OOP PART-1

		internal	external
BTCS305	L3, T1,	40	60
BTCS309 Lab	P4	30	20

1-4, 5-7, 8-11

3rd CSE

15/07/15

CH-1

1. OOP Concepts → ⓐ Procedural programming

(I) problem is viewed as sequence of instruction that to be executed in step by step.

(II) gp of instruction is known as function, primary focus on fuction.

(III) Large program is divided into smaller program known as function

(IV) each function has its own local data and they can share global data among themselves data move openly around the system from one function to another function.

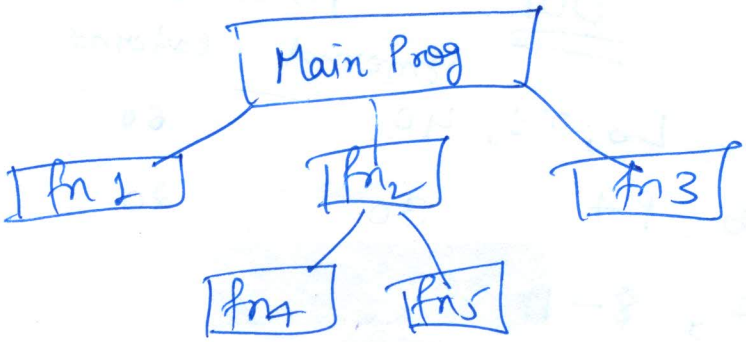
(V) It uses top down approach in program.

(VI) It does not Model real world problem very well.

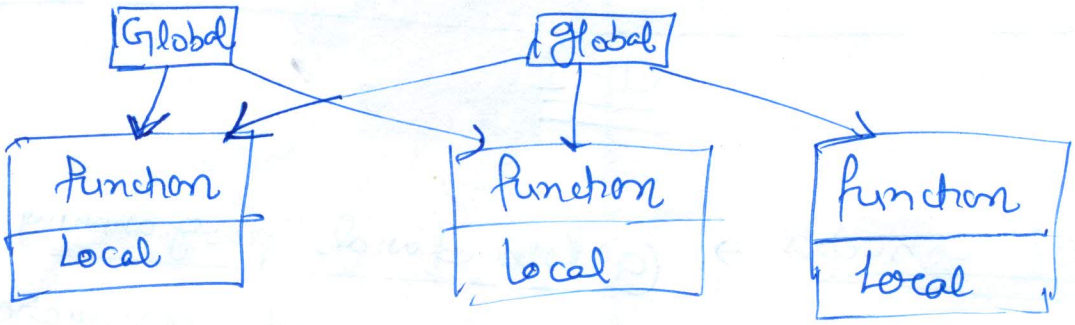
(VII) If program is too large, It is very difficult to identify what data is used by which function.

drawback

Eg COBOL, FORTRAN & C



Structure of procedural lang



function & data

(b) Object oriented Programming →

(I) It Remove the drawbacks of PPL such used to solve real life world problem with help of programing. & It does not allow data to flow freely arround the system.

