فصل چهارم

چندریختی (Polymorphism)

مقدمه

- با توابع مجازی(Virtual Functions) و چندریختی (Polymorphism) می توان سیستم هایی را طراحی و پیاده سازی کرد که به آسانی قابل توسعه باشند.
- در c++ چندریختی هم در زمان ترجمه و هم در زمان اجرا امکان پذیر است.
  - چندریختی زمان ترجمه: تعریف مجدد توابع و عملگرها
  - چندریختی زمان اجرا: وراثت و توابع مجازی

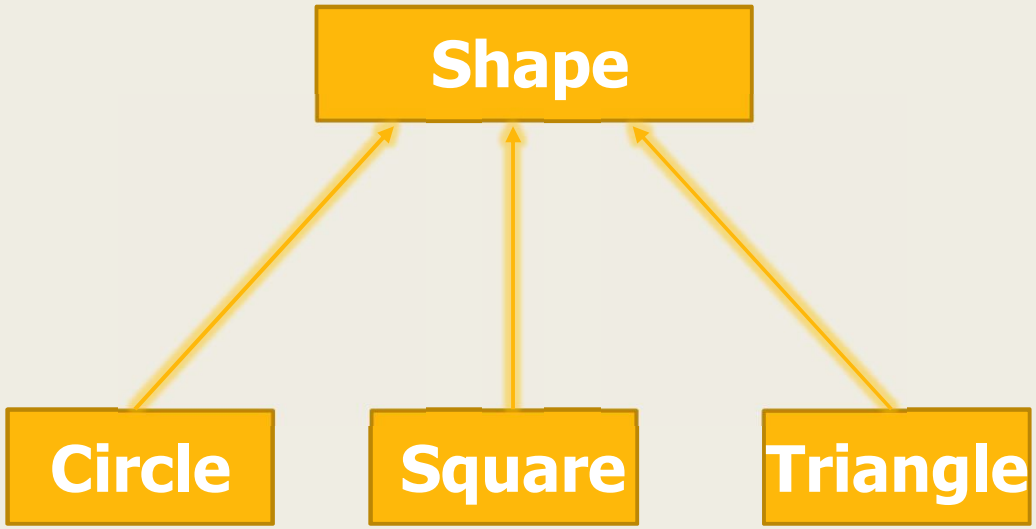
تابع مجازی

- یک تابع عضو است که در یک کلاس پایه اعلان می شود و دوباره در کلاس مشتق تعریف می گردد.

```
virtual OutputType FunctionName(Inputs)
{
      //Function Codes
};
```

تابع مجازی

- در شی گرائی می توان به هرکدام از کلاس های مشتق شده زیر قابلیتی اضافه کرد که خودشان را رسم کنند. برای انجام چنین کاری تابع رسم در کلاس پایه به صورت مجازی تعریف می شود و هر یک از توابع موجود در کلاس های مشتق را با اعضای جدیدی می نویسیم تا شکل مناسب را رسم کند.



تابع مجازی

```
class Base
{
    public:
      virtual void vFunction(){
            cout <<  "This is vFunction in Base class . " <<
endl;
      }
};
class Derived1:public Base
{
    public:
      void vFunction(){
            cout << "This is vFunction in Derived1 . " << endl;
      }
};

class Derived2:public Base
{
    public:
      void vFunction(){
            cout << "This is vFunction in Derived2 . " << endl;
      }
};
```

تابع مجازی

```
int main()
{
 Base *p, b;
 Derived1 d1;
 Derived2 d2;
 cout << endl << "in Base class..." << endl;
 p = &b;
 p -> vFunction();
 cout <<endl<<"in Derived1 class..." <<endl;
 p = &d1;
 p -> vFunction();
 cout <<endl<<"in Derived2 class..." <<endl;
 p = &d2;
 p -> vFunction();
}
```

فراخوانی تابع مجازی از طریق مرجع کلاس پایه

- مرجع یک اشاره گر ضمنی است بنابراین، مرجع کلاس می تواند به شیئی از کلاس پایه یا هر شیئی از کلاسی که از کلاس پایه مشتق شده اشاره کند.
- وقتی تابع مجازی به وسیله مرجع کلاس پایه فراخوانی می شود، نسخه ای از تابع که اجرا می گردد، با توجه به شیئی که در هنگام فراخوانی به آن مراجعه می کند، تعیین می شود.
- متداول ترین حالتی که تابع مجازی از طریق مرجع کلاس پایه فراخوانی می شود، هنگامی است که مرجع به عنوان پارامتر تابع است.

