* Defⁿ OOP

Date : / /
Page No.:

1. Principle of object oriented programming 14m

Introduction -

Object oriented programming is new way of organizing code and data that provides increased reliability and excellent control over the software development process is called as OOP.

OOP is most recent and popular concept among different programming languages. OOP incapsulate data and functions into a package are called as object.

Where data and functions are tide together.

OOP takes the advantage of class relationship where the entire set of data and code of an object can be made userdefined datatypes with the help of class.

A class is the collection of similar type of object.

The most important facility provided by OOP are: (OOP's concept)

① Data abstraction, data incapluation, polymorphism, inheritance.

OOP also model communication between objects, objects are communicate with each other through the messages.

OOP is used to represent real life entities in system design.

It ensure reuseability and extensibility of already developed and designed modules.

Date: / /
Page No.:

- POP - Procedure oriented programming (C lang)

POP approach, the ~~program~~ problem is view as segence of things to written such as reading (reft to right and top to bottom), calculating, printing.

A number of functions are written to acomplish ~~this~~ this task.

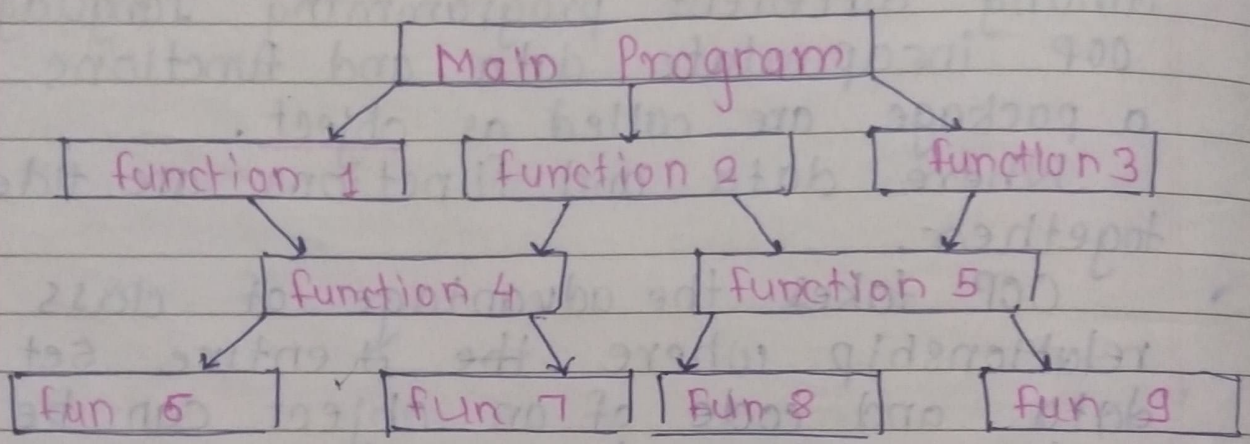


fig name - Structure of POP

In diagram, the techqunic of hierarchical decomposition has been used to specify the task to be completed in order to solve a problem.

In multifunction program many important data items are placed at global so that they may be accessed by all the functions.

Each function may have its own local data.

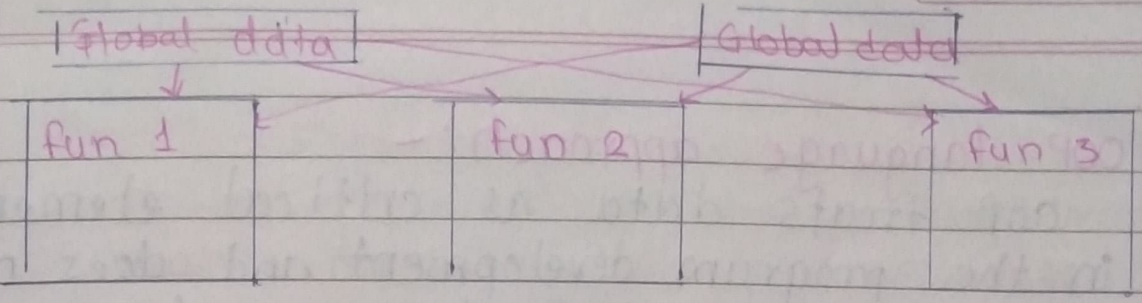


fig - relation betⁿ data and function in POP

Global data are more open to an accidental change by a function.

In a large program, it is very difficult to identify what data is used by which function.

Therefore, if we need to revise an external data structure have to revise all the functions that access the data.

The major drawback of POP is that does not module real world problem very well.

