First Year First Semester Course

M.Tech. (CS) [Batch 2021-23]

Lecture #15

Introduction to Programming

C++: Inheritance

Course Instructor:

Dr. Monidipa Das

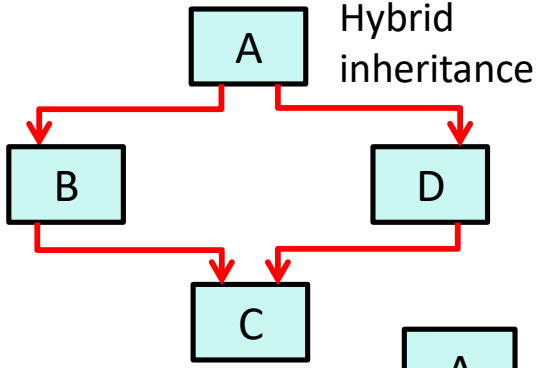
DST-INSPIRE Faculty

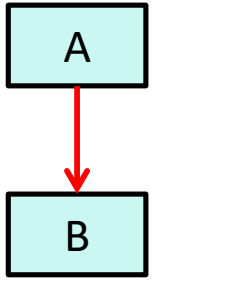
Machine Intelligence Unit (MIU), Centre for Artificial Intelligence and Machine Learning (CAIML)

Indian Statistical Institute (ISI) Kolkata, India

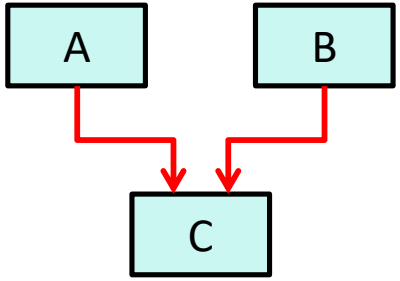
Inheritance

- The mechanism of deriving a new class from an old one
- Offers reusability
- Old class: Base class/Parent class/Super class
- New class: Derived class/Child class/Sub class

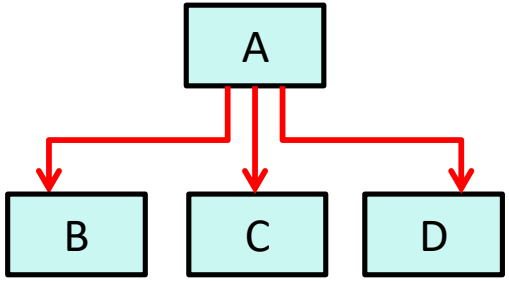




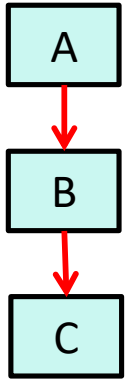
Single inheritance



Multiple inheritance



Hierarchical inheritance



Multilevel inheritance

Defining Derived Classes

General Form:

```
class derived-class-name : visibility-mode base-class name
{
      -------//
      -------//          members of derived class
      -------//
};
```

Private members are not inherited

Example:

```
class ABC : private XYZ
{
        members of ABC
};
```

```
class ABC : public XYZ
{
        members of ABC
};
```

