

Lab Report on CSE 1204

Submitted by

Name: Md. Shabir Khan Akash.

Class: 1st year, even semester.

Dept.: Department of CSE.

Roll: 1603108

Section: B

Submitted to

Shyla Afroge Madam

Assistant Professor

Department of CSE, RUET

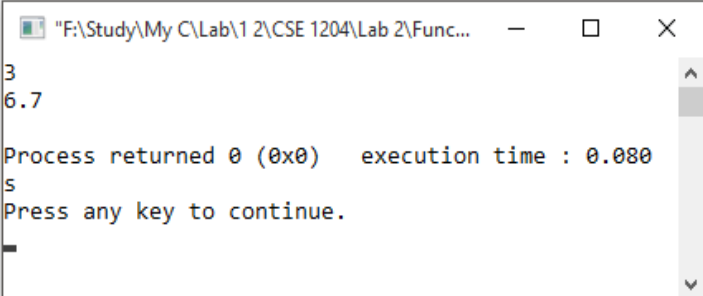
Theory:

One of the most important and pervasive C++ feature is **function overloading**. Not only does function overloading provide the mechanism by which C++ achieves one type of polymorphism, it also forms the basis by which the C++ programming environment can be dynamically extended. Function overloading mainly indicates declaring multiple functions with a same function-name but the data types of the parameters are not same, they are different in each function. Actually when two or more functions share the same function-name, they are said to be **overloaded**. In C++ compiler, it will automatically select the correct version based upon the number or data type of the arguments used to call the function.

Code

```
1  #include<iostream>
2  using namespace std;
3  class test {
4  public:
5      int min(int m, int n)
6      {
7          if(m<n)
8              return m;
9          else
10             return n;
11     }
12     int min(int x, int y, int z)
13     {
14         if(x<y)
15         {
16             if(x<z)
17                 return x;
18             else if (z<y)
19                 return z;
20         }
21         else if (y<z)
22             return y;
23         else
24             return z;
25     }
26     double min(double p, double q)
27     {
28         if (p<q)
29             return p;
30         else
31             return q;
32     }
33 };
34
35 int main()
36 {
37     int s;
38     double p;
39     test ob;
40     s = ob.min(5,10);
41     cout<<s<<endl;
42     s = ob.min(3,6,9);
43     cout<<s<<endl;
44     p = ob.min(12.4,6.7);
45     cout<<p<<endl;
46     return 0;
47 }
48
```

Output



```
"F:\Study\My C\Lab\1 2\CSE 1204\Lab 2\Func...
3
6.7

Process returned 0 (0x0)   execution time : 0.080
s
Press any key to continue.
```

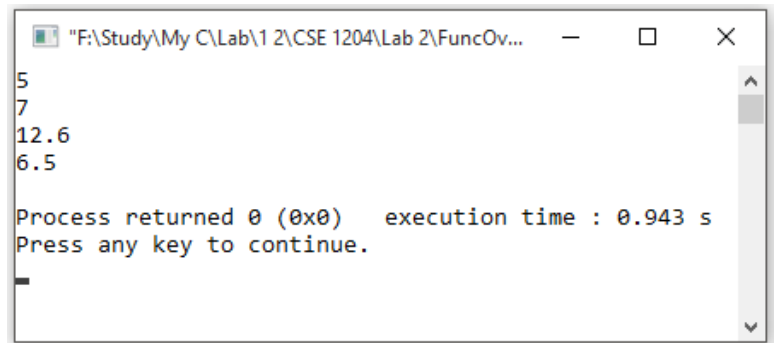
Comment

In this above code, in a class named *test*, three functions named *min* was declared. But these three functions don't have the same return type and the same parameters. One of those three functions were used to find out the minimum number between two integers and the other was used to find out the minimum number among three integer numbers and finally the last one was used to find out the minimum number among three *double* numbers.

Code

```
1  #include<iostream>
2  using namespace std;
3  class MoreTest {
4  public:
5      int abs(int x) {
6          if(x>=0)
7              return x;
8          else
9              return (0-x);
10     }
11     double abs(double z) {
12         if(z>=0)
13             return z;
14         else
15             return (0-z);
16     }
17 };
18
19 int main()
20 {
21     int a;
22     double b;
23     MoreTest ob;
24     a = ob.abs(5);
25     cout<<a<<endl;
26     a = ob.abs(-7);
27     cout<<a<<endl;
28     b = ob.abs(12.6);
29     cout<<b<<endl;
30     b = ob.abs(-6.5);
31     cout<<b<<endl;
32     return 0;
33 }
```

Output



```
"F:\Study\My C\Lab\1 2\CSE 1204\Lab 2\FuncOv...
5
7
12.6
6.5

Process returned 0 (0x0)   execution time : 0.943 s
Press any key to continue.
```

Comment

In this above code, two functions named **abs** was declared in a class named **MoreTest**. But these two functions don't have the same return type and the same parameters. One of those two functions were used to find out the absolute value of an integer (both positive and negative) and the other **abs** function was used to find out the absolute value of a **double** number.