2m * Characteristics of POP :- (Importance)

① Emphases are given on doing things.
② large programs are divided into smaller programs (functions).
③ Most of the function share the Global data.
④ Data moves openly around the system from function to function.
⑤ Follows top down approach in program design.

Date : / /
Page No.:

* OOP language approach :-

OOP treats data as critical element in the program development and does not allow it to flow freely around the system.

It ties data more closely functions that operate on it and protects it from accidental modifications from outside the functions.

OOP allows decomposing the problem into number of entities called as object.

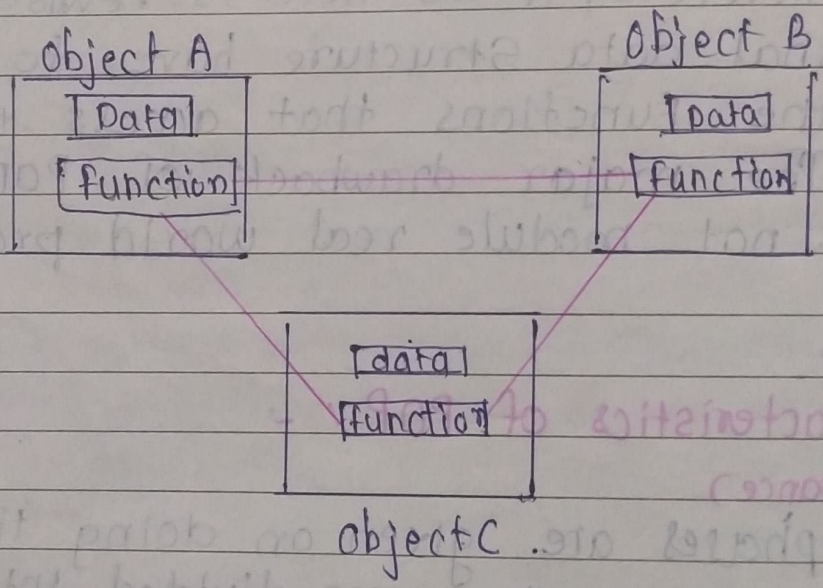


fig - organization of data and function in OOP.

The data of an object can be accessed only by the funtions associated with that object.

Date : / /
Page No.:

- characteristics of OOP: features / uses / properties
① Emphases are on data rather than procedure
② Programs are divided into objects. Data is hidden and cannot be accessed by outside the functions.
③ Objects may communicate through functions.
④ New data and functions easily added whenever necessary.
⑤ Follows bottom up approach in program design.

Imp - Applications of OOP:
① Real time system -
for eg - traffic signal, Banking, Aeroplane, staellite launching etc.
② Simulation and modeling.
③ Object oriented Database.
④ Hypertext, Hypermedia.
⑤ AI (Artifical Integillance and neural networks).
⑥ Parallel programming.
⑦ CIM, CAM, CAD system.

Date: / /
Page No.:

4m: Difference betn POP and OOP

POP	OOP
① large programs are divided into smaller programs called as function.	① large programs are divided into objects.
② It follows the top-down approach in program design.	② It follows the bottom-up approach in program design.
③ Emphases is given to doing things	③ Emphases is given to data rather than procedure.
④ The function share the Global data.	④ The data cannot be accessed by outside the function
⑤ Data is not hidden.	⑤ Data is hidden.
⑥ It does not module real world ~~##~~ programmery well	⑥ It is used in real life system

Date: / /
Page No.:

- Basic concept of OOP

IMP ① Object :-

Object is basic run time entity in object oriented system. is called as object. They may represent a place, barek account, a table of data or any item that the program must handle.

For e-g - If coustmer and account are two objects in a program then the ~~account~~ coustmer object can send the message to the account object requesting for the bank balance.

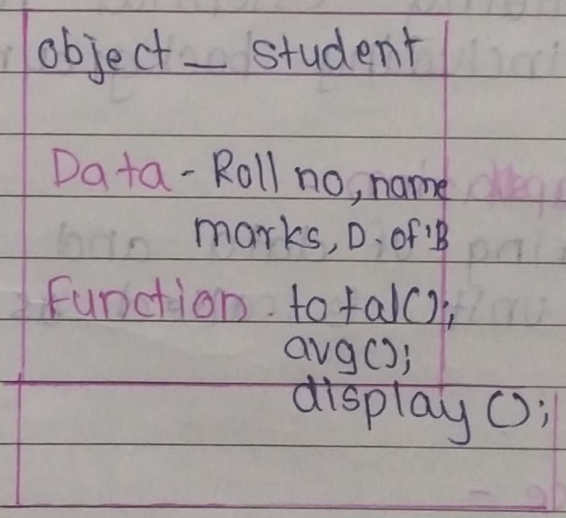


fig- Representation of an object

objects can interacts without knowing details of each other data or code.

