i == -1))  
        {  
            for(j=i+1;j<=p;j++)  
            {  
                Str2[k]=str1[j];  
                k++;  
            }  
            Str2[k]=' '  
            k++;  
            p=i-1;  
        }  
    }  
    for(i=0;i<=l;i++)  
        cout<<str2[i];
```



```
break;
```

```
default:
```

```
cout<<"Wrong Choice"<<endl;
```

```
break;
```

```
}
```

```
getch();
```

```
}
```

## **2. //The sum and average of two integers .**

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

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

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

```
void main()
```

```
{
```

```
clrscr();
```

```
int x,y,sum;
```

```
float average;
```

```
cout << "Enter 2 integers : " << endl;
```

```
cin>>x>>y;
```

```
sum=x+y;
```

```
average=sum/2;
```

```
cout << "The sum of " << x << " and " << y << " is " << sum << "." << endl;
```

```
cout << "The average of " << x << " and " << y << " is " << average << "." << endl;
```

```
getch();
```

```
}
```

3. //This program takes in the velocity, acceleration and the time as a screen input from the user.

//The final velocity is calculated using the formula  $v = u + a * t$ , and then outputted using the //'cout' command.

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

void main()
{
    clrscr();
    int x,y,sum;
    float average;
    cout << "Enter 2 integers : " << endl;
    cin >> x >> y;
    sum = x + y;
    average = sum / 2;
    cout << "The sum of " << x << " and " << y << " is " << sum << "." << endl;
    cout << "The average of " << x << " and " << y << " is " << average << "." << endl;
    getch();
}
```

4. //This program takes in an integer num as a screen input from the user. //It then calculates the total value of the integer based on the formula  $x - \frac{1}{3!}x^3 + \frac{1}{5!}x^5 - \frac{1}{7!}x^7 + \frac{1}{9!}x^9$ .

//It then outputs the final answer using the 'cout' command.

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

```
#include <conio.h>
#include <math.h>
int main()
{
    clrscr();
    float factorial=1;
    float num,tot,term,total;
    int i,n=20,index,j=1;
    cout << "Enter a single-digit integer : \n";
    cin>>num;
    tot=num;
    total=num;
    for(i=2,index=3;i<=n;i++,index+=2)
    {
        for(j=1,factorial=1;j<=index;j++)
            factorial*=j;
        tot=tot*pow((double)(-1),(double)(2*i-1))*num*num;
        term=tot/factorial;
        total+=term;
    }
    cout << "Total = " << total << endl;
    getch();
    return 0;
}
```

5. //This program does not take in any screen inputs from the user.  
//It just prints out the first ten lines of the pascal's triangle using the 'cout' command.

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

```
#include <conio.h>
#include <iomanip.h>
long triangle(int x,int y);
int main()
{
clrscr();
const lines=10;
for (int i=0;i<lines;i++)
for (int j=1;j<lines-i;j++)
cout << setw(2) << " ";
for (int j=0;j<=i;j++)
cout << setw(4) << triangle(i,j);
cout << endl;
getch();
}
long triangle(int x,int y)
{
if(x<0 || y<0 || y>x)
return 0;
long c=1;
for (int i=1;i<=y;i++,x--)
c=c*x/i;
return c;
}
```

**6. //This program takes in the elements A[x][y] of the 4 x 4 matrix as a screen input from the user.  
//It then calculates the sum of either of its diagonals and outputs it using the 'cout' command.**

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

```
void main()
{
    clrscr();
    int x;
    int A[4][4],sum=0; //Reading the matrix.
    cout << "Enter the elements of the matrix : " << endl;
    for(int y=0;y<4;y++)
    for (int x=0;x<4;x++)
    {
        cout << "Element " << x+1 << ", " << y+1 << " : ";
        cin>>A[x][y];
    }
    //Sum of either of the diagonal elements.
    for(x=0;x<4;x++)
    for(y=0;y<4;y++)
    if(x==y)
        sum+=A[x][y];
    else if(y==4-(1+1));
        sum+=A[x][y];
    cout << "Sum of either of the diagonal elements is : " << sum;
    getch();
}
```



## Keywords

**actual\_parameter** = Any parameter in the call of a subprogram.

**algorithm** = A description in precise but natural and mathematical language of how a problem is solved.

**argument** = A set of premises connected by correct deductive steps to one or more conclusions.

**arithmetic operators** = Operator is a symbol that is used to perform mathematical operations. addition, subtraction, multiplication and division ideally forming an

**array data type**=Each array associates each value of one data type with a unique object of another type

**assignment statement**=A statement with an expression and one or more variables. The expression is evaluated and the result is stored in the variables

**binding** =A relationship between two things, typically an identifier and some one of its properties or attributes.

**bit** = A bit has one of two values traditionally named 0 and 1. Eight bits make a byte. and four bits a nibble. So 2 nibbles = 1 byte.

**block** =A piece of source code that has one or more declarations in it.

**byte** =eight bits.

**call** =to make use of something by writing its name and the correct protocol.

**class** =a description of a collection of objects that have similar states and behaviors.

**compile** =translate source code into executable object code.

**compound** =a single statement or object that can have any number of other statements as its parts.

**conditional** =an expression or statement that selects one out of a number of alternative subexpressions.

**control\_statement** = statements that permit a processor to select the next of several possible computations according to various conditions.

**data\_type** = A collection of values together with the operations that use them and produce them.

**declaration** = A piece of source code that adds a name to the program's environment and binds it to a class of meanings, and may also define the name. **default** = An item provided in place of an omitted item.

**dynamic\_binding** = A binding that can be made at any time as a program runs and may

**encapsulation** = The ability to hide unwanted details inside an interface so that the result works like a black box or vending machine - providing useful services to many clients (programs or people).

**enumerated data\_type** = data defined by listing its possible values.

**equation** = A formula stating the equality of two or more formula which may be true if the correct values of the variables are found.

**exception** = a mechanism for handling abnormal situations.

**expression** = A shorthand description of a calculation

**formal\_parameter** = The symbol used inside a subprogram in place of the actual\_parameter provided when the subprogram is called.

**function** = A subprogram that returns a value but can not change its parameters or have side effects.

**garbage** = A piece of storage that has been allocated but can no longer be accessed by a program.

**goto** = usable in all practical languages to indicate an unconditional jump.

**header\_file** = A collection of function headers, class interfaces, constants and definitions that is read by a compiler and changes the interpretation of the rest of the program by defining operation for handling strings.

**identifier** := a name used in a programming language to identify something else - a variable, function, procedure, etc.`.

**implementation** =the way something is made to work.

**inheritance** =The ability to easily construct new data types or classes by extending existing structures, data types, or classes.

**inout** =A way of handling parameters that lets a subprogram both use and change the values of an actual parameter. It can be implemented by pass\_by\_reference, pass\_by\_name, or pass\_by\_value\_result.

**input** =data supplied to some program, subprogram, OS, machine, system, or abstraction.

**Int data\_type**=fixed\_point **data** representing a subset of the whole numbers.

**interpreter program**=A program that translates a single instruction of a program and executes it before moving on to the next one.

**keyword\_parameter** =A parameter with a keyword that indicates the formal parameter to which the actual parameter is to be bound.

**late\_binding** =dynamic\_type\_binding **between** pointers to a general type and objects of a more specific type.

**long data type**=A Fixed\_point data type that may have more bits than ints.

**Matrix** = that can be implemented by rectangular arrays and has many of the arithmetic\_operations defined on them.

**method** =something that an object can perform or suffer often determined by the class of the object rather than the specific object. In C++ a method is called a member function.

**natural\_numbers** =The numbers 1,2,3,4...

**object** =Object is uniquely identifiable by the user.

**operation** =One of a set of functions with special syntax and semantics that can be used to construct an expression.

**operator** =A symbol for an operation. Operators can infix, prefix, or postfix.

**overloading** =giving multiple meanings to a symbol depending on its context.

**parameter** =Parameter is used in a subprogram that can be changed when the subprogram is called.



**parameter\_passing** =the means by which the actual\_parameters in a call of a subprogram are connected with the formal\_parameters in the definition of the subprogram.

**pass\_by\_value** =parameter\_passing **where** The actual parameter is evaluated (if necessary) and the value placed in a location bound to the formal parameter.

**pass\_by\_reference** =parameter\_passing **where** The parameter is implemented by providing an access path to the actual parameter from the formal parameter. Actions written as if they use or change the formal parameter use or change the actual parameter instead. **pointer** =data\_type with values that are addresses of other items of data.

**polymorphism objects**=The ability of a function to apply to more than one type of object or data.

**postfix** =An operator that is placed after its single operand.

**prefix** r=An operator that is placed in front of its single operand.

**recursion** =A technique of defining something in terms of a smaller or simpler object of the same type. If you don't understand this then see recurse.

**relational\_operator** =an infix operator that returns a Boolean value when given non-Boolean operands.

**scope** =the parts of a program where a particular identifier has a particular meaning (set of bindings).

**selection** =a statement that chooses between several possible executions paths in a program.

**semantics** =A description of how the meaning of a valid statement or sentence can be worked out from its parsed form.

**stack** =A collection of data items where new items are added and old items retrieved at the same place, so that the last item added is always the first item retrieved, and so on. **structure**

**data\_type**=A finite collection of named items of data of different types. **subprogram** =A piece of code that has been named and can be referred to by that name (called) as many times as is needed. Either a procedure or a function.

**subtype** =A type S is a subtype of type T if every valid operation on an object of type T is also a a valid operation of type S.

**syntax** =A description of the rules that determine the validity and parsing of sentences or statements in a language.

**ternary** = Ternary operators have two operands. Ternary numbers have base 3 and use 3 symbols.

**tree** =A collection of connected objects called nodes with all nodes connected indirectly by precisely one path. An ordered tree has a root and the connections lead from this root to all other nodes. Nodes at the end of the paths are called leaves. The connections are called branches. All computer science trees are drawn upside-down with the root at the top and the leaves at the bottom.

