C++

PROGRAMMING LANGUAGE

L09 – CLASSES, P2

Mohammad Shaker
mohammadshaker.com
@ZGTRShaker
2010, 11, 12, 13, 14

Classes

Composition

Classes

Composition

Class has object of other classes as members

"has – a " relation

Composition

```
#include <iostream>
using namespace std;

#ifndef date_h
#define date_h

class date
{
public:
            date(int = 1, int = 1, int = 1990);
            void print()const;
            ~date();
private:
            int day, month, year;
};

date::date(int d, int m, int y)
{           day = d; month = m, year = y;
            cout << "Constructor runs for date class ";   print();}

void date::print() const
{           cout << day << "/" << month << "/" << year << endl; }

date::~date()
{           cout << "destructor runs for date class, which date is:" << endl;     print();
}
#endif
```

Composition

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

#ifndef employee_h
#define employee_h

class employee
{
public:
            employee();
            employee(char *First,
                        char *Last,
                        const date& d1,
                        const date& d2);
            void print()const;
            ~employee();

private:
            char FirstName [20];
            char LastName [20];
            const date hiredate;
            const date birthdate;
};
```

```
employee::employee(char *First, char *Last,const date& d1, const date&
d2):birthdate(d1), hiredate(d2)
{
            int Length = strlen(First);
            strncpy(FirstName,First,20);
            FirstName[Length]='\0';
            strncpy(LastName,Last,20);
            LastName[Length]='\0';
            cout<<"Constructor runs for employee:"<<FirstName<<' '<< LastName<<endl;
            // ' ' space char;)
}

void employee::print() const
{
            cout << "Employee: "<< FirstName << ' ' << LastName << endl;
            cout << "printing dates for this employee" << endl;
            birthdate.print();
            hiredate.print();
            cout << "_________________" << endl;
}

employee::~employee()
{
            cout << "destructor runs for employee class, for ";
            print();
}
#endif
```

Composition

```
#include <iostream>
#include "employee.h"
#include "date.h"
using namespace std;


void main()
{
            date Birth (15,12,1990);
            date Hire (28,7,2010);
            employee manager ("mee", "loo", Birth, Hire );
            manager.print();
}
```

```
Constructor runs for date class 15/12/1990
Constructor runs for date class 28/7/2010
Constructor runs for employee:mee Hee
Employee: mee Hee
printing dates for this employee
15/12/1990
28/7/2010

________________
destructor runs for employee class, for Employee: mee Hee
printing dates for this employee
15/12/1990
28/7/2010

________________
destructor runs for date class, which date is:
15/12/1990
destructor runs for date class, which date is:
28/7/2010
destructor runs for date class, which date is:
28/7/2010
destructor runs for date class, which date is:
15/12/1990
Press any key to continue
```

Composition

```
class employee
{
public:
            employee();
            employee(char *First,
                        char *Last,
                        const date& d1,
                        const date& d2);
            void print()const;
            ~employee();

private:
            char FirstName [20];
            char LastName [20];
            const date hiredate;
            const date birthdate;
};
```

Change this to

```
class employee
{
public:
            employee();
            employee(char *First,
                        char *Last,
                        const date& d1,
                        const date& d2);
            void print()const;
            ~employee();

private:
            char FirstName [20];
            char LastName [20];
            const date birthdate;    // we change this
            const date hiredate;     // we change this
};
```

Composition

```
#include <iostream>
#include "employee.h"
#include "date.h"
using namespace std;


void main()
{
            date Birth (15,12,1990);
            date Hire (28,7,2010);
            employee manager ("mee", "loo", Birth, Hire );
            manager.print();
}
```

```
Constructor runs for date class 15/12/1990
Constructor runs for date class 28/7/2010
constructor runs for employee:mee Hee
Employee: mee Hee
printing dates for this employee
15/12/1990
28/7/2010

_________________
destructor runs for employee class, for Employee: mee Hee
printing dates for this employee
15/12/1990
28/7/2010

_________________
destructor runs for date class, which date is:
28/7/2010
destructor runs for date class, which date is:
15/12/1990
destructor runs for date class, which date is:
28/7/2010
destructor runs for date class, which date is:
15/12/1990
Press any key to continue
```

So!
Member objects are constructed in order declared and not in order of constructor's member initializer list!

friend Functions

`friend` Functions

- friend Functions
  - A function that is defined outside the scope of the class
  - A non-member function of class
    - But has access to its private data members!
  - The world “friend ” is appears only in the function prototype (in the class definition)

  - Escaping data-hiding restriction
  - Benefits:
    - friend function can be used to access more than one class!

friend Functions

```
#include <iostream>
using namespace std;

#ifndef Count_H
#define Count_H

class Count
{
public:
            Count(): x(0) {}
            void print()const{cout << x << endl;};
private:
            int x;
};

void Wrong()
{
            Count C;
            C.x = 3;
}

#endif
```

```
Compiler error Can't access private date
members
```

friend Functions

```
#include <iostream>
using namespace std;

#ifndef Count_H
#define Count_H

class Count
{
public:
            Count(): x(0) {}
            void print()const{cout << x << endl;};
private:
            int x;
};

#endif
```

```
#include <iostream>
#include "count.h"
using namespace::std;


void Wrong()
{
            Count C;
            C.x = 3;
}


void main()
{
            Wrong();
}
```

Compiler error Can’t access private date members

friend Functions

```
#include <iostream>
using namespace std;

