4.1 How to inherit one class from another

Source Code

Student.h

```
#ifndef STUDENT_H
#define STUDENT_H

#include<string>
using namespace std;

class Student
{
public:
    Student();
    Student(string name, string address, int rollNo);

    virtual void PrintInfo();

    string getName();
    void setName(string name);
    string getAddress();
    void setAddress(string address);
    int getRollNo();
    void setRollNo(int rollNo);
private:
    string name;
    string address;
    int rollNo;
};

#endif
```

Student.cpp

```
#include"Student.h"
#include<iostream>

Student::Student()
{
}
Student::Student(string name, string address, int rollNo)
{
    this->name = name;
    this->address = address;
    this->rollNo = rollNo;
}

void Student::PrintInfo()
{
    cout << "***Student***" << endl;
    cout << "Name: " << name << endl;
    cout << "Address: " << address << endl;
```

```
    cout << "RollNo: " << rollNo << endl;
}

string Student::getName()
{
    return name;
}
void Student::setName(string name)
{
    this->name = name;
}
string Student::getAddress()
{
    return address;
}
void Student::setAddress(string address)
{
    this->address = address;
}
int Student::getRollNo()
{
    return rollNo;
}
void Student::setRollNo(int rollNo)
{
    this->rollNo = rollNo;
}
```

BSStudent.h

```
#ifndef BSSTUDENT_H
#define BSSTUDENT_H

#include"Student.h"

class BSStudent: public Student
{
public:
    BSStudent();
    BSStudent(string name, string address, int rollNo, int year);

    void PrintInfo();

    int getYear();
    void setYear(int year);
private:
    int year;
};

#endif
```

BSStudent.cpp

```
#include"BSStudent.h"
#include<iostream>

BSStudent::BSStudent()
```

```
{
}
BSStudent::BSStudent(string name, string address, int rollNo, int year)
{
    setName(name);
    setAddress(address);
    setRollNo(rollNo);
    this->year = year;
}

void BSStudent::PrintInfo()
{
    cout << "***BSStudent***" << endl;
    cout << "Name: " << getName() << endl;
    cout << "Address: " << getAddress() << endl;
    cout << "RollNo: " << getRollNo() << endl;
    cout << "Year: " << year << endl;
}

int BSStudent::getYear()
{
    return year;
}
void BSStudent::setYear(int year)
{
    this->year = year;
}
```

MSStudent.h

```
#ifndef MSSTUDENT_H
#define MSSTUDENT_H

#include"Student.h"

class MSStudent: public Student
{
public:
    MSStudent();
    MSStudent(string name, string address, int rollNo, string advisor);

    void PrintInfo();

    string getAdvisor();
    void setAdvisor(string advisor);
private:
    string advisor;
};

#endif
```

MSStudent.cpp

```
#include"MSStudent.h"
#include<iostream>

MSStudent::MSStudent()
```

```
{
}
MSStudent::MSStudent(string name, string address, int rollNo, string advisor)
{
    setName(name);
    setAddress(address);
    setRollNo(rollNo);
    this->advisor = advisor;
}

void MSStudent::PrintInfo()
{
    cout << "***MSStudent***" << endl;
    cout << "Name: " << getName() << endl;
    cout << "Address: " << getAddress() << endl;
    cout << "RollNo: " << getRollNo() << endl;
    cout << "Advisor: " << advisor << endl;
}

string MSStudent::getAdvisor()
{
    return advisor;
}
void MSStudent::setAdvisor(string advisor)
{
    this->advisor = advisor;
}
```

MSByCourse.h

```
#ifndef MSBYCOURSE_H
#define MSBYCOURSE_H

#include"MSStudent.h"

class MSByCourse: public MSStudent
{
public:
    MSByCourse();
    MSByCourse(string name, string address, int rollNo, string advisor);

    void PrintInfo();
};

#endif
```

