MA122 - Computer Programming and Applications

Indian Institute of Space Science and Technology

April 26, 2017

Lecture 33

example 1


```
1  #include<iostream>
2  using namespace std;
3  int main( )
4  {
5      enum CarTyre { normal, wet, snow, sports };
6      int     Number    = 0;
7      bool    IsValid   = false;
8      CarTyre FrontLeft = normal;
9
10
11     //int    Number2    = "A"     //as the character "
           A" is not a valid integer;
12     //bool   IsValid2   = 1.42;   //as 1.42 is a
           floating point value and not a boolean
13     //CarTyre FrontLeft2 = giraffe //as giraffe is not
            a valid CarTyre;
```

example 1


```
1
2       if( Number == 1 )
3           cout<<Number<<endl;
4
5       if( IsValid == false )
6           cout<<"It is valid"<<endl;
7
8       if( FrontLeft == normal )
9           cout<<" It works"<<endl;
10
11
12      return 0;
13  }
```

example 2


```
1  #include<iostream>
2  using namespace std;
3  int main( )
4  {
5      enum Color { red, green, blue };
6      Color r = red;
7      switch(r)
8      {
9          case red : std::cout << "red\n"; break;
10         case green: std::cout << "green\n"; break;
11         case blue : std::cout << "blue\n"; break;
12     }
13
14    return 0;
15 }
```

example 3


```
1  #include<iostream>
2  using namespace std;
3  int main( )
4  {
5
6      enum Foo { a, b, c = 10, d, e = 1, f, g = f + c };
7      //a = 0, b = 1, c = 10, d = 11, e = 1, f = 2, g =
           12
8      Foo r = b;
9      switch(r)
10     {
11         case a : std::cout << a<<endl; break;
12         case b: std::cout << b<<endl; break;
13         case c : std::cout << c<<endl; break;
14     }
15
16    return 0;
17 }
```

Lecture 33

Pointer to a data member


```
1 #include <iostream>
2
3 using namespace std;
4 class Data
5 {
6  public:
7  int a;
8  void print() { cout << "a is="<< a<<endl; }
9 };
```

pointer to a object


```
1  int main()
2  {
3   Data d, *dp;
4   dp = &d;    // pointer to object
5
6   int Data::*ptr=&Data::a; // pointer to data member 'a
        '
7
8   d.*ptr=10;
9   d.print();
10
11  dp->*ptr=20;
12  dp->print();
13 }
```

Lecture 33

Pointer to a data member


```
1  #include <iostream>
2  using namespace std;
3  class Test
4  {
5  private:
6      ~Test() { cout << "Destroying Object\n"; }
7  public:
8       Test() { cout << "Object Created\n"; }
9  friend void destructTest(Test* );
10 };
11
12 // Only this function can destruct objects of Test
13 void destructTest(Test* ptr)
14 {
15     delete ptr;
16     cout << "Object Destroyed\n";}
```

object using new


```
1
2  int main()
3  {
4      /* Uncommenting following following line would
           cause compiler error */
5      // Test t1;
6
7      // create an object
8      Test *ptr = new Test;
9
10     // destruct the object to avoid memory leak
11     destructTest(ptr);
12
13     return 0;
14 }
```

Lecture 33

Type Casts


```
1  #include<iostream>
2  int main()
3  {
4    long count = 8;     // int value 8 converted to type
         long
5    double time = 11;   // int value 11 converted to type
         double
6    int side = 3.33;    // double value 3.33 converted to
         type int 3
7    //int * p = 10; // type clash
8    int *p=(int *) 10;
9    return 0;
10 }
```

Example


```
1  #include <iostream>
2  #include <fstream>
3  using std::cout;
4  class Stonewt
5    {
6    private:
7      enum {Lbs_per_stn = 14};    // pounds per stone
8      //static const int Lbs_per_stn = 14;
9      int stone;                  // whole stones
10     double pds_left;            // fractional pounds
11     double pounds;              // entire weight in
           pounds
```

Example


```
1  public:
2    Stonewt(double lbs);        // constructor for
         double pounds
3    Stonewt(int stn, double lbs); // constructor for
         stone, lbs
4    Stonewt();                  // default constructor
5    ~Stonewt();
6    void show_lbs() const;      // show weight in
         pounds format
7    void show_stn() const;      // show weight in stone
          format
8  };
```

Example


```
1  // construct Stonewt object from double value
2  Stonewt::Stonewt(double lbs)
3  {
4    stone = int (lbs) / Lbs_per_stn; // integer division
5    pds_left = int (lbs) % Lbs_per_stn + lbs - int(lbs);
6    pounds = lbs;
7  }
8  // construct Stonewt object from stone, double values
9  Stonewt::Stonewt(int stn, double lbs)
10 {
11   stone = stn;
12   pds_left = lbs;
13   pounds = stn * Lbs_per_stn +lbs;
14 }
```

Example


```
1  Stonewt::Stonewt()        // default constructor, wt =
       0
2  {
3    stone = pounds = pds_left = 0;
4  }
5  Stonewt::~Stonewt()       // destructor
6  {
7  }
8  // show weight in stones
9  void Stonewt::show_stn() const
10 {
11   cout << stone << " stone, " << pds_left << " pounds\
       n";
12 }
```