**unary** = unary operators have one operand, unary numbers use base 1 and one symbol.

**BACHELOR OF COMPUTER APPLICATIONS**  
**(Part-I) EXAMINATION**  
**(Faculty of Science)**  
(Three – Year Scheme of 10+2+3 Pattern)  
**PAPER 119**  
**PRINCIPLES OF PROGRAMMING LANGUAGES**

---

**OBJECTIVE PART- I**

**Year - 2011**

**Time allowed : One Hour**

**Maximum Marks : 20**

*The question paper contains 40 multiple choice questions with four choices and student will have to pick the correct one (each carrying ½ mark).*

1. The C language was developed in:  
(a) 1987 (b) 1972  
(c) 1946 (d) None of the above ( )
2. The C language was developed by:  
(a) Dennis Ritchie (b) Tim Bemer Lee  
(c) Ken Thompson (d) None of the above ( )
3. BCPL is:  
(a) Beginner Computer Programming Language  
(b) Basic Computer Programs Language  
(c) Basic Combined Programming Language  
(d) None of the above ( )
4. Which of the following is a keyword?  
(a) Include  
(b) Void  
(c) Struct  
(d) All of the above ( )
5. The name given to any memory location is called.....  
(a) Keywords (b) Constant  
(c) Character Set (d) Variable ( )

6. Which of the following is not a data type in C?  
(a) Float (b) Double  
(c) Char (d) Struct ( )
7. The range of unsigned char data type in C is.....  
(a) -128 to 127 (b) 0 to 255  
(c) - 32768 to 32767 (d) 0 to 65535 ( )
8. ....are data values that never change their values during program execution.  
(a) Variable  
(b) Constant  
(c) Function  
(d) None of the above ( )
9. % \ d format specifier is used for.....  
(a) Signed decimal integer  
(b) String  
(c) Single character  
(d) Singed long decimal integer ( )
10. Which of the following is a Run time Error?  
(a) Division by 0 (b) Stack overflow  
(c) Dreaded null pointer assignment (d) None of the above ( )
11. Which ascape sequence represents the newline character?  
(a) \t (b) \a  
(c) \n (d) \b ( )
12. Which of the following are valid variables?  
(a) x (b) area  
(c) sum (d) all of the above ( )
13. What will be the output of the following program segment? include <stdio.h>  
void main ( ) {  
int a ; a = 25 + 4 - 4 ; printf (" %d", a) ; }  
(a) 25 (b) 29  
(c) 33 (d) None of the above ( )
14. Language used for business purpose is:

- (a) C (b) C +  
(c) COBOL (d) FORTRAN ( )
15. If  $a = 10$  and  $b = 3$  what is the value of  $a/b$  ?  
(a) 3.33 (b) -3  
(c) 3 (d) None of the above ( )
16. Element of an array are stored in:  
(a) Continuous memory location  
(b) Non-continuous memory location  
(c) It depends on the program  
(d) None of the above ( )
17. The switch feature:  
(a) Can not always be replaced by a nested if then else clause  
(b) Can always be replaced by a nested if then else clause  
(c) Can not enhance logical clarity  
(d) None of the above ( )
18. The conditional operator,  $?:$  is a .....  
(a) Unary Operator  
(b) Binary Operator  
(c) Ternary operator  
(d) None of the above ( )
19. What is the output of  $\text{sqrt}(24 + 6/5)$ ?  
(a) 24 (b) 6/5  
(c) 6 (d) 5 ( )
20. The.....function is used to return a single character from a standard input device :  
(a) `putch ( )` (b) `putchar ( )`  
(c) `getchar ( )` (d) none of the above ( )
21. Which of the following is a storage class specifiers in „C“?  
(a) `auto`  
(b) `extern`  
(c) `static`  
(d) all of the above ( )
22. The `cin` and `cout` objects required the header file:



- (a) `stdio.h`  
(b) `iostream.h`  
(c) `iostream.h`  
(d) none of the above ( )
23. The members of a class can be:  
(a) Private  
(b) Public  
(c) Protected  
(d) All of the above ( )
24. Inheritance is a way to:  
(a) make a new class  
(b) pass arguments to objects of a class  
(c) add features to existing classes without rewriting them  
(d) improve data hiding and encapsulation ( )
25. `<< endl` to `cout` is equivalent is:  
(a) `"\n"`  
(b) `„\t“`  
(c) `„\b“`  
(d) None of the above ( )
26. The binding of data and function together into a single class type variable is known to as.....  
(a) Encapsulation  
(b) Inheritance  
(c) Polymorphism  
(d) None of the above ( )
27. The memory space for object is allocated.....  
(a) When the class is specified  
(b) When they are declared  
(c) Both of the above  
(d) None of the above ( )
28. A..... is a special member function whose task is to initialize the object of its class :  
(a) Constructor  
(b) Destructor  
(c) Polymorphism  
(d) None of the above ( )

29. LISP means:  
(a) Logic Programming  
(b) List Programming  
(c) List Function  
(d) Logic Function ( )
30. Choose the storage class type which access the variable fastly:  
(a) Automatic storage class  
(b) Static storage class  
(c) Register storage class  
(d) External storage class ( )
31. A destructor :  
(a) has a return type  
(b) can have parameters  
(c) has same name as class  
(d) All of the above ( )
32. A C++ class can hold:  
(a) only data  
(b) only function  
(c) both data and function  
(d) none of the above ( )
33. Which of the following are manipulators in C++ ?  
(a) endl  
(b) new  
(c) both (a) and (b)  
(d) none of the above ( )
34. ....refers to the act of representing essential features without including the background details or explanations.  
(a) Inheritance  
(b) Data abstraction  
(c) Encapsulation  
(d) None of the above ( )
35. The ability to take more than one form in C ++ is.....  
(a) Message Passing  
(b) Dynamic Binding  
(c) Inheritance



- (d) Polymorphism ( )
36. We can give several meaning to an operator. This process is known as.....  
 (a) Polymorphism  
 (b) Inheritance  
 (c) Operator overloading  
 (d) Function overloading ( )
37. Which of the following is a memory management operator?  
 (a) malloc ( )  
 (b) new  
 (c) delete  
 (d) all of the above ( )
38. Which of the following is a token in C++ ?  
 (a) Keywords (b) Constant  
 (c) Strings (d) All of the above ( )
39. Cout is:  
 (a) a keywords (b) an object  
 (c) a library function (d) a variable ( )
40. Among the following which one is the scope resolution operator?  
 (a) \*^ (b) →  
 (c) :: (d) ! ( )

**Answer Key**

1. ( b )	2. ( a )	3. ( c )	4. ( d )	5. ( d )	6. ( d )	7. ( b )	8. ( b )	9. ( d )	10. ( b )
11. ( c )	12. ( d )	13. ( a )	14. ( c )	15. ( c )	16. ( a )	17. ( a )	18. ( c )	19. ( d )	20. ( c )
21. ( d )	22. ( b )	23. ( d )	24. ( c )	25. ( a )	26. ( a )	27. ( b )	28. ( a )	29. ( b )	30. ( c )
31. ( c )	32. ( c )	33. ( c )	34. ( b )	35. ( d )	36. ( c )	37. ( d )	38. ( d )	39. ( c )	40. ( c )

## SOLVED DESCRIPTIVE PART-II

Year- 2011

**Time allowed : 2 Hours**

**Maximum Marks : 30**

*Attempt any four questions out of the six. All questions carry 7½ marks each.*

**Q.1a What is meant by scope and lifetime of variables?**

**Ans** A storage class defines the scope (visibility) and life time of variables or functions within a Program. There are following storage classes which can be used in a C Program

- auto
- register
- static
- extern

### **The auto Storage Class**

The auto storage class is the default storage class for all local variables.

```
{  
  int mount;  
  auto int month;  
}
```

The example above defines two variables with the same storage class, auto can only be used within functions, i.e. local variables.

### **The register Storage Class**

The register storage class is used to define local variables that should be stored in a register instead of RAM. This means that the variable has a maximum size equal to the register size (usually one word) and can't have the unary '&' operator applied to it (as it does not have a memory location).

```
{  
  register int miles;  
}
```

The register should only be used for variables that require quick access such as counters. It should also be noted that defining 'register' goes not mean that the variable will be stored in a register. It means that it **MIGHT** be stored in a register depending on hardware and implementation restrictions.

**The static Storage Class**

The static storage class instructs the compiler to keep a local variable in existence during the lifetime of the program instead of creating and destroying it each time it comes into and goes out of scope. Therefore, making local variables static allows them to maintain their values between function calls.

The static modifier may also be applied to global variables. When this is done, it causes that variable's scope to be restricted to the file in which it is declared.

**The extern Storage Class**

The extern storage class is used to give a reference of a global variable that is visible to ALL the program files. When you use 'extern' the variable cannot be initialized as all it does is point the variable name at a storage location that has been previously defined.

When you have multiple files and you define a global variable or function which will be used in other files also, then extern will be used in another file to give reference of defined variable or function. Just for understanding extern is used to declare a global variable or function in another files.

The extern modifier is most commonly used when there are two or more files sharing the same global variables or functions as explained below. First File: main.c

```
#include <stdio.h>
```

```
int count ;  
extern void write_extern();
```

```
main()  
{  
    write_extern();  
}
```

Second File: write.c

```
#include <stdio.h>
```

```
extern int count;
```

```
void write_extern(void)  
{  
    count = 5;  
    printf("count is %d\n", count);  
}
```

Here extern keyword is being used to declare count in the second file where as it has its definition in the first file. Now compile these two files as follows:

```
$gcc main.c write.c
```

This will produce a.out executable program, when this program is executed, it produces following result: 5

### Summary

Storage class	Storage	Default initial value	Scope	Life
Automatic	Memory	Garbage	Local to the block in which variable is defined	Till the control remains within the block in which the variable is defined
Register	CPU registers	Garbage	Local to the block in which variable is defined	Till the control remains within the block in which the variable is defined
Static	Memory	Zero	Local to the block in which variable is defined	Value of variable persists between different function calls.
External	Memory	Zero	Global	As long as program execution doesn't come to end.

### Q1.b Discuss the unconditional branching.

A **goto** :-It allows to make an absolute jump to another point in the program. We should use this feature with caution since its execution causes an unconditional jump ignoring any type of nesting limitations.

The destination point is identified by a label, which is then used as an argument for the goto statement. A label is made of a valid identifier followed by a colon (:). goto loop example

```
#include <stdio.h>
int main ()
{
    int n=10; loop:
    printf("%d", n);
```

```
n--;  
if (n>0)  
goto loop;  
printf( "FIRE");  
}  
10, 9, 8, 7, 6, 5, 4, 3, 2, 1, FIRE!
```

**Q1.c Discuss the various parameter passing methods.**

A There are two ways to pass parameters to a function:

**Pass by Value:**

This mechanism is used when we don't want to change the value of passed parameters. When parameters are passed by value then functions in C create copies of the passed variables and do required processing on these copied variables.

```
int main()  
{  
    int a = 10;  
    int b = 20;  
  
    printf("Before: Value of a = %d and value of b = %d\n", a, b  
); swap( a, b );  
    printf("After: Value of a = %d and value of b = %d\n", a, b );  
}  
  
void swap( int p1, int p2 )  
{  
    int t;  
  
    t = p2;  
    p2 = p1;  
    p1 = t;  
    printf("Value of a (p1) = %d and value of b(p2) = %d\n", p1, p2 );  
}
```

Before: Value of a = 10 and value of b = 20  
Value of a (p1) = 20 and value of b(p2) = 10  
After: Value of a = 10 and value of b = 20

**Pass by Reference :**

This mechanism is used when you want a function to do the changes in passed parameters and reflect those changes back to the calling function. In this case only addresses of the variables are passed to a function so that function can work directly over the addresses.

```

void swap( int *p1, int *p2 );

int main()
{
    int a = 10;
    int b = 20;

    printf("Before: Value of a = %d and value of b = %d\n", a, b
); swap( &a, &b );
    printf("After: Value of a = %d and value of b = %d\n", a, b );
}

void swap( int *p1, int *p2 )
{
    int t;

    t = *p2;
    *p2 = *p1;
    *p1 = t;
    printf("Value of a (p1) = %d and value of b(p2) = %d\n", *p1, *p2 );
}
before: Value of a = 10 and value of b = 20
Value of a (p1) = 20 and value of b(p2) = 10
After: Value of a = 20 and value of b = 10

```

### Q2.a Discuss various operators in C

A An operator is a symbol that operates on a certain data type and produces the output as the result of the operation.

Eg. expression 4 + 5 is equal to 9. Here 4 and 5 are called operands and + is called operator.

#### Category of operators

**Unary Operators:-**A unary operator is an operator, which operates on one operand.

**Binary:-**A binary operator is an operator, which operates on two operands

**Ternary:-**A ternary operator is an operator, which operates on three operands.

### C contains the following operator groups

#### 1 Arithmetic Operator

The arithmetic operator is a binary operator, which requires two operands to perform its operation of arithmetic. Following are the arithmetic operators that are available.

Operator	Description	Eg.
+	Addition	a+b
-	Subtraction	a-b

/	Division	a/b
*	Multiplication	a*b
%	Modulo or remainder	a%b

## 2 Relational Operators

Relational operators compare between two operands and return in terms of true or false i.e. 1 or 0. In C and many other languages a true value is denoted by the integer 1 and a false value is denoted by the integer 0. Relational operators are used in conjunction with logical operators and conditional & looping statements.