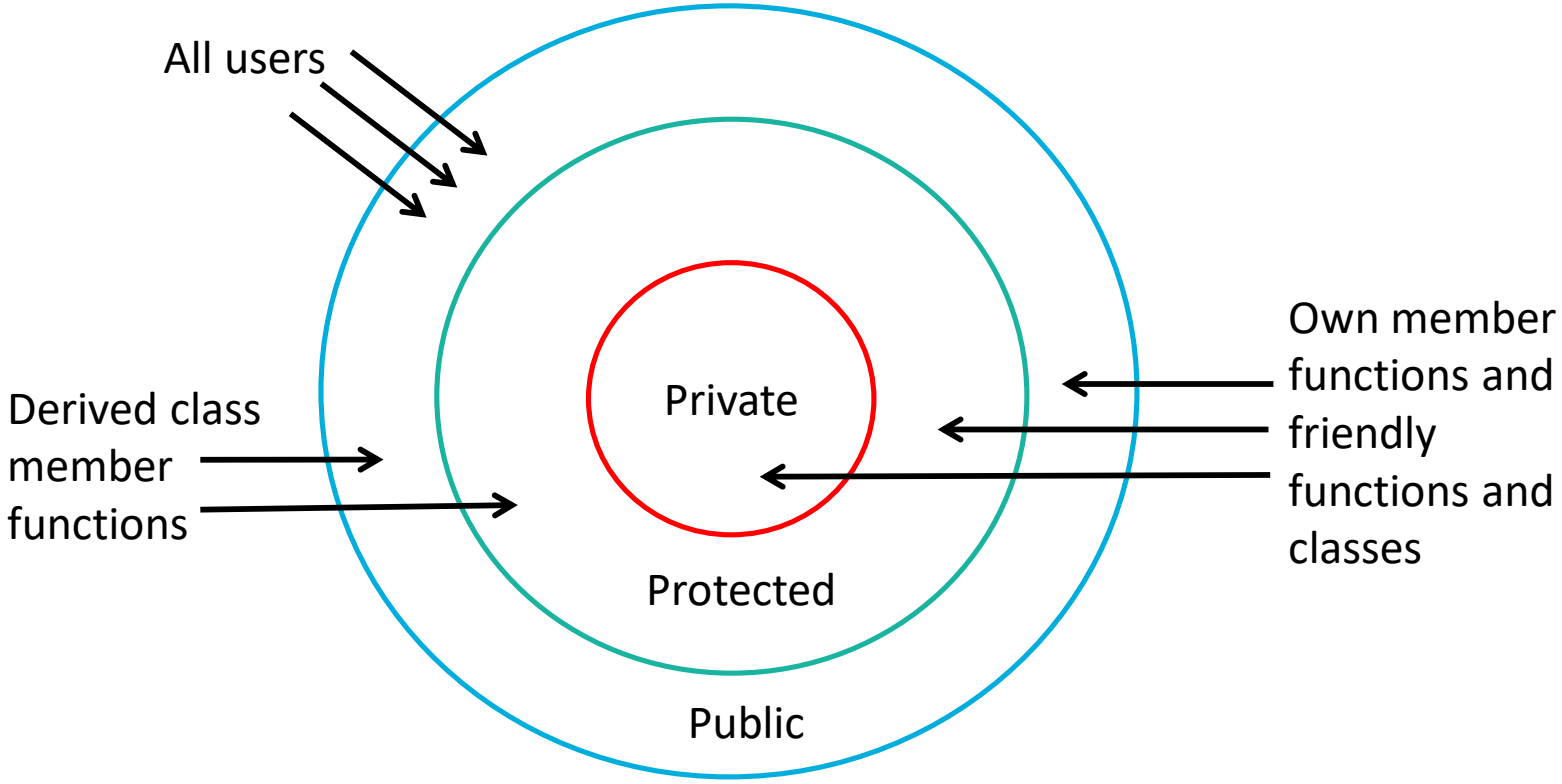
```
class ABC : XYZ
{
        members of ABC
};
```

Private derivation by default

Visibility of Inherited Members

Base class visibility	Derived Class Visibility		
	Public derivation	Private derivation	Protected derivation
Private	Not inherited	Not inherited	Not inherited
Protected	Protected	Private	Protected
Public	Public	Private	Protected

Access Control of the Members



Single Inheritance (public derivation)


```
class worker
{
          int age;
          char name [10];
          public:
                    void get();
                    void show();
};
class manager : public worker
{
          int now;
          public:
          void read();
          void display() ;
};
void worker :: get(){
  cout <<"\nYour name please: ";
  cin >> name;
  cout <<"\nYour age please: ";
  cin >> age;
}
```

```
void worker :: show(){
          cout <<"\nAs Worker: My name is "<<name<<". My
age is "<<age;
}
void manager :: read(){
          get() ;
          cout << "\nNumber of workers under you: ";
          cin >> now;
}
void manager :: display(){
          show();
          cout <<"\nAs Manager: No. of workers under me
is " << now;
}
int main(){
          manager M1;
          M1.get();
          M1.show();
          M1.read();
          M1.display() ;      return 0;
}
```