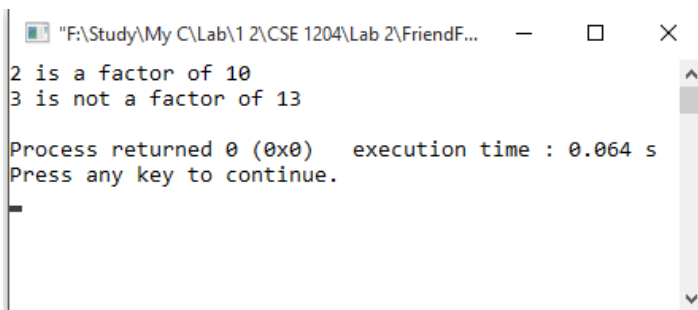
Theory

Friend function is a special feature of C++ language. A **friend** is not a member of a class but still has the access to its private elements. Two reasons that friend functions are useful have to do with operator overloading and the creation of certain types of I/O functions. A friend function is defined as a regular, non-member function. However, inside the class declaration for which it will be friend, its prototype is also included, prefaced by the keyword **friend**.

Code

```
1  #include<iostream>
2  using namespace std;
3  class akash {
4      int a,b;
5  public:
6      akash(int x, int y) {
7          a = x;
8          b = y;
9      }
10     friend int factor(akash ob);
11 };
12
13 int factor (akash ob) {
14     if (!(ob.a % ob.b))
15         return 1;
16     else
17         return 0;
18 }
19
20 int main()
21 {
22     akash ob1(10,2), ob2(13,3);
23     if (factor(ob1))
24         cout<<"2 is a factor of 10"<<endl;
25     else
26         cout<<"2 is not a factor of 10"<<endl;
27     if (factor(ob2))
28         cout<<"3 is a factor of 13"<<endl;
29     else
30         cout<<"3 is not a factor of 13"<<endl;
31     return 0;
32 }
```

Output



```
"F:\Study\My C\Lab\1 2\CSE 1204\Lab 2\FriendF...  -  □  ×
2 is a factor of 10
3 is not a factor of 13

Process returned 0 (0x0)   execution time : 0.064 s
Press any key to continue.
```

Comment

In the above code, two integer a, b was declared in a class named **akash** in private mode and a function named **akash** was also declared where two variables assigning them to the private variable and friend function's prototype was included there. After that a friend function was declared outside that class. And in that function a checking operation was performed whether first number is a factor of second or not. Then in main function it was being printed by declaring two object (ob1, ob2).

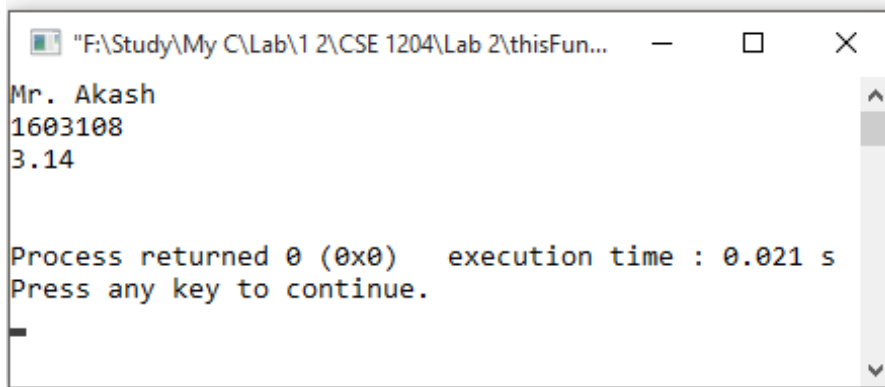
Theory

C++ contains a special pointer that is called **this**. **this** is a pointer that is automatically passed to any member function when it is called and it is a pointer to the object that generates the call. It is noted that only member functions are passed to **this** pointer. The **this** pointer has several uses, including aiding in overloading operators. By default, all member functions are automatically passed a pointer to the invoking object.

Code

```
1  #include<iostream>
2  #include<string.h>
3  using namespace std;
4  class student {
5      char name[50];
6      int roll;
7      double cgpa;
8  public:
9      student(char *p, int i, double d)
10     {
11         strcpy(this->name,p);
12         this->roll = i;
13         this->cgpa = d;
14     }
15     void show()
16     {
17         cout<<this->name<<"\n"<<this->roll<<"\n"<<this->cgpa<<"\n"<<endl;
18     }
19 };
20
21 int main()
22 {
23     student ob("Mr. Akash",1603108,3.14);
24     ob.show();
25
26     return 0;
27 }
```

Output



```
"F:\Study\My C\Lab\1 2\CSE 1204\Lab 2\thisFun...  -  □  X
Mr. Akash
1603108
3.14

Process returned 0 (0x0)   execution time : 0.021 s
Press any key to continue.
_
```

Comment

In this above code, three information i.e. *name*, *roll*, *cgpa* was declared in a class named *student* in private mode. Then in that class a function named *student* was declared in public where three parameters were present and in the body of that function, these three arguments were assigned to the private member using *this* pointer. After that, in main function, a *student* type object was declared where name, roll, cgpa of a student was assigned from the user and then a *show()* function was used to print that information on the console.