

Polymorphism in C++

Russell Feldhausen
russfeldh@gmail.com

Previously...

- Classes and Objects in C++
- `private` and `public` Members
- Constructors
- Function Overloading
- Pointers and Dereferencing

```
class Person{
    private:
        int age;
    public:
        Person(int);
        int happyBirthday();
};

Person::Person(int anAge){
    age = anAge;
}

int Person::happyBirthday(){
    age = age + 1;
    return age;
}
```

Class Hierarchy

- Person
 - Student
 - Teacher
 - Administrator
- Shape
 - Rectangle
 - Square
 - Circle

Think-Pair-Share

What other examples can you come up with?

Polymorphism

noun: *The ability to assume different forms or shapes*

In programming, polymorphism allows an object to assume different forms and perform different operations based on its type.

An object can assume multiple types based on its class hierarchy (or inheritance).

```
class Person{
public:
    int age;
    void happyBirthday();
};
class Student : public Person{
public:
    int major;
    void changeMajor();
};
class Teacher: public Person{
public:
    int title;
    void promotion();
};
class Administrator: public Teacher{
public:
    int salary;
    void promotion();
};

void Person::happyBirthday() {
    age = age + 1;
}
void Student::changeMajor() {
    major = major + 1;
}
void Teacher::promotion() {
    title = title + 1;
}
void Administrator::promotion() {
    title = title + 1;
    salary = salary + 1;
}
```

```
class Person{
public:
    int age;
    void happyBirthday();
};
class Student : public Person{
public:
    int major;
    void changeMajor();
};
class Teacher: public Person{
public:
    int title;
    void promotion();
};
class Administrator: public Teacher{
public:
    int salary;
    void promotion();
};
void Person::happyBirthday() {
    age = age + 1;
}
void Student::changeMajor() {
    major = major + 1;
}
void Teacher::promotion() {
    title = title + 1;
}
void Administrator::promotion() {
    title = title + 1;
    salary = salary + 1;
}

int main(){
    Person person;
    person.age = 32
    person.happyBirthday();
    cout << person.age << endl;
    return 0;
}
```

```
class Person{
public:
    int age;
    void happyBirthday();
};
class Student : public Person{
public:
    int major;
    void changeMajor();
};
class Teacher: public Person{
public:
    int title;
    void promotion();
};
class Administrator: public Teacher{
public:
    int salary;
    void promotion();
};
void Person::happyBirthday() {
    age = age + 1;
}
void Student::changeMajor() {
    major = major + 1;
}
void Teacher::promotion() {
    title = title + 1;
}
void Administrator::promotion() {
    title = title + 1;
    salary = salary + 1;
}

int main(){
    Person person;
    person.age = 32
    person.happyBirthday();
    cout << person.age << endl; // 33
    return 0;
}
```

```
class Person{
public:
    int age;
    void happyBirthday();
};
class Student : public Person{
public:
    int major;
    void changeMajor();
};
class Teacher: public Person{
public:
    int title;
    void promotion();
};
class Administrator: public Teacher{
public:
    int salary;
    void promotion();
};
void Person::happyBirthday() {
    age = age + 1;
}
void Student::changeMajor() {
    major = major + 1;
}
void Teacher::promotion() {
    title = title + 1;
}
void Administrator::promotion() {
    title = title + 1;
    salary = salary + 1;
}

int main(){
    Student student;
    Person* person = &student;
    student.age = 42;
    student.major = 2;
    student.happyBirthday();
    student.changeMajor();
    cout << student.age << endl;
    cout << person->age << endl;
    cout << student.major << endl;
    return 0;
}
```

```
class Person{
public:
    int age;
    void happyBirthday();
};
class Student : public Person{
public:
    int major;
    void changeMajor();
};
class Teacher: public Person{
public:
    int title;
    void promotion();
};
class Administrator: public Teacher{
public:
    int salary;
    void promotion();
};
void Person::happyBirthday() {
    age = age + 1;
}
void Student::changeMajor() {
    major = major + 1;
}
void Teacher::promotion() {
    title = title + 1;
}
void Administrator::promotion() {
    title = title + 1;
    salary = salary + 1;
}

int main(){
    Student student;
    Person* person = &student;
    student.age = 42;
    student.major = 2;
    student.happyBirthday();
    student.changeMajor();
    cout << student.age << endl; // 43
    cout << person->age << endl; // 43
    cout << student.major << endl; // 3
    return 0;
}
```

```
class Person{
public:
    int age;
    void happyBirthday();
};
class Student : public Person{
public:
    int major;
    void changeMajor();
};
class Teacher: public Person{
public:
    int title;
    void promotion();
};
class Administrator: public Teacher{
public:
    int salary;
    void promotion();
};
void Person::happyBirthday() {
    age = age + 1;
}
void Student::changeMajor() {
    major = major + 1;
}
void Teacher::promotion() {
    title = title + 1;
}
void Administrator::promotion() {
    title = title + 1;
    salary = salary + 1;
}
```

```
int main(){
    Administrator admin;
    Teacher* teacher = &admin;
    Person* person = &admin;
    admin.age = 25;
    admin.title = 5;
    admin.salary = 12;
    teacher->happyBirthday();
    admin.promotion();
    cout << person->age << endl;
    cout << teacher->title << endl;
    cout << admin.salary << endl;
    return 0;
}
```

```
class Person{
public:
    int age;
    void happyBirthday();
};
class Student : public Person{
public:
    int major;
    void changeMajor();
};
class Teacher: public Person{
public:
    int title;
    void promotion();
};
class Administrator: public Teacher{
public:
    int salary;
    void promotion();
};
void Person::happyBirthday() {
    age = age + 1;
}
void Student::changeMajor() {
    major = major + 1;
}
void Teacher::promotion() {
    title = title + 1;
}
void Administrator::promotion() {
    title = title + 1;
    salary = salary + 1;
}
```

```
int main(){
    Administrator admin;
    Teacher* teacher = &admin;
    Person* person = &admin;
    admin.age = 25;
    admin.title = 5;
    admin.salary = 12;
    teacher->happyBirthday();
    admin.promotion();
    cout << person->age << endl; // 26
    cout << teacher->title << endl; // 6
    cout << admin.salary << endl; // 13
    return 0;
}
```

