LAB SHEET - 10

[Type conversion]

Date: 2023/01/26

#*Index Page:-*

#Topics: #Page No.

1. Title:

Type conversion

2. Objective:

- **To be familiar with type conversion**

3. Theory:

Introduction to type conversion

Type conversion or typecasting refers to **changing an entity of one datatype into another**. There are two types of conversion: implicit and explicit. The term for implicit type conversion is coercion. Explicit type conversion in some specific way is known as casting.

- ***Basic to class type***

  This constructor builds a string type object from a char* type variable a. The variables length and name are data members of the class string. Once you define the constructor in the class string, it can be used for conversion from char* type to string type.

  Syntax:

- ***Class to basic type***

  C++ allows us to define a overloaded casting operator that convert a class type data to basic type. The general form of an overloaded casting operator function, also referred to as a conversion function, is:

  Syntax:

  operator typename( ) { //Program statement. }

- ***Class to class type***

  Through class conversion, one can assign data that belongs to a particular class type to an object that belongs to another class type. where '=' has been overloaded for objects of class type 'B'. Class conversion can be achieved by conversion function which is done by the use of operator overloading.

  Syntax:

4. Lab Exercise:

a. Write a program to read height of person in meter and convert into feet and inches by using suitable type conversion method.

#SourceCode:

```
#include <iostream>
using namespace std;
class Height{
   private:
      short int ft,in;
   public:
      Height(float m){
         ft=(int)m*3.281;
         float t=(m*3.281)-ft;
         cout<<ft<<"\t"<<t<<endl;
         in=(int)(t*12);
      }
      void display(void){
         cout<<"Feet: "<<ft<<", inches: "<<in<<endl;
      }
};
int main()
{
   Height H1(5);
   H1.display();
   return 0;
}
```

#Output:

```
16      0.405
Feet: 16, inches: 4

Process returned 0 (0x0)   execution time : 0.347 s
Press any key to continue.
```

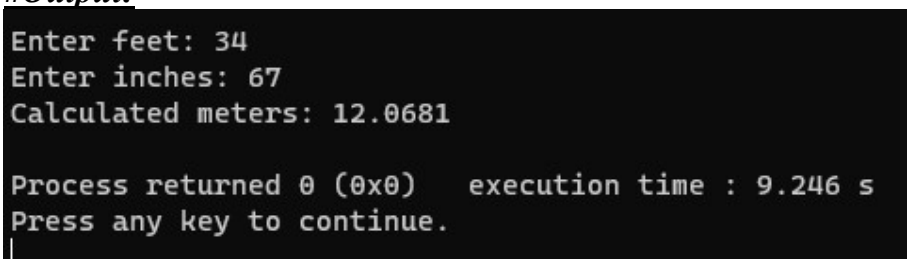
b. Write a program to read height of person in feet and inches and convert into meter by using suitable type conversion method.

#SourceCode:

```
#include <iostream>
using namespace std;
class Meter{
   private:
      float m;
   public:
```

```
        Meter(int ft, int in){
            float temp=(float)(ft+(in/12.0));
            m=temp/3.28;
        }
        void display(void){
            cout<<"Calculated meters: "<<m<<endl;
        }
};
int main(){
    cout<<"Enter feet: ";
    short int in,ft;
    cin>>ft;
    cout<<"Enter inches: ";
    cin>>in;
    Meter M1(ft,in);
    M1.display();
    return 0;
}
```

#Output:



c. ***Make a class called memory with member data to represent bytes, kilobytes and megabytes .Read the value of memory in bytes from the user as basic types and display the result in user defined memory type. Like for m (basic type) = 108766, your program should display as: 1 megabyte 38 kilobytes 177 bytes. [Hint: Use basic to user defined (basic-to-class) conversion method].***

#SourceCode:

```
#include <iostream>
using namespace std;
class Memory{
    private:
        short int mb,kb,b;
    public:
        Memory(int r){
            kb=r/1024; //converting bytes to kilobytes
            mb=kb/1024; //converting  kilobytes to megabytes
            kb=kb-(mb*1024);
            b=r-(kb*1024);
```

```
    }
    void display(void){
       cout<<"Calculated memory: "<<endl;
       cout<<mb<<" megabyte "<<kb<<" kilobytes "<<b<<" bytes"<<endl;
    }
};
int main(){
   cout<<"Enter bytes: ";
   unsigned read;
   cin>>read;
   Memory M1(read);
   M1.display();
   return 0;
}
```

#Output:

```
Enter bytes: 425618881
Calculated memory:
21 megabyte 923 kilobytes 449 bytes

Process returned 0 (0x0)   execution time : 4.046 s
Press any key to continue.
```

d. Write a program to convert total memory in bytes into mb, kb and bytes using type conversion. (1kb=1024bytes,1mb=1024kb)

#SourceCode:

```
#include <iostream>
using namespace std;
class Memory{
   private:
      unsigned byte;
   public:
      Memory(int mb,int kb,int b){
         kb+=mb*1024;
         b+=kb*1024;
         byte=b;
      }
      void display(void){
         cout<<"Calculated memory: "<<endl;
         cout<<"Total bytes:"<<byte <<endl;
      }
};
int main(){
   cout<<"Enter MB: ";
   int mb,kb,b;
```

```
    cin>>mb;
    cout<<"Enter KB: ";
    cin>>kb;
    cout<<"Enter bytes: ";
    cin>>b;
    Memory M1(mb,kb,b);
    M1.display();
    return 0;
}
```

#Output:

```
Enter MB: 34
Enter KB: 6432
Enter bytes: 643792
Calculated memory:
Total bytes:42881744

Process returned 0 (0x0)   execution time : 25.371 s
Press any key to continue.
```

e. Write a complete program to convert the polar coordinates into rectangular coordinates.(hint: polar-coordinates(radius, angle) and rectangular co-ordinates (x,y) where x=r*cos(angle) and y=r*sin(angle)).

