Lecture 3 –More on Member Functions

Inline function

- If the function definition is inside the class, it is called inline function.
- It acts like a macro ie, code is expanded at the point of each invocation.(function call is replaced by function definition)

Advantages of inline function

- In the case of function call, significant amount of overhead is generated by calling and return mechanism.
- Typically arguments are pushed to stack and various registers are saved when a function is called and restored when function is returned.
- When a function is expanded in line, none of those operations occur.
- Faster run time

Restriction

- Number of codes should be less
- Inline is a request to the compiler to consider it as a macro. Compiler can accept it or ignore it.

Compiler wont consider it as inline in the following cases

- If function has many statements
- If it is recursive function
- If function returns any value
- If function has loop, switch, goto

How to make function which is defined outside the class as inline

```
class item
{ int cost;
  public:
      void getdata();
};
inline void item::getdata()
{
cin>>cost;
}
```

Normal functions

```
# include< iostream>
using namespace std;
largest(int a, int b);
int main ( )
{
        int a,b,c;
        cin>>a>>b;
        c=largest(a,b);
        cout<<“largest no is”<<c;
}
largest(int a, int b)
{
        if(a>b) return a;
        else return b;
}
```

NESTING OF MEMBER FUNCTION

```
class set
{
      int m,n;
      public:
      void input(void);
      void display (void);
      int largest(void);
};
int set::largest (void)
{
      if(m>n)   return m;
      else  return n;
}
```

```
void set::input(void)
{
           cin>>m>>n;
}
void set::display(void)
{
         cout<<"largestvalue="<<largest()<<"\n";
}
int main()
{
set A;
A.input( );
       A.display( );
}
```

Array of objects

```
class emp
{
        char name[20];
        int age,sal;
        public:
        void getdata( );
        void putdata( );
};
void emp : : getdata( )
{    cin>>name>>age>>sal;
}
void emp :: putdata ()
{
        cout<<”emp
        name:”<<name<<endl<<age<<endl<<sal;
}
```

```
int main()
{       int i;
        emp foreman[5];
        emp engineer[5];
        for(i=0;i<5;i ++)
        {  foreman[i] . getdata();  }
        for(i=0;i<5;i++)
        {  foreman[i].putdata(); .
        }
        for(int i=0;i<5;i ++)
        {  engineer[i].getdata();
        }
        for(i=0;i<5;i++)
        {
        engineer[i].putdata();
        }
}
```

Object as function arguments

```
class time
{
        int hours;
        int minutes;
        public:
        void gettime(int h, int m)
        {
        hours=h;
        minutes=m;
        }
        void puttime(void)
        {
        cout<< hours<<”hours and:”;
        cout<<minutes<<”minutes:”<<end;
        }
        void sum( time ,time);
};
```

```
void time :: sum (time t1,time t2)
{
        minutes=t1.minutes + t2.minutes;
        hours=t 1.hours+t2.hours;
        hours=hours+ minute/60;
        Minutes=minutes%60;
}
int main()
{
        time X1,X2,X3;
        X1.gettime(2,60);
        X2.gettime(3,30);
        X3.sum(X1,X2);
        cout<<”X1=”;
        X1.puttime( );
        cout<<”X2=”;
        X2.puttime( );
        cout<<”X3=”;
        X3.puttime( );
}
```