MSByCourse.cpp

```
#include"MSByCourse.h"
#include<iostream>

MSByCourse::MSByCourse()
{
}
MSByCourse::MSByCourse(string name, string address, int rollNo, string advisor)
{
    setName(name);
```

```
    setAddress(address);
    setRollNo(rollNo);
    setAdvisor(advisor);
}

void MSByCourse::PrintInfo()
{
    cout << "***MSByCourse***" << endl;
    cout << "Name: " << getName() << endl;
    cout << "Address: " << getAddress() << endl;
    cout << "RollNo: " << getRollNo() << endl;
    cout << "Advisor: " << getAdvisor() << endl;
}
```

MSByResearch.h

```
#ifndef MSBYRESEARCH_H
#define MSBYRESEARCH_H

#include"MSStudent.h"

class MSByResearch: public MSStudent
{
public:
    MSByResearch();
    MSByResearch(string name, string address, int rollNo, string advisor, bool thesisStatus);

    void PrintInfo();

    bool getThesisStatus();
    void setThesisStatus(bool thesisStatus);
private:
    bool thesisStatus;
};

#endif
```

MSByResearch.cpp

```
#include"MSByResearch.h"
#include<iostream>

MSByResearch::MSByResearch()
{
}
MSByResearch::MSByResearch(string name, string address, int rollNo, string advisor, bool
thesisStatus)
{
    setName(name);
    setAddress(address);
    setRollNo(rollNo);
    setAdvisor(advisor);
    this->thesisStatus = thesisStatus;
}

void MSByResearch::PrintInfo()
{
```

```
    cout << "***MSByCourse***" << endl;
    cout << "Name: " << getName() << endl;
    cout << "Address: " << getAddress() << endl;
    cout << "RollNo: " << getRollNo() << endl;
    cout << "Advisor: " << getAdvisor() << endl;
    cout << "ThesisStatus: " << thesisStatus << endl;
}

bool MSByResearch::getThesisStatus()
{
    return thesisStatus;
}
void MSByResearch::setThesisStatus(bool thesisStatus)
{
    this->thesisStatus = thesisStatus;
}
```

4.2

Source Code

TVChannel.h

```
#ifndef TVCHANNEL_H
#define TVCHANNEL_H
using namespace std;

class TVChannel
{
public:
    TVChannel(){};

    virtual void DisplayName()=0;
};

#endif
```

NewsChannel.h

```
#ifndef NEWSCHANNEL_H
#define NEWSCHANNEL_H

#include"TVChannel.h"

class NewsChannel: public TVChannel
{
public:
    NewsChannel();

    void DisplayName();
};

#endif
```

NewsChannel.cpp

```
#include"NewsChannel.h"
#include<iostream>

NewsChannel::NewsChannel()
{
}

void NewsChannel::DisplayName()
{
    cout << "This is NewsChannel" << endl;
}
```

MusicChannel.h

```
#ifndef MUSICCHANNEL_H
#define MUSICCHANNEL_H

#include"TVChannel.h"

class MusicChannel: public TVChannel
{
public:
    MusicChannel();

    void DisplayName();
};

#endif
```

MusicChannel.cpp

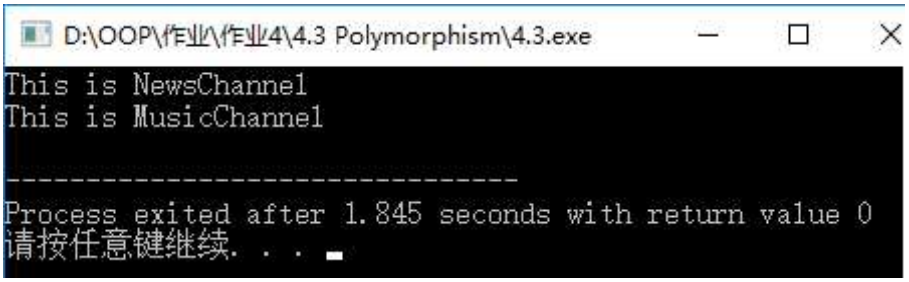
```
#include"MusicChannel.h"
#include<iostream>

MusicChannel::MusicChannel()
{
}

void MusicChannel::DisplayName()
{
    cout << "This is MusicChannel" << endl;
}
```

main.cpp

```
#include <iostream>

#include"TVChannel.h"
#include"NewsChannel.h"
#include"MusicChannel.h"

int main(int argc, char** argv)
```

```
{
    TVChannel *p;
    NewsChannel ob1;
    MusicChannel ob2;

    p=&ob1;
    p->DisplayName();

    p=&ob2;
    p->DisplayName();

    return 0;
}
```