#ifndef Count_H
#define Count_H
class Count
{
public:
            friend void Play ();
            Count(): x(0) {};
            void print()const{cout << x << endl;};
private:
            int x;
};

void Play()
{
            Count C;
            C.x = 3;
}
#endif
```

```
#include <iostream>
#include "count.h"
using namespace::std;


void Wrong()
{
            Count C;
            C.x = 3;
}


void main()
{
            Wrong();
}
```

Compile and run

friend Functions

```
#include <iostream>
using namespace std;
#ifndef Count_H
#define Count_H

class Count
{
public:
            friend void Play ();
            Count(): x(0) {};
            void print()const{cout << x << endl;};
private:
            int x;
};

#endif
```

```
#include <iostream>
#include "count.h"
using namespace::std;
void Play()
{
            Count C;
            C.x = 3;
}

void main()
{

}
```

Compile and run

friend Functions

```
#include <iostream>
using namespace std;
#ifndef Count_H
#define Count_H

class Count
{
public:
            friend void Play (Count &C);
            Count(): x(0) {};
            void print()const{cout << x << endl;};
private:
            int x;
};
void Play(Count &C)
{
            C.x = 3;
}
#endif
```

```
#include <iostream>
#include "count.h"
using namespace::std;

void main()
{
            Count CooCoo;
            CooCoo.print();
            Play(CooCoo);
            CooCoo.print();
}
```

```
0
3
Press any key to continue
```

friend Functions

```
#include <iostream>
using namespace std;
#ifndef Count_H
#define Count_H

class Count
{
public:
            friend void Play (Count &C);
            Count(): x(0) {};
            void print()const{cout << x << endl;};
private:
            int x;
};
void Play(Count &C)
{
            C.x = 3;
}
#endif
```

```
#include <iostream>
#include "count.h"
using namespace::std;


friend void StillWrong()
{
            Count C;
            C.x = 3;
}



void main()
{
            StillWrong();
}
```

Compiler error we write the “friend” key word in the function defintion!

`friend` Classes

friend Classes

- Properties of friendship
  - Granted not taken
    - class B is a friend of class A
      - class A must explicitly declare class B as friend
  - Not symmetric
    - class B friend of class A
      - class A not necessarily friend of class B!
  - Not transitive
    - A friend of B
    - B friend of C
      - A not necessarily friend of C!

friend Classes

In _1stClass header

```
#include <iostream>
#include "_2ndClass.h"
using namespace std;


class _1stClass
{
public:
            friend class _2nClass; // declaring _2ntClass
                                   // as friend to _1stClass
private:
            int x;
};
```

In _2ndClass header

```
#include <iostream>
using namespace std;

#ifndef _2ndClass_h
#define _2ndClass_h

class _2ndClass
{
public:

private:
            int x;
};
#endif
```

```
#include <iostream>
using namespace std;

class _1stClass
{
public:
            friend class _2nClass;
private:
            const int x;
};

class _2ndClass
{
public:
            void PlayWith_1stClass(_1stClass C1);

private:
            int y;
};

void _2ndClass::PlayWith_1stClass(_1stClass C1)
{
            y = C1.x;
}
```

Now it's all okay

`this` Keyword

`this` Keyword

- "this" pointer
  - Represent the address memory of the object
  - Its values is always the memory address of the object
  - Can be used
    - Explicitly \ implicitly
  - What's used for?
    - Can be used to check if a parameter passed to a member function of an object is the object itself
    - Cascading
      - Multiple function invoked in the same statement

this Keyword

```
#include <iostream>
using namespace std;

class Test
{
public:
            Test():x(){};
            void Change(int);
            void print() const;

private:
            int x;
};

void Test::print() const
{
            cout << x << endl;
            cout << this->x << endl;
            cout << (*this).x << endl;
}

void Test::Change (int newValue)
{
            x = newValue;
}
```

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

void main()
{
            Test T;
            T.print();
            T.Change(6);
            T.print();

}
```

```
0
0
0
6
6
6
Press any key to continue
```

this Keyword

```
#include <iostream>
using namespace std;

class Test
{
public:
            Test():x(){};
            void Change(int);
            void print() const;

private:
            int x;
};

void Test::print() const
{
            cout << x << endl;
            cout << this->x << endl;
            cout << *this.x << endl;
}

void Test::Change (int newValue)
{
            x = newValue;
}
```

*this.x != (*this).x because the (.) has higher precedence than (*) So,Compiler error!

this Keyword

```
#include <iostream>
using namespace std;

class Test
{
public:
            Test():x(){};
            void IsSame(Test &Tempo);
            void print() const;

private:
            int x;
};

void Test::print() const
{
            cout << x << endl;
}

void Test::IsSame (Test &Tempo)
{
            if (this == &Tempo)
            {
                        cout << "References are the same" << endl;
            }
}
```

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

void main()
{
            Test T;
            Test *Tptr = &T;
            T.print();
            Tptr->IsSame(T);

}
```

```
0
References are the same
Press any key to continue
```

this Keyword

```
#include <iostream>
using namespace std;

class Test
{
public:
            Test():x(){};
            void IsSame(Test &Tempo);
            void print() const;

private:
            int x;
};

void Test::print() const
{
            cout << x << endl;
}

void Test::IsSame (Test Tempo)
{
            if (this == &Tempo)
            {
                        cout << "References are the same" << endl;
            }
}
```

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

void main()
{
            Test T;
            Test *Tptr = &T;
            T.print();
            Tptr->IsSame(T);

}
```

```
0
Press any key to continue
```

Cause it's passed by value not reference

this Keyword

```
#include <iostream>
using namespace std;

class Test
{
public:
            Test():x(){};
            void IsSame(Test &Tempo);
            void print() const;

private:
            int x;
};

void Test::print() const
{
            cout << x << endl;
}

void Test::IsSame (Test &Tempo)
{
            if (this == &Tempo)
            {
                        cout << "References are the same" << endl;
            }
}
```