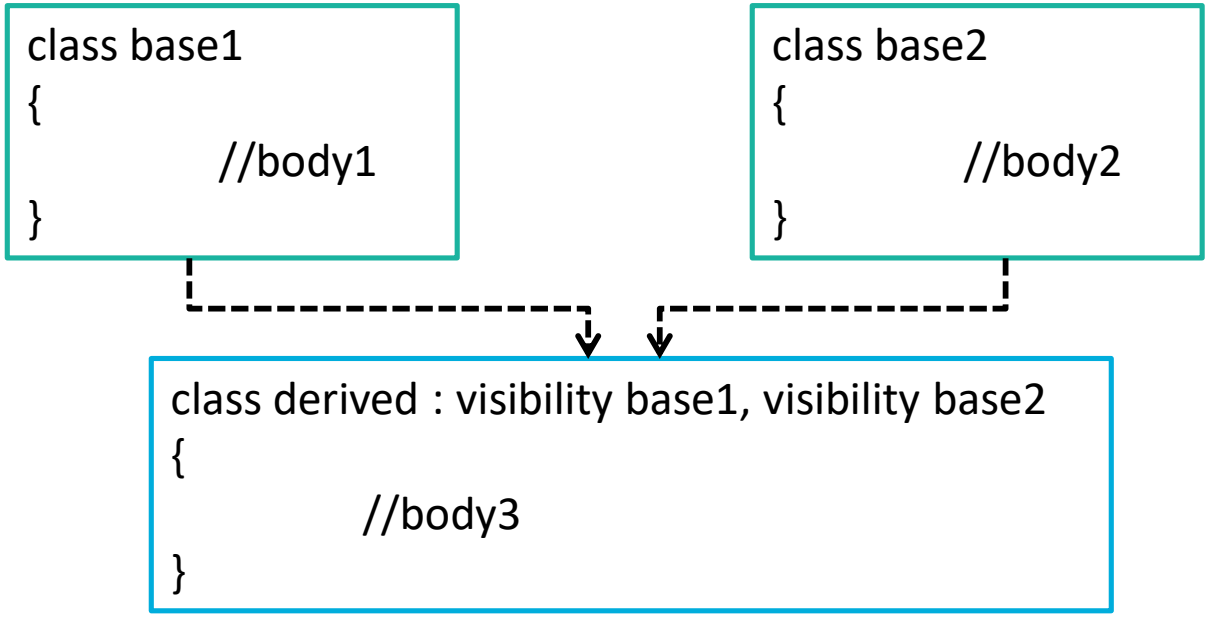
Single Inheritance (private derivation)


```
class worker
{
          int age;
          char name [10];
          public:
                    void get();
                    void show();
};
class manager : private worker
{
          int now;
          public:
          void read();
          void display() ;
};
void worker :: get(){
  cout <<"\nYour name please: ";
  cin >> name;
  cout <<"\nYour age please: ";
  cin >> age;
}
```

```
void worker :: show(){
          cout <<"\nAs Worker: My name is "<<name<<". My
age is "<<age;
}
void manager :: read(){
          get() ;
          cout << "\nNumber of workers under you: ";
          cin >> now;
}
void manager :: display(){
          show();
          cout <<"\nAs Manager: No. of workers under me
is " << now;
}
int main(){
          manager M1;
          //M1.get();
          //M1.show();
          M1.read();
          M1.display() ;      return 0;
}
```

Multiple Inheritance

- The syntax of the derived class here is



Multiple Inheritance [contd.]


```
class father
{
    int age ;
    char name [20] ;
  public:
    void get ( ) ;
    void show ( ) ;
};
void father :: get ( )
{
    cout << "Enter the name of father: ";
    cin >> name;
    cout << "Enter the age of father: ";
    cin >> age;
}
void father :: show ( )
{
    cout<< "\nFather's name is "<<name<<
". Father's age is "<<age;
}
```

```
class mother
{
    int age ;
    char name [20] ;
   public:
    void get ( )
    {
          cout << "Enter the name of mother: ";
          cin >> name;
          cout << "Enter the age of mother: ";
          cin >> age;
    }
    void show ( )
    {
          cout << "\nMother's name is "<<name;
          cout << ". Mother's age is "<<age;
    }
};
```