Subclass (Child or Derived Class)

- Can access superclass (parent or base class)
`public` members
- Can access own `public` & `private` members

What if I want something accessible by subclasses but NOT everyone?

Subclass (Child or Derived Class)

- Can access superclass (parent or base class)
`public` members
- Can access own `public` & `private` members

What if I want something accessible by subclasses but NOT everyone? **Use the `protected` access modifier!**

```
class Person{
    protected:
    int age;
    void happyBirthday();
};
class Student : public Person{
    public:
    int major;
    void setAge(int);
    int getAge();
    void getOlder();
    void changeMajor();
};
```

```
void Person::happyBirthday() {
    age = age + 1;
}
void Student::setAge(int anAge) {
    age = anAge;
}
int Student::getAge() {
    return age;
}
void Student::getOlder() {
    happyBirthday();
}
void Student::changeMajor() {
    major = major + 1;
}
```

```
class Person{
protected:
    int age;
    void happyBirthday();
};
class Student : public Person{
public:
    int major;
    void setAge(int);
    int getAge();
    void getOlder();
    void changeMajor();
};
void Person::happyBirthday() {
    age = age + 1;
}
void Student::setAge(int anAge) {
    age = anAge;
}
int Student::getAge() {
    return age;
}
void Student::getOlder() {
    happyBirthday();
}
void Student::changeMajor() {
    major = major + 1;
}
```

```
int main(){
    Student student;
    student.age = 42;
    student.major = 2;
    student.happyBirthday();
    student.changeMajor();
    cout << student.age << endl;
    cout << student.major << endl;
    return 0;
}
```

```
class Person{
protected:
    int age;
    void happyBirthday();
};
class Student : public Person{
public:
    int major;
    void setAge(int);
    int getAge();
    void getOlder();
    void changeMajor();
};
void Person::happyBirthday() {
    age = age + 1;
}
void Student::setAge(int anAge) {
    age = anAge;
}
int Student::getAge() {
    return age;
}
void Student::getOlder() {
    happyBirthday();
}
void Student::changeMajor() {
    major = major + 1;
}
```

```
int main(){
    Student student;
    student.age = 42; // protected!
    student.major = 2;
    student.happyBirthday(); // protected!
    student.changeMajor();
    cout << student.age << endl; // protected
    !
    cout << student.major << endl;
    return 0;
}
```

```
class Person{
protected:
    int age;
    void happyBirthday();
};
class Student : public Person{
public:
    int major;
    void setAge(int);
    int getAge();
    void getOlder();
    void changeMajor();
};
void Person::happyBirthday() {
    age = age + 1;
}
void Student::setAge(int anAge) {
    age = anAge;
}
int Student::getAge() {
    return age;
}
void Student::getOlder() {
    happyBirthday();
}
void Student::changeMajor() {
    major = major + 1;
}
```

```
int main(){
    Student student;
    student.setAge(42);
    student.major = 2;
    student.getOlder();
    student.changeMajor();
    cout << student.getAge() << endl;
    cout << student.major << endl;
    return 0;
}
```

```
class Person{
protected:
    int age;
    void happyBirthday();
};
class Student : public Person{
public:
    int major;
    void setAge(int);
    int getAge();
    void getOlder();
    void changeMajor();
};
void Person::happyBirthday() {
    age = age + 1;
}
void Student::setAge(int anAge) {
    age = anAge;
}
int Student::getAge() {
    return age;
}
void Student::getOlder() {
    happyBirthday();
}
void Student::changeMajor() {
    major = major + 1;
}
```

```
int main(){
    Student student;
    student.setAge(42);
    student.major = 2;
    student.getOlder();
    student.changeMajor();
    cout << student.getAge() << endl; // 43
    cout << student.major << endl; // 3
    return 0;
}
```