Sample Output



4.3 Overloading operators_Vector

Source Code

Vector.h

```
#ifndef VECTOR_H
#define VECTOR_H
using namespace std;

class Vector
{
public:
    Vector();
    Vector(double x, double y, double z);

    double getXComponent() const;
    void setXComponent(double x);
    double getYComponent() const;
    void setYComponent(double y);
    double getZComponent() const;
    void setZComponent(double z);

    void display();

    Vector operator+(const Vector& secondVector) const;
    double operator*(const Vector& secondVector) const;
private:
    double XComponent;
    double YComponent;
```

```
    double ZComponent;
};

#endif
```

Vector.cpp

```
#include"Vector.h"
#include <iostream>

Vector::Vector()
{
}
Vector::Vector(double x, double y, double z)
{
    this->XComponent = x;
    this->YComponent = y;
    this->ZComponent = z;
}

double Vector::getXComponent() const
{
    return XComponent;
}
void Vector::setXComponent(double x)
{
    this->XComponent = x;
}
double Vector::getYComponent() const
{
    return YComponent;
}
void Vector::setYComponent(double y)
{
    this->YComponent = y;
}
double Vector::getZComponent() const
{
    return ZComponent;
}
void Vector::setZComponent(double z)
{
    this->ZComponent = z;
}

void Vector::display()
{
    cout << this->XComponent << "i";
    if(this->YComponent >= 0)
    {
        cout << "+";
    }
    cout << this->YComponent << "j";
    if(this->ZComponent >= 0)
    {
        cout << "+";
    }
    cout << this->ZComponent << "k";
    cout << endl;
```

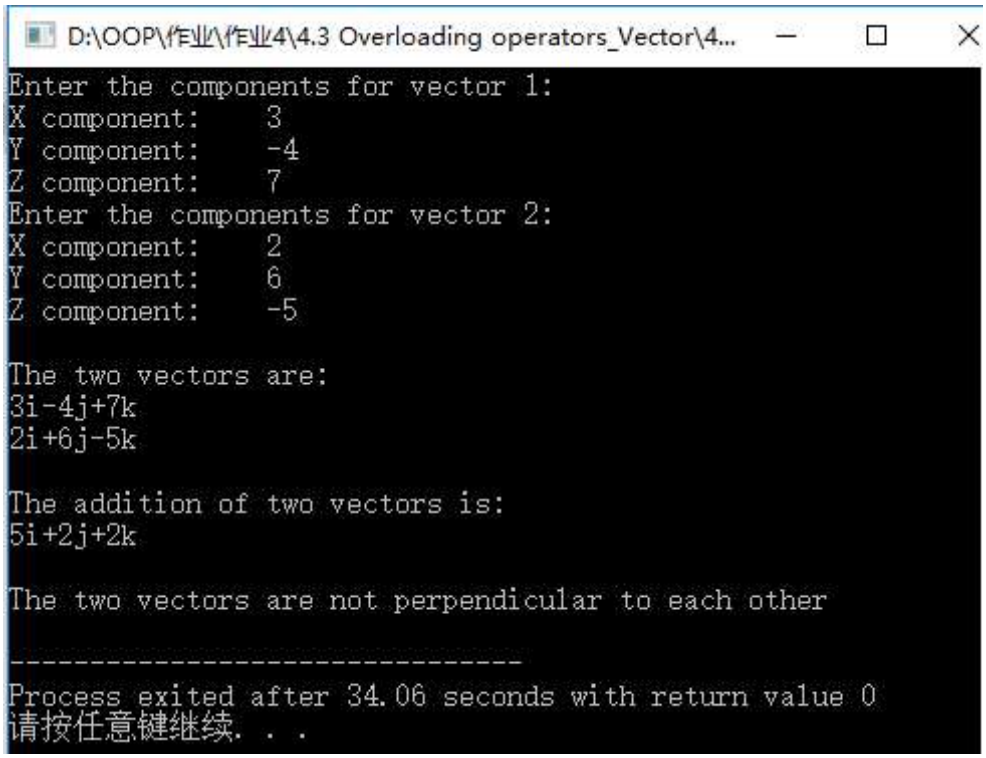
```
}

Vector Vector::operator+(const Vector& secondVector) const
{
    double x = this->XComponent + secondVector.getXComponent();
    double y = this->YComponent + secondVector.getYComponent();
    double z = this->ZComponent + secondVector.getZComponent();
    return Vector(x,y,z);
}
double Vector::operator*(const Vector& secondVector) const
{
    double r1 = this->XComponent * secondVector.getXComponent();
    double r2 = this->YComponent * secondVector.getYComponent();
    double r3 = this->ZComponent * secondVector.getZComponent();
    return r1 + r2 + r3;
}
```