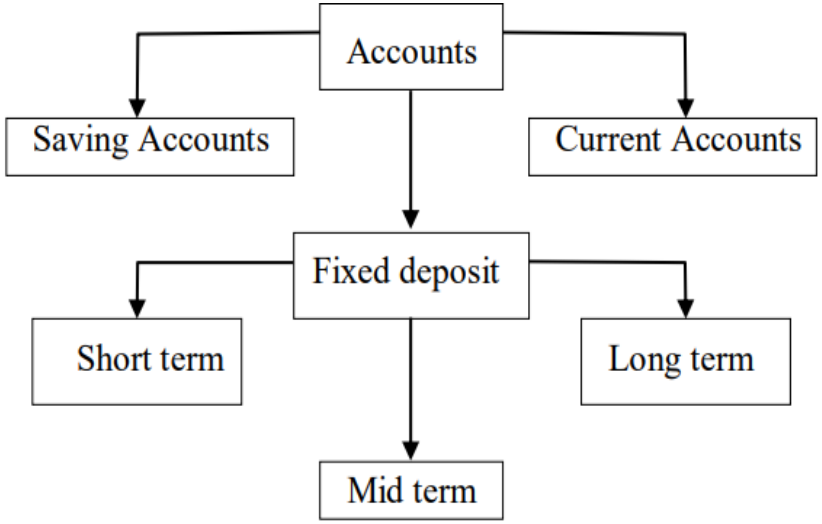
Multiple Inheritance [contd.]


```
class daughter : public father, public mother
{
    char name [20] ;
    int std;
  public:
    void get ( ) ;
    void show ( ) ;
};
void daughter :: get ( )
{
    father :: get ( ) ;
    mother :: get ( ) ;
    cout << "Enter the child's name: ";
    cin >> name;
    cout << "Enter the child's standard: ";
    cin >> std;
}
```

```
void daughter :: show ( )
{
    father :: show ( );
    mother :: show ( ) ;
    cout << "\nChild's name is "<<name;
    cout << ". Child's standard is "<< std;
}
int main ( )
{
          daughter d;
          d.get();
          d.show();
          return 0;
}
```

Hierarchical Inheritance

- The syntax of the derived class here is




```
class A
{
        //body A
}
```

```
class D1: public A
{
        //body1
}
```

```
class D2: public A
{
        //body2
}
```

Hierarchical Inheritance


```
class father //Base class declaration
{
    int age;
    char name [15];
  public:
    void get ( )
    {
      cout<< "Father's name please: ";
      cin >> name;
      cout<< "Father's age please: ";
      cin >> age;
    }
    void show ( )
    {
      cout << "\nFather's name is : "<<name;
      cout << ". Father's age is: "<< age;
    }
};
```

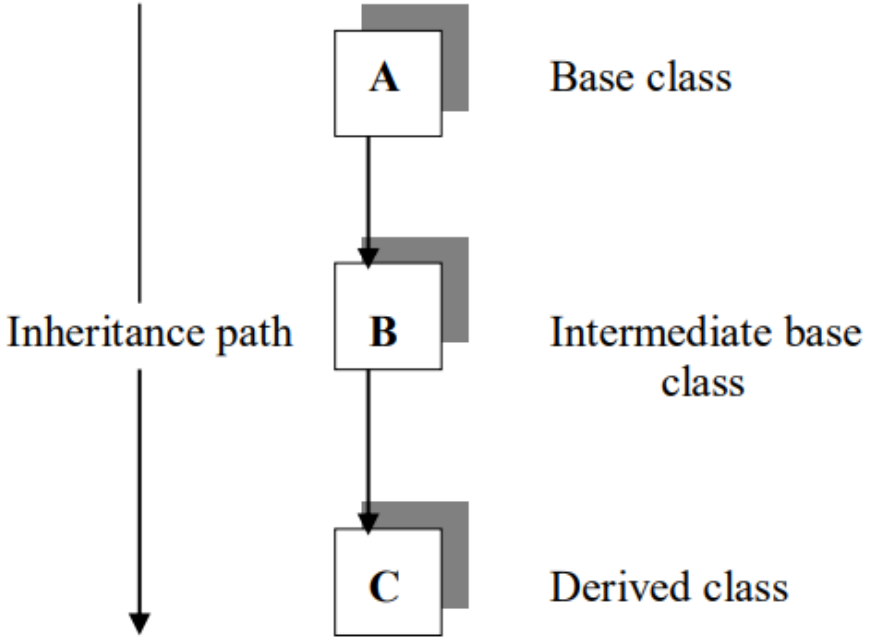
```
class son : public father
{
   char name [20] ;
   int age ;
  public:
   void get ( ) ;
   void show ( ) ;
} ;
void son :: get ( ){
   father :: get ( ) ;
   cout << "Enter the name of son: ";
   cin >>name;
   cout << "Enter the age of son: ";
   cin>>age;
}
void son :: show ( ){
   father :: show ( ) ;
   cout << ". Son's name is : " <<name;
   cout << ". Son's age is : " <<age;
}
```