(II) It emphasis on data rather than ~~fn~~ procedure

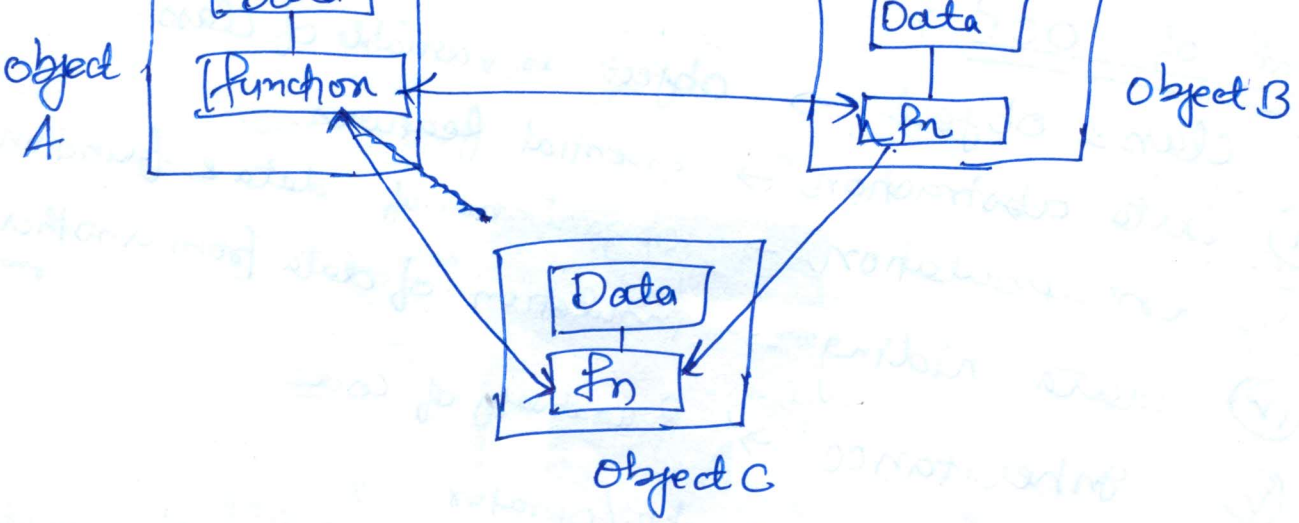
(III) data & function are organised in one entity called class.

(IV) Data can not be accesed by external function.

(V) New data & function can be easily added.

(VI) It follow bottom up approach.

(VII) decomposition of problem into no. of entities called objects.

