TRIBHUVAN UNIVERSITY

INSTITUTE OF ENGINEERING

PURWANCHAL CAMPUS

DHARAN

LABSHEET 6 : [TYPE CONVERSION AND INHERITANCE]

BY

Rabin Poudel [PUR079BCT056]

DEPARTMENT OF COMPUTER & ELECTRONICS ENGINEERING

PURWANCHAL CAMPUS

DHARAN, NEPAL

JULY, 2024

0.1 Question 1 :

Write a program that can convert the Distance (meter, centimeter) to meters measurement in float and vice versa. Make a class distance with two data members, meter and centimeter. You can add function members as per your requirement.

Code:

```
#include <iostream>
using namespace std;
class Distance{
   int meter;
   int centimeter;
public:
   Distance(int m,int cm):meter(m),centimeter(cm){};
   Distance(float no)
   {
      meter=no;
      centimeter=(no−int(no))    *100;
   }
   operator float()
   {
      return float(meter) + float(centimeter)/100.0;
   }
   Distance operator+(Distance a)
   {
      return
           Distance(meter+a.meter+(centimeter+a.centimeter)/100,(centimeter+a.centimeter)%10
   }
   void display()
   {
      cout << meter << " m " << centimeter << "cm" << endl;
   }
};
int main()
{
   Distance d(10.34);
   d.display();
   cout << "Distance typecasting to float ";
```

```
    cout << d << endl;
    Distance c(10,6);
    Distance s = d+c;
    cout << "addition of two distances" << endl;
    d.display();
    c.display();
    cout << "Gives";
    s.display();
    return 0;
}
```

0.2 Question 2:

Write two classes to store distances in meter-centimeter and feet-inch system respectively. Write conversions functions so that the program can convert objects of both types.

Code:

```
#include <iostream>
using namespace std;
class Feet
{
private:
   int feet;
   int inches;
public:
   Feet(int f, int i):feet(f),inches(i){};
   Feet(float no)
   {
      feet = int (no);
      inches = int( (no − int (no))          *  12 );
   }
   float tofloat()
   {
      return float(feet + float(inches) / 12);
   }
   void display()
   {
      cout << feet << " f " << inches << " in " << endl;
   }

};
class Metric{
private:
   int meter;
   int centimeter;
public:
   Metric(int m,int cm):meter(m),centimeter(cm){};
   Metric(float no)
```

```
    {
        meter = int (no);
        centimeter = int( ( no − int(no) )              *  100) ;
    }
    Metric(Feet f)
    {
        Metric(f.tofloat()          *  3.28);
    }
    void display()
    {
        cout << meter << " m " << centimeter << " cm " << endl;
    }
    float tofloat()
    {
        return float(meter + float(centimeter)/100.0);
    }
    operator Feet()
    {
        return Feet(tofloat() / 3.28);
    }
};
int main()
{
    cout << "Distance in Metric" << endl;
    Metric d(10.34);
    d.display();
    cout << "Converting into Feet" << endl;
    Feet(d).display();
    cout << "Distance in Feet" << endl;
    Feet f(20.34);
    f.display();
    cout << "Converting into Metric" << endl;
    Metric(f).display();
    return 1;
}
```

0.3 Question 3:

Create a class called Musicians to contain three methods string(), wind(), and perc(). Each of these methods should initialize a string array to contain the following instruments:

- veena, guitar, sitar, sarod and mandolin under string()
- flute, clarinet saxophone, nadhaswaram, and piccolo under wind()
- tabla, mridangam, bangos, drums and tambour under perc()

It should also display the contents of the arrays that are initialized. Create a derived class called TypeIns to contain a method called get() and show(). The get() method must display a menu as follows:

Type of instruments to be displayed

a. String instruments

b. Wind instruments

c. Percussion instruments

The show() method should display the relevant detail according to our choice. The base class variables must be accessible only to their derived classes.

