(OOP PROGRAMS OF BOOK)

(MADE BY ALI AKBER)

(BSCS 2ND SS1)

SIMPLE CLASS:

```
#include <iostream>

using namespace std;


class smallobj{

        private:

                int somedata;

                public:

                        void setdata(int d){

                                somedata=d;

                        }

                        void showdata(){

                                cout<<"data is "<<somedata<<endl;

                        }

};

int main (){

smallobj s1,s2;

s1.setdata(1200);

s2.setdata(1300);

s1.showdata();

s2.showdata();

return 0;

}
```

WIDGETS PARTS AS OBJECTS:

```
#include <iostream>

using namespace std;


class part{

        private:

                int modalnumber;

                int partnumber;

                float cost;

                public:

                        void setpart(int mn,int pn, int c){

                                modalnumber=mn;

                                partnumber=pn;

                                cost=c;

                        }

                        void getpart (){

                        cout<<"Modal is "<<modalnumber<<endl;

                        cout<<"Part number is "<<partnumber<<endl;

                        cout<<"Cost is "<<cost<<endl;}

};

int main (){

part part1;

part1.setpart(6244,3723,217.78F);

part1.getpart();


return 0;
```

```
}
```

//Objects as function arguments.

```
#include <iostream>
#include <string.h>
using namespace std;

class Distance{
        private:
                int feet;
                float inches;
                public:
                        Distance():feet(0),inches(0.0f){                //constructor with no arguments.
                        }
                        Distance (int ft, float in):feet(ft),inches(in){          //constructor with two
arguments.
                        }
                        void getdist(){
                                cout<<"Enter feets:"<<endl;
                                cin>>feet;
                                cout<<"Enter inches: "<<endl;
                                cin>>inches;
                        }
                        void showdist(){
                                cout<<feet<<"\'-"<<inches<<'\"';
                        }
                        void add_dist(Distance,Distance);
```

```
};
void Distance::add_dist(Distance d2,Distance d3){
        inches=d2.inches+d3.inches;
        feet=0;
        if (inches>=12.0){
        inches -=12.0;
        feet++;
        }
        feet +=d2.feet+d3.feet;
}
int main (){
Distance dist1,dist3;
Distance dist2(11,6.25);
dist1.getdist();
dist3.add_dist(dist1,dist2);
cout<<"Distance 1 is :";
dist1.showdist();
cout<<endl;
cout<<"Distance 2 is :";
dist2.showdist();
cout<<endl;
cout<<"Distance 3 is: "<<endl;
dist3.showdist();
return 0;
}
```

//CONST MEMBER FUNCTIONS.

```
#include <iostream>
#include <string.h>
using namespace std;
class Distance{
        private:
                int feet;
                float inches;
                public:
                        Distance ():feet(0),inches(0.0){        //no argument constructor
                        }
                        Distance (int ft,float in):feet(ft),inches(in){      //two argument constructor.
                        }
                        void getdist(){
                                cout<<"Enter feet:"<<endl;
                                cin>>feet;
                                cout<<"Enter inches:"<<endl;
                                cin>>inches;
                        }
                                void showdist()const{
                        cout<<feet<<"\'-"<<inches<<'\"';
                                }
                                Distance add_dist(const Distance&)const;
                        };
                        Distance Distance::add_dist(const Distance& d2)const {
                        Distance temp;
                        temp.inches=inches+d2.inches;
```

```
                        if (temp.inches>=12.0){
                                temp.inches -=12.0;
                                temp.feet=1;
                        }
                        temp.feet +=feet+d2.feet;
                        return temp;
                        }
int main(){
Distance dist1,dist3;
Distance dist2(11,6.25);

dist1.getdist();
dist3= dist1.add_dist(dist2);

dist1.showdist();
dist2.showdist();
dist3.showdist();
        return 0;
}
```

//UNARY OPERATOR.

//UNARY OPERATOR OVERLOADING.

//PREFIX AND POSTFIX NOTATION.