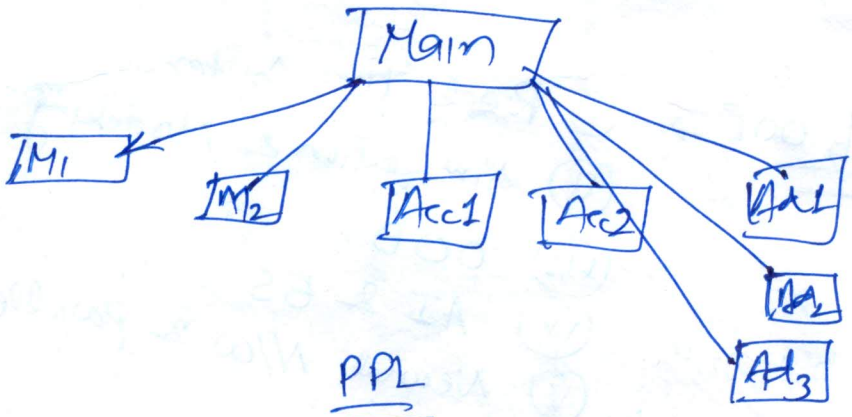


OOP

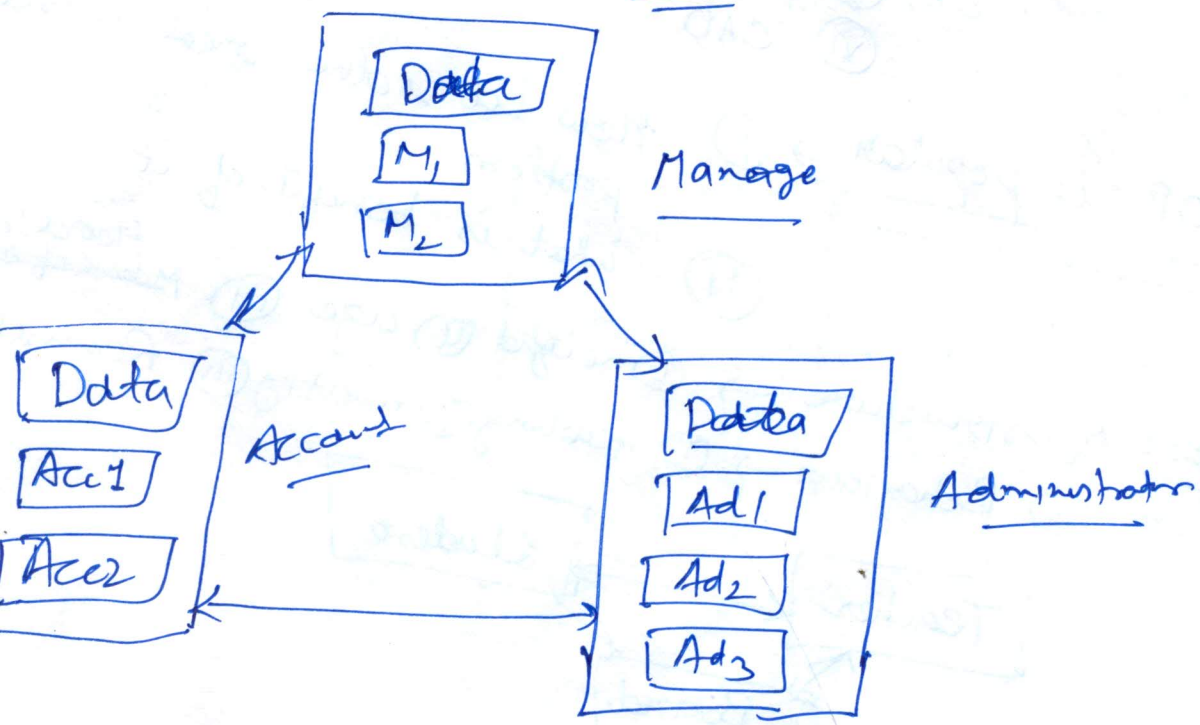
Example

education institute

(I) Management (II) Accountant (III) Administrator



PPL



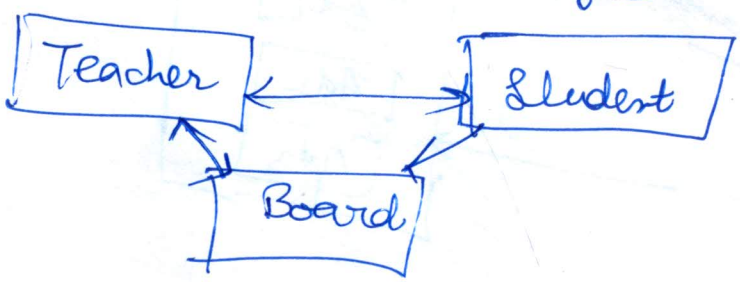
OOP

(I) Class & object → object is variable of class.
(II) Data abstraction → essential features
(III) encapsulation → organisation of data & function
(IV) data hiding → insulation of data from another function/program
(V) Inheritance → Reusability of code
(VI) overloading → fn/operator
(VII) polymorphism → Many form of single prog/function
(VIII) Messaging → comm^n of object with each other when object call the function of class.

Application of OOP → (I) Real Time System
(II) Simulation & Modeling
(III) OOD
(IV) AI & ES
(V) Neural N/w & parallel programming
(VI) CAD

Why OOP is popular → (I) How it solve real world problem
(II) what is benefit of it

Object → (I) structure → (I) height (II) age (III) ~~Mark of sign~~ gender
(II) Behaviour → (I) teaching (II) writing (III) Research



CH-2

Begining with C++ → 1970 — C by Dennis Ritchie
1980 — C++ by Bjarne stroustrup at AT&T Bell Laboratories

BCPL, Ken Thompson — B language

C++ → C with classes, 1983 → Changed to C++

Simula67 & C

① <iostream.h> ⇒ cout & cin
cout << ⇒ insertion operator
cin >> ⇒ extraction operator
<stdio.h> ⇒ std I/o Library function

② I/o function

ⓐ formatted I/o → scanf() & printf()
> I/p and O/p formatted as per requirements.
> such space & width we require, new line

ⓑ unformatted I/o →
> character I/o → take one character at a time

(i) getchar() → Return a char that has been recently typed

(ii) getch() → " " same " ". But neither the user required to type enter key after typing a character nor the typed character is echoed to the computer screen.