IMP ② Classes :-

Class is a collection of similar objects with similar properties, common behaviour and common relationship to each object

is called as classes.

Once class has been defined we can create any number of objects belonging that classes.

Eg -

Mango, Apple, orange are objects of class fruits.

If fruit has been defined as a class then the statement is,

class fruit mango, apple, orange

③ Data abstraction :-

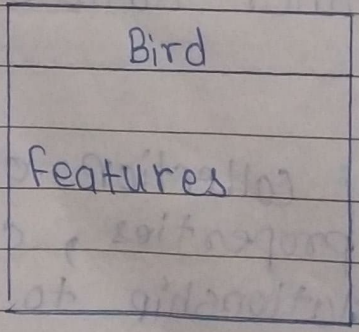
Data abstraction refers to the act of representaing essential features without including background details.

④ Data incapsulation -

Wrapping of data and function into a single unit is called as data incapsulation.

⑤ Inheritance -

Object of one class aqulre the properties of objects of another class is called as inheritance.



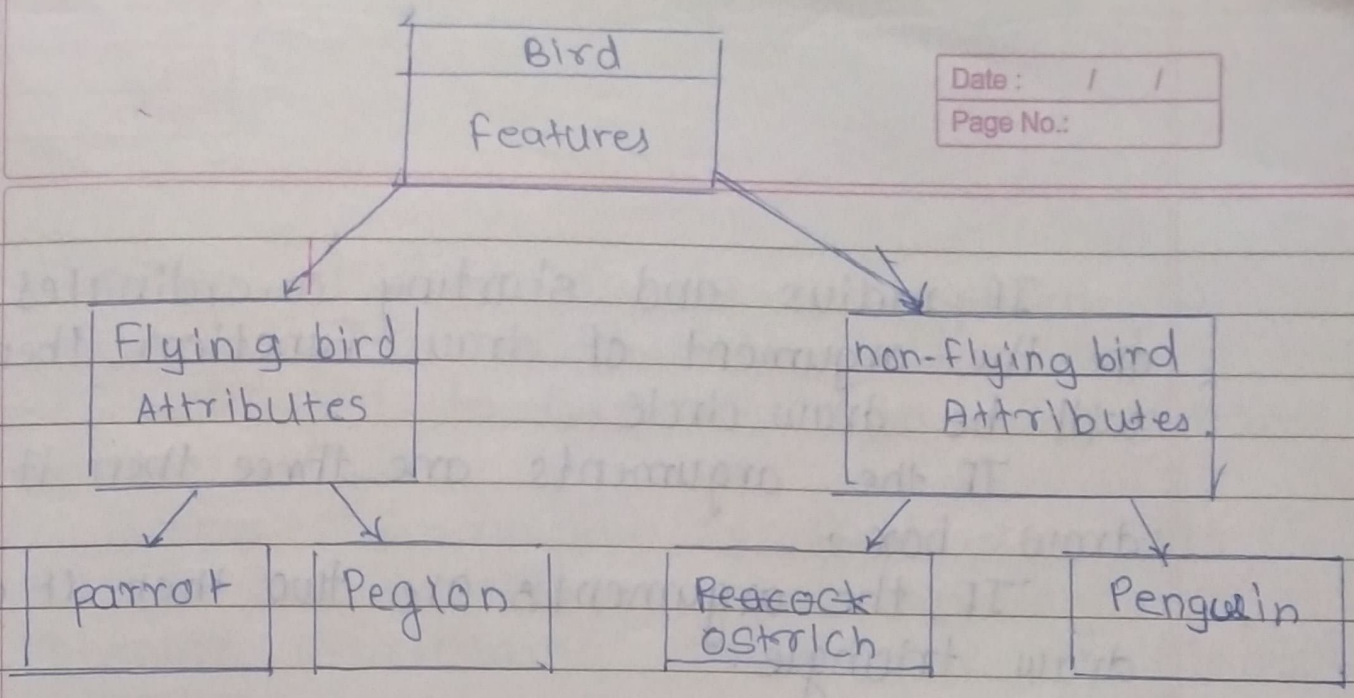


fig-Name - properties of inheritance.

Inheritance provides important features of OOP that is reusability this means we can add additional features to an existing class without modification.

The old class is refered as base class and new dass is called as derived class.

⑥ Polymorphism -

Polymorphism means ability to take more than one form is called as Polymorphism.