//NAMELESS TEMPORARY OBJECTS.

```
#include <iostream>
using namespace std;
class Counter{
```

```
        private:
                int count;
                public:
                        Counter(): count(0){      }    //no arg constructor.
                        Counter(int c): count(c){}    //one arg constructor.
                        int getcount(){
                                return count;
                        }
                Counter operator ++(){
                        ++count;
                        return Counter(count);
                }
                Counter operator ++(int){
                        count++;
                        return Counter(count);
                }
                Counter operator --(int){
                        count--;
                        return Counter(count);
                }
                Counter operator --(){
                        --count;
                        return Counter(count);
                }
};
```

```
int main (){
Counter c1;
cout<<"Result before pre increment is:"<<c1.getcount()<<endl;
++c1;
++c1;
cout<<"Result after pre increment is:"<<c1.getcount()<<endl;
c1++;
c1++;
c1++;
cout<<"Result after post increment is:"<<c1.getcount()<<endl;
c1--;
c1--;
cout<<"Result after post decrement is:"<<c1.getcount()<<endl;
--c1;
--c1;
cout<<"Result after pre decrement is:"<<c1.getcount()<<endl;
        return 0;
}
```

//Unary operator.

(YOUTUBE EXAMPLE).

```
#include <iostream>
#include <string.h>
using namespace std;
class Weight{
        private:
        int kg;
```

```
        public:
                Weight():kg(0){    //no arguments constructor.
                }
        Weight(int w):kg(w){       //two arguments constructor.
                }
                void showWeight(){
                        cout<<"Weight in kg is:"<<kg<<endl;
                }
                void operator ++(){
                        ++kg;
                }
                        void operator --(){
                        --kg;
                }
                        void operator ++(int){
                        kg++;
                }
                        void operator --(int){
                        kg--;
                }
};

int main (){
Weight w1;
w1.showWeight();
++w1;
```

```
w1.showWeight();
w1++;      //w1.operator ++();
w1.showWeight();

--w1;
w1.showWeight();
w1--;
w1.showWeight();
return 0;
}
```

//Unary operator Overloading.

//OPERATOR RETURN VALUES.

(YOUTUBE EXAMPLE).

```
#include <iostream>
#include <string.h>
using namespace std;
class Weight{
        private:
        int kg;
        public:
                Weight():kg(0){    //no arguments constructor.
                }
        Weight(int w):kg(w){       //two arguments constructor.
                }
                void showWeight(){
                        cout<<"Weight in kg is:"<<kg<<endl;
```

```
        }
        Weight operator ++(){
                Weight temp;
                temp.kg= ++kg;
                return temp;
        }
                void operator --(){
                --kg;
        }
                void operator ++(int){
                kg++;
        }
                void operator --(int){
                kg--;
        }
};


int main (){
Weight w1,w2;
w2= ++w1;
w2.showWeight();
return 0;
}
```

//CONCATENATING STRINGS.

//USING STRING.

```
#include <iostream>
```

```
#include <string.h>
#include <stdlib.h>
using namespace std;
class String {
	private:
		string str;
		public:
		String():str(""){
		}
		String (string s):str(s){
		}
		void getstr(){
			cout<<"Enter string:"<<endl;
			getline(cin,str);
		}
			void showstr()const{
			cout<<"string is:"<<str<<endl;
		}
		String operator +(const String&S)const{
		return String(str+S.str);
		}
			String operator +=(const String&S){
			str+= S.str;
			return String(str);
		}
			bool operator >(const String&S)const{
```

```
                return str>S.str?true:false;
        }
        bool operator <(const String&S)const{
                return str<S.str?true:false;
        }
};

int main (){
String s1, s2("Akber");
        s1.getstr();

        String s3 =     s1 + s2;
        cout<<"S3 ";
        s3.showstr();

        String s4 = s1 += s3;

        s1.showstr();
        s4.showstr();
        if(s1 > s4)
                cout<<"S1 is largest string"<<endl;
        else if (s1 < s4)
                cout<<"S2 is largest string"<<endl;
        else
                cout<<"Both are equals"<<endl;
return 0;
```

```
}
```

//CONCATENATING STRINGS.