(iii) getche() → " " . User not required to type enter key after typing character, required in getchar.

putchar() → outputs the character variable to std o/p.

for eg →

```
void main()
{
 char ch;
 Printf("\nEnter a character:");
 ch = getchar();
 Printf("\n output1 is");
 Putchar(ch);

 ch = getche();
 Printf("\n output2 is");
 putchar(ch);

 ch = getch();
 Printf("output3 is");
 putchar(ch);
}
```

String I/O → gets and puts

for eg

```
void main
{
 char s[10];
 Printf(" Enter the string");
 gets(s);
 Printf(" output is");
Puts(s);
}
```

stream

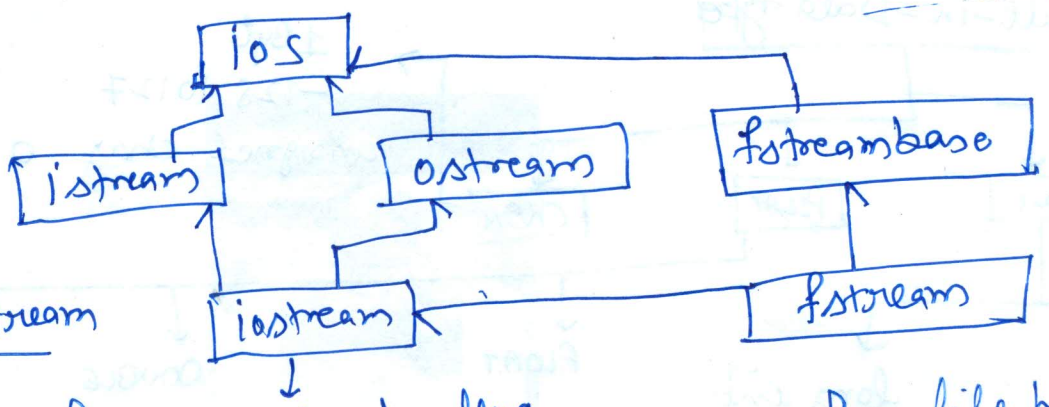


fig: Hierarchy of stream

for character handling
i.e cin, cout

for file handling
i.e
fout-open
fout-close

for eg → #include <iostream.h>

C:\tc\Bin\Include

I/o using Manipulator → used to manipulate the I/o using <iomanip.h>

(I) endl;
(II) hex, dec, oct
(III) setbase
(IV) setw
(V) setfill
(VI) setprecision
(VII) ends.

(I) cursor will ~~transfer~~ to new line
cout<<sunil; cout<<kumar; } cout<<"sunil"<<endl;
sunil kumar
sunil kumar

(II) cout<<"value in decimal"<<dec<<a
<<hex<<
<<oct<<a

(III) setbase(8), setbase(16)
Cout<<"value in decimal"<<setbase(10)<<value;