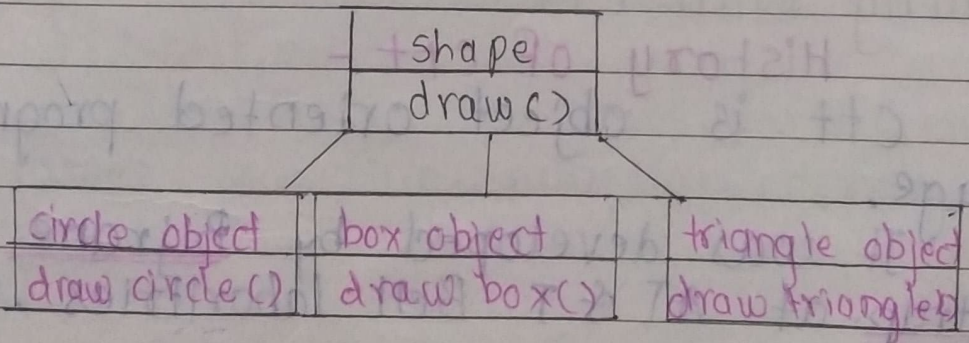


fig - Polymorphism.

If radius and starting co-ordinates are the argument of draw function then it will be draw circle.

If the arguments are three then it will draw box.

If the arguments are two then it will draw triangle.

⑦ Dynamic binding :- / late binding.

Compiler should match function call with correct function defination at the run time is called as dynamic binding or late binding.

⑧ Message passing -

Object are communicate with one another by sending and receiving messages or information.

- Beginning With C++ :-

History of C++ -

C++ is object oriented programming language.

It was developed by Bijrane Stroustrup at AT & T Bell laboratories, ~~USA~~ in USA in 1980.

C++ is an extention C language with addition of C language ~~and~~ of classes and objects.

Class is addition to c language so it is called as c with classes and it was changed to c++ with its incremented version of C.

Important facility is that c++ adds classes and objects, function overloading, operator overloading, inheritance, polymorphism, etc.

- Tokens:-

Token is a smallest individual unit in the program.

for eg - keywords, Identifiers, constants, string operators etc.

①keywords -

keywords are the reserved words and cannot be used as variables.

for eg - Integer, character, Auto, double, break.

②Identifiers -

Identifiers are refers the name of variable classes, structure, function, array name etc.

③ Constant -

Constant is an entity whose value are not changed.

for eg -

```
# define PI 3.14
```

④ String -
String is the collection of characters String are enclosed into double Quotes (" ").
for eg -
" shreya"

- Rules :-
① C reconigzes 32 characters but in c++ has no limit.
② Upper case and lower case letters are different.
③ The name cannot be start from digit.
④ Declared keywords cannot be used as Identifiers or variables.
⑤ Only Alphabtes characters, digit and underscore are permitted.
⑥

