コールバック関数としてのC++仮想関数

阿　部　　司*

Virtual Function in C ++ as a Call Back Function

Tsukasa ABE

Abstruct

Call back functions are used to handle exceptions in operating systems. In window systems, call back function are used to redraw window. Call back functions are depends on operating systems and window systems. Virtual function in C ++ can be used as a call back function, and independent form operating systems and window systems.

1．はじめに

一般に関数呼び出しは，ある関数から他の関数，さらにその関数からまた次の関数を呼び出すというように，入れ子状に関数を呼び出していくのが普通である。同様に，通常はアプリケーションプログラムがオペレーティングシステムを呼び出す。

逆にオペレーティングシステムからアプリケーションプログラムが呼び出される場合もあり，例外処理が典型的な例である。アプリケーションプログラムは実行開始直後に，例外発生時に呼び出して欲しいコールバック関数をオペレーティングシステムに登録する。オペレーティングシステムは例外を検出すると，このコールバック関数を呼び出す。つまり，オペレーティングシステムが標準的に提供する例外処理ではない，アプリケーションプログラム独自の例外処理が可能である。

イベント駆動型システムの典型であるウィンドウシステムでも，画面上の破壊