Example


```
1  // show weight in pounds
2  void Stonewt::show_lbs() const
3  {
4    cout <<pounds << " pounds\n";
5  }
6  void display(const Stonewt & st, int n);
7  int main()
8  {
9    Stonewt incognito = 275; // uses constructor to
         initialize
10   Stonewt wolfe(285.7);  // same as Stonewt wolfe =
         285.7;
11   Stonewt taft(21, 8);
12   cout << "The celebrity weighed ";
13   incognito.show_stn();
```

Example


```
1  cout << "The detective weighed ";
2  wolfe.show_stn();
3  cout << "The President weighed ";
4  taft.show_lbs();
5  incognito = 276.8;    // uses constructor for
       conversion
6  taft = 325;           // same as taft = Stonewt(325)
       ;
7  cout << "After dinner, the celebrity weighed ";
8  incognito.show_stn();
9  cout << "After dinner, the President weighed ";
10 taft.show_lbs();
```

Example


```
1   display(taft, 2);
2   cout << "Thewrestler weighed even more.\n";
3   display(422, 2);
4   cout << "No stone left unearned\n";
5   return 0;
6 }
7 void display(const Stonewt & st, int n)
8 {
9   for (int i = 0; i < n; i++)
10  {
11    cout << "Wow! ";
12    st.show_stn();
13  }
14 }
```

Lecture 33

enumeration

Pointer to a Data Member

Object using new

Automatic Conversions and Type Casts

overloading << operator

overloading << operator: General Method

Allowed and Not Allowed

friend2


```
1  #include <iostream>
2  class Time
3    {
4    private:
5      int hours;
6      int minutes;
7    public:
8      Time();
9      Time(int h, int m = 0);
10     Time operator*(double n) const;
11     friend Time operator*(double m, const Time & t);
12     friend void operator<<(std::ostream & os, const
           Time & t);
13   };
```

friend2


```
1  Time::Time()
2  {
3    hours = minutes = 0;
4  }
5  Time::Time(int h, int m )
6  {
7    hours = h;
8    minutes = m;
9  }
10 Time Time::operator*(double mult) const
11 {
12   Time result;
13   long totalminutes = hours * mult * 60 + minutes *
         mult;
14   result.hours = totalminutes / 60;
15   result.minutes = totalminutes % 60;
16   return result;
17 }
```

friend2


```
1
2  Time operator*(double m, const Time & t)
3  {
4    return t * m;
5  }
6
7  void operator<<(std::ostream & os, const Time & t)
8  {
9    os << t.hours << " hours, " << t.minutes << "
         minutes"<<std::endl;
10 }
```

friend2


```
1  int main()
2  {
3    using std::cout;
4    using std::endl;
5    Time weeding(4, 35);
6
7
8    Time adjusted;
9    //adjusted = weeding * 1.5;
10   adjusted = 1.5*weeding;
11   cout << "adjusted work time = ";
12   cout << adjusted;
13   return 0;
14 }
```

Lecture 33

enumeration

Pointer to a Data Member

Object using new

Automatic Conversions and Type Casts

overloading << operator

overloading << operator: General Method

Allowed and Not Allowed

friend3


```
1  #include <iostream>
2  class Time
3    {
4    private:
5      int hours;
6      int minutes;
7    public:
8      Time();
9      Time(int h, int m = 0);
10     Time operator*(double n) const;
11     friend Time operator*(double m, const Time & t);
12     friend std::ostream & operator<<(std::ostream & os
           , const Time & t);
13
14   };
```

friend3


```
1  Time::Time()
2  {
3    hours = minutes = 0;
4  }
5  Time::Time(int h, int m )
6  {
7    hours = h;
8    minutes = m;
9  }
10 Time Time::operator*(double mult) const
11 {
12   Time result;
13   long totalminutes = hours * mult * 60 + minutes *
         mult;
14   result.hours = totalminutes / 60;
15   result.minutes = totalminutes % 60;
16   return result;
17 }
```

friend3


```
1
2  Time operator*(double m, const Time & t)
3  {
4    return t * m;
5  }
6
7  std::ostream & operator<<(std::ostream & os, const
       Time & t)
8  {
9    os << t.hours << " hours, " << t.minutes << "
         minutes";
10   return os;
11 }
```

friend3


```
1  int main()
2  {
3    using std::cout;
4    using std::endl;
5    Time weeding(4, 35);
6
7
8    Time adjusted;
9    //adjusted = weeding * 1.5;
10   adjusted = 1.5*weeding;
11   cout <<"Hello"<<endl<< adjusted<<endl;
12   return 0;
13 }
```

Lecture 33

Table 11.1 **Operators That Can Be Overloaded**

+	-	*	/	%	^
&	|	~	!	=	<
>	+=	-=	*=	/=	%=
^=	&=	|=	<<	>>	>>=
<<=	==	!=	<=	>=	&&
||	++	--	,	->*	->
()	[]	new	delete	new []	delete []

Only Member Functions

Operator	Description
=	Assignment operator
()	Function call operator
[]	Subscripting operator
->	Class member access by pointer operator

Not Allowed

Operator	Description
`sizeof`	The `sizeof` operator
`.`	The membership operator
`.*`	The pointer-to-member operator
`::`	The scope-resolution operator
`?:`	The conditional operator