- C++ data types -

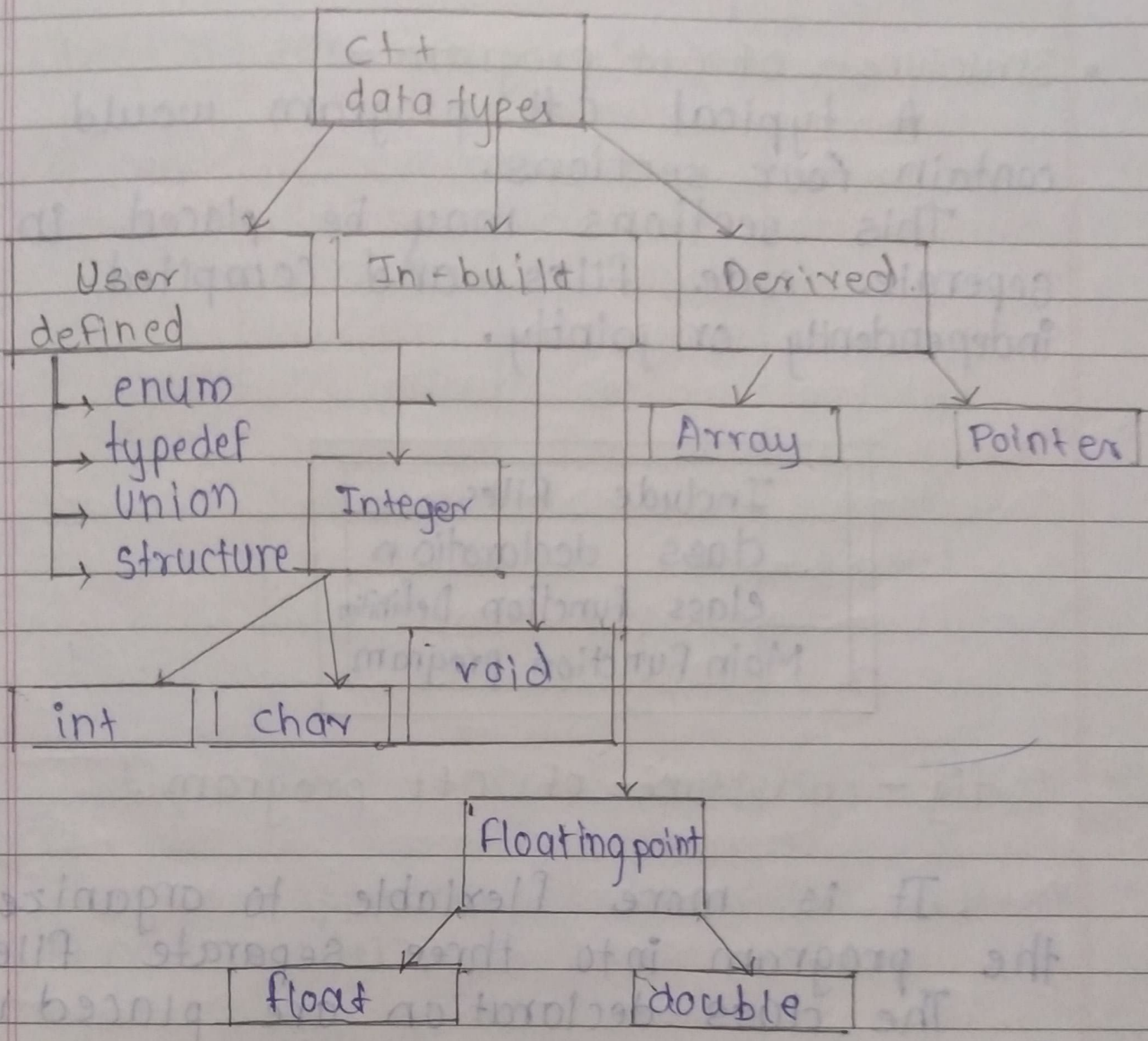


fig name - datatype in c++

~~Array~~

- Structure of c++ program -

A typical c++ program would contain four sections.

This sections may be placed in seperate code files and compiled independently or jointly.

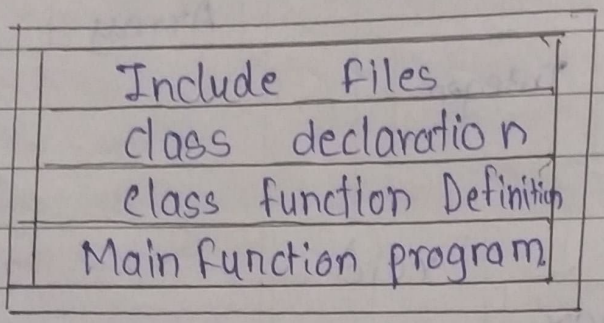


fig - Structure of c++ program.

It is more flexiable to organize the program into three separate files.

The class declaration are placed in header files and the defination of member function goes to the next file. This approach enables the program to separate the abstract (Hide) specification of the interface from the implementation details.

Date : / /
Page No.:

- Client server structure in C++ -

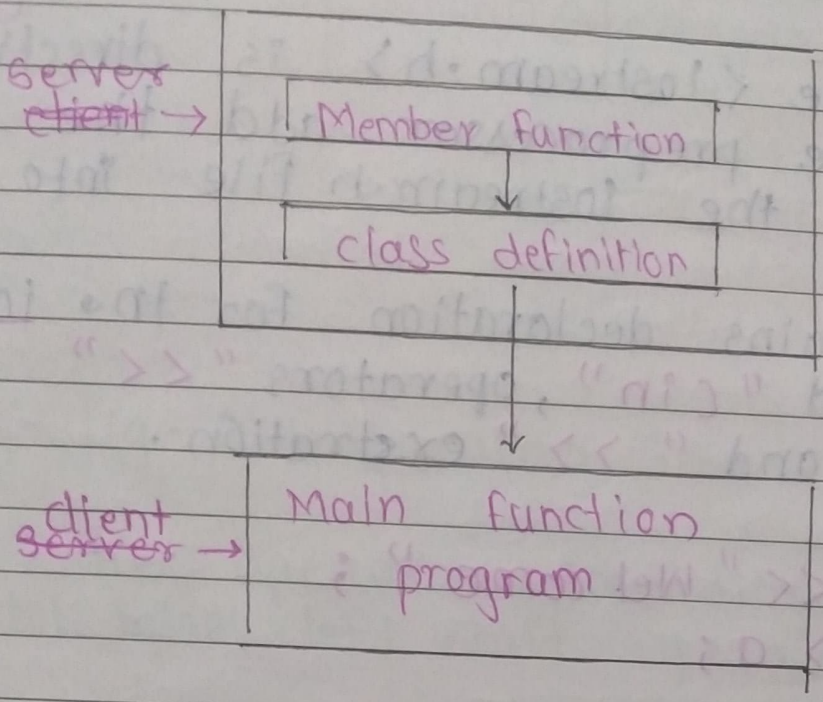


Fig - Client Server structure in C++

This approach is based on client server model.

Client -

Client sends the request to server for getting different services.

Server -

Server provides services as per requirement of client.

In above figure main function program act as a client and Member function and class definition act as a server means that member function and class definition provieds the serives to main function.

Date : / /
Page No.:

input output
- The iostream.h file :-

(Library)
include <iostream.h> is directive causes the preprocessor add the content of the iostream.h file into the program.

It contains declaration for the identifiers "cout" and "cin", operators "<<" insertion and ">>" extration.

for eg -

```
cout << "Welcome";
cin >> a;
```

Imp
- Output operator -

The output statement in the program is cout ~~(insertion~~

```
cout << "Welcome";
```

The identifier cout is pre defined object that represent the standard output strem in C++.

The operation insertion "<<" is called as put to operator. It inserts or sends the content of variable to its right to the object on its left

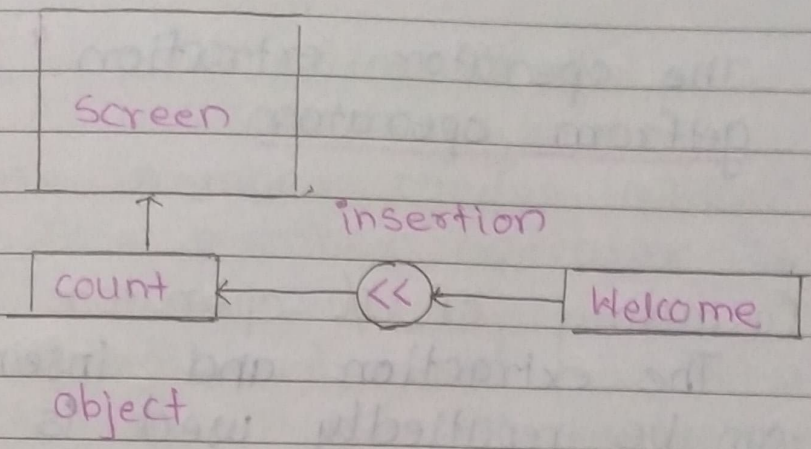


fig - Output using insertion operator.

The operator insertion "<<" is the bit wise left shift operator.

4m. Input operator -

The statement cin extr >> a; causes the program to wait for the user to type a number.

eg -

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

The identifier cin >> predefined object in c++ corresponds to standard input stream.

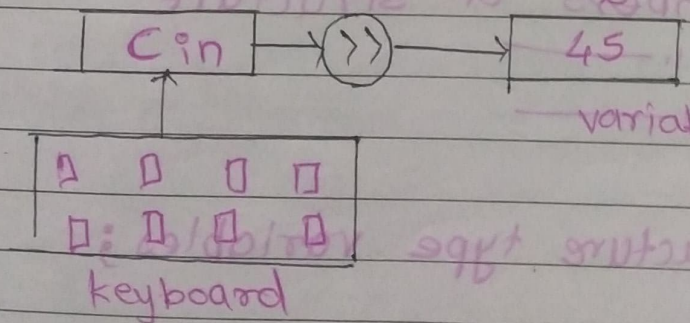


fig - input using extraction operator.

Def^n Cascading

Date: / /
Page No.:

The operator extraction ">>" is called getfrom operator.

2m • Cascading of I/O operator
The extraction and insertion operators can be repatiedly used is a single cin & cout statement is called as cascading

for eg -

```
cin >> x >> y;
cout << "Addition = " << sum;
```

• Concept of structure -

Def^n -

Structure is collection of different datatypes grouped ~~data~~ together under a single name is called as structure.

• Declaring and initializing the structure

syntax -

```
struct name of structure
{
  members of structure
} structure type variables;
```

Date: / /
Page No.:

- Accessing structure member.

Structure member can be accessed using structure member operator is called as dot operator (.).