//USING CSTRING.

```
#include <iostream>
#include <string.h>
#include <stdlib.h>
using namespace std;
const int size=50;
class String {
        private:
                char str[50];
                public:
                String (){
                        strcpy(str,"");
                }
                String (char s[]){
                        strcpy(str,s);
                }
                void getstr(){
                        cout<<"Enter string:"<<endl;
                        cin.getline(str,50);
                }
                        void showstr()const{
                        cout<<"string is:"<<str<<endl;
                }
                String operator +(const String&S)const{
```

```
        String temp;
if (strlen(str)+strlen(S.str)>=size-1){
        cout<<"String length is overflow:"<<endl;
        exit(1);
}
strcpy(temp.str,str);
strcat(temp.str,S.str);
return temp;
}
        String operator +=(const String&S){
        if (strlen(str)+strlen(S.str)>=size-1){
        cout<<"String length is overflow:"<<endl;
        exit(1);
}
strcat(str,S.str);
return String(str);
}
        bool operator >(const String&S)const{
                return strcmp(str,S.str)==1?true:false;
        }
        bool operator <(const String&S)const{
                return strcmp(str,S.str)==-1?true:false;
        }
        bool operator ==(const String&S)const{
                return strcmp(str,S.str)==-0?true:false;
        }
```

```
};


int main (){
String s1, s2("Akber");
        s1.getstr();


        String s3 =     s1 + s2;
        cout<<"S3 ";
        s3.showstr();


        String s4 = s1 += s3;


        s1.showstr();
        s4.showstr();
        if(s1 > s4)
                cout<<"S1 is largest string"<<endl;
        else if (s1 < s4)
                cout<<"S2 is largest string"<<endl;
        else
                cout<<"Both are equals"<<endl;
return 0;
}
```

//CONVERSION BETWEEN OBJECTS AND BASIC TYPES.

```
#include <iostream>
#include <stdlib.h>
using namespace std;
```

```
class Distance{
        private:
                const float MTF;    //meters to feet.
                int feet; float inches;
                public:
                        Distance ():feet(0),inches(0),MTF(3.280833F){    }    //constructor with no
arguments.
                        Distance (float meters):MTF(3.280833F)                  //convert meters to distance.
one argument constructor .
                        {
                        float f1tfeet=MTF*meters;    //convert to float feet.
                        feet=int(f1tfeet);                  //feet is integer part.
                        inches=12*(f1tfeet-feet);           //inches is what's left.
                        }
                Distance (int ft,float in)  : feet(ft),inches(in),MTF(3.280833){        }    //two arguments
constructor.
                void getdist()
                {
                cout<<"Enter feet:"       <<endl; cin>>feet;
                cout<<"Enter inches:"<<endl; cin>>inches;         }
                void showdist()const{
                cout<<feet<<"\'-"<<inches<<'\" ';}
                operator float()const          //conversion operator.
        {
                float fracfeet=inches/12;    //convert to inches.
                fracfeet +=static_cast<float>(feet); //add the feet.
                return fracfeet/MTF;        //convert to meters.
```

```
                }       };
int main (){
float mtrs;
Distance dist1=2.35F;    //uses one argument constructor to convert meters to distance.
cout<<"\ndist1="; dist1.showdist();
mtrs= static_cast<float>(dist1);    //uses conversion operator for distance to meters.
cout<<"\n dist1="<<mtrs<<"meters\n";

Distance dist2(5,10.25);        //uses two argument constructor.
mtrs=dist2;                             //also uses conversion operator.
cout<<"\n dist2="<<mtrs<<"meters\n";
return 0;
}
```

//CONVERSION BETWEEN CSTRING AND STRING OBJECTS.

```
#include <iostream>
#include <string.h>
#include <stdlib.h>
using namespace std;
class String{
        private:
                char cstr[50];
                public:
                String (){
        cstr[0]='0';        }       //no argument constructor.
        String (char s[]){
                strcpy(cstr,s);
```

```
        }                               //one arg constructor.
        void display()const{
        cout<<"string is :"<<cstr<<endl;}      //display.
        operator char*(){
                return cstr;
        }                               //conversion operator.
};
int main (){
String s1;
char xstr[]="ali akber";
s1=xstr;                    //use one arg constructor to convert cstring to string.
s1.display();

String s2="ahmed ali";
cout<<static_cast<char*>(s2);      //use conversion operator to convert string to cstring.
cout<<endl;
return 0;
}
```