main.cpp

```
#include <iostream>
#include"Vector.h"

int main(int argc, char** argv) {
    double x,y,z;

    cout << "Enter the components for vector 1:" << endl;
    cout << "X component:   ";
    cin >> x;
    cout << "Y component:   ";
    cin >> y;
    cout << "Z component:   ";
    cin >> z;
    Vector vector1(x,y,z);

    cout << "Enter the components for vector 2:" << endl;
    cout << "X component:   ";
    cin >> x;
    cout << "Y component:   ";
    cin >> y;
    cout << "Z component:   ";
    cin >> z;
    Vector vector2(x,y,z);

    cout << endl << "The two vectors are:" << endl;
    vector1.display();
    vector2.display();

    Vector vector_sum;
    vector_sum = vector1 + vector2;
    cout << endl << "The addition of two vectors is:" << endl;
    vector_sum.display();

    double vector_dotproduct = vector1 * vector2;
    if(vector_dotproduct == 0)
    {
        cout << endl << "The two vectors are perpendicular to each other" << endl;
    }
    else
    {
```

```
        cout << endl << "The two vectors are not perpendicular to each other" << endl;
    }

    return 0;
}
```

Sample Output



4.4 Overloading operators_FeetInches

Source Code

FeetInches.h

```
#ifndef FEETINCHES_H
#define FEETINCHES_H
using namespace std;

class FeetInches
{
public:
    FeetInches();
    FeetInches(int feet, int inches);

    int getFeet() const;
    void setFeet(int feet);
    int getInches() const;
    void setInches(int inches);

    bool operator>=(const FeetInches& secondFeetInches) const;
    bool operator!=(const FeetInches& secondFeetInches) const;
    FeetInches operator+(const FeetInches& secondFeetInches) const;
```

```
    FeetInches operator*(const FeetInches& secondFeetInches) const;
    FeetInches operator/(const FeetInches& secondFeetInches) const;

    void display();
private:
    int feet;
    int inches;

    void convert();
};

#endif
```