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

void main()
{
            Test T;
            Test *Tptr = &T;
            T.print();
            T.IsSame(*Tptr);

}
```

```
0
References are the same
Press any key to continue
```

this Keyword

```
class Time
{public:
            Time();
            Time &SetTime(int,int,int);          // set functions
            Time &SetHour(int);
            Time &SetMinute(int);
            Time &SetSecond(int);
            int GetHour();                                  // get functions
            int GetMinute();
            int GetSecond();
            void PrintTime();
private:    int hour;   int minute; int second;             };
```

```
Time::Time()
{           hour = minute = second = 0; };

Time &Time::SetTime(int xhour,int xminute, int xsecond)
{           SetHour(xhour);         SetMinute(xminute);     SetSecond(xsecond);
            return *this;           // enabling cascading
};

Time &Time::SetHour(int h) { hour = (h>=0 && h<24)? h: 0;
            return *this;           // enabling cascading
};
Time &Time::SetMinute(int m) { minute = (m>=0 && m<60)? m: 0;
            return *this;           // enabling cascading
};
Time &Time::SetSecond(int s) { second = (s>=0 && s<60)? s: 0;
            return *this;           // enabling cascading
};

int Time::GetHour() {return hour;};
int Time::GetMinute() {return minute;};
int Time::GetSecond() {return second;};

void Time::PrintTime()
{           cout << hour << ":" << minute << ":" << second << endl; }
```

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

void main()
{
            Time T;
            //cascade member calls
            T.SetHour(12).SetMinute(5).SetSecond(2);
            T.PrintTime();
}
```

`static` Class Members

static Class Members

- Static members variables are shared among all instances of class
  - Keeps track of how many objects have been created
    - Incrementing with each constructing
    - Decrementing with each destructing
  - Must be initialized
    - Only once
    - Out side the class
  - Can be
    - public, private or protected

static Class Members

- Accessible through public member function or friends
  - Public static variables
    - When no object of class exists
  - Private static variables
    - When no object of class exists
    - Can be accessed via public static member functions

- Can't access non-static data or functions, why?
- No **`this`** pointer for **`static`** functions, why?

static Class Members

```
#include <iostream>
using namespace std;

class Test
{
public:
            Test()
            {           cout << SVariable << endl;          SVariable++;};
            void print() const;
            ~Test()
            {           cout << SVariable << endl;          SVariable--;};

private:
            static int SVariable;
};

// initialize just once outside the scope of the class
int Test::SVariable = 5;

void Test::print() const
{           cout << SVariable << endl;}
```

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

void main()
{
            Test T;
}
```

```
5
6
Press any key to continue
```

static Class Members

```
#include <iostream>
using namespace std;

class Test
{
public:
            Test()
            {           cout << SVariable << endl;                  SVariable++;};
            void print() const;
            ~Test()
            {           cout << SVariable << endl;                  --SVariable;};
            static int SVariable;

private:

};

int Test::SVariable = 5;                        // initialize just once outside the scope of
the class

void Test::print() const
{           cout << SVariable << endl;}
```

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

void main()
{
Test T;
cout << T.SVariable<<endl;
}
```

```
5
6
6
Press any key to continue
```

static Class Members

```
#include <iostream>
using namespace std;

class Test
{
public:
            Test()
            {           cout << "const " << SVariable << endl;
            SVariable++;};
            void print() const;
            ~Test()
            {           cout << "dest " << SVariable << endl;                    SVariable-
-;};
            static int SVariable;

private:

};

int Test::SVariable = 5;                        // initialize just once outside the scope of
the class

void Test::print() const
{           cout << SVariable << endl;}
```

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

void main()
{
    {
            Test T;
            cout << T.SVariable << endl;
    }
            cout << Test::SVariable << endl;
}
```

```
const 5
6
dest 6
5
Press any key to continue
```

static Class Members

```
#include <iostream>
using namespace std;

class Test
{
public:
            Test()
            {           cout << "const " << SVariable << endl;
            SVariable++;};
            void print() const;
            ~Test()
            {           cout << "dest " << SVariable << endl;                    SVariable-
-;};
            static int SVariable;

private:

};

int Test::SVariable = 5;                        // initialize just once outside the scope of
the class

void Test::print() const
{           cout << SVariable << endl;}
```

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

void main()
{
  {
            Test T[5];
            cout << Test::SVariable << endl;
  }
            cout << Test::SVariable << endl;
}
```

```
const 5
const 6
const 7
const 8
const 9
10
dest 10
dest 9
dest 8
dest 7
dest 6
5
Press any key to continue
```

static Class Members

```
#include <iostream>
using namespace std;

class Test
{
public:
            Test()
            {           cout << "const " << SVariable << endl;
            SVariable++;};
            void print() const;
            ~Test()
            {           cout << "dest " << SVariable << endl;                    SVariable-
-;};
            static int SVariable;

private:

};

int Test::SVariable = 5;                        // initialize just once outside the scope of
the class

void Test::print() const
{           cout << SVariable << endl;}
```

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

void main()
{
  {
            Test *T = new Test[5];
            cout << Test::SVariable << endl;
  }
            cout << Test::SVariable << endl;

}
```

```
const 5
const 6
const 7
const 8
const 9
10
10
Press any key to continue
```

static Class Members

```
#include <iostream>
using namespace std;

class Test
{
public:
            Test()
            {           cout << "const " << SVariable << endl;
            SVariable++;};
            void print() const;
            ~Test()
            {           cout << "dest " << SVariable << endl;                    SVariable-
-;};
            static int SVariable;

private:

};

int Test::SVariable = 5;                        // initialize just once outside the scope of
the class

void Test::print() const
{           cout << SVariable << endl;}
```

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

void main()
{
  {
            Test *T = new Test[5];
            cout << Test::SVariable << endl;
            delete  [] T;
  }
  cout << Test::SVariable << endl;
  cout << Test::SVariable << endl;
}
```

```
const 5
const 6
const 7
const 8
const 9
10
dest 10
dest 9
dest 8
dest 7
dest 6
5
5
Press any key to continue
```

static Class Members

```
#include <iostream>
using namespace std;

class Test
{
public:
            Test()
            {           cout << "const " << SVariable << endl;
            SVariable++;};
            void print() const;
            ~Test()
            {           cout << "dest " << SVariable << endl;                    SVariable-
-;};
            static int SVariable;

private:

};

int Test::SVariable = 5;                        // initialize just once outside the scope of
the class

void Test::print() const
{           cout << SVariable << endl;}
```

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

void main()
{
  {
            Test *T = new Test[5];
            cout << Test::SVariable << endl;
  }
            cout << Test::SVariable << endl;
            delete  [] T;
            cout << Test::SVariable << endl;
}
```

Compile error. Undeclared identifier T
We did allocate memory and we have never been de-allocated it!

static Class Members

```
#include <iostream>
using namespace std;

class Test
{
public:
            Test()
            {           ++SVariable;
                        cout << "const " << SVariable << endl;
            };
            void print() const;
            ~Test()
            {           --SVariable;
                        cout << "dest " << SVariable << endl;
             };
            static int SGet();      // static member function
private:
            static int SVariable;   // static date member
};

int Test::SVariable = 0;            // initialize just once outside the scope of the class

void Test::print() const
{           cout << SVariable << endl;}

int Test::SGet()
{
            return SVariable;
}
```

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

void main()
{
            Test T1;
            Test T2;
}
```

```
const 1
const 2
dest 1
dest 0
Press any key to continue
```

static Class Members

```
#include <iostream>
using namespace std;

class Test
{
public:
            Test()
            {           ++SVariable;
                        cout << "const " << SVariable << endl;
            };
            void print() const;
            ~Test()
            {           --SVariable;
                        cout << "dest " << SVariable << endl;
             };
            static int SGet();      // static member function
private:
            static int SVariable;   // static date member
};

int Test::SVariable = 0;            // initialize just once outside the scope of the class

void Test::print() const
{           cout << SVariable << endl;}

int Test::SGet()
{
            return SVariable;
}
```

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

void main()
{
            Test *T1;
            Test *T2;

}
```

```
Press any key to continue
```

static Class Members

```
#include <iostream>
using namespace std;

class Test
{
public:
            Test()
            {           ++SVariable;
                        cout << "const " << SVariable << endl;
            };
            void print() const;
            ~Test()
            {           --SVariable;
                        cout << "dest " << SVariable << endl;
             };
            static int SGet();      // static member function
private:
            static int SVariable;   // static date member
};

int Test::SVariable = 0;            // initialize just once outside the scope of the class

void Test::print() const
{           cout << SVariable << endl;}

int Test::SGet()
{
            return SVariable;
}
```

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

void main()
{
            Test TSource;
            Test *T1;
            Test *T2;
            //cout << T1->SGet() << endl;
}
```

```
const 1
dest 0
Press any key to continue
```

static Class Members

```
#include <iostream>
using namespace std;

class Test
{
public:
            Test()
            {           ++SVariable;
                        cout << "const " << SVariable << endl;
            };
            void print() const;
            ~Test()
            {           --SVariable;
                        cout << "dest " << SVariable << endl;
             };
            static int SGet();      // static member function
private:
            static int SVariable;   // static date member
};

int Test::SVariable = 0;            // initialize just once outside the scope of the class

void Test::print() const
{           cout << SVariable << endl;}

int Test::SGet()
{
            return SVariable;
}
```

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

void main()
{
            Test TSource;
            Test *T1 = &TSource;
            Test *T2;

}
```

```
const 1
dest 0
Press any key to continue
```

static Class Members

```
#include <iostream>
using namespace std;

class Test
{
public:
            Test()
            {           ++SVariable;
                        cout << "const " << SVariable << endl;
            };
            void print() const;
            ~Test()
            {           --SVariable;
                        cout << "dest " << SVariable << endl;
             };
            static int SGet();      // static member function
private:
            static int SVariable;   // static date member
};

int Test::SVariable = 0;            // initialize just once outside the scope of the class

void Test::print() const
{           cout << SVariable << endl;}

int Test::SGet()
{
            return SVariable;
}
```

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

void main()
{
            Test TSource;
            Test *T1 = new Test;
            Test *T2;
}
```

```
const 1
const 2
dest 1
Press any key to continue
```

static Class Members

```
#include <iostream>
using namespace std;

class Test
{
public:
            Test()
            {           ++SVariable;
                        cout << "const " << SVariable << endl;
            };
            void print() const;
            ~Test()
            {           --SVariable;
                        cout << "dest " << SVariable << endl;
             };
            static int SGet();      // static member function
private:
            static int SVariable;   // static date member
};

int Test::SVariable = 0;            // initialize just once outside the scope of the class

void Test::print() const
{           cout << SVariable << endl;}

int Test::SGet()
{
            return SVariable;
}
```

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

void main()
{
            Test TSource;
            Test *T1 = new Test;
            Test *T2;
            cout << T1->SGet() << endl;
}
```