This operator is used in between structure variable and structure members.

eg -

```
# include <iostream.h>
# include <conio.h>
 struct  student
 {
   int roll_no;
    char  name[20];
     float  percentage;
 } s1;
 void main()
{
 cout<<"enter student roll_no, name
 and percentage";
 cin>>s1.roll_no>>s1.name>>s1.percentage;

 cout<<"roll_no, name, percentage"<<s1.
Roll_no
name<<s1.name<<s1.percentage;
   getch();

 }
```

eg ②

```
# include <iostream.h>
# include <conio.h>
struct book
{
   int Id_no;
   char book_name[20];
   float price;
} b1;
void main()
{
  cout<<"enter book Id_no, book_name,
   price";

  cin>>b1.Id_no>>b1.book_name>>
  b1.price;

  cout<<"book Id_no, book_name, price
  are"<< b1.Id_no <<b1.book_name<<
  b1.price;

  getch();
}
```

Date : / /
Page No.:

Imp.

- Difference betⁿ structure and class.

structure	class
① structure is the collection of different data types.	① class is the collection of objects.
② In structure "struct" keyword is used while declaring the structure	② "class" keyword is used while declaring the class.
③ Structure variables can be created is called as structure type variable.	③ class variables can be created is called as objects. or class type variable
④ Functions are not included in structure definitation.	④ Functions are included in class definitation.
⑤ eg- struct student { int rollno; char name[20]; } s1;	⑤ eg- class student { int rollno; void get(); };
⑥ No public and private declaration	⑥ In class the data member and member functions can be declared as private and public.

Date / /
Page No.

① Data hiding is not achelved

② Data hideing is achelved.

* Enumeric datatype -

An enumerated datatype is user defined datatype which provides way for attaching name to number.

The enum keyword automatically enumerate list of words by assigning them values from 0, 1, 2 - - - - n.

Syntax -

① enum shape { circle, square, rectangle };

② enum colour { Red, Green, Blue };

The tag names shape and colour becomes user defined datatypes.

enum shape { circle, square = 4, rectangle };

for eg -

```
# include <iostream.h>
# include <conio.h>
# include <iomanip.h>
enum game { win = 1, lost, tie };
void main ()
{
```

```
game   status;
int   choice;
cout<<"enter your choice";
cin>>choice;
switch(choice)
{
 case 1:
  status=win;
  cout<<"congrats you win the game"<<
     status;
     break;

 case 2:
   status=lost;
  cout<<"sorry you lost the game"<<
  status;
   break;

 case 3:
    status=tie;
   cout<<"game is tied"<<status;
    break;

   default:
   cout<<"sorry wrong choice";
}
 getch();
}
```

Imp

★ Reference variable -

C++ introduces a new kind of variable known as reference variable.

It provides an alternating name for previsouly defined variables.

Syntax-

Data type &reference variable name = variable name ;

eg-

```
float total = 10.50;
float &sum = total;
total = total + 10;
cout << " total is = " << total; // 20.50
cout << "sum is = " << sum; // 20.50
```

If we can change the value of one variable then it changes in both variables

V. Imp

★ Operators in C++ -

① Insertion operator : ("<<").
② extraction operator : (">>")
③ Scope resolution operator : ("::")
④ Pointer member declaration ("::*")
⑤ Pointer to member operator : ("->*", ".*")
⑥ delete operator - It is an memory release operator.

Date : / /
Page No.:

① new - memory allocation operator.
② setw(size) - Field width operator.

Imp ① scope resolution operator
4m C++ is block structured language.
The scope of the variable declared in block entends from point of its declaration till the end of the block
The variable declare inside the block
② is said to be local variable declaration.
In 'C' global version of variable - cannot be accessed within inner block.
In "C++" scope resolution operator is used to uncovered the hidden variable.
Syntax -
:: variable name ;

4m eg -

```
# include <iostream.h>
# include <conio.h>
void main()
int m=10; // Global declaration.
void main()
{
  int m=20; // local declaration
  {
    int k=m;
    int m=30;
    cout << " k = " << k; // o/p=20
    cout << "m = " << m; // o/p=30
```

```
    cout<<"::m"<<::m; // o/p = 10
    }
    cout<<"m="<<m; // o/p = 20
    cout<<"::m"<<::m; // o/p = 10
    getch();
}
```

* Memory management operator.
In "C" wes "malloc()" and "calloc()" functions allocate the memory dynamically at run time.
The function "Free()" is used to free dynamically allocated memory
In "c++" also defined two binary operators "new" and "delete" that performing the task of allocating and freeing the memory

Syntax -

```
pointer_variable = new datatype;
```

The "new" operator allocate the sufficien memory to hold a datatype.
The pointer variable holds the addres of memory space allocated.

eg -

```
a = new int;
b = new float;
```

endl = line feed operator:

Date: / /
Page No.:

In above eg - a is the pointer variable having integer datatype
In above eg - b is the pointer variable having float datatype.
When the datatype no longer needed it destroyed to release the memory.

