C & C++

- C++ supports stream-based I/O.
- #include<iostream.h>

```
 void main()
{
  cout<<“hello world”;
}
```

iostream.h supports stream programming features by including predefined stream objects

<< : insertion operator.

This operator sends the message to the predefined object, *cout*.

- Filename : hello.cpp ⟶ extension

complete syntax: (output stream)

cout << variable1<< variable2<<……..

<< variable n<<endl;

New line = "\n"

Input stream :

c.in>> variable

Stream object

Extraction operator

Complete syntax :

c.in >> variable 1>> variable 2>>…….>>variable n

```
#include<iostream.h>
 void main()
{
  float number1, number2;
  float  sum, average;
  cout<<“enter two numbers:” ; // prompt
  cin>> number1>> number2;
  sum=number1+number2;
  average=sum/2;
  cout<<“sum=”<<sum<<endl;
  cout<<“average=”<<average<<endl;
}
```

Scope resolution operator::

The scope resolution operator(::) permits a program to reference an identifier in the global scope that has been hidden by another identifier with the same name in the local scope

::(2 colons without space)

```
#include<iostream.h>
 int num=20;
 void main()
{
  int num=10;
  cout<<“Local=”<<num<<endl;//local variable
  cin<<“Global=”<<::num<<endl;//global variable
  cout<<“Global+Local=”<<::num+num<<endl;
}
```

Output :

Local=10

Global=20

Global+Local= 30

- An example with class:

```
#include<iostream.h>
class person
{
  char name[30];
  int age;
  public:
    void getdata (void);
    void display (void);
 };
  void person:: getdata (void);
  {
```

```
void person:: display (void)
{
 cout<<"\n name: "<<name;
 cout<<"\n age: "<<age;
}
void main()
{
 person p;
 p.getdata();
 p.dislpay();
}
```

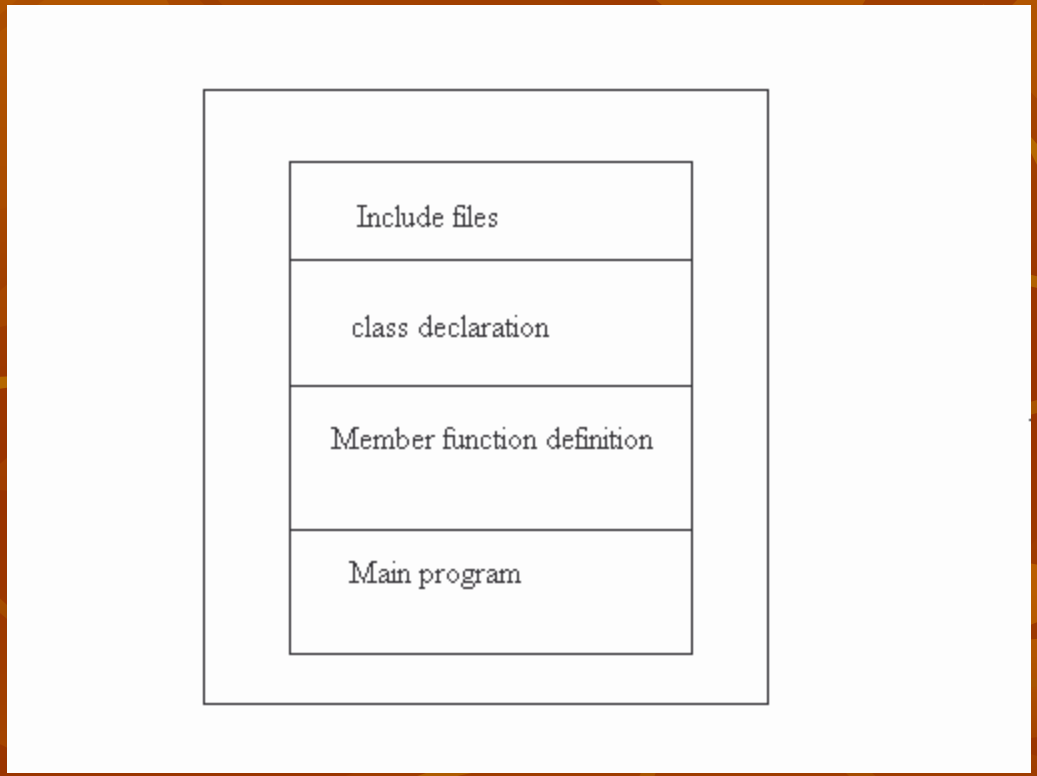
output:

```
enter name: John
enter age: 30
name: John
age: 30
```

P ⟶ is an object belonging to the class person

Note:

cin can read only one word and therefore we cannot use names with blank spaces



Variable Aliases -Reference Variables

C++ supports one more type of variable, in addition to the value variable and pointer variable in c.

General Declaration:

Data Type & Reference Variable = Value Variable

```
# include<iostream.h>
Void main()
{
  int a=1,b=2,c=3;
  int & z=a;//variable z becomes an alias of a
  cout<<"a="<<a<<"b="<<b
      <<"c="<<c<<"z="<<z<endl;
  z=b;// changes value of a to that of b
  cout<<"a="<<a<<"b="<<b
  <<"c="<<c<<"z="<<z<<endl;
  a++;
  cout<<"a="<<a<<"b="<<b
  <<  "c="<<c<<"z="<<z<<endl;
}
```

output:

a=1 b=2 c=3 z=1
a=2 b=2 c=3 z=2
a=3 b=2 c=3 z=3