Code:

```
#include <iostream>
#include <cstring>
#define SUCCESS 0
class Musicians
{
 protected:
   std::string str[5];
   std::string wnd[5];
   std::string per[5];
 public:
   void string()
   {
```

```
    str[0] ="vern";
    str[1]="guitar";
    str[2]="sitar";
    str[3]="sarod";
    str[4]="mandolin";
    std::cout<< "string instrument has been initialized to" <<
         std::endl;
    for(int i = 0 ; i < 5; i++)
    {
       std::cout << str[i] << std::endl;
    }
}
void wind(){

    wnd[0]="flute";
    wnd[1]="mridangam";
    wnd[2]="bangos";
    wnd[3]="drums";
    wnd[4]="tambour";
    std::cout<< "wind instrument has been initialized to" <<
         std::endl;
    for(int i = 0 ; i < 5; i++)
    {
       std::cout << wnd[i] << std::endl;
    }
}
void perc()
{
    per[0]="tabla";
    per[1]="mridangam";
    per[2]="bangos";
    per[3]="drums";
    per[4]="tambour";
    std::cout<< "percussion instrument has been initialized to" <<
         std::endl;
    for(int i = 0 ; i < 5; i++)
    {
       std::cout << per[i] << std::endl;
    }
```

```
    }
};
class TypeIns:public Musicians
{
  public:
    std::string     *  get()
    {
       std::cout<<"Type of instrument to be displayed" << std::endl;
       std::cout << "a. String instruments" << std::endl;
       std::cout << "b. Wind instruments" << std::endl;
       std::cout << "c. Percussion instruments" << std::endl;
       char choice;
       std::cin >> choice;
       show(choice);
    }
    void show(char choice)
    {
       if(choice == 'a')
       {
          for(int i = 0 ; i < 5; i++)
          {
             std::cout << str[i] << std::endl;
          }
       }
       else if(choice == 'b')
       {
          for(int i = 0 ; i < 5; i++)
          {
             std::cout << wnd[i] << std::endl;
          }
       }
       else
       {
          for(int i = 0 ; i < 5; i++)
          {
             std::cout << per[i] << std::endl;
          }
       }
    }
```

```
};
int main()
{
   TypeIns t;
   t.string();
   t.perc();
   t.wind();
   t.get();
   return SUCCESS;
}
```

0.4 Question 4:

Write three derived classes inheriting functionality of base class person (should have a member function that asks to enter name and age) and with added unique features of student, and employee, and functionality to assign, change and delete records of student and employee. And make one member function for printing address of the objects of classes (base and derived) using this pointer. Create two objects of base class and derived classes each and print the addresses of individual objects.Using calculator, calculate the address space occupied by each object and verify this with address spaces printed by the program.

Code:

```
#include<iostream>
using namespace std;
class Person
{
	string name;
	int age;
public:
	void display()
	{
		cout<<"The address of the object in person class is:
			"<<this<<endl;
		cout<<"The size of the object in person class is:
			"<<sizeof(this)<<endl;
	}
};
class Student:public Person
{
	int roll_no;
	int grade;
public:
	void display()
	{
		cout<<"The address of the object in student class is:
			"<<this<<endl;
		cout<<"The size of the object in student class is:
```

```
                "<<sizeof(this)<<endl;
        }
};
class Employee:public Person
{
        int employee_id;
        float salary;
public:
        void display()
        {
                cout<<"The address of the object in employee class is:
                    "<<this<<endl;
                cout<<"The size of the object in employee class is:
                    "<<sizeof(this)<<endl;
        }
};
int main()
{
        Person pobj1,pobj2;
        Student sobj1,sobj2;
        Employee eobj1,eobj2;
        pobj1.display();
        pobj2.display();
        sobj1.display();
        sobj2.display();
        eobj1.display();
        eobj2.display();
        return 0;
}
```

0.5 Question 5:

Write base class that ask the user to enter a complex number and make a derived class that adds the complex number of its own with the base. Finally make third class that is friend of derived and calculate the difference of base complex number and its own complex number.

Code:

```
#include <iostream>
using namespace std;
class Complex
{
 public:
   int real, img;
   Complex()
   {
      cout << "Enter real part";
      cin >> real;
      cout << "Enter imaginary part";
      cin >> img;
   }
   void display ()
   {
      cout << real << "+i" << img;
   }
};

class ComplexMath:public Complex
{
 private:
 public:
   ComplexMath(){
   }

   ComplexMath& operator+(const Complex a)
   {
      real += a.real;
      img += a.img;
```

```
        return   *this;
    }
};

int main()
{
    Complex a;
    ComplexMath c;
    a.display();
    std::cout << "+";
    c.display();
    cout << "=";
    (c+a).display();
    return 0;
}
```