```
const 1
const 2
2
dest 1
Press any key to continue
```

static Class Members

```
#include <iostream>
using namespace std;

class Test
{
public:
            Test()
            {           ++SVariable;
                        cout << "const " << SVariable << endl;
            };
            void print() const;
            ~Test()
            {           --SVariable;
                        cout << "dest " << SVariable << endl;
             };
            static int SGet();      // static member function
private:
            static int SVariable;   // static date member
};

int Test::SVariable = 0;            // initialize just once outside the scope of the class

void Test::print() const
{           cout << SVariable << endl;}

int Test::SGet()
{
            return SVariable;
}
```

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

void main()
{
            Test TSource;
            Test *T1 = new Test;
            Test *T2;
            cout << Test::SGet() << endl;
}
```

```
const 1
const 2
2
dest 1
Press any key to continue
```

static Class Members

```
#include <iostream>
using namespace std;

class Test
{
public:
            Test()
            {           ++SVariable;
                        cout << "const " << SVariable << endl;
            };
            void print() const;
            ~Test()
            {           --SVariable;
                        cout << "dest " << SVariable << endl;
             };
            static int SGet();      // static member function
private:
            static int SVariable;   // static date member
};

int Test::SVariable = 0;            // initialize just once outside the scope of the class

void Test::print() const
{           cout << SVariable << endl;}

int Test::SGet()
{
            return SVariable;
}
```

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

void main()
{
            Test TSource;
            Test *T1 = new Test;
            Test *T2;
            delete T1;
            cout << Test::SGet() << endl;
}
```

```
const 1
const 2
dest 1
1
dest 0
Press any key to continue
```

static Class Members

```
#include <iostream>
using namespace std;

class Test
{
public:
            Test()
            {           ++SVariable;
                        cout << "const " << SVariable << endl;
            };
            void print() const;
            ~Test()
            {           --SVariable;
                        cout << "dest " << SVariable << endl;
             };
            static int SGet();      // static member function
private:
            static int SVariable;   // static date member
};

int Test::SVariable = 0;            // initialize just once outside the scope of the class

void Test::print() const
{           cout << SVariable << endl;}

int Test::SGet()
{
            return SVariable;
}
```

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

void main()
{
            Test TSource;
            Test *T1 = new Test;
            Test *T2;
            delete T1;
            cout << T1->SGet() << endl;
}
```

```
const 1
const 2
dest 1
1
dest 0
Press any key to continue
```

static Class Members

```
#include <iostream>
using namespace std;

class Test
{
public:
            Test()
            {           ++SVariable;
                        cout << "const " << SVariable << endl;
            };
            void print() const;
            ~Test()
            {           --SVariable;
                        cout << "dest " << SVariable << endl;
             };
            static int SGet();      // static member function
private:
            static int SVariable;   // static date member
};

int Test::SVariable = 0;            // initialize just once outside the scope of the class

void Test::print() const
{           cout << SVariable << endl;}

int Test::SGet()
{
            return SVariable;
}
```

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

void main()
{
            Test TSource;
            Test *T1 = new Test;
            Test *T2;
            delete []T1;
            cout << T1->SGet() << endl;
}
```

Runtime error coz of [] not compiler error for sure!

static Class Members

```
#include <iostream>
using namespace::std;

class TestStaticClass
{
	public:
		static int i;
		static void StaticPublicMethodTest()
		{
			cout << "You called a static method!" << endl;
		}
};
int TestStaticClass::i = 0;

void main()
{
	TestStaticClass T;
	T.StaticPublicMethodTest();
	TestStaticClass::StaticPublicMethodTest();
	system("pause");
}
```

```
You called a static method!
You called a static method!
Press any key to continue
```

static Class Members

```
#include <iostream>
using namespace::std;

class TestStaticClass
{
	public:
	void GetMyName()
	{
		cout << "You are in \"TestStaticClass\" class" << endl;
	}
	static void StaticPublicMethodTest()
	{
		cout << "You called a static method!" << endl;
	}
};

void main()
{
	TestStaticClass T;
	T.GetMyName();
	TestStaticClass::GetMyName();
	system("pause");
}
```

```
Compiler error!, TestStaticClass::GetMyName(); is not static!
```

static Class Members

```
#include <iostream>
using namespace::std;

class TestStaticClass
{
	public:
	void GetMyName()
	{
		cout << "You are in \"TestStaticClass\" class" << endl;
	}
	static void StaticPublicMethodTest()
	{
		cout << "You called a static method!" << endl;
	}
};

void main()
{
	TestStaticClass T;
	T.GetMyName();
	system("pause");
}
```

```
You are in "TestStaticClass" class
Press any key to continue
```

static Class Members

```
#include <iostream>
using namespace::std;

class TestStaticClass
{
	public:
	void GetMyName()
	{
		cout << "You are in \"TestStaticClass\" class" << endl;
		StaticPublicMethodTest();
	}
	static void StaticPublicMethodTest()
	{
		cout << "You called a static method!" << endl;
	}
};

void main()
{
	TestStaticClass T;
	T.GetMyName();
	system("pause");
}
```

```
You are in "TestStaticClass" class
You called a static method!
Press any key to continue
```

```
class TestClass
{
	public:
	void PublicNonStaticMethodTest()
	{
		cout << "You called a non static method!" << endl;
		PublicStaticMethodTest();
	}
	static void PublicStaticMethodTest()
	{
		cout << "You called a static method!" << endl;
	}
};

void main()
{
	TestClass T;
	T.PublicNonStaticMethodTest();
	system("pause");
}
```