Subclass (Child or Derived Class)

- Can access superclass (parent or base class)
`public & protected members`
- Can access own `public, protected & private` members

Can we have the same function in both a subclass and its superclass, but with different code?

Subclass (Child or Derived Class)

- Can access superclass (parent or base class)
`public & protected members`
- Can access own `public, protected & private` members

Can we have the same function in both a subclass and its superclass, but with different code? **Yes!**

```
class Person{
    protected:
        int age;
    public:
        void whoAmI();
};

class Student : public Person{
    public:
        void whoAmI();
};
```

```
void Person::whoAmI() {
    cout << "Parent" << endl;
}

void Student::whoAmI() {
    cout << "Student" << endl;
}
```

```
class Person{
protected:
    int age;
public:
    void whoAmI();
};

class Student : public Person{
public:
    void whoAmI();
};

void Person::whoAmI() {
    cout << "Parent" << endl;
}

void Student::whoAmI() {
    cout << "Student" << endl;
}
```

```
int main(){
    Person person;
    Student student;
    person.whoAmI();
    student.whoAmI();
    return 0;
}
```

```
class Person{
protected:
    int age;
public:
    void whoAmI();
};

class Student : public Person{
public:
    void whoAmI();
};

void Person::whoAmI() {
    cout << "Parent" << endl;
}

void Student::whoAmI() {
    cout << "Student" << endl;
}
```

```
int main(){
    Person person;
    Student student;
    person.whoAmI(); // Parent
    student.whoAmI(); // Student
    return 0;
}
```

```
class Person{
protected:
    int age;
public:
    void whoAmI();
};

class Student : public Person{
public:
    void whoAmI();
};

void Person::whoAmI() {
    cout << "Parent" << endl;
}

void Student::whoAmI() {
    cout << "Student" << endl;
}
```

```
int main(){
    Student student;
    Person* person = &student;
    student.whoAmI();
    person->whoAmI();
    return 0;
}
```

```
class Person{
protected:
    int age;
public:
    void whoAmI();
};

class Student : public Person{
public:
    void whoAmI();
};

void Person::whoAmI() {
    cout << "Parent" << endl;
}

void Student::whoAmI() {
    cout << "Student" << endl;
}
```

```
int main(){
    Student student;
    Person* person = &student;
    student.whoAmI(); // Student
    person->whoAmI(); // Parent
    return 0;
}
```

Function Overriding

The function in the superclass is
overridden
by the function in the subclass.

Can we have it *always* use the subclass code, even if the current type is the superclass?

Function Overriding

The function in the superclass is
overridden
by the function in the subclass.

