Arshita
21115024 / P2

① The given code will throw an error.

We declared b in line 3 int b = 20;

& in the very next line we declared int &b = a;

These are confiding declarations.

So, we get an error.

b was int in line 3. in line 4 → reference variable.

② Yes we can;

Structs can have inheritance relations also;.

Ex

We can define a struct in a class in C++

```
class Class{
  struct Struct{
    };
  };
```

→ The inheritance in case of struct is exactly like class except the default accessibility → public for struct while it is private for class.

Ex

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

(3)

```
#include<iostream>
#include<string>
using namespace std;
class Complex
{
public:
double real, imag;
Complex (double a, double b)
 {
  real=a;
  imag=b;
 }
Complex operator+(complex a2)
 {
  real = real +a2.real;
  imag= imag +a2.imag;
  return *(this);
 }
Complex operator*(complex a2)
 {
  real = (real*(a2.real)) - (imag*(a2.imag));
  imag = (real*(a2.imag)) + (imag*(a2.real));
  return *(this);
 }
 Void display()
 {
  cout<<real<<"+"<<imag<<"i"<<endl;
 }};
int main()
{
 int x1, y1, x2, y2;
 cin>>x1>>y1>>x2>>y2;
 Complex a1(x1,y1);
 Complex a2(x2,y2);
 Complex a3 = a1 + a2;
 Complex a4 = a1 * a2;
 a3.display(); a4.display();
 }
```

④ Inline and Friend functions

Friend function:

Due to data hiding, we can hide certain data-values from outside the class.

If you are a member function, then only you can change the values of the hidden data.

But in some other cases, you can access the hidden data from non-member function.

Such functions are called friend functions.

Ex→

```
class Ctest
{
 private:
 int x,y;
 public:
  void put()
  {
   x = 1;
   y = 2;
  }
 friend int calculate (Ctest obj1);
};
int calculate (ctest obj1)
{
 return (obj1.x + obj1.y);
}
void main()
{
 Ctest obj;
 obj.put();
 cout << " The sum is:" << calculate (obj);
}
```

output = 3.

Since friend function are not members of the class, they can't access the variables x & y directly. They can only be accessed through an object of the respective class.

Inline function

A function that is expanded in line when it is called when the inline function is called, whole code of the inline function gets inserted or substituted at the point of inline function call. This substitution is performed by the C++ compiler

Ex
```
#include<iostream>
using namespace std;
inline int square(int t)
{
 return t;
}
int main()
{
cout<<"square is" << square(4) <<endl;
return 0;
}
```

(5) 'this' pointer / keyword

C++ uses a unique keyword called this to represent an object that invokes a member function. this is a pointer that points to the object for which function was called. This pointer is passed implicitly to point the object of that class. It is not user specified.

Ex:

```
#include<iostream>
using namespace std;
Class obj
{
int x;
public:
Void setValue(int x)
 {
 this->x = x;
 }
 Void print()
 {
 cout << x <<endl;
 }
int main()
{
 obj a;
 int x=5;
  a.set(x);
  a.print();
 }
```

↑ When local variable's name is same as member's name

→ To return reference to the calling object

→ Test is a class

```
Test & Test:: func()
 {
  return *this;
 }
```