```
You called a non static method!
You called a static method!
Press any key to continue
```

static Class Members

```
class TestClass
{
	public:
		int iNonStatic;
		static int iStatic;
		void PublicNonStaticMethodTest()
		{
			cout << "You called a non static method!"
		<< endl;
			PublicStaticMethodTest();
		}
		static void PublicStaticMethodTest()
		{
			cout << "You called a static method!" <<
		endl;
			iStatic = 5;
		}
};
int TestClass::iStatic = 0;

void main()
{
	TestClass T;
	T.PublicNonStaticMethodTest();
	cout << TestClass::iStatic << endl;
	system("pause");
}
```

```
You called a non static method!
You called a static method!
5
Press any key to continue
```

```
class TestClass
{
public:
	int iNonStatic;
	static int iStatic;
	void PublicNonStaticMethodTest()
	{
		cout << "You called a non static method!" << endl;
		PublicStaticMethodTest();
	}
	static void PublicStaticMethodTest()
	{
		cout << "You called a static method!" << endl;
		iStatic = 5;
		iNonStatic = 5;
	}
};
int TestClass::iStatic = 0;

void main()
{
	TestClass T;
	T.PublicNonStaticMethodTest();
	cout << TestClass::iStatic << endl;
	cout << T.iNonStatic << endl;
	system("pause");
}
```

```
Compile error iNonStatic = 5;
In “PublicStaticMethodTest”
Static method!
```

static Class Members

```
class TestClass
{
public:
	int iNonStatic;
	static int iStatic;
	void PublicNonStaticMethodTest()
	{
		cout << "You called a non static method!" << endl;
		iNonStatic = 5;
		PublicStaticMethodTest();
	}
	static void PublicStaticMethodTest()
	{
		cout << "You called a static method!" << endl;
		iStatic = 5;
	}
};
int TestClass::iStatic = 0;

void main()
{
	TestClass T;
	T.PublicNonStaticMethodTest();
	cout << TestClass::iStatic << endl;
	cout << T.iNonStatic << endl;
	system("pause");
}
```

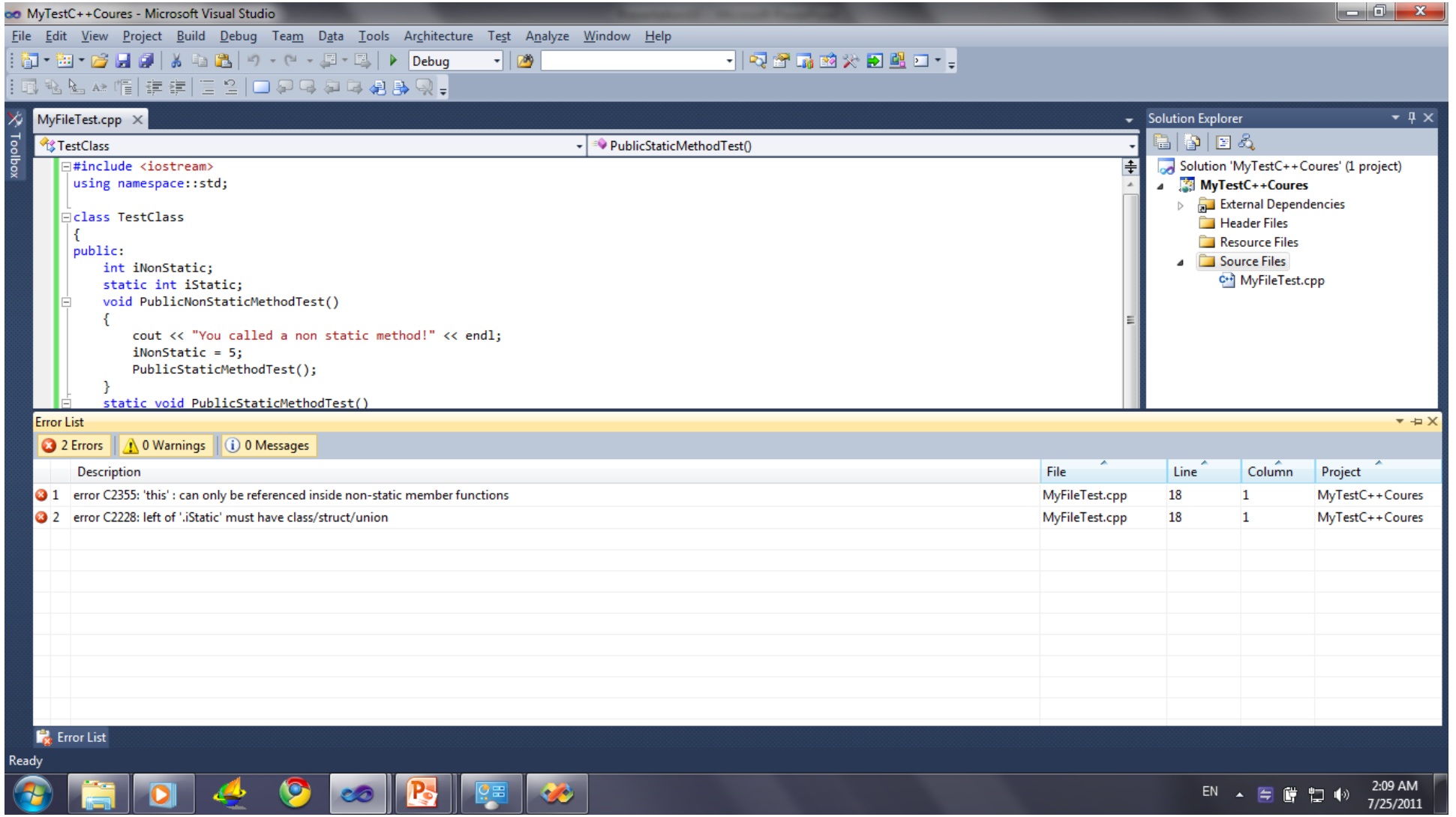
```
You called a non static method!
You called a static method!
5
5
Press any key to continue
```