#SourceCode:

```
//Here the conversion routine is in the source
#include <iostream>
#include<math.h>
using namespace std;
class Rectangular{
   private:
      float xco, yco;
   public:
      Rectangular(float xco=0,float yco=0){
         this->xco=xco;
         this->yco=yco;
      }
      void display(void){
         cout<<"X co-ordinate: "<<xco<<", Y co-ordinate: "<<yco<<endl;
      }
};
class Polar{
   private:
      float rad, ang;
   public:
```

```
        Polar(float rad=0, float ang=0){
           this->rad=rad;
           this->ang=(ang);
        }
        void display(void){
           cout<<"Radius: "<<rad<<", angle: "<<ang<<endl;
        }
        operator Rectangular(){
           float t1,t2;
           t1=rad*(cos(ang));
           t2=rad*(sin(ang));
           return Rectangular(t1,t2);
        }
};
int main(){
   Polar P(6,56);
   Rectangular R=P;
   R.display();
   return 0;
}
```

#Output:

```
X co-ordinate: 5.11932, Y co-ordinate: -3.12931

Process returned 0 (0x0)   execution time : 0.489 s
Press any key to continue.
```

f. Write a program to convert Rectangle coordinate to Polar coordinate by using:

- ***conversion routine in source class***
- ***conversion routine in destination class***

#SourceCode:

```
//Here the conversion routine is the source class
#include <iostream>
#include<math.h>
using namespace std;
class Rectangular{
   private:
      float xco, yco;
   public:
      Rectangular(float xco=0,float yco=0){
         this->xco=xco;
         this->yco=yco;
      }
      void display(void){
```

```
            cout<<"X co-ordinate: "<<xco<<", Y co-ordinate: "<<yco<<endl;
        }
};
class Polar{
   private:
      float rad, ang;
   public:
      Polar(float rad=0, float ang=0){
         this->rad=rad;
         this->ang=ang;
      }
      void display(void){
         cout<<"Radius: "<<rad<<", angle: "<<ang<<endl;
      }
      operator Rectangular(){
         float t1,t2;
         t1=rad*(cos(ang));
         t2=rad*(sin(ang));
         return Rectangular(t1,t2);
      }
};
int main(){
   Polar P(2,9);
   Rectangular R=P;
   R.display();
   return 0;
}

//here the conversion routine is in the destination class
#include <iostream>
#include<math.h>
using namespace std;
class Rectangular;
class Polar{
   private:
      float rad, ang;
   public:
      Polar(float rad=0, float ang=0){
         this->rad=rad;
         this->ang=ang;
      }
      void display(void){
         cout<<"Radius: "<<rad<<", angle: "<<ang<<endl;
      }
      friend class Rectangular;
};
```

```
class Rectangular{
   private:
      float xco, yco;
   public:
      Rectangular(float xco=0,float yco=0){
         this->xco=xco;
         this->yco=yco;
      }
      Rectangular(Polar P){
         xco=(P.rad)*cos(P.ang);
         yco=(P.rad)*sin(P.ang);
      }
      void display(void){
         cout<<"X co-ordinate: "<<xco<<", Y co-ordinate: "<<yco<<endl;
      }
};
int main(){
   Polar P(7,3);
   Rectangular R(P);
   R.display();
   return 0;
}
```

#Output:

```
X co-ordinate: -1.82226, Y co-ordinate: 0.824237

Process returned 0 (0x0)   execution time : 0.459 s
Press any key to continue.
```

```
X co-ordinate: -6.92995, Y co-ordinate: 0.98784

Process returned 0 (0x0)   execution time : 0.659 s
Press any key to continue.
```

g. Write a program to convert degree Celsius to degree Fahrenheit by using class to class type conversion.

#SourceCode:

```
#include <iostream>
using namespace std;
class Fahrenheit;
class Celcius{
   private:
      float cel;
   public:
      Celcius(int x=0){
```

```
            cel=x;
        }
        void display(void){
            cout<<"Temerature in Celcius: "<<cel<<endl;
        }
        friend class Fahrenheit;
};
class Fahrenheit{
    private:
        float far;
    public:
        Fahrenheit(int t=0){
            far=t;
        }
        Fahrenheit(Celcius D){
            far=((9/5.0)*D.cel)+32;
        }
        void display(void){
            cout<<"Temperature in Fahrenheit: "<<far<<endl;
        }
};
int main(){
    Celcius C1(99);
    Fahrenheit F(C1);
    F.display();
    return 0;
}
```

#Output:

```
Temperature in Fahrenheit: 210.2

Process returned 0 (0x0)   execution time : 0.479 s
Press any key to continue.
```

5. Conclusion:

In this LabSheet, We learned to be familiar with type conversion in C++. We also become familiar with different type conversion and their uses.