//ROUTINE IN SOURCE OBJECT.

```
#include <iostream>
#include <string.h>
#include <stdlib.h>
using namespace std;
class time12{
        private:
                bool pm;
```

```
        int hrs;
        int mins;
        public:
        time12():pm(true),hrs(0),mins(0){}      //no arg constructor.
        time12(bool ap,int h,int m){
                pm=ap;          hrs=h;          mins=m;         }
//3 arg constructor.
                void display()const{
                cout<<hrs<<":";
                if (mins<10)
                cout<<'0';    //extra 0 for 01.
                cout<<mins<<' ';
                string am_pm=pm? "p.m":"a.m";
                cout<<am_pm;
                }
};
class time24{
    private:
        int hours;
        int minutes;
          int seconds;
          public:
                time24(){
                        hours=0;        minutes=0;      seconds=0;
}    //no arg constructor.
                        time24(int h,int m,int s){      hours=h;
    minutes=m;      seconds=s;      } //3 arg constructor.
```

```
                                        void display()const{
                                          if (hours<10)    cout<<'0';
                                          cout<<hours<<":";
                                          if (minutes<10)    cout<<'0';
                                          cout<<minutes<<":";
                                          if (seconds<10)    cout<<'0';
                                          cout<<seconds;}
                                          operator time12()const;
};
time24::operator time12()const{
int hrs24=hours;
bool pm=hours<12?false:true;
int roundMins=seconds<30?minutes:minutes+1;
if (roundMins==60){
        roundMins=0;
        ++hrs24;
        if (hrs24==12 || hrs24==24)
        pm=(pm==true)? false:true;
}
int hrs12 =(hrs24<13)?hrs24:hrs24-12;
if (hrs12==0){
        hrs12=12; pm=false;
}
return time12(pm,hrs12,roundMins);
}
int main (){
```

```
int h,m,s;
while (true){
        cout<<"Enter 24-hours time:"<<endl;
        cout<<"Hours 0-23:"<<endl;
        cin>>h;
        if (h>23)
        return (1);
        cout<<"Minutes:";
        cin>>m;
        cout<<"Seconds:";
        cin>>s;
time24 t24(h,m,s);
cout<<"You entered:"<<endl;
t24.display();
time12 t12=t24;
cout<<"\12 hours time:"          ;
t12.display();
cout<<endl;
}
return 0;
}
```

//ROUTINE IN DESTINATION OBJECT.

```
#include <iostream>
#include <string.h>
#include <stdlib.h>
using namespace std;
```

```
class time24{
        private:
                int hours;
                int minutes;
                  int seconds;
                  public:
                        time24(){
                                hours=0;                minutes=0;              seconds=0;
}    //no arg constructor.
                                time24(int h,int m,int s){          hours=h;
        minutes=m;      seconds=s;          } //3 arg constructor.
                                void display()const{
                                  if (hours<10)    cout<<'0';
                                  cout<<hours<<":";
                                  if (minutes<10)    cout<<'0';
                                  cout<<minutes<<":";
                                  if (seconds<10)    cout<<'0';
                                  cout<<seconds;}
                                  int gethrs()const{
                                  return hours;}
                                  int getmins()const{
                                  return minutes;}
                                  int getseconds()const{
                                  return seconds;}
};
class time12{
        private:
```

```
                bool pm;
                int hrs;
                int mins;
                public:
                time12():pm(true),hrs(0),mins(0){}      //no arg constructor.
                time12(bool ap,int h,int m){
                        pm=ap;                  hrs=h;                  mins=m;                 }
//3 arg constructor.
                        void display()const{
                        cout<<hrs<<":";
                        if (mins<10)
                        cout<<'0';    //extra 0 for 01.
                        cout<<mins<<' ';
                        string am_pm=pm? "p.m":"a.m";
                        cout<<am_pm;
                        }
                        time12( time24 t24 );
};


time12::time12( time24 t24 ) {
  int hrs24=t24.gethrs();
    pm=t24.gethrs()<12?false:true;
    mins=(t24.gethrs()<30)?
    t24.getmins():t24.getmins()+1;
    if (mins==60) {
            mins=0;
```

```
        ++hrs24;
        if (hrs24==12 || hrs24==24)
        pm=(pm==true)?false:true;
    }
    hrs=(hrs24<13)?hrs24:hrs24-12;
    if (hrs==0){
        hrs=12; pm=false; }
}
int main (){
int h,m,s;
while (true){
        cout<<"Enter 24-hours time:"<<endl;
        cout<<"Hours 0-23:"<<endl;
        cin>>h;
        if (h>23)
        return (1);
        cout<<"Minutes:";
        cin>>m;
        cout<<"Seconds:";
        cin>>s;
time24 t24(h,m,s);
cout<<"You entered:"<<endl;
t24.display();
time12 t12=t24;
cout<<"\12 hours time:"          ;
t12.display();
```

```
cout<<endl;
}
return 0;
}
```