<	Less than
>	Greater than
<=	Less than or equal to
>=	Greater than or equal to
!=	Not equal to
==	Equal to

## 3 Logical Operators

A logical operator is used to compare or evaluate logical and relational expressions. There are three logical operators available in the C language.

&& Logical AND  
|| Logical OR  
! Logical NOT

## 4 Assignment operator

An assignment operator (=) is used to assign a constant or a value of one variable to another.

Example:

```
a = 5;  
b = a;  
rate = 10.5  
net = (a/b) * 100;
```

- \* There is always difference between the equality operator (==) and the assignment operator (=).

## 5 Conditional or Ternary Operator

A conditional operator checks for an expression, which returns either a true or a false value. If the condition evaluated is true, it returns the value of the true section of the operator, otherwise it returns the value of the false section of the operator.

Its general structure is as follows:

Expression1 ? expression 2 (True Section): expression3 (False Section)

Example:

```
a=3,b=5,c;
```



`c = (a>b) ? a+b : b-a;`

The variable `c` will have the value 2, because when the expression `(a>b)` is checked, it is evaluated as false. Now because the evaluation is false, the expression `b-a` is executed and the result is returned to `c` using the assignment operator.

## 6 Bitwise Operators:

These are used to perform bitwise operations such as testing the bits, shifting the bits to left or right, one's complement of bits. This operator can be applied on only `int` and `char` data type.

& AND

| Inclusive OR

^ Exclusive OR

<< Shift Left

>> Shift Right

~ One's complement

`~A = 1100 0011`

## 7 Increment and Decrement Operators

These operators are unary operators.

The increment and decrement operators are very useful in C language. They are extensively used in `for` and `while` loops. The syntax of these operators is given below.

`++`

`--`

## 8 Comma operator ( , ):-

The comma operator (,) is used to separate two or more expressions that are included where only one expression is expected. When the set of expressions has to be evaluated for a value, only the rightmost expression is considered.

## Q2.b Differentiate between structures and unions.

**Ans** Union allocates the memory equal to the maximum memory required by the member of the union but structure allocates the memory equal to the sum of the memory allocated to its each individual members.

In Union, one block is used by all the members of union but in case of structure, each member has their own memory space.

Example:

Structure:

```
#include <stdio.h>
```

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

```
int main()
```

```
{
```

```
    struct testing
```

```
{
int a;
char b;
float c;
}var;
clrscr();
printf("Size of var=%d\n",sizeof(var));
printf("Size of a=%d\n",sizeof(var.a));
printf("Size of b=%d\n",sizeof(var.b));
printf("Size of c=%d\n",sizeof(var.c));
var.a=10;
var.b="w";
var.c=3.1422;
printf("value of a=%d\n",var.a);
printf("value of b=%c\n",var.b);
printf("value of c=%f\n",var.c);
getch();
return 0;
}
Union:
#include <stdio.h>
#include <conio.h>
int main()
{
union testing
{
int a;
char b;
float c;
}var;
clrscr();
printf("Size of var=%d\n",sizeof(var));
printf("Size of a=%d\n",sizeof(var.a));
printf("Size of b=%d\n",sizeof(var.b));
printf("Size of c=%d\n",sizeof(var.c));
var.a=10;
var.b="w";
var.c=3.1422;
printf("value of a=%d\n",var.a);
printf("value of b=%c\n",var.b);
printf("value of c=%f\n",var.c);
```

```

getch();
return 0;
} Output of both these programs will be same. The difference lies in the storage.

```

**Q3.a Write a program to input any number and print them in reverse order using 'WHILE statement' in C.**

Ans

```

#include<stdio.h>
void main()
{
    long int number,reverse,n;
    clrscr();
    printf("Enter the number");
    scanf("%ld",&number);
    reverse=0;
    while(number!=0)
    {
        n=number%10;
        reverse=reverse*10+n;
        number=number/10;
    }
    printf("The reverse number=%ld\n",reverse);
    getch();
}

```

**Q3.b Differentiate between constructors and destructors.**

Ans **A constructor** is a special kind of a function which is the member of the class. The name of the constructor is same as name of the class. A constructor is automatically called when object is created. A constructor does not have a return type. There is a special type of constructor called Default constructor which has no parameters. If no constructor is defined by the user then compiler supplies the default constructor. Once the constructor is defined by the user then compiler does not supply default constructor and then user is responsible for defining default constructor.

**A destructor** works opposite to that of the constructor. It is used to destroy the objects. The objects are destroyed in order to deallocate the memory occupied by them. The name of the destructor is same as the name of the constructor as is preceded by a tilt operator „~“. A destructor for objects is executed in the reverse order of the constructor functions.

Example:

```

#include<iostream.h>
class cube
{
    public:

```

```
double side;
double volume()
{
    return(side*side*side);
}
cube(double side1)
{
    cout << "A constructor is called" << endl;
    side=side1;
}
cube()
{
    cout << "A default constructor is called " << endl;
}
~cube()
{
    cout << "Destructing " << side << endl;
}

};

int main()
{
    cube c1(2.34);
    cube c2;
    cout << "The side of the cube is: " << c1.side << endl;
    cout << "The volume of the first cube is : " << c1.volume() << endl;
    cout << "Enter the length of the second cube : " ;
    cin >> c2.side;
    cout << "The volume of second cube is : " << c2.volume() <<
    endl; return(0);
}
```

**Q4.a Write short notes on:****Ans Exceptions**

Exceptions provide a way to react to exceptional circumstances (like runtime errors) in our program by transferring control to special functions called handlers.

To catch exceptions we must place a portion of code under exception inspection. This is done by enclosing that portion of code in a try block. When an exceptional circumstance arises within that block, an exception is thrown that transfers the control to the exception handler. If no exception is thrown, the code continues normally and all handlers are ignored.

An exception is thrown by using the `throw` keyword from inside the `try` block. Exception handlers are declared with the keyword `catch`, which must be placed immediately after the `try` block:

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

int main () {
    try
    {
        throw 20;
    }
    catch (int e)
    {
        cout << "An exception occurred. Exception Nr. " << e << endl;
    }
    return 0;
}
```

The code under exception handling is enclosed in a `try` block. In this example this code simply throws an exception:

```
throw 20;
```

A `throw` expression accepts one parameter (in this case the integer value 20), which is passed as an argument to the exception handler.

The exception handler is declared with the `catch` keyword. As you can see, it follows immediately the closing brace of the `try` block. The `catch` format is similar to a regular function that always has at least one parameter. The type of this parameter is very important, since the type of the argument passed by the `throw` expression is checked against it, and only in the case they match, the exception is caught.

We can chain multiple handlers (catch expressions), each one with a different parameter type. Only the handler that matches its type with the argument specified in the `throw` statement is executed.

#### Q4b Stack and Queues

A **stack** is a container of objects that are inserted and removed according to the last-in first-out (LIFO) principle. In the pushdown stacks only two operations are allowed: push the item into the stack, and pop the item out of the stack. A stack is a limited access data structure - elements can be added and removed from the stack only at the



top. push adds an item to the top of the stack, pop removes the item from the top. A helpful analogy is to think of a stack of books; you can remove only the top book, also you can add a new book on the top. stack is a recursive data structure.

#### Queues

A queue is a container of objects (a linear collection) that are inserted and removed according to the first-in first-out (FIFO) principle. An excellent example of a queue is a line of students in the food court of the UC. New additions to a line made to the back of the queue, while removal (or serving) happens in the front. In the queue only two operations are allowed enqueue and dequeue. Enqueue means to insert an item into the back of the queue, dequeue means removing the front item. The picture demonstrates the FIFO access

The difference between stacks and queues is in removing. In a stack we remove the item the most recently added; in a queue, we remove the item the least recently added.

#### Q4c Enumerated Data Types

A Enumerated data type variables can only assume values which have been previously declared.

```
enum month { jan = 1, feb, mar, apr, may, jun, jul, aug, sep, oct, nov, dec
}; enum month this_month;
this_month = feb;
```

In the above declaration, month is declared as an enumerated data type. It consists of a set of values, jan to dec. Numerically, jan is given the value 1, feb the value 2, and so on. The variable this\_month is declared to be of the same type as month, then is assigned the value associated with feb. This\_month cannot be assigned any values outside those specified in the initialization list for the declaration of month.

#### Q5a Discuss the basic concepts of object oriented programming.

A OOP stands for Object Oriented Programming and the language that support this Object Oriented programming features is called Object oriented Programming Language. An example of a language that support this Object oriented features is C++.

#### Features of Object oriented Programming

The Objects Oriented programming language supports all the features of normal programming languages. In addition it supports some important concepts and terminology which has made it popular among programming methodology.

The important features of Object Oriented programming are:

- Inheritance
- Polymorphism
- Data Hiding
- Encapsulation
- Overloading
- Reusability

But before that it is important to know some new terminologies used in Object Oriented programming namely

- Objects
- Classes

Objects:

In other words object is an instance of a class.

Classes:

These contain data and functions bundled together under a unit. In other words class is a collection of similar objects. When we define a class it just creates template or Skelton. So no memory is created when class is created. Memory is occupied only by object.

Example:

```
Class classname
{
    Data
    Functions
};
main ( )
{
    classname objectname1,objectname2,...;

}
```

In other words classes acts as data types for objects.

**Member functions:**

The functions defined inside the class as above are called member functions.

**Data Hiding:**

This concept is the main heart of an Object oriented programming. The data is hidden inside the class by declaring it as private inside the class. When data or functions are defined as private it can be accessed only by the class in which it is defined. When data or functions are defined as public then it can be accessed anywhere outside the class. Object Oriented programming gives importance to protecting data which in any system. This is done by declaring data as private and making it accessible only to the class in which it is defined. This concept is called data hiding. But one can keep member functions as public.

So above class structure becomes

**Example:**

```
Class classname
{
    private:
    datatype data;
```



```
public:
    Member functions
};
main ( )
{
    classname objectname1,objectname2,...;
}
```

**Encapsulation:**

The technical term for combining data and functions together as a bundle is encapsulation.

**Inheritance:**

Inheritance as the name suggests is the concept of inheriting or deriving properties of an existing class to get new class or classes. In other words we may have common features or characteristics that may be needed by number of classes. So those features can be placed in a common tree class called base class and the other classes which have these characteristics can take the tree class and define only the new things that they have on their own in their classes. These classes are called derived class. The main advantage of using this concept of inheritance in Object oriented programming is it helps in reducing the code size since the common characteristic is placed separately called as base class and it is just referred in the derived class. This provides the users the important usage of terminology called as reusability.

**Reusability:**

This usage is achieved by the above explained terminology called as inheritance. Reusability is nothing but re-usage of structure without changing the existing one but adding new features or characteristics to it. It is very much needed for any programmers in different situations. Reusability gives the following advantages to users.

It helps in reducing the code size since classes can be just derived from existing one and one need to add only the new features and it helps users to save their time.

For instance if there is a class defined to draw different graphical figures say there is a user who wants to draw graphical figure and also add the features of adding color to the graphical figure. In this scenario instead of defining a class to draw a graphical figure and coloring it what the user can do is make use of the existing class for drawing graphical figure by deriving the class and add new feature to the derived class namely add the feature of adding colors to the graphical figure.

**Polymorphism and Overloading:**

Poly refers many. So Polymorphism as the name suggests is a certain item appearing in different forms or ways. That is making a function or operator to act in different forms depending on the place they are present is called Polymorphism. Overloading is a kind of polymorphism. In other words say for instance we know that +, - operate on

integer data type and is used to perform arithmetic additions and subtractions. But operator overloading is one in which we define new operations to these operators and make them operate on different data types in other words overloading the existing functionality with new one. This is a very important feature of object oriented programming methodology which extended the handling of data type and operations.

**Q5.b Write a program to check whether the number is prime or not.**

**A**