(IV) setw cout<<a<<b;
cout<<setw(2)<<a<<setw(3)<<b;
10-20

(V) cout<<setfill('*');
cout<<setw(2)<<a<<setw(3)<<b;
10*20

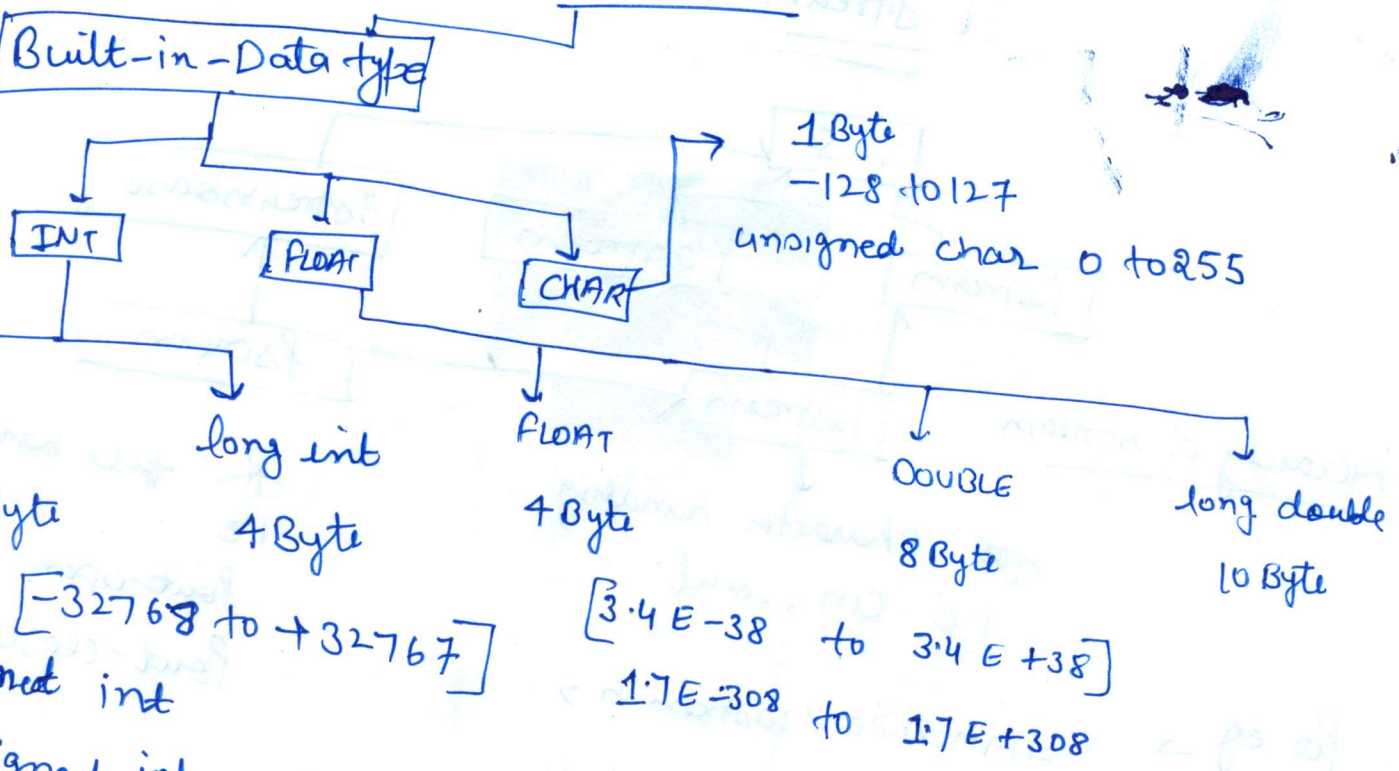
(VI) up to decimal point
Cout<<setprecision(6)<<c;
" " " (2)<<c;

(VII) to add null ~~string~~ at ~~terminator~~ end of I/o character ('\0')

→ right justified
setw(5)<<a; | | 1 | 2 | 3 |
<<a

| 1 | 2 | 3 | left justified

(a) Built-in-Data type



(b) Structured / user defined data type

- Structure
- Union
- class
- Enumerated

(c) Other Data types

- Array
- Pointer

Operator in C++ →

(a) Arithmetic, +, -, /, *, %
(b) Relational ⇒ <, >, ≤, ≥, ==
(c) Logical ⇒ &&, ||, !
(d) Bitwise ⇒ &, |, ~, <<, >> ;
(e) Assignment ⇒ =, +=, -=, /=, %=
(f) INC/Dec ⇒ ++, --,
(g) special operator ⇒ ,, ::, sizeof, ? → ternary operator

Unary operator → applied on single variable, +, -, ++, --

(11) greatest of two number using ternary operator. (5)

(1) Comments in C++ → (a) // single line comment
(b) /* multiple line comment */

(2) Keywords
(3) Identifers
(4) Constants
(5) Strings.
(6) character

Type Casting (1) to convert lowest type to highest type is called automatic conversion.