Hierarchical Inheritance


```
class daughter : public father //derived class 2.
{
   char name [15] ;
   int age;
  public:
   void get ( )
   {
   father :: get ( ) ;
   cout << "Enter the name of daughter: ";
   cin>>name;
   cout << "Enter the age of daughter: "; cin >>age;
   }
   void show ( )
   {
   father :: show ( ) ;
   cout << ". Daughter's name is : " <<name;
   cout << ". Daughter's age is : " <<age;
   }
};
```

```
int main ( )
{
          son S1;
          daughter D1;
          S1.get();
          D1.get();
          S1.show();
          D1.show();
          return 0;
}
```

Multilevel Inheritance




```
class A
{
        //body
}
class B : public A
{
        //body
}
class C : public B
{
        //body
}
```

Multilevel Inheritance [contd.]


```
class worker // Base class declaration
{
  int age;
  char name [20] ;
 public:
  void get( ) ;
  void show( ) ;
};
void worker:: get ( )
{
  cout << "Your name please: ";
  cin >> name;
  cout << "Your age please: ";
  cin>> age;
}
void worker :: show ( )
{
          cout <<"\nAs worker: My name is
"<<name<<". My age is "<<age;
}
```

```
class manager : public worker
{
    int now;
  public:
    void get ( ) ;
    void show( ) ;
};
void manager :: get ( )
{
    worker ::get () ;
    cout << "No. of workers under you: ";
    cin >> now;
}
void manager :: show ( )
{
          worker :: show ( ) ;
          cout << "\nAs manager: No. of
workers under me are "<< now;
}
```

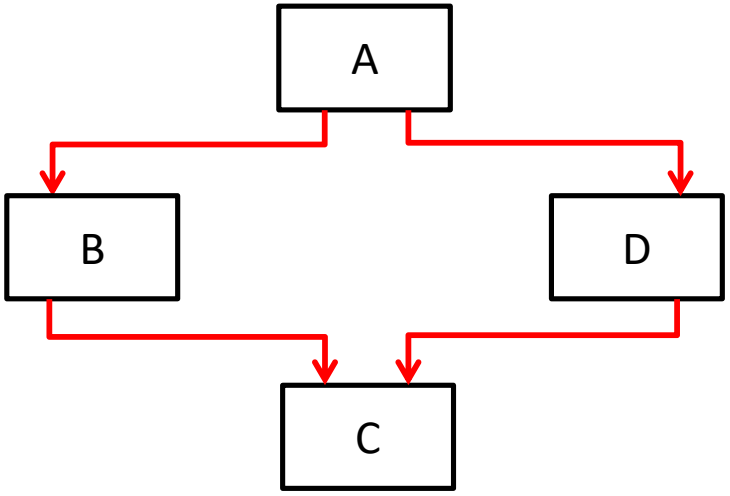
Multilevel Inheritance [contd.]


```
class ceo: public manager
{
    int nom;
  public:
    void get ( ) ;
    void show ( ) ;
};
void ceo :: get ( ){
    manager :: get ( ) ;
    cout << "No. of managers under you are: ";
         cin >> nom;
}
void ceo :: show ( ) {
         manager :: show ( ) ;
    cout << "\nAs CEO: No. of managers under me are ";
    cout << nom;
}
```

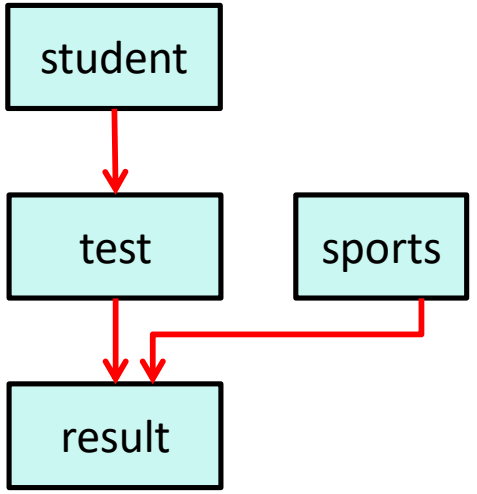
```
int main ( )
{
         ceo c1 ;
         c1.get ( ) ; cout <<"\n\n";
         c1.show ( ) ;
}
```