```
#include<stdio.h>

main()
{
    int n, i = 3, count, c;

    printf("Enter the number of prime numbers required\n");
    scanf("%d",&n);

    if ( n >= 1 )
    {
        printf("First %d prime numbers are :\n",n);
        printf("\n");
    }

    for ( count = 2 ; count <= n ; )
    {
        for ( c = 2 ; c <= i - 1 ; c++ )
        {
            if ( i%c == 0 )
                break;
        }
        if ( c == i )
        {
            printf("%d\n",i);
            count++;
        }
        i++;
    }

    return 0;
}
```

**Q6.a Write a program to generate the following pattern :**

A

```

      1
    2 2
  3 3 3
4 4 4 4

```

A

```

#include<stdio.h>
main()
{
  int count = 40,i;
  clrscr();
  for(i=1;i<=u;i++)
  {
    printf("\n\n");
    for(int j=1;j<=count;j++)
    {
      printf(" ");
    }
    for(int k=1;k<=i;k++)
    {
      printf("%8d",i);
    }
    count=count-4;
  }
  getch();
}

```

**Q6b. Write a short note on linked lists and virtual functions**

Ans **Linked list**

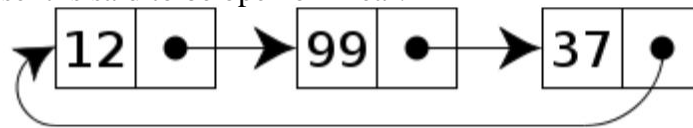
A linked list is a technique of creating a list with the ability to add, delete, or retrieve items. Additional operations can also be provided to a more elaborate list such as finding an item, deleting an item, etc.

Linked lists are among the simplest and most common data structures. They can be used to implement several other common abstract data types, including stacks, queues, associative arrays, and symbolic expressions, though it is not uncommon to implement the other data structures directly without using a list as the basis of implementation.

#### **Linear and circular lists**

In the last node of a list, the link field often contains a null reference, a special value used to indicate the lack of further nodes. A less common convention is to make it

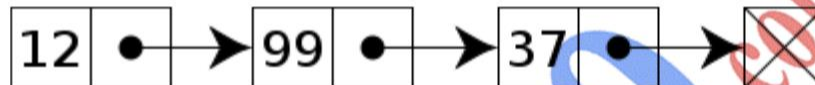
point to the first node of the list; in that case the list is said to be circular or circularly linked; otherwise it is said to be open or linear.



A circular linked list

### Singly, doubly, and multiply linked lists

Singly linked lists contain nodes which have a data field as well as a next field, which points to the next node in the linked list.



A singly linked list whose nodes contain two fields: an integer value and a link to the next node

In a doubly linked list, each node contains, besides the next-node link, a second link field pointing to the previous node in the sequence. The two links may be called forward(s) and backwards, or next and prev(ious).



A doubly linked list whose nodes contain three fields: an integer value, the link forward to the next node, and the link backward to the previous node

A technique known as XOR-linking allows a doubly linked list to be implemented using a single link field in each node. However, this technique requires the ability to do bit operations on addresses, and therefore may not be available in some high-level languages.

**In a multiply linked list**, each node contains two or more link fields, each field being used to connect the same set of data records in a different order (e.g., by name, by department, by date of birth, etc.). (While doubly linked lists can be seen as special cases of multiply linked list, the fact that the two orders are opposite to each other leads to simpler and more efficient algorithms, so they are usually treated as a separate case.)

In the case of a circular doubly linked list, the only change that occurs is that end, or "tail", of the said list is linked back to the front, or "head", of the list and vice versa.

### Virtual Function

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

Base class pointer can point to derived class object. In this case, using base class pointer if we call some function which is in both classes, then base class function is invoked. But if we want to invoke derived class function using base class pointer, it can be achieved by defining the function as virtual in base class, this is how virtual functions support runtime polymorphism.

Consider following program code:

```
Class A
{
    int a;
    public:
    A()
    {
        a = 1;
    }
    virtual void show()
    {
        cout <<a;
    }
};
Class B: public A
{
    int b;
    public:
    B()
    {
        b = 2;
    }
    virtual void show()
    {
        cout <<b;
    }
};
int main()
{
    A *pA;
    B oB;
    pA = &oB;
    pA->show();
    return 0;
}
```

Output is 2 since pA points to object of B and show() is virtual in base class A.



## PRINCIPLES OF PROGRAMMING LANGUAGES

### PAPER 119

#### OBJECTIVE PART- I

Year - 2010

**Time allowed : One Hour**

**Maximum Marks : 20**

*The question paper contains 40 multiple choice questions with four choices and student will have to pick the correct one (each carrying ½ mark).*

1. The statement `char ch = 'z'` would store in `ch`:  
(a) the character `z`  
(b) ASCII value of `z`  
(c) `z` character along with single inverted commas  
(d) Both (a) and (b) ( )
2. The maximum value that an single constant can have is:  
(a) `- 32767` (b) `32767`  
(c) `1.7014e +38` (d) `-1.7001e +38` ( )
3. .NET language are implemented with:  
(a) Pure interpretation  
(b) Compiler Implementation  
(c) JIT Hybrid implementation  
(d) None of the above ( )
4. Which one of the following language is not implemented using pure interpretation?  
(a) Syntax  
(b) Semantic of programming languages  
(c) Both a and b  
(d) None of the above ( )
5. A context – free grammar is useful to describe:  
(a) Syntax of programme languages  
(b) Semantic of programming languages  
(c) both a and b  
(d) None of the above ( )
6. LISP means:  
(a) Logic processing (b) List processing

- (c) List function (d) Logic programming ( )
7. Language used for business purpose is:  
(a) C (b) C++  
(c) COBOL (d) FORTRAN ( )
8. C language has been developed by:  
(a) ken Thompson (b) Dennis Ritchie  
(c) Peter Norton (d) Martin Richards ( )
9. Ambiguity means:  
(a) Two distinct left parse tree for a sentence  
(b) Two distinct right parse tree for sentence  
(c) More than one parse tree for a sentence  
(d) None of the above ( )
10. Which of the following statement is true after execution of the program:  
Int a [10], I, \*p;  
A[0]=I;  
A p1[ =2;  
P=a;  
(\*p)++;  
(a) a [0] =2 (b) a [1]= 3  
(c) a [1] =2 (d) a [0]=1 ( )
11. if a = -11 and b =3, what is the value of a % b?  
(a) -3 (b) -2  
(c) 2 (d) 3 ( )
12. Choose the storage class type which access the variable fastly:  
(a) Automatic storage class  
(b) Static storage class  
(c) Register storage class  
(d) External storage class ( )
13. Which operator has the highest precedence?  
(a) Unary (b) Binary  
(c) Ternary (d) All of the above ( )
14. int \* arr [5]; this statement declares:  
(a) a pointer to an array (b) an array of pointer  
(c) an array of pointer (d) none of the above ( )



15. A structure member by default is:  
(a) Public  
(b) Private  
(c) Protected  
(d) Any of the above ( )
16. Elements of an array are stored in:  
(a) Continuous memory location  
(b) Noncontinuous memory location  
(c) It depends on the program  
(d) None of the above ( )
17. enum (x,y,z) defines as:  
(a) x=0, y=1, z=2, (b) x=1, y=2, z=3  
(c) x=0, y=0, z=0, (d) None of the above ( )
18. In the statement ; int \*\* a; a is used to store:  
(a) address of an integer variable  
(b) address of an integer pointer variable  
(c) both a and b  
(d) None of the above ( )
19. What will the following code do:  
main ( )  
{ unsigned char ch; }  
for (ch=0; ch<256; ch++)  
printf("%c",ch);  
}  
(a) Print number from 0 to 255 (b) go a never ending loop  
(c) illegal code (d) print all ASCII chara. ( )
20. In the statement; fprintf(fpt,%i);  
(a) A character variable  
(b) Arbitrary assigned value  
(c) Special kind of variable called "file"  
(d) A pointer to file ( )
21. Length of the string "Programming" is:  
(a) 11 (b) 12  
(c) implementation (d) A pointer to file ( )

22. In n has the value 3, output of the statement `printf ("%d",n++, ++n);` is:  
(a) 3,4 (b) 4,5  
(c) 4,4 (d) Implementation dependent ( )
23. The switch feature:  
(a) Can't always be replaced by a nested if then else clause  
(b) can always be replaced by a nested if then else clause  
(c) not enhances logical clarity  
(d) None of the above ( )
24. The member of a class can be:  
(a) Private (b) Public  
(c) Protected (d) All of the above ( )
25. The cin and cout object required the header file:  
(a) `stdio.h` (b) `Iostream.h`  
(c) `iostream.h` (d) None of the above ( )
26. A destructor :  
(a) has a return type (b) can have parameter  
(c) has same name as class (d) All of the above ( )
27. Use of functions:  
(a) Helps to avoid repeating a set instructions many times  
(b) enhance the logical clarity of the program  
(c) helps to avoid repeating programming across program  
(d) All of the above ( )
28. A C++ class can hold:  
(a) only data (b) only function  
(c) both data functions (d) none of the above ( )
29. Among the following which one is the scope resolution operator?  
(a) \* (b) →  
(c) :: (d) ! ( )
30. Templates can be used:  
(a) functions only (b) classes and function  
(c) classes only (d) None of the above ( )
31. Cout is :  
(a) a key word (b) an object

- (c) a library function (d) a variable ( )
32. Operator overloading concepts are used to:  
(a) Extending capability of operators to operate on use defined data  
(b) Data conversion  
(c) Both the above a and b  
(d) None of the above ( )
33. Which of the following operators can not be overloaded?  
(a) Scope resolution (b) Operator overloading  
(c) Binary (d) Logical ( )
34. Dynamic binding implemented through:  
(a) Function overloading (b) Operation  
(c) Rs 500 (d) None of the above ( )
35. Base class binding implemented through:  
(a) Object of base class  
(b) Object of derived class  
(c) Both object of base and derived classes  
(d) None of the above ( )
36. << end 1 in cout is equivalent to:  
(a) '\n'  
(b) '\t'  
(c) '\b'  
(d) None of the above ( )
37. A virtual base class is useful when:  
(a) different functions in base and derived classes have the same name  
(b) there are multiple paths from one derived class to another  
(c) the identification of a function in a base class is ambiguous  
(d) it makes sense to use a base class with no body ( )
38. Inheritance is a way to:  
(a) make a new class  
(b) pass arguments to object of a class  
(c) add feature to existing classes without rewriting them  
(d) improve data hiding and encapsulation ( )
39. The new operator :  
(a) returns address of a variable

- (b) creates a variable called new  
 (c) obtains memory for a new variable  
 (d) tells how much memory is available ( )
40. A data file must be closed using:  
 (a) Library function fprintf (b) Library function fclose  
 (c) Exit function (d) None of the above ( )

**Answer Key**

1. ( b )	2. ( b )	3. ( c )	4. ( a )	5. ( c )	6. ( b )	7. ( c )	8. ( a )	9. ( c )	10. ( a )
11. ( b )	12. ( c )	13. ( a )	14. ( c )	15. ( a )	16. ( a )	17. ( a )	18. ( b )	19. ( d )	20. ( d )
21. ( b )	22. ( c )	23. ( b )	24. ( a )	25. ( b )	26. ( c )	27. ( d )	28. ( c )	29. ( c )	30. ( b )
31. ( b )	32. ( a )	33. ( a )	34. ( c )	35. ( c )	36. ( a )	37. ( b )	38. ( c )	39. ( c )	40. ( b )

**DESCRIPTIVE PART-II****Year- 2010****Time allowed : 2 Hours****Maximum Marks : 30*****Attempt any four questions out of the six. All questions carry 7½ marks each.***

- Q.1 (a) Describe language evaluation criterion in detail.  
(b) What do you mean by semantic analysis? Explain static and dynamic semantics.
- Q.2 (a) Explain recursive descent parsing process with example.  
(b) Write short note on the following:  
(i) Context – Free grammar  
(ii) Attribute grammar  
(iii) Extended BNF.
- Q.3 (a) What are contract structure in language? Explain with syntax.  
(b) Write a program in C to accept 10 elements of an array and find highest element and average of elements of the array.
- Q.4 (a) What is an operator? Describe several different types of operators that are included in C.  
(b) Write a program to calculate factorial of a given integer number n.
- Q.5 (a) What is pointer? How is it declared and initialized? Explain the difference between "Call by value".  
(b) Write short notes on the following  
(i) Data encapsulation  
(ii) Operator overloading
- Q. 6 (a) What is function overloading? Write a program to calculate area of a triangle, circle, square using function overloading.  
(b) Differentiate the following:  
(i) Class and object  
(ii) Structure and Union  
(iii) Contractor and destructor



## PRINCIPLES OF PROGRAMMING LANGUAGES

### PAPER 119

#### OBJECTIVE PART- I

Year - 2009

**Time allowed : One Hour**

**Maximum Marks : 20**

*The question paper contains 40 multiple choice questions with four choices and student will have to pick the correct one (each carrying ½ mark).*

1. A destructor :  
(a) Has a return type (b) Have parameters  
(c) has same name as class (d) None of the above ( )
2. The default parameter is passed with:  
(a) Call by value (b) Call by reference  
(c) Callby value result (d) None of the above ( )
3. Void is used as:  
(a) A data type of a function that returns nothing to its calling environment  
(b) Inside the parentheses of a function that doesn't have any argument  
(c) In an expression  
(d) In a print statement ( )
4. Extension of C++ program file is:  
(a) .exe (b) .c  
(c) .cpp (d) None of the above ( )
5. A C++ class can hold:  
(a) only data (b) only function  
(c) Both data and function (d) None of the above ( )
6. Extension of a header file is:  
(a) 'h' (b) 'c'  
(c) 'cpp' (d) 'exe' ( )
7. Prolog means:  
(a) programming logic (b) Program logic  
(c) Programming logic method (d) program function ( )

8. Language used for scientific manner is:  
(a) C (b) C ++  
(c) FORTRAN (d) None of the above ( )
9. The output of the following is :  
void main ( )  
{  
float a; b =5;  
  
c=10; a =b/c;  
  
printf("%f",a);  
}  
  
(a) 0.000000 (b) 0.500000  
(c) Both a and b (d) None of the above ( )
10. What is an output statement:  
(a) Getch ( )  
(b) Scan ( )  
(c) getchar ( )  
(d) putchar ( ) ( )
11. A new class can be derived from an existing class, that is done by:  
(a) Inheritance  
(b) overloading  
(c) polymorphism  
(d) overriding ( )
12. A pointer is:  
(a) Address of a variable  
(b) Indication of a variable to be accessed next  
(c) A variable for storing address  
(d) None of the above ( )
13. A class member by default is:  
(a) Public (b) Private  
(c) Protected (d) Any of the above ( )
14. A variable which is accessible by all function is called:  
(a) Local (b) Global



- (c) Static (d) None of the above ( )
15. Which is not a primitive type?  
(a) int (b) string  
(c) float (d) char ( )
16. Switch statement is used for  
(a) Multiple selection  
(b) one way selection  
(c) two way selection  
(d) None of the above ( )
17. Structure is used to store:  
(a) Similar type of data (b) Different type of data  
(c) Both a and b (d) None of the above ( )
18. A function that calls itself is called:  
(a) Looping (b) Iteration  
(c) Recursion (d) None of the above ( )
19. The Operator overloading function can be:  
(a) Member function only (b) non member function only  
(c) Both A and B (d) None of the above ( )
20. The number of copies created for a static data member of a class, when 10 class object created would be:  
(a) 0 (b) 1  
(c) 9 (d) 10 ( )
21. "Break" statement is used to exit from;  
(a) If statement (b) For loop  
(c) Switch (d) Both A and B ( )
22. File \*fp:  
(a) Disk data file to be opened (b) Structure file  
(c) Both a and b (d) None of the above ( )
23. Structure elements are stored in:  
(a) Contiguous memory stored in  
(b) Non-contiguous on the program  
(c) It depends on the program  
(d) None of the above ( )

24. If a file is opened in read mode, and it does not exist:  
(a) There will be compile error (b) File will get created  
(c) Link array of pointers (d) none of the above ( )
25. `Int * arr [3];` this declares:  