Syntax-

delete operator pointer_variable_name;

eg -

```
delete a;
delete b;
```

* Manipulators -

Manipulators are operators that are used to format the data display.
The most commonly used manipulator are "endl" and "setw()".
The "endl" manipulator when used in output statement it causes a line feed to be inserted. It is same as "\n"

eg-

```
cout << "sum =" << x ;
cout << endl << "sub = " << y;
```

o/p :-

```
sum = 10
sub = 5.
```

If number is right justified the "setw" manipulator allow to specify the common width of all the numbers and forces them to printed right justified.

syntax -

```
setw(size);
```

eg- sum = 345

```
cout << setw(5) << sum;
```

The manipulator setw(5) specifies in field width "5" or printing the value of variable sum.

eg -

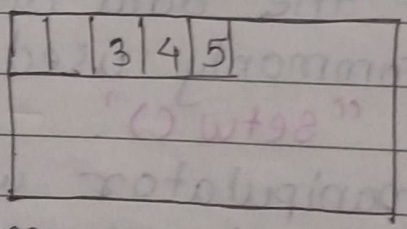


fig - memory representation.

Date : / /
Page No.:

• Functions -

Functions is the self contained block of statements that performe specific well defined task and may written a value to the calling program is kniown as Functions.

* Advantages of functions :-

① By following top down approach, the main can be kept very small and also task can be designed in various functions

② Troubleshooting and debugging becomes easier in structured program.

③ Individual functions can be easily build and tested.

④ Multiple function can be developed and tested simultaneously to reduce program developent life cycle time.

⑤ Function can call the other functions.

⑥ The function can call itself is called direct recursion.

Date: / /
Page No.:

with ex-

FAQ
VVImp

* inline function.

One of the object is used in function the program is saved to memory space. Every time the function is called it takes lot of extra time in excauting the series of instructions or task such as jumping to the function, saving the registor, pushing the arguments in a stack and written into the calling function.

When function is small the time may not be spend.

The ctt processes a new function is called "inline function" i-e compiler replaces the function call with corresponding function definition.

Syntax-

```
inline return_type functionname()
{
    body of function
}
```

When we prefix the keyword "inline" it becomes inline function.

All inline function must be defined before they are called.

Date : / /
Page No.:

Inline keyword just send the request not a command to the compiler.

The compiler may ignore if the function is too large, to complicated it compiles the function as a normal function

The inline expression makes program to run faster.

program -

```
#include <iostream.h>
#include <conio.h>
inline float multi(float x, float y)
{
    return (x*y);
}
void main()
{
  float x=5.3, y=2.3;
  cout<<" multiplication is ="<<multi
  (x,y);
  getch();
}
```

- Default arguments -

C++ allows us to call function without specifing all its arguments.

In such cases the function assigns default value to the parameter.

A default argument is a value provided in function declaration.

i.e automatically assigned function defination by the compiler.

eg-

```
void amount (float Pr, int year, float rate = 0.12); // function declaration

amount (10,000, 3); // function calling
```

passes the value 10000 to principal and 3 to no. of years and the function uses default value 0.12 for the rate of interest.

We can add default value from left to right.

eg- program -

```
# include <iostream.h>
# include <conio.h>
Float bvolume (float l=1, float b=1,
float h=1);
void main()
{
  cout << "Default bvolume =" << bvolume
  getch();
}
float bvolume (float l, float b, float h)
{
  return l*b*h;
}
```

Date : / /
Page No.:

② #include <iostream.h>
#include <conio.h>
Float bvolume (float l=10, float b=1,
float h=1);
void main()
{
cout << "Defaut value of l=" <<
bvolume (10);
getch();
}
Float bvolume (float l, float b, float
h)
{
return l *b*h;
}.

#include <iostream.h>
#include <conio.h>
Float bvolume (float l=1, float b=1,
float h=1);
void main()
{
cout <<' Default bvolume = " << bvolume();
cout <<" default value of l is 10 & two def"<<
bvolume (10);
cout <<" default value of l and b "<<bvolume
(10,20);
cout <<" default value of l, b and h"<<
bvolume (10, 20, 30);
getch();
}

```
float bvolume (float l, float b, float h)
{
    return l*b*h;
}
```

***