//DERIVED CLASS AND BASE CLASS.

```
#include <iostream>
using namespace std;
class Counter{
        protected:
                int count;
                public:
                        Counter():count(0){     }   //no arg constructor.
                        Counter (int c):count(c){}   // one arg constructor.
                        int get_count()const{
                        return count;}\
                        Counter operator ++(){
                                return Counter(++count);        }
};
class Dcounter : public Counter{
        public:
                Counter operator --(){
                        return Counter(--count);                }
};
int main (){
        Dcounter c1;
        cout<<"Ist result is:"<<c1.get_count();
```

```
	cout<<endl;
	++c1;++c1;++c1;
	cout<<"2nd result is:"<<c1.get_count();
	cout<<endl;
	--c1;--c1;
	cout<<"3rd result is:"<<c1.get_count();
	cout<<endl;
	return 0;
}
```

//DERIVED CLASS CONSTRUCTOR.

```
#include <iostream>
using namespace std;
class Counter{
	protected:
		int count;
		public:
			Counter():count(0){	}	//no arg constructor.
			Counter (int c):count(c){}	// one arg constructor.
			int get_count()const{
			return count;}
			Counter operator ++(){
				return Counter(++count);	}
};
class Dcounter : public Counter{
	public:
			Dcounter():Counter(){	}	//no arg constructor.
```

```
        Dcounter (int c):Counter(c){       }    // one arg constructor.
        Dcounter operator --(){
                return Dcounter(--count);                    }
};
int main (){
    Dcounter c1;
    Dcounter c2(100);
    cout<<"Ist result of c1 is:"<<c1.get_count();
    cout<<endl;
        cout<<"Ist result of c2 is:"<<c2.get_count();
    cout<<endl;
    ++c1;++c1;++c1;
    cout<<"2nd result of c1 is:"<<c1.get_count();
    cout<<endl;
    --c2;--c2;
    cout<<"2nd result of c2 is:"<<c2.get_count();
    cout<<endl;
    Dcounter c3 = --c2;
    cout<<"Ist result of c3 is:"<<c3.get_count();
    cout<<endl;
    return 0;
}
```

//DERIVED CLASS CONSTRUCTOR.

```
#include <iostream>
#include<string.h>
using namespace std;
```

```
class Person{
        private:
                int id;
                char name[50];
                public:
                        Person(){
                        id=0;
                        strcpy(name,"");          }      //no argument constructor.
                        Person(int i,char na[]){
                                id=i;
                                strcpy(name,na);
                        }                                          //two arg constructor.
                        void showdata1(){
                                cout<<"Id is :"<<id<<endl;
                                        cout<<"Name is :"<<name<<endl;
                        }
                        void getdata1(){
                                cout<<"Enter id of the person:"<<endl;
                                cin>>id;
                                cout<<"Enter name of the person:"<<endl;
                                cin>>name;
                        }
};
class Student:public Person{
        private:
                float gpa;
```

```
        public:
        Student():Person(),gpa(0){          }
        Student(int i,char na[],float gp):Person(i,na),gpa(gp){      }
        void getdata2(){
                cout<<"Enter gpa of the student:"<<endl;
                cin>>gpa;
        }
        void showdata2(){
                cout<<"Gpa of the student is:"<<gpa<<endl;
        }
};
int main (){
Student s1;
s1.getdata1();
s1.getdata2();

cout<<endl;
s1.showdata1();
s1.showdata2();
    return 0;
}
```