(a) for eg → int c=7;
float a= 10.5;
double t= c*a;

longdouble double float longint int char short

Highest ← to lowest

```
Int x; 20
float y; 10.5
x=y;
```
Ans → 10

```
int a =200;
Int b = 300;
float c;
c= float(a) *b;
```

```
Int a= 5;
Int b= 2;
float c;
c = a/b;
```
Ans= 2

c = (float)a/b
Ans = 2.5

Implicit / automatic conversion →

Control Structure ⇒

- Conditional Statnet
  - if Statment
    - if else
    - nested if else
  - Switch case
- Loop Statment
  - while
  - do while
  - For loop
- Breaking control statement
  - Break
  - Continue
  - go to

Conditional Statment

(I) WAP to calculate ~~Divi~~ Division of Numbe if first No is greater than second.

(II) greatest of three Number

(III) WAP to find day of week with Switch case.

Loops

(I) while loop → for loop is best if we know fixed no. of repition but if we dont know how many time you want to do something before you start the loop, then while loop is used.

```
while (condition)
  body of loop;
```

WAP to find fibonacci series

$f_3 = f_2 + f_1$, $f_4 = f_2 + f_3$

```
cin>>n;
f1=1;
f2=1;
cout<<f1<<endl;
cout<<f2<<endl;
int c=2
while(c<n)
{
  f3=f1+f2;
  f1=f2
  f2=f3
  cout<<f3<<endl;
}
```

(I) Why break statment is used in Switch.

(II) Switch ~~loop~~ use only int & character value.

(III)

(II) For loop.

```
for(int i=0; i<=9; i++)
{
  statement 1;
  statement 2;
}
```

foreg →

```
for(int i=0, f=2; i<=5; i++, f--)        ——— (I)
for(i=0, j=3; j2=5; j++)                 ——— (II)
for(;;)   infinite loop                  ——— (III)
```

Program

(I) WAP to find table of 10.

(II) WAP with the help of nested loop.

```
for(i=1; i<=6; i++)
{
for(j=1; j<=i; j++)
cout<<j;
}
```

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

(III) Do while → body of loop execute at least one time.

```
do
{
statement 1;
  ", " 2
}
while(condition);
```

WAP to calculate sum of odd No & even No.

```
cin>>n
int sum=0, i=1;
do
{
sum = sum+i;
  i = i+2;  ⇒
}
while(i<=n) cout<<sum;
```

i=2;

WAP to find sum of n different values avg of it.

```
sum=0; i=0; cin>>n
do
{
 cin>>x;
 sum=sum+x;
 i++;
}
while(x<=n-1);
avg = sum/n;
```

Breaking control statements

(I) **Break statement** → It terminates the loop in which that is written.

```
for(i=0; i<=5; i++)
{
  Statm 1;
  break;
  Statm 2;
}
Statmat 3;
```

loop terminate

(II) **Continue** → inverse operation of break statment.

```
while(condn)
{
Statment 1;
Statment 2;
if(a<b)
continue;
statmat 3;
}
```

Que WAP which reads an integer & then calculate the sum of individual digits in number.

Ans

```
cin >> n;
int sum=0, d=0;
do
{
sum = sum + n%10;
 n = n/10;
 d++
}
while(n>0);
cout<<"no. of digits in the"<<n<<"are"<<d;
cout<<" sum of digits "<<sum<<endl;
getch();
}
```

(III) **goto** →

```
statment 1;
goto: x;
Statement 2;
X: statmat 3;
```

local and global variable

(1) Life time → time period b/w creation & destruction of variable is called its life time.

(11) Scope → where within a program it can be accessed

Storage classes → (1) Automatic ie Local →

```
auto int a,b;
auto int c,d;
```

```
void f()
{
auto int a,b;
}
```

```
void f()
{
int a,b;
}
```

Life time → variable created only when function is called & destroyed when we return from function

(2) external ie global

```
int a,b;
void main()
{
statement1;
   ____ a=10;
   ____
void f()
{
   ___ a=5;
   ___
}
}
```

external variable

```
extern int a,b;
```

Life time → exist for life of program.

if it is not initialized then it is initialized to zero.

(3) Static →

```
static int a;
```

It has scope of local variable but life time of an external variable

(4) Register →

```
void f()
{
register int a;
}
```