(a) A pointer to an array  
(b) A pointer to an integer  
(c) An array of pointers  
(d) None of the above ( )
26. Union can store:  
(a) The values of its entire member  
(b) only values of one member can be stored  
(c) Only according to the size of its members  
(d) None of the above ( )
27. `f close ( )` is a function:  
(a) The values of its entire member  
(b) only values of one member can be stored  
(c) Only according to the size of its member  
(d) None of the above ( )
28. `Error ( )` function is used for:  
(a) return 1 if error (b) return 0 if error  
(c) return 0 if no error (d) None of the above ( )
29. Enum is defined as:  
(a) keyword (b) identifier  
(c) Data type (d) None of the above ( )
30. Scope resolution operator is used to:  
(a) Access global version of the variable  
(b) Define member function outside that class  
(c) Both a and B  
(d) None of the above ( )
31. Friend function is a:  
(a) Member function of class  
(b) Not member function but they can access private and public member of the class  
(c) Same friend function can be used in two different classes  
(d) None of the above ( )

32. Which of the following operator does not require any arguments while overloading the operator ?  
(a) ++  
(b) +  
(c) [ ]  
(d) < ( )
33. Formal arguments are:  
(a) Stored main program  
(b) Declared in subprogram header  
(c) Assigned when calls the subprogram  
(d) None of the above ( )
34. Type conversion includes:  
(a) Implicit type conversion (b) Explicit type conversion  
(c) Both a and b (d) None of the above ( )
35. Strcpy (s1, s2); what would be the output;  
(a) Copies string S2 to S2 (b) Copies string S1 to S2  
(c) Copies into a third string (d) None of the above ( )
36. Choose the storage class type which accesses the variable fast:  
(a) Automatic storage (b) Static storage class  
(c) Register storage class (d) External storage class ( )
37. The malloc ( ) function:  
(a) Returns a pointer to the allocated memory  
(b) Returns a pointer to the first byte of the region of memory  
(c) Changes the size of the allocated memory  
(d) Deallocates the memory ( )
38. Templates can be used to produce:  
(a) Function only (b) Classes only  
(c) Both a and B (d) None of the above ( )
39. Literal is a/an:  
(a) String (b) String constant  
(c) Character (d) Apphabat ( )
40. The & & and !! are :  
(a) Arithmetic (b) Equality Operator

(c) Logical operator

(d) Relational operator

( )

**Answer Key**

1. ( c )	2. ( a )	3. ( a )	4. ( c )	5. ( c )	6. ( a )	7. ( a )	8. ( c )	9. ( b )	10. ( d )
11. ( a )	12. ( c )	13. ( b )	14. ( b )	15. ( b )	16. ( a )	17. ( b )	18. ( c )	19. ( a )	20. ( b )
21. ( d )	22. ( c )	23. ( a )	24. ( c )	25. ( c )	26. ( c )	27. ( c )	28. ( a )	29. ( a )	30. ( c )
31. ( b )	32. ( d )	33. ( b )	34. ( c )	35. ( a )	36. ( a )	37. ( a )	38. ( c )	39. ( b )	40. ( c )

**Gurukpo.com**  
No. 1 Educational Web Portal in India

**DESCRIPTIVE PART-II****Year- 2009****Time allowed : 2 Hours****Maximum Marks : 30***Attempt any four questions out of the six. All questions carry 7½ marks each.*

- Q.1 (a) Define syntax and semantics. Distinguish between static and dynamic semantics.  
(b) What are the advantages and disadvantages of implicit declarations?
- Q.2 (a) What are the different of data types available in language? Explain in detail with an example.  
(b) Write a program in C language to check the implicit and explicit type conversion.
- Q.3 (a) Write a program in C language to calculate the sum of the given number ( $34567 = 3+4+5+6+7 = 25$ )  
(b) What do you mean by nested loop? Explain break statement with the suitable example.
- Q.4 (a) What is recursion? Explain recursion with a suitable examples.  
(b) What is an array? How can we declare an array? Explain multidimensional array.
- Q.5 (a) Explain the concept of pointer. What is smart pointer? Discuss virtual function?  
(b) What is structure? Explain the array of structure with an example?
- Q.6 (a) What constructor? Explain constructor overloading and copy constructor.  
(b) What is inheritance? Explain different levels of inheritance.



## PRINCIPLES OF PROGRAMMING LANGUAGES

### PAPER 119

#### OBJECTIVE PART- I

Year - 2008

**Time allowed : One Hour**

**Maximum Marks : 20**

*The question paper contains 40 multiple choice questions with four choices and student will have to pick the correct one (each carrying 1/2 mark).*

1. The language used in making web pages is:  
(a) C++ (b) Java  
(c) Prolog (d) PASCAL ( )
2. Multithreading concept in a programming language was introduced in:  
(a) 5 GL (b) 3 GL  
(c) 1 GL (d) 4 GL ( )
3. To reference an object using a pointer to object use the:  
(a)  $\rightarrow$  Operator  
(b) dop operator  
(c) & (ampersand) operator  
(d)  $\diamond$  Operator ( )
4. You are not allowed to call a ..... member functions from anywhere other than member function of the same class:  
(a) Global (b) Public  
(c) Local (d) Private ( )
5. A class is a (n) .....that is defined by the programme:  
(a) User defined variable type (b) Attributes  
(c) Method (d) Function ( )
6. A suitable place to store class declaration is.....  
(a) auxiliary .cpp file (b) Floppy disks  
(c) Their own header files (d) None ( )
7. The cin and cout function require the header file:  
(a) Stdio.h (b) iostream.h

- (c) iomanip.h (d) None ( )
8. The member of a class can be:  
(a) Private (b) Public  
(c) protected (d) All of the above ( )
9. Which of these keywords are access specifies:  
(a) near and far (b) table and row  
(c) Private and public (d) opened and closed ( )
10. When the code a member function is defined inside a class declaration, it is called:  
(a) Purchases book (b) Sales book  
(c) Purchases returns book (d) Journal proper ( )
11. You can redefine the way.....work when used with objects:  
(a) Standard operators  
(b) Preprocessor directives  
(c) Whitespace characters  
(d) Undefined variables ( )
12. Each and every operator of C++ can be overloaded. This statement is.....  
(a) True  
(b) False  
(c) Conditional  
(d) None ( )
13. The principles of .....allows as to create new classes based on existing classes:  
(a) Inheritance  
(b) Function over loading  
(c) Copy construction  
(d) Polymorphism ( )
14. When you desire a class privately a protected base class member becomes;  
(a) Public (b) Private  
(c) Explicit (d) Inherited ( )
15. Ability to take many forms is.....  
(a) Polymorphism (b) Encapsulation  
(c) Member function (d) Inheritance ( )
16. The run time system performs.....on virtual functions.  
(a) Additional error checking (b) Static binding



- (c) Dynamic binding (d) None ( )
17. A method is declared virtual using keyword:  
(a) Private (b) Public  
(c) Protected (d) Virtual ( )
18. A .....pointer can point a .....function .  
(a) Derived class base class  
(b) Void, derived class  
(c) Void, Null  
(d) Base class, derived function ( )
19. Every C program required a .....function.  
(a) void ( ) (b) main ( ) function  
(c) inline function (d) user defined function ( )
20. Every statement string constants in C .....are used:  
(a) colon (b) Comma  
(c) inline function (d) Full stop ( )
21. For defining string constants in C.....are used:  
(a) Single quotes (b) Double quotes  
(c) Apostrophes (d) None ( )
22. The purpose of a file buffer is:  
(a) use memory (b) to speed up input/output  
(c) to speed up arithmetic operations (d) to protect data ( )
23. To read a single character from a file, we use the member function.....  
(a) input (b) get  
(c) put (d) read ( )
24. The .....member function returns the true value when the end of file has been found:  
(a) input (b) get  
(c) put (d) read ( )
25. What of the following is not a windows accessories:  
(a) Paint (b) Calculator  
(c) Desktop (d) Notepad ( )
26. ALU stands for:

- (a) Atomic line unit (b) Analog logic unit  
(c) Arithmetic and logic unit (d) automatic logic unit ( )
27. Garbage collector is the concept of:  
(a) Java (b) Basic  
(c) Prolog (d) Ada ( )
28. Friend keyword can be used for:  
(a) A function (b) A class  
(c) Both function and class (d) A data member ( )
29. Switch statement is used for:  
(a) Two Set Selection  
(b) Three Set Selection  
(c) Multiple selection  
(d) A data member ( )
30. 1 GB is equal to:  
(a) 1024 bits (b) 1.24 bytes  
(c) 1024 kB (d) 104 MB ( )
31. A data type with user specified value is:  
(a) a marco (b) pointer  
(c) Rs.1,44,000 (d) Enumerated ( )
32. A function that does not return anything can be declared by:  
(a) float (b) int  
(c) void (d) char ( )
33. Which of the following C type is not a primitive data structure?  
(a) Float (b) int  
(c) void (d) char ( )
34. Which of the following C type is not a primitive data structure?  
(a) int (b) float  
(c) char (d) none ( )
35. Malloc is used for:  
(a) Fast speed  
(b) Parallel operation  
(c) Sequential operation  
(d) Memory management ( )

36. C is a:  
 (a) Middle level language (b) High level language  
 (c) Low level language (d) None ( )
37. Inline functions .....call overload:  
 (a) Increase (b) Reduce  
 (c) Depends on situation (d) None ( )
38. The following is a data structure:  
 (a) Queue (b) Stack  
 (c) Tree (d) None ( )
39. In C ++, the stream base class is:  
 (a) iostream (b) Iofstream  
 (c) ios (d) Stdio ( )
40. The identifiers cout and cin are predefined:  
 (a) Classes  
 (b) Objects  
 (c) Functions  
 (d) All of the above ( )

**Answer Key**

1. ( b )	2. ( d )	3. ( a )	4. ( d )	5. ( a )	6. ( a )	7. ( b )	8. ( d )	9. ( c )	10. ( b )
11. ( b )	12. ( b )	13. ( b )	14. ( b )	15. ( a )	16. ( c )	17. ( d )	18. ( a )	19. ( b )	20. ( c )
21. ( b )	22. ( a )	23. ( b )	24. ( c )	25. ( b )	26. ( c )	27. ( a )	28. ( c )	29. ( c )	30. ( d )
31. ( d )	32. ( b )	33. ( c )	34. ( d )	35. ( d )	36. ( a )	37. ( b )	38. ( d )	39. ( a )	40. ( a )

**DESCRIPTIVE PART-II****Year- 2008****Time allowed : 2 Hours****Maximum Marks : 30***Attempt any four questions out of the six. All questions carry 7½ marks each.*

- Q.1 (a) Explain the features of good programming language. Write a note on generations of computer languages.  
(b) What are different types of function available in C language? Write a note on subroutines.
- Q.2 (a) Write a program in c language to count the number of characters in a string.  
(b) What are control structures in C language? What is method overloading?
- Q.3 (a) What are the construction and destructors? Write a note on classes and objects.  
(b) Write a program in C language to calculate the real roots of the quadratic equation:  $ax^2 + bx + c = 0$   
Using the quadratic formula :  
$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$
- Q.4 (a) What is an array? How it is defined in a C program? Write a program in C showing the use of an array?  
(b) What are loop control structures? Write a note on interrupts.
- Q.5 (a) What are the feature of object oriented programming languages? How a class is defined inside a class?  
(b) What are different types of data structures in C++? What is dynamic allocation?
- Q.6 (a) Write an algorithm for inserting an element in a link list. What are class constructors?  
(b) Write a program in C ++ to calculate the income tax based in yearly pay of an employee.

**PRINCIPLES OF PROGRAMMING LANGUAGES****PAPER 119****OBJECTIVE PART- I****Year - 2007****Time allowed : One Hour****Maximum Marks : 20**

*The question paper contains 40 multiple choice questions with four choices and student will have to pick the correct one (each carrying 1/2 mark).*

1. C was primarily developed as a:  
(a) System programming  
(b) General purpose language  
(c) data structure  
(d) none of the above ( )
2. Choose the correct answer:  
(a) use of goto enhances the logical clarity of a code  
(b) use of goto makes the debugging task easier  
(c) use of goto when you want to jump out of a nested loop  
(d) never use goto ( )
3. Length of the string " correct" is:  
(a) 7 (b) 8  
(c) 6 (d) implementation dependent ( )
4. 