فراخوانی تابع مجازی از طریق مرجع کلاس پایه

```
class baseClass
{
    public:
      virtual void vFunction(){
            cout <<  "This is vFunction in Base class . " <<
endl;
      }
};
class derivedClass1 : public baseClass
{
    public:
      void vFunction(){
            cout << "This is vFunction in Derived1 . " << endl;
      }
};

class derivedClass2 : public baseClass
{
    public:
      void vFunction(){
            cout << "This is vFunction in Derived2 . " << endl;
      }
};
```

فراخوانی تابع مجازی از طریق مرجع کلاس پایه

```
void refParameter(baseClass &ref)
{
    ref.vFunction();
}
int main()
{
    baseClass baseObject;
    derivedClass1 derivedObject1;
    derivedClass2 derivedObject2;
    refParameter(baseObject);
    refParameter(derivedObject1);
    refParameter(derivedObject2);
}
```

ارث بری صفت مجازی

- وقتی تابع مجازی توسط کلاسی به ارث برده می شود، ماهیت مجازی بودن آن نیز به ارث برده خواهد شد. یعنی اگر یک کلاس مشتق، یک تابع مجازی را از کلاس پایه ای به ارث ببرد و این کلاس مشتق به عنوان کلاس پایه ای برای کلاس مشتق دیگری محسوب شود، تابع مجازی در این کلاس مشتق نیز قابل استفاده است.

ارث بری صفت مجازی

```
class Base
{
    public:
      virtual void vFunction(){
            cout <<  "This is vFunction in Base class . " <<
endl;
      }
};
class Derived1:public Base
{
    public:
      void vFunction(){
            cout << "This is vFunction in Derived1 . " << endl;
      }
};

class Derived2 : public Derived1
{
    public:
      void vFunction(){
            cout << "This is vFunction in Derived2 . " << endl;
      }
};
```

ارث بری صفت مجازی

```
int main()
{
 Base *p, b;
 Derived1 d1;
 Derived2 d2;
 cout <<endl <<"in Base class... " <<endl;
 p = &b;
 p -> vFunction();
 cout <<endl <<"in Derived1 ... " <<endl;
 p = &d1;
 p -> vFunction();
 cout <<endl <<"in Derived2 ... " <<endl;
 p = &d2;
 p -> vFunction();
}
```

سلسله مراتبی بودن توابع مجازی

- وقتی تابعی در کلاس پایه به صورت مجازی تعریف شد، توسط کلاس مشتق مجددا قابل تعریف است. اما لازم نیست کلاس مشتق حتما آن را دوباره تعریف نماید.
- اگر کلاس مشتق، تابع مجازی را دوباره در خودش تعریف نکند، هنگامی که شیئی از آن کلاس مشتق به آن تابع مراجعه کند، تابعی که در کلاس پایه تعریف شده مورد استفاده قرار می گیرد

ارث بری صفت مجازی

```
class Base
{
    public:
     virtual void vFunction(){
          cout <<  "This is vFunction in Base class . "
<< endl;
      }
};
class Derived1:public Base
{
    public:
     void vFunction(){
          cout << "This is vFunction in Derived1  . " <<
endl;
      }
};

class Derived2 : public Derived1
{
    public:
};
```

ارث بری صفت مجازی

```
int main()
{
 Base *p, b;
 Derived1 d1;
 Derived2 d2;
 cout <<endl <<"in Base class... " <<endl;
 p = &b;
 p -> vFunction();
 cout <<endl <<"in Derived1 ... " <<endl;
 p = &d1;
 p -> vFunction();
 cout <<endl <<"in Derived2 ... " <<endl;
 p = &d2;
 p -> vFunction();
}
```