```
class TestClass
{
public:
	int iNonStatic;
	static int iStatic;
	void PublicNonStaticMethodTest()
	{
		cout << "You called a non static method!" << endl;
		iNonStatic = 5;
		PublicStaticMethodTest();
	}
	static void PublicStaticMethodTest()
	{
		cout << "You called a static method!" << endl;
		this.iStatic = 5;
	}
};
int TestClass::iStatic = 0;

void main()
{
	TestClass T;
	T.PublicNonStaticMethodTest();
	cout << TestClass::iStatic << endl;
	cout << T.iNonStatic << endl;
	system("pause");
}
```

```
This can’t be used in a static method!
It’s logically wrong!
```

static Class Members



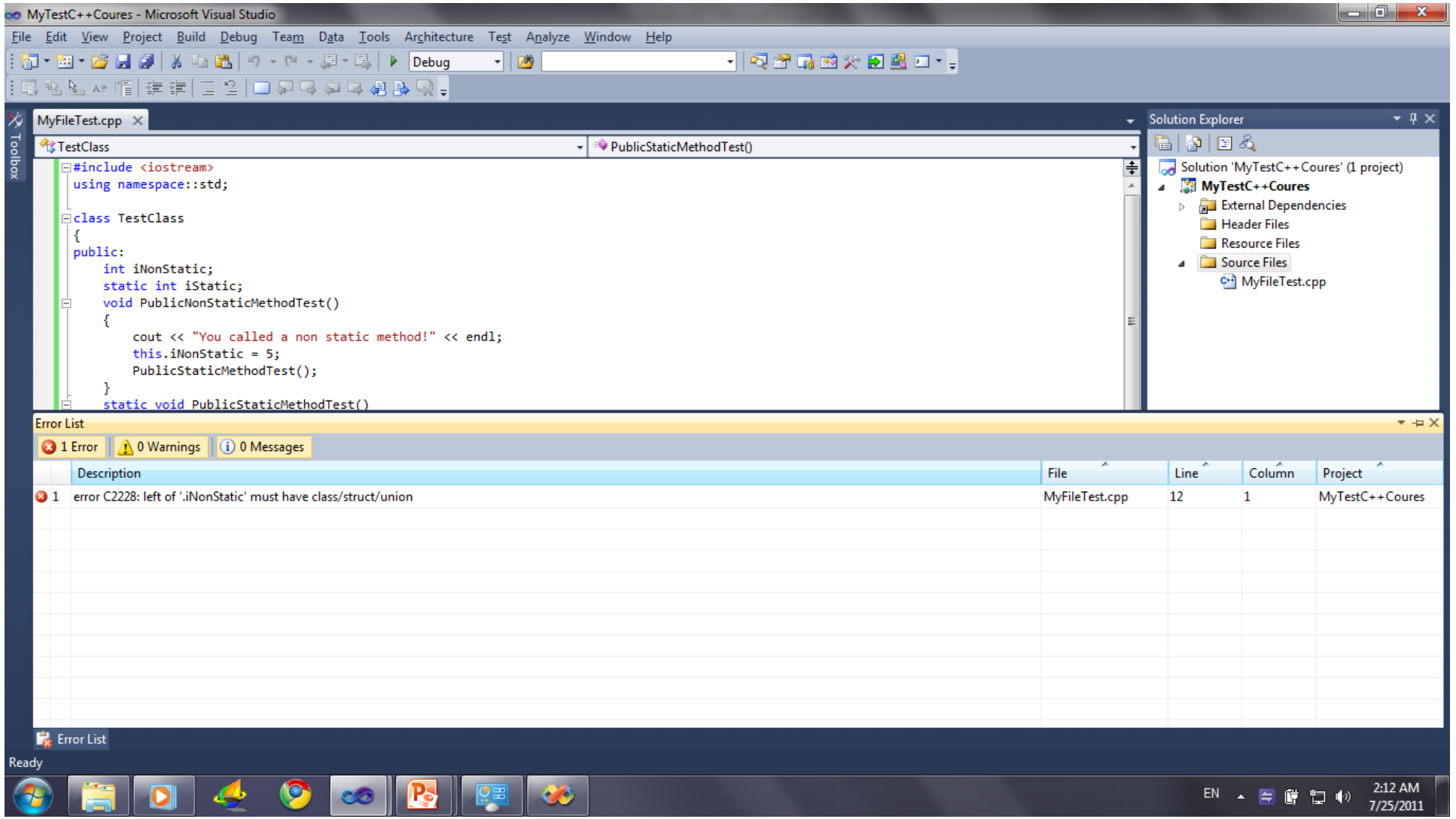
static Class Members

```
class TestClass
{
public:
	int iNonStatic;
	static int iStatic;
	void PublicNonStaticMethodTest()
	{
		cout << "You called a non static method!" << endl;
		this.iNonStatic = 5;
		PublicStaticMethodTest();
	}
	static void PublicStaticMethodTest()
	{
		cout << "You called a static method!" << endl;
		iStatic = 5;
	}
};
int TestClass::iStatic = 0;

void main()
{
	TestClass T;
	T.PublicNonStaticMethodTest();
	cout << TestClass::iStatic << endl;
	cout << T.iNonStatic << endl;
	system("pause");
}
```

