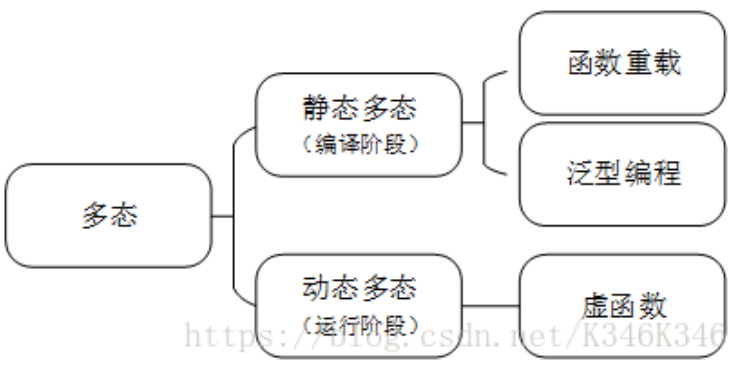
詳解C++ 多型的兩種形式（靜態、動態）

阿新 • 來源：網路 • 發佈：2020-08-31

1.多型的概念與分類

多型（Polymorphisn）是面向物件程式設計（OOP）的一個重要特徵。多型字面意思為多種狀態。在面嚮物件語言中，一個介面，多種實現即為多型。C++中的多型性具體體現在編譯和執行兩個階段。編譯時多型是靜態多型，在編譯時就可以確定使用的介面。執行時多型是動態多型，具體引用的介面在執行時才能確定。



靜態多型和動態多型的區別其實只是在什麼時候將函式實現和函式呼叫關聯起來，是在編譯時期還是執行時期，即函式地址是早繫結還是晚繫結的。靜態多型是指在編譯期間就可以確定函式的呼叫地址，並生產程式碼，這就是靜態的，也就是說地址是早繫結。靜態多型往往也被叫做靜態聯編。 動態多型則是指函式呼叫的地址不能在編譯器期間確定，需要在執行時確定，屬於晚繫結，動態多型往往也被叫做動態聯編。

2.多型的作用

為何要使用多型呢？封裝可以使得程式碼模組化，繼承可以擴充套件已存在的程式碼，他們的目的都是為了程式碼重用。而多型的目的則是為了介面重用。靜態多型，將同一個介面進行不同的實現，根據傳入不同的引數（個數或型別不同）呼叫不同的實現。動態多型，則不論傳遞過來的哪個類的物件，函式都能夠通過同一個介面呼叫到各自物件實現的方法。

3.靜態多型

靜態多型往往通過函式過載和模版（泛型程式設計）來實現，具體可見下面程式碼：

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

//兩個函式構成過載
int add(int a,int b)
{
  cout<<"in add_int_int()"<<endl;
  return a + b;
}
double add(double a,double b)
{
  cout<<"in add_double_doube()"<<endl;
  return a + b;
}

//函式模板（泛型程式設計）
template <typename T>
T add(T a,T b)
{
  cout<<"in func tempalte"<<endl;
  return a + b;
}

int main()
{
  cout<<add(1,1)<<endl;                  //呼叫int add(int a,int b)
  cout<<add(1.1,1.1)<<endl;              //呼叫double add(double a,double b)
  cout<<add<char>('A',' ')<<endl;      //呼叫模板函式，輸出小寫字母a
}
```

程式輸出結果：

```
in add_int_int()
2
in add_double_doube()
2.2
in func tempalte
a
```

4.動態多型

動態多型最常見的用法就是宣告基類的指標，利用該指標指向任意一個子類物件，呼叫相應的虛擬函式，可以根據指向的子類的不同而呼叫不同的方法。如果沒有使用虛擬函式，即沒有利用C++多型性，則利用基類指標呼叫相應函式的時候，將總被限制在基類函式本身，而無法呼叫到子類中被重寫過的函式。因為沒有多型性，函式呼叫的地址將是一定的，而固定的地址將始終呼叫同一個函式，這就無法“實現一個介面，多種實現”的目的了。

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

class Base
{
public:
  virtual void func()
  {
    cout << "Base::fun()" << endl;
  }
};

class Derived : public Base
{
public:
  virtual void func()
  {
   cout << "Derived::fun()" << endl;
  }

};

int main()
{
  Base* b=new Derived;       //使用基類指標指向派生類物件
  b->func();              //動態繫結派生類成員函式func

  Base& rb=*(new Derived);     //也可以使用引用指向派生類物件
  rb.func();
}
```

程式輸出結果：

```
Derived::fun()
Derived::fun()
```

通過上面的例子可以看出，在使用基類指標或引用指向子類物件時，呼叫的函式是子類中重寫的函式，這樣就實現了執行時函式地址的動態繫結，即動態聯編。動態多型是通過“繼承+虛擬函式”來實現的，只有在程式執行期間（非編譯期）才能判斷所引用物件的實際型別，根據其實際型別呼叫相應的方法。具體格式就是使用virtual關鍵字修飾類的成員函式時，指明該函式為虛擬函式，並且派生類需要重新實現該成員函式，編譯器將實現動態繫結。

以上就是詳解C++ 多型的兩種形式（靜態、動態）的詳細內容，更多關於C++ 靜態多型和動態多型的資料請關注我們其它相關文章！

動態多型 多型 靜態多型

« 上一篇 IDEA 錯誤之找不到或無法載入主類的問題

下一篇 » SpringCloud容器化服務發現及註冊實現方法解析