Can we have it *always* use the subclass code, even if the current type is the superclass? **Use the `virtual` keyword!**

```
class Person{
    protected:
        int age;
    public:
        virtual void whoAmI();
};

class Student : public Person{
    public:
        void whoAmI();
};
```

```
void Person::whoAmI() {
    cout << "Parent" << endl;
}

void Student::whoAmI() {
    cout << "Student" << endl;
}
```

```
class Person{
protected:
    int age;
public:
    virtual void whoAmI();
};

class Student : public Person{
public:
    void whoAmI();
};

void Person::whoAmI() {
    cout << "Parent" << endl;
}

void Student::whoAmI() {
    cout << "Student" << endl;
}
```

```
int main(){
    Person person;
    Student student;
    person.whoAmI();
    student.whoAmI();
    return 0;
}
```

```
class Person{
protected:
    int age;
public:
    virtual void whoAmI();
};

class Student : public Person{
public:
    void whoAmI();
};

void Person::whoAmI() {
    cout << "Parent" << endl;
}

void Student::whoAmI() {
    cout << "Student" << endl;
}
```

```
int main(){
    Person person;
    Student student;
    person.whoAmI(); // Parent
    student.whoAmI(); // Student
    return 0;
}
```

```
class Person{
protected:
    int age;
public:
    virtual void whoAmI();
};

class Student : public Person{
public:
    void whoAmI();
};

void Person::whoAmI() {
    cout << "Parent" << endl;
}

void Student::whoAmI() {
    cout << "Student" << endl;
}
```

```
int main(){
    Student student;
    Person* person = &student;
    student.whoAmI();
    person->whoAmI();
    return 0;
}
```

```
class Person{
protected:
    int age;
public:
    virtual void whoAmI();
};

class Student : public Person{
public:
    void whoAmI();
};

void Person::whoAmI() {
    cout << "Parent" << endl;
}

void Student::whoAmI() {
    cout << "Student" << endl;
}
```

```
int main(){
    Student student;
    Person* person = &student;
    student.whoAmI(); // Student
    person->whoAmI(); // Student
    return 0;
}
```

Virtual Functions

Virtual functions can be
redefined
within subclasses.

Do we always have to implement the function in the superclass?

Virtual Functions

Virtual functions can be
redefined
within subclasses.

Do we always have to implement the function in the superclass? **No! Make the superclass an *abstract* class with *pure virtual* functions**

```
class Person{
    protected:
        int age;
    public:
        virtual void whoAmI() = 0;
};
```

```
class Student : public Person{
    public:
        void whoAmI();
};
```

```
void Student::whoAmI() {
    cout << "Student" << endl;
}
```

```
class Person{
protected:
    int age;
public:
    virtual void whoAmI() = 0;
};

class Student : public Person{
public:
    void whoAmI();
};

void Student::whoAmI() {
    cout << "Student" << endl;
}
```

```
int main(){
    Student student;
    Person person;
    person.whoAmI();
    student.whoAmI();
    return 0;
}
```

```
class Person{
protected:
    int age;
public:
    virtual void whoAmI() = 0;
};

class Student : public Person{
public:
    void whoAmI();
};

void Student::whoAmI() {
    cout << "Student" << endl;
}
```

```
int main(){
    Student student;
    Person person; // cannot instantiate
    person.whoAmI();
    student.whoAmI();
    return 0;
}
```

```
class Person{
protected:
    int age;
public:
    virtual void whoAmI() = 0;
};

class Student : public Person{
public:
    void whoAmI();
};

void Student::whoAmI() {
    cout << "Student" << endl;
}
```

```
int main(){
    Student student;
    Person* person = &student;
    student.whoAmI();
    person->whoAmI();
    return 0;
}
```

```
class Person{
protected:
    int age;
public:
    virtual void whoAmI() = 0;
};

class Student : public Person{
public:
    void whoAmI();
};

void Student::whoAmI() {
    cout << "Student" << endl;
}
```

```
int main(){
    Student student;
    Person* person = &student; // pointer ok
    student.whoAmI(); // Student
    person->whoAmI(); // Student
    return 0;
}
```

Review

- **Polymorphism** - class hierarchies
- **protected keyword** - self & subclass access only
- **Function Overriding** - different code in superclass and subclass
- **virtual keyword** - redefine functions in subclass
- **Abstract Class** - virtual function with no definition
 - Cannot be instantiated!

Next Steps

- Constructor Chaining
- Multiple Inheritance
 - The Diamond Problem
- Dynamic Allocation - `new` and `delete`
- Changing Type - `dynamic_cast`
- Generic Types & Templates

Assignment - Vehicles

- Vehicle (abstract)
 - Car
 - Truck
 - Motorcycle
 - Scooter

Methods for vehicle type and tax rate

Practice with `virtual` functions and `protected` members