FeetInches.cpp

```
#include"FeetInches.h"
#include <iostream>

FeetInches::FeetInches()
{
}
FeetInches::FeetInches(int feet, int inches)
{
    this->feet = feet;
    this->inches = inches;

    this->convert();
}

void FeetInches::convert()
{
    if(inches > 12)
    {
        feet += inches / 12;
        inches = inches % 12;
    }
}

int FeetInches::getFeet() const
{
    return feet;
}
void FeetInches::setFeet(int feet)
{
    this->feet = feet;
}
int FeetInches::getInches() const
{
    return inches;
}
void FeetInches::setInches(int inches)
{
    this->inches = inches;
}

bool FeetInches::operator>=(const FeetInches& secondFeetInches) const
{
    if(this->feet > secondFeetInches.getFeet())
        return true;
```

```
    else if(this->feet == secondFeetInches.getFeet())
    {
        if(this->inches >= secondFeetInches.getInches())
            return true;
        else
            return false;
    }
    else
        return false;
}
bool FeetInches::operator!=(const FeetInches& secondFeetInches) const
{
    if(this->feet == secondFeetInches.getFeet() && this->inches == secondFeetInches.getInches())
        return false;
    else
        return true;
}
FeetInches FeetInches::operator+(const FeetInches& secondFeetInches) const
{
    int feet = this->feet + secondFeetInches.getFeet();
    int inches = this->inches + secondFeetInches.getInches();
    FeetInches sumFeetInches(feet,inches);
    sumFeetInches.convert();
    return sumFeetInches;
}
FeetInches FeetInches::operator*(const FeetInches& secondFeetInches) const
{
    int feet = this->feet * secondFeetInches.getFeet();
    int inches = this->inches * secondFeetInches.getInches();
    FeetInches timesFeetInches(feet,inches);
    timesFeetInches.convert();
    return timesFeetInches;
}
FeetInches FeetInches::operator/(const FeetInches& secondFeetInches) const
{
    int inches1 = this->feet * 12 + this->inches;
    int inches2 = secondFeetInches.getFeet() * 12 + secondFeetInches.getInches();
    FeetInches dividedFeetInches(0,inches1/inches2);
    dividedFeetInches.convert();
    return dividedFeetInches;
}

void FeetInches::display()
{
    cout << this->feet << " feet, ";
    cout << this->inches << " inches";
}
```

main.cpp

```
#include"FeetInches.h"
#include <iostream>

int main(int argc, char** argv) {
    int feet,inches;

    cout << "Enter a length in feet and inches:" << endl;
    cout << "Feet:  ";
    cin >> feet;
```

```
    cout << "Inches:    ";
    cin >> inches;
    FeetInches feetinches1(feet,inches);

    cout << "Enter a length in feet and inches:" << endl;
    cout << "Feet:  ";
    cin >> feet;
    cout << "Inches:    ";
    cin >> inches;
    FeetInches feetinches2(feet,inches);

    cout << endl;
    FeetInches sumfeetinches = feetinches1 + feetinches2;
    feetinches1.display();
    cout << " sum ";
    feetinches2.display();
    cout << " is ";
    sumfeetinches.display();
    cout << endl;

    cout << endl;
    FeetInches timesfeetinches = feetinches1 * feetinches2;
    feetinches1.display();
    cout << " times ";
    feetinches2.display();
    cout << " is ";
    timesfeetinches.display();
    cout << endl;

    cout << endl;
    feetinches1.display();
    if(feetinches1 >= feetinches2)
        cout << " is greater than and equal to ";
    else
        cout << " is not greater than and equal to ";
    feetinches2.display();
    cout << endl;

    cout << endl;
    if(feetinches1 != feetinches2)
        cout << "Two are not equal";
    else
        cout << "Two are equal";
    cout << endl;

    cout << endl;
    FeetInches dividedfeetinches = feetinches1 / feetinches2;
    feetinches1.display();
    cout << " divided ";
    feetinches2.display();
    cout << " is ";
    dividedfeetinches.display();
    cout << endl;

    return 0;
}
```

Sample Output

D:\OOP\作业\作业4\4.4 Overloading operators_FeetInches\4.4.exe

```
Enter a length in feet and inches:
Feet:   4
Inches: 5
Enter a length in feet and inches:
Feet:   6
Inches: 9

4 feet, 5 inches sum 6 feet, 9 inches is 11 feet, 2 inches

4 feet, 5 inches times 6 feet, 9 inches is 27 feet, 9 inches

4 feet, 5 inches is not greater than and equal to 6 feet, 9 inches

Two are not equal

4 feet, 5 inches divided 6 feet, 9 inches is 0 feet, 0 inches

--------------------------------
Process exited after 14.14 seconds with return value 0
请按任意键继续. . .
```