Hybrid Inheritance

- Hybrid Inheritance is the combination of one or more types of the inheritance



Hybrid inheritance



Hybrid inheritance

Hybrid Inheritance [contd.]


```
class student //base class declaration
{
  protected:
    int r_no;
  public:
    void get_n (int a){
      r_no =a;
    }
    void put_n (void){
      cout << "Roll No. : "<< r_no;
    }
};
```

```
class test : public student
{ //Intermediate base class
  protected : int part1, part2;
  public :
    void get_m (int x, int y) {
         part1 = x; part2 = y; }
    void put_m (void) {
         cout << "\nMarks obtained: "
                          << "Part 1 = " << part1
              << " Part 2 = " << part2;
        }
};
```

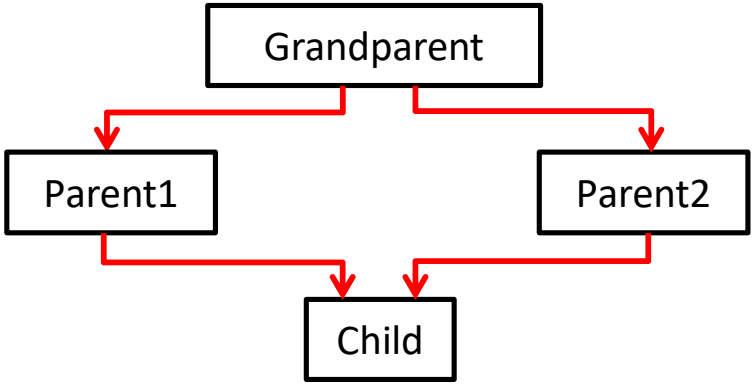
```
class sports // base for result
{
  protected : int score;
  public:
    void get_s (int s) {score = s; }
    void put_s (void) {cout << "\nSports wt. : "<< score << "\n\n";}
};
```

Hybrid Inheritance [contd.]


```
class result : public test, public sports //Derived from test & sports
{
    int total;
  public:
    void display (void);
};
void result :: display (void){
    total = part1 + part2 + score;
    put_n();
    put_m();
    put_s();
    cout <<"Total score: "<<total<< "\n";
}
int main ( ){
    result S1;
    S1.get_n (1042) ;
    S1.get_m (80, 85);
    S1.get_s (8) ;
    S1.display ( ) ; return 0;
}
```

Virtual Base Classes

- When a class is **virtual base class**, C++ takes necessary care to see that only one copy of that class is inherited



Multipath inheritance

- Note that keywords 'virtual' and 'public' can be used in either order.

```
class  g_parent
{
        //Body
};
class parent1: virtual public g_parent
{
        // Body
};

class parent2: public virtual g_parent
{
        // Body
};
class child : public parent1, public parent2
{
        // body
};
```

Virtual Base Classes [contd.]


```
class student //base class declaration
{
  protected:
    int r_no;
  public:
    void get_n (int a)
    {
      r_no =a;
    }
    void put_n (void)
    {
      cout << "Roll No. : "<< r_no;
    }
};
```

```
class test : virtual public student
{ //Intermediate base class

  protected :
    int part1, part2;
  public :
    void get_m (int x, int y) {
         part1 = x; part2 = y; }
    void put_m (void) {
         cout << "\nMarks obtained: "
          << "Part 1 = " << part1
         << " Part 2 = " << part2;
        }
};
```

Virtual Base Classes [contd.]


```
class sports: public virtual student
{
  protected :
          int score;
  public:
    void get_s (int s) {score = s; }
    void put_s (void) {
    cout << "\nSports wt. : "
    << score << "\n\n";
    }
};
class result : public test, public sports
//Derived from test & sports
{
    int total;
  public:
    void display (void);
};
```

```
void result :: display (void)
{
    total = part1 + part2 + score;
    put_n();
    put_m();
    put_s();
    cout <<"Total score: "
    <<total<< "\n";
}
int main ( )
{
    result S1;
    S1.get_n (1042) ;
    S1.get_m (80, 85);
    S1.get_s (8) ;
    S1.display ( ) ;
}
```

Questions?