```
int x,y = 2, z, a;  
X = (y * =2) + (z =a=y)  
printf("d",x);
```

  
(a) prints 7  
(b) prints 6  
(c) prints 8  
(d) is syntactically warning ( )
5. If n has the value 3, then the output of the statement:  

```
printf("%d%d", n++, ++n);
```

  
(a) is 3 4 (b) is 4 5  
(c) is 4 4 (d) is implementation dependent ( )



6. Printf ("%C",100):  
(a) Prints 100  
(b) Prints the ASCII equivalent of 100  
(c) prints garbage  
(d) none of the above ( )
7. In a for loop, if the condition is missing then:  
(a) It is assumed to be present and taken to be false  
(b) It is assumed to be present and taken to be true  
(c) it results in a syntax error  
(d) execution will be terminated abruptly ( )
8. Choose the correct answer:  
(a) 0 stands for a false condition  
(b) non-zero value for a false condition  
(c) 1 stands for a false condition  
(d) anything's that is not 1, stands for a false condition ( )
9. C is a :  
(a) high level language  
(b) low level language  
(c) high level language with some low level languages  
(d) machine language ( )
10. The switch feature:  
(a) Can always be replaced by a nested if then else clause  
(b) not enhances logical clarity  
(c) can't always be replaced by a nested if then else clause  
(d) none of the above ( )
11. A destructor :  
(a) has a return type  
(b) can have parameter  
(c) has same name as class  
(d) none of the above ( )
12. The members of a class can be:  
(a) Private (b) public  
(c) protected (d) all of the above ( )
13. The cin and cout functions require the header file:

- (a) `stdio.h` (b) `iostream.h`  
(c) `io manip.h` (d) none of the above ( )
14. The default parameter passing mechanism is:  
(a) Call by value  
(b) call by reference  
(c) call by value result  
(d) none of the above ( )
15. Use of functions:  
(a) helps to avoid repeating a set of instructions many times  
(b) enhance the logical clarity programming across program  
(c) helps to avoid repeated programming across program  
(d) all of the above ( )
16. Void can be used:  
(a) As a data type of a function that returns nothing to its calling environment  
(b) Inside the brackets of a function that does not any argument  
(c) in an expression  
(d) in a print statement ( )
17. If `max` is a function that returns the large of the two integers given as arguments, then which of the following statements finds the largest of three given numbers?  
(a) `max (max (a,b), max (a,c) )` (b) `max(msx(a,b), max (b,c))`  
(c) `max (b, max(a,b))` (d) all of the above ( )
18. Any C program :  
(a) Must contain at least one function (b) none of the above  
(c) need input data (d) none of the above ( )
19. `main ( )`  
{  
Int a = 5, b =2;  
Printf ("d,a +++b);  
}  
(a) results in syntax (b) prints 7  
(c) prints 8 (d) none of the above ( )
20. `printf ("ab","cd","ef")` prints:  
(a) ab (b) abcdef  
(c) none of the above (d) abdcef ( )



21. The statement : `int ** a;`  
(a) is illegal  
(b) is legal but meaningless  
(c) is syntactically and semantically correct  
(d) None of the above ( )
22. Consider the declaration :  
`Char X [ ] = "WHATIZIT";`  
`Char * Y = " WHATIZIT";`  
Pick the correct answer:  
(a) The output of puts (X) and puts (Y) will be the sasme  
(b) The output of puts (X) and puts (Y) will be the different  
(c) The output of puts (Y) is implementation dependent  
(d) None of the above comments is true ( )
23. A C++ class can hold:  
(a) Only data  
(b) only function  
(c) both same name as class  
(d) None of the above ( )
24. A constructor :  
(a) has a return type (b) may take parameters  
(c) both data and function (d) both b and a ( )
25. A new class can be derived from an existing class. This concept is called as:  
(a) inheritance (b) Polymorphism  
(c) overlaoding (d) dynamic binding ( )
26. Among the following, which one is the scope resolution operator?  
(a) \* (b)  $\longrightarrow$   
(c) :: (d) ! ( )
27. Garbage collector is the concept of:  
(a) JAVA (b) BASIC  
(c) PROLOG (d) ADA ( )
28. Templates can be used:  
(a) Functions only  
(b) classes only  
(c) functions and classes both  
(d) none of the above ( )

29. Which of the following operators can be over-loaded?  
(a) » (b) ? :  
(c) both a and b (d) no such operator exists ( )
30. Cout is:  
(a) a key word (b) an object  
(c) a library function (d) a variable ( )
31. The pure virtual functions are defined in:  
(a) base class (b) derived class  
(c) main program (d) both a and b ( )
32. Inheritance is a way to:  
(a) make new classes  
(b) pass arguments of objects of class  
(c) add feature to existing classes without rewriting them  
(d) Improve to existing classes without rewriting them ( )
33. By default, a class members are:  
(a) Public (b) Private  
(c) Protected (d) Any of the above ( )
34. Friend keyword can be used for :  
(a) a function  
(b) a class  
(c) both function and class  
(d) a data member ( )
35. – operator used when:  
(a) to access structure's member  
(b) to access structure member when pointer declaration is using  
(c) to store data item from one structure to another structure  
(d) none of the above ( )
36. Which operator has the highest precedence?  
(a) Unary (b) Binary  
(c) Ternary (d) all of the above ( )
37. A pointer is:  
(a) Address of a variable  
(b) An indication of a variable to be accessed next

- (c) A variable for storing address  
(d) None of the above ( )
38. Switch statement is used for:  
(a) Multiple selection (b) Two set selection  
(c) three set selection (d) none of the above ( )
39. Which of the following operator takes only integer operands?  
(a) + (b) X  
(c) / (d) % ( )
40. In an expression involving !! operator, evaluation:  
(a) Will be stopped if one its components evaluates to false  
(b) Will be stopped if one of its components evaluates to true  
(c) takes place from left to right  
(d) both a and b ( )

**Answer Key**

1. ( b )	2. ( c )	3. ( a )	4. ( d )	5. ( c )	6. ( d )	7. ( b )	8. ( a )	9. ( c )	10. ( a )
11. ( c )	12. ( d )	13. ( b )	14. ( a )	15. ( d )	16. ( a )	17. ( d )	18. ( a )	19. ( b )	20. ( b )
21. ( c )	22. ( a )	23. ( c )	24. ( d )	25. ( a )	26. ( c )	27. ( a )	28. ( c )	29. ( a )	30. ( b )
31. ( b )	32. ( c )	33. ( d )	34. ( c )	35. ( b )	36. ( a )	37. ( c )	38. ( a )	39. ( d )	40. ( b )

## DESCRIPTIVE PART-II

Year- 2007
------------

Time allowed : 2 Hours

Maximum Marks : 30

*Attempt any four questions out of the six. All questions carry 7½ marks each.*

- Q.1 (a) What is an operator? Describe several different types of operators that are included in C.  
 (b) Define the following  
 (i) Syntax  
 (ii) Semantics  
 (iii) Grammar.
- Q.2 (a) Write a C program calculate the real roots of the quadratic equation  $ax^2 + bx + c = 0$   
 Using a quadratic formula :  $\frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$   
 (b) Point out the error, if nay, in the following program:
- ```
(i) main ( )
{
    ini i=1; if
    (i > 10)
    break;
}
```
- (ii) main ( )  
 {  
     int i = 1  
     while ( ; ; )  
     {  
         printf("%d", i++);  
         If (i>10)  
         Break;  
     }  
 }

(iii) f(int a, int b)  
 {  
     int a: a  
     = 20;  
     return a;