```
Compiler error
```

static Class Members



Important Example

Constructors and Destructors

```
#include <iostream>
using namespace::std;

class TestClass
{
public:
		char* _name;
		TestClass(char *name)
		{
				this->_name = name;
				cout << "Constructor for" << _name << endl;
		}
		~TestClass()
		{
				cout << "Destructor for" << _name << endl;
		}
};

void main()
{
		TestClass T1("T1");
		{
				TestClass T2("T2");
				static TestClass TStaticInMainInnerScope("TStaticInMainInnerScope");
		}
		static TestClass TStaticInMainOuterScope("TStaticInMainOuterScope");
		TestClass T4("T4");
}

TestClass TGlobal("TGlobal");
static TestClass TStaticGlobal("TStaticGlobal");
```

```
Constructor forTGlobal
Constructor forTStaticGlobal
Constructor forT1
Constructor forT2
Constructor forTStaticInMainInnerScope
Destructor forT2
Constructor forTStaticInMainOuterScope
Constructor forT4
Destructor forT4
Destructor forT1
Destructor forTStaticInMainOuterScope
Destructor forTStaticInMainInnerScope
Destructor forTStaticGlobal
Destructor forTGlobal
Press any key to continue
```

Quiz

Quiz #1

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

#ifndef employee_h
#define employee_h

class employee
{
public:
            employee();
            employee(char *First,
                        char *Last,
                        const date& d1,
                        const date& d2);
            void print()const;
            ~employee();

private:
            char FirstName [20];
            char LastName [20];
            const date hiredate;
            const date birthdate;
};
```

```
employee::employee(char *First, char *Last,const date& d1, const date&
d2):birthdate(d1), hiredate(d2)
{
            int Length = strlen(First);
            strncpy(FirstName,First,20);
            FirstName[Length]='\0';
            strncpy(LastName,Last,20);
            LastName[Length]='\0';
            cout<<"constructor runs for employee:"<<FirstName<<' '<< LastName<<endl;
            // ' ' space char;)
}

void employee::print() const
{
            cout << "Employee: "<< FirstName << ' ' << LastName << endl;
            cout << "printing dates for this employee" << endl;
            birthdate.print();
            hiredate.print();
            cout << "_________________" << endl;
}

employee::~employee()
{
            cout << "destructor runs for employee class, for ";
            print();
}
#endif
```

Quiz #1, What's the Output?

```
#include <iostream>
using namespace std;

#ifndef date_h
#define date_h

class date
{
public:
            date(int = 1, int = 1, int = 1990);
            void print()const;
            ~date();
private:
            int day, month, year;
};

date::date(int d, int m, int y)
{           day = d; month = m, year = y;
            cout << "Constructor runs for date class ";    print();}

void date::print() const
{           cout << day << "/" << month << "/" << year << endl; }

date::~date()
{           cout << "destructor runs for date class, which date is:"<<endl;
            print();
}
#endif
```

```
#include <iostream>
#include "employee.h"
#include "date.h"
using namespace std;


void main()
{
            date Birth1 (8,8,1990);
            date Hire1 (4,4,2010);
            date Birth2 (10,10,1990);
            date Hire2 (5,5,2010);
            employee manager1 ("mee", "loo", Birth1, Hire1 );
            employee manager2 ("zee", "HooHoo", Birth2, Hire2 );
            manager1.print();
            manager2.print();
}
```

Quiz #1

```
Constructor runs for date class 8/8/1990
Constructor runs for date class 4/4/2010
Constructor runs for date class 10/10/1990
Constructor runs for date class 5/5/2010
constructor runs for employee:mee Hee
constructor runs for employee:zee HooHoo
Employee: mee Hee
printing dates for this employee
8/8/1990
4/4/2010

_________________
Employee: zee HooHoo
printing dates for this employee
10/10/1990
5/5/2010

_________________
```

```
destructor runs for employee class, for Employee: zee HooHoo
printing dates for this employee
10/10/1990
5/5/2010

_________________
destructor runs for date class, which date is:
5/5/2010
destructor runs for date class, which date is:
10/10/1990
destructor runs for employee class, for Employee: mee Hee
printing dates for this employee
8/8/1990
4/4/2010

_________________
destructor runs for date class, which date is:
4/4/2010
destructor runs for date class, which date is:
8/8/1990
destructor runs for date class, which date is:
5/5/2010
destructor runs for date class, which date is:
10/10/1990
destructor runs for date class, which date is:
4/4/2010
destructor runs for date class, which date is:
8/8/1990
```

Quiz #2, What's the Output?

```
#include <iostream>
using namespace std;

class Test
{
public:
            Test()
            {
                        cout << "const " << SVariable << endl;
                        SVariable++;
            };
            void print() const;
            ~Test()
            {           cout << "dest " << SVariable << endl;
                        SVariable--;
            };
            static int SVariable;

private:

};
int Test::SVariable = 5;
void Test::print() const
{
            cout << SVariable << endl;
}
```

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

void main()
{
            Test T1;
            {
                        Test T2[5];
                        Test TTemp;
                        cout << TTemp.SVariable << endl;
            }
            cout << Test::SVariable << endl;
            Test *T3 = &T1;
            static Test T4;
            cout << Test::SVariable << endl;
            Test* &T5 = T3;
            cout << Test::SVariable << endl;
}
```

```
const 5
const 6
const 7
const 8
const 9
const 10
const 11
12
dest 12
dest 11
dest 10
dest 9
dest 8
dest 7
6
const 6
7
7
dest 7
dest 6
```