```
    }  
(iv) will the following function work ?  
    <Yes/No.> F1 (int a, int b)  
    {  
        return (f2(20));  
    }  
    F2 (int a  
{return (a * a);  
}
```

- Q.3 (a) What is an array in C? Explain, with example, how an array is declared and initialized.  
(b) Write a complete c program to print integer array of 10 elements and find initialized.
- Q.4 (a) Differentiate between each of the following :  
(i) Private and public member of a class  
(ii) Normal function and friend function  
(iii) Class and object  
(iv) Constructor and destructor
- Q.5 (a) Write short notes on any three of the following:  
(i) Switch – statement  
(ii) Data – Encapsulation  
(iii) Structure  
(iv) Function  
(b) Write a C program that calculates factorial of a given integer number n.
- Q.6 (a) What is pointer? How is it declared and initialized? Explain the difference between "Call by value and call by reference?"  
(b) What is function? Describe the various category of function with examples.



## PRINCIPLES OF PROGRAMMING LANGUAGES

### PAPER 119

#### OBJECTIVE PART- I

Year - 2006

**Time allowed : One Hour**

**Maximum Marks : 20**

*The question paper contains 40 multiple choice questions with four choices and student will have to pick the correct one (each carrying 1/2 mark).*

1. Extensive of a header file, by default, is:  
(a) 'h' (b) 'c' (c) 'cpp' (d) 'exe' ( )
2. A long integer normally occupies a memory of:  
(a) 1 byte (b) 2 byte (c) 4 byte (d) 8 byte ( )
3. Which one is incorrect variable name?  
(a) Pay (b) basic pay (c) age (d) female ( )
4. Value of (! ! ) is :  
(a) 0 (b) 1 (c) 1 (d) none of the above ( )
5. The results of the expression  $10 \% 3 * 3 + 1$  is :  
(a) 1 (b) 10 (c) 11 (d) none of the above ( )
6. Which of the following is not a primitive type?  
(a) int (b) string (c) float (d) char ( )
7. Execution of a C-program always begins from:  
(a) the first statement  
(b) the main ( ) function  
(c) The declaration statement  
(d) None of the above ( )



8. A variable which is accessible by all functions is called:  
(a) local (b) global  
(c) static (d) none of the basic ( )
9. A variable which is accessible by all functions is called:  
(a) + (b) -  
(c) \* (d) % ( )
10. The statement `print ("This\n\na\nc-\n program");` would give.....  
(a) 3 (b) 6  
(c) 4 (d) 5 ( )
11. The conversion specifier `1% d'` in `printf` – statement is used to print:  
(a) unsigned decimal integer (b) signed decimal integer,  
(c) floating point value (d) character ( )
12. The function declaration `double ABC (int, float);` denotes that the function ABC:  
(a) Returns int value (b) returns float value  
(c) Returns double value (d) declaration is incorrect ( )
13. A function calling itself is called as:  
(a) invalid call  
(b) looping  
(c) iteration  
(d) recursion ( )
14. Consider the declaration `int a [5] = {0,1}`; the value of the element `a [2]` :  
(a) 0 (b) 2  
(c) 1 (d) can not predict ( )
15. How many times will take `printf` statement in the following code will be executed ?  
`for (i=1< 1; i=i +1)`  
`print ("hellow");`  
(a) 0 (b) 10  
(c) 6 (d) 11 ( )
16. The function prototype for a function `smallest` that takes 3 integers `x,y,z` and returns an integer would be:  
(a) `void smallest (int);` (b) `void smallest (int x, int y, int z)`  
(c) `int smallest (x,y,z)` (d) `int smallest (int, int, int);` ( )

17. Which one can not be passed as a parameter in a function?  
(a) array (b) structure  
(c) function (d) pointer ( )
18. What is the value of sum when the following code is executed?  
`int a [5] = {0,1,2,3,4};  
int sum = 1 =  
i=0 while (i< 4)  
{  
sum + = a [i];  
i ++ ;  
}`  
(a) 10  
(b) 7  
(c) 6  
(d) 5 ( )
19. Consider the declaration `int a [ ] = {0,1,2,3,4}` what will be the value of `a (0)` `a p5[ ]` ?  
(a) 0  
(b) 4  
(c) error message  
(d) unpredictable ( )
20. To specify the number of columns to be used to print a number, one should use in `printf`-statement?  
(a) conversion specified  
(b) field width specifier  
(c) flag  
(d) Precision ( )
21. A C++ Class can hold?  
(a) only data  
(b) only function  
(c) both data and function  
(d) none of the above ( )
22. A class in C++ program can be treated as :  
(a) function  
(b) user defined data type  
(c) object  
(d) none of the above ( )

23. The member of a class can be:  
(a) private  
(b) user defined data type  
(c) object  
(d) None of the above ( )
24. For a class to be meaningful, at least:  
(a) one member should be private  
(b) one member should be public  
(c) one member should be protected  
(d) all of above ( )
25. For a function to access the private data of a class, it should be made:  
(a) member function  
(b) a friend function  
(c) both a and b  
(d) none of the above ( )
26. Protected members have their roles in:  
(a) Inheritance  
(b) polymorphism  
(c) overloading  
(d) none of the above ( )
27. A new class can be derived from an existing class. This concept is called us:  
(a) inheritance  
(b) polymorphism  
(c) overloading  
(d) dynamic binding ( )
28. The cin and cout functions require the header file:  
(a) stdio.h  
(b) iostream.h  
(c) iomanip.h  
(d) none of the above ( )
29. endl in cout is equivalent to :  
(a) '\t' (b) '\b'  
(c) '\n' (d) none of the above ( )
30. Name of a constructor is:  
(a) user defined

- (b) same as class name  
(c) same as program file name  
(d) none of the above ( )
31. Constructor is involved :  
(a) when class object is created  
(b) when class object is initialized  
(c) through an explicit call  
(d) none of the above ( )
32. Number of constructor in a class can be:  
(a) minimum one  
(b) zero  
(c) two  
(d) as many as required ( )
33. For operator overloading the operands should be of type:  
(a) Minimum one (b) zero  
(c) user defined (d) any type ( )
34. The word which makes the name of an operator overloading function is:  
(a) the operator symbol  
(b) the key word operator  
(c) the keyword operator followed by the operator symbol  
(d) user defined ( )
35. The operator overloading function can be :  
(a) member function only  
(b) non member function only  
(c) both a or b  
(d) none of the above ( )
36. The number of copies created for a static data member of a class, when 10 class objects are created would be:  
(a) 0 (b) 1  
(c) 2 (d) 4 ( )
37. The number of copies created for a static data member of class, when 10 class objects are created, would be:  
(a) 0 (b) 1  
(c) 9+ (d) 10 ( )

38. A destructor :
- (a) has a return type
  - (b) may take parameters
  - (c) has same name of class
  - (d) both b and a
- ()
39. A constructor:
- (a) has a return type
  - (b) may take parameter
  - (c) has same name as class
  - (d) both a and b
- ()
40. The member of a class by default are:
- (a) Private
  - (b) Protected
  - (c) Public
  - (d) no default exists
- ()

**Answer Key**

|           |           |           |           |           |           |           |           |           |           |
|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|
| 1. ( a )  | 2. ( c )  | 3. ( b )  | 4. ( a )  | 5. ( d )  | 6. ( b )  | 7. ( b )  | 8. ( b )  | 9. ( d )  | 10. ( d ) |
| 11. ( b ) | 12. ( c ) | 13. ( d ) | 14. ( a ) | 15. ( b ) | 16. ( c ) | 17. ( c ) | 18. ( b ) | 19. ( d ) | 20. ( c ) |
| 21. ( c ) | 22. ( b ) | 23. ( d ) | 24. ( b ) | 25. ( c ) | 26. ( a ) | 27. ( a ) | 28. ( b ) | 29. ( c ) | 30. ( b ) |
| 31. ( d ) | 32. ( d ) | 33. ( c ) | 34. ( c ) | 35. ( c ) | 36. ( b ) | 37. ( b ) | 38. ( c ) | 39. ( d ) | 40. ( a ) |



**DESCRIPTIVE PART - II****Year – 2006****Time allowed : 2 Hours****Maximum Marks : 30***Attempt any four questions out of the six. All questions carry 7½ marks each.*

- Q.1 Write C- statement (s) to accomplish each of the following tasks:
- (i) Compute the product of three integers contained in x,y and z and assign the result to variable
  - (ii) Write four different C-statement that each add 1 to integer variable x.
  - (iii) Calculate the remainder after q is divided by d and assign the result to q.
  - (iv) Print the floating point value 3.14159 with three digits to the right of decimal point.
  - (v) Sum of odd integers between 1 to 99 using a for statement.
  - (vi) Initialize each of the 5 elements of single subscribed integer array A to 8.
  - (vii) Convert the character stored in variable C to an uppercase letter and assign it to C.
- Q.2 (a) What is an array? Explain with examples, how an array is declared and initialized.  
(b) Identify and correct the error in each of the following:
- (i) `Print ("The value is & d/n". number);`
  - (ii) `if (C<7);  
printf("C is less than 7/n");`
  - (iii) `while (C≤5) {  
P*=C  
C++;`
  - (iv) `int sum (int x, int y)  
{  
  
int z;  
z = x + y ;  
}`
  - (v) `struct person  
{  
  
char name [20];  
int age;  
}`
- Q.3 (a) Write a C program that calculate and print sum and average of 5 integer number.



- (b) Write a C- Program that calculates factorial of a given integer number.
- Q.4 Differentiate between each of the following :
- (i) Structure and union
  - (ii) Class and object s
  - (iii) Constructor and destructor
  - (iv) Private and public member of class
  - (v) Normal function and friend function.
- Q.5 (a) Define a class to represent a complex number. The class has features of setting and printing the values. Write a program to test the class.
- (b) Define inheritance and explain its types.
- Q.6 Write short notes on any three of the following;
- (a) Evaluation of programming language
  - (b) Logical operator
  - (c) Switch statement
  - (d) Defining member function of a class
  - (e) Data encapsulation

\*\*\*\*\*

# Bibliography

## Books Referred-

- Let us C by Yashwant Kanetkar,
- Complete Reference by Herberld Sheild,
- C++ programming by E.Balaguruswami.

## Websites Referred-

- Wikipedia,
- [www.cprogramming.com](http://www.cprogramming.com)
- [www.eskimo.com](http://www.eskimo.com)
- [www.cprogrammingexpert.com](http://www.cprogrammingexpert.com),
- [www.compgeom.com](http://www.compgeom.com)
- [www.physicsforums.com](http://www.physicsforums.com)

**Gurukpo.com**  
No.1 Educational Web Portal in India

## ● Notes

**Gurukpo**  
No. 1 Educational Web Portal in India .com

## Notes

GU  
No. 1 Education

*This question paper contains 3 printed pages/*

Roll No. \_\_\_\_\_

Sl.No.

**134**

**B.C.A. (Part - I)**

**B.C.A. (Part - I) EXAMINATION, 2017**  
**(Faculty of Science)**  
**(Three - Year Scheme of 10 +2 + 3 Pattern)**  
**Paper - 134**  
**PRINCIPLES OF PROGRAMMING**  
**LANGUAGE (THROUGH 'C')**

*Time : Three Hours/*

*[Maximum Marks : 100]*

Answer of all the questions (short answer as well as descriptive) are to be given in the main answer -book only. Answers of short answer type questions must be given in sequential order. Similarly all the parts of one question of descriptive part should be answered at one place in the answer-book. One complete question should not be answered at different places in the answer-book. Write your roll numbers on question paper before start writing answers of questions.

- PART - I:** *(Very Short Answer) consists of 10 questions of 2 marks each. Maximum limit for each question is up to 40 words.*
- PART - II:** *(Short answer) consists of 5 questions of 4 marks each. Maximum limit for each question is up to 80 words.*
- PART - III:** *(Long answer) consists of 5 questions of 12 marks each with internal choice.*

**PART - I**

Attempt all Questions

Each questions carries 2 marks

**[10 × 2 = 20]**

1.
  - a) What is algorithm?
  - b) Give flow chart symbols for I/O, processing terminal and flow lines.
  - c) How do we create constants in 'C'? Give syntax.
  - d) What are local variables?
  - e) Discuss purpose and syntax of goto statement.
  - f) How do we read and write strings in 'C' Explain.
  - g) What are formal parameters?

- h) Define pointers
- i) How do we create structures in 'C'? Explain.
- j) Differentiate between `fprintf ( )` and `printf ( )`.

### PART - II

Attempt all questions

Each questions carries 4 marks

[5 × 4 = 20]

2. ~~a)~~ Draw a flow chart to find out sum and average of any 3 nos.  
~~b)~~ Discuss any 2 (two) data types of 'C' with suitable examples.  
~~c)~~ Differentiate between break and continue statements with the help of appropriate example(s).  
~~d)~~ Differentiate between call by value and call by reference.  
~~e)~~ Discuss the purpose of following functions:  
i) `putch ( )`  
ii) `puts ( )`  
iii) `putchar ( )`  
iv) `scanf ( )`

### PART - III

3. Discuss machine level, Assembly and high level languages in detail. [12]

OR

Write pseudocodes to find out:

- a) factorial of a given no.
- b) sum of 1<sup>st</sup> 10 natural no's.

[6 + 6 = 12]

4. Discuss the various operators of 'C'. [12]

OR

Write a program in 'C' to find out grade of a student based on the following criterias:

- a) Percentage is  $\leq 40$ ; grade is 'D'.
- b) Percentage is  $\geq 40$  but  $< 50$ ; grade is 'C'.
- c) Percentage is  $\geq 50$  but  $< 60$ ; grade is 'B'.
- d) Percentage is  $\geq 60$  but  $< 75$ ; grade is 'A'.
- e) Percentage is  $\geq 75$  grade is 'A+'.



5. Explain the following functions:

- a) `streat ( )`
- b) `strempr ( )`
- c) `strempr ( )`
- d) `strlen ( )`
- e) `strstr ( )`
- f) `strchr ( )`

[6 × 2 = 12]

OR

Discuss single Dimensional and double Dimensional arrays of 'C' in brief. [12]

6. What is recursion? Why do we use recursion? Explain Also write a code to print fibonacci series with recursive function. <https://www.uoronline.com> [12]

OR

Write a 'C' program that uses of function to search a no with in an array. [12]

7. Explain the following:

- a) File modes.
- b) Steps of file handling in 'C'.
- c) Stream I/O model.

[3 × 4 = 12]

OR

Create a structure containing five members : rollno, name - of- student, marks1, marks2 & marks3. Write a program to access those members using structure variable or pointer. [12]

<https://www.uoronline.com>

Whatsapp @ 9300930012

Send your old paper & get 10/-

अपने पुराने पेपर्स भेजे और 10 रुपये पायें,

Paytm or Google Pay से

R-680

**B.C.A. (PART-I) EXAMINATION - 2018**  
(Faculty of Science)  
(Three year Scheme of 10-2+3 Pattern)  
Paper - 134

**PRINCIPLES OF PROGRAMMING  
LANGUAGE (THROUGH 'C')**

**Time Allowed : Three Hours**

**Maximum Marks - 100**

**PART I:** (Very short answer) consists of 10 questions of 2 marks each. Maximum limit for each question is up to 40 words.

**PART II:** (Short answer) consists of 5 questions of 4 marks each. Maximum limit for each question is up to 80 words.

**PART III:** (Long answer) consists of 5 questions of 12 marks each with internal choice.

**PART - I**

1. Attempt all questions. Each question carries 2 marks. [10 x 2 = 20]
- (a) What is Pseudo Code?
  - (b) Explain Programming Domains.
  - (c) What is operator precedence?
  - (d) What are the different methods to declare a constant in 'c'? Give example.
  - (e) What is the difference between a do-while loop and a while loop?
  - (f) What is the difference between a structure and a Union?
  - (g) What is a NULL pointer? Give example.
  - (h) What is enumerated data type? Give an example.
  - (i) What is the difference between actual and formal parameters?
  - (j) How do we calculate the size of a union in C?

**PART - II**

2. Attempt all questions. Each question carries 4 marks. [5 x 4 = 20]
- (a) Draw a flowchart to calculate factorial of a given integer number.
  - (b) Describe the skeleton of a 'C' program.
  - (c) Write a program to read N values in an array and then find highest value.
  - (d) Explain scope, visibility and lifetime of a variable in context to functions.

(c) Discuss the purpose of the following library functions :

- (i) fseek()
- (ii) rewind()
- (iii) feof()
- (iv) ftell

**PART - III**

3. (a) Write pseudo code to find the sum of first 100 even numbers. [12]

OR

(b) Write an algorithm to check whether a number entered by user is prime or not. [12]

4. (a) Explain different data types available in 'C'. [12]

OR

(b) Write a program to input basic salary of an employee and calculate its gross salary according to the following :

(i) Basic Salary  $\leq$  10000 : HRA=20%, DA=80%

(ii) Basic Salary  $\leq$  20000 : HRA=25%, DA=90%

(iii) Basic Salary  $\geq$  20000 : HRA=30%, DA=95% [12]

5. (a) Write a 'C' program to find the length of a string without using built-in functions. [12]

OR

(b) Discuss the following :

(i) Declaration of one dimensional and two dimensional arrays.

(ii) Initialization of one dimensional and two dimensional arrays.

(iii) Accessing of elements from one dimensional and two dimensional arrays.

(iv) Why array name is called a constant pointer? [3 x 4 = 12]

6. (a) What do you mean by Recursion? Write a recursive program in 'C' to print all the elements of an array. [2 + 10 = 12]

OR

(b) What are Pointers? How a function can be called by using a pointer to it? Explain with an example. [2 + 10 = 12]

7. (a) Write a program in C to copy content of a file to another file. [12]

OR

(b) Explain the following :

(i) Declaring a structure

(ii) Accessing members of a structure using pointer

(iii) Self-referential structure

(iv) Difference between a structure and a union [3 x 4 = 12]

This question paper contains 2 printed pages.

Roll No. ....

B.C.A. (Pt. -I)

Pri. of Pro. Lan. (Through C)

134

B.C.A (Part-I) EXAMINATION, 2019

(Faculty of Science)

(Three Year Scheme of 10+2+3 Pattern)

**PRINCIPLES OF PROGRAMMING  
LANGUAGE (THROUGH 'C') - 134**

Time Allowed : Three Hours

Maximum Marks : 100

Answer all the questions (short answer as well as descriptive) are to be given in the main answer-book only. Answers of short answer type questions must be given in sequential order. Similarly, all the parts of one question of descriptive part should be answered at one place in the answer-book. One complete question should not be answered at different places in the answer-book.

Write your roll numbers on question paper before start writing answers of questions.

PART - I : (Very short answer) consists of 10 questions of 2 marks each. Maximum limit for each question is up to 40 words.

PART - II : (Short answer) consists of 5 questions of 4 marks each. Maximum limit for each question is up to 80 words.

PART - III : (Long answer) consists of 5 questions of 12 marks each with internal choice.

**PART - I**

1. Attempt all questions. Each question carries 2 marks.

10x2=20

- (i) What is an algorithm ?
- (ii) Draw and list any 5 components used in a flow chart.
- (iii) Give the skeleton/basic outline of a C program.
- (iv) List logical and relational operators.
- (v) Give syntax of a while loop. Describe its features.
- (vi) Define an array. Declare an array to hold 5 real number values.
- (vii) What is a function prototype ? What are its elements ?
- (viii) What is a pointer ? Declare a pointer and an array and store the address of the array in the pointer.
- (ix) Describe using a diagram how the memory is allocated for each member of a structure.
- (x) How is a file opened for reading in 'read-only' mode ?

**PART - II**

2. Attempt all questions. Each question carries 4 marks.

5x4=20

- (i) Write an algorithm to compute factorial of a number.
- (ii) Write a C program to check if the year entered is a Leap year or not. Leap year is defined as every 4<sup>th</sup> year, if it is a non-century year, and every 400<sup>th</sup> year, otherwise.
- (iii) Differentiate between for, while and do-while loops.
- (iv) What is recursion ? Write a recursive function to calculate HCF/GCD of two numbers.
- (v) Differentiate between structure and unions.

**PART - III**

3. What is :

4x3=12

- (i) A Compiler
- (ii) An Interpreter
- (iii) An Assembler
- (iv) A Linker

**OR**

Write pseudo-code and draw flow-chart to compute sum of digits of a positive integer.

12

4. Discuss about different operators available in C language. What is meant by operator precedence and associativity ?

8+4=12

**OR**

Write a C program using switch-case to print marks range given a student's grade as per the following table :

12

| Grade Letter | Min. Marks | Max. Marks |
|--------------|------------|------------|
| D            | 0          | 40         |
| C            | 40         | 60         |
| B            | 60         | 80         |
| A            | 80         | 100        |

5. Write a program to find all prime numbers between 1 and N.

12

**OR**

Write a C program to input and sort an array of integers using linear sort.

12

6. What is function definition ? Write a custom C function and use it in a program to find all occurrences of a character in a string.

5+7

**OR**

Write a program to transpose a matrix using custom function. Access the matrix using pointer notation.

12

7. (i) What are Structures and Unions ? How are they declared ?

4

(ii) Write a program to declare and use a structure to hold student data - roll no, name, program, and semester. Input details of 3 students and print them sequentially.

8

**OR**

Write a program to input multiple lines one-by-one and store them in a file. The input will end when the user types "STOP". Then read the file and print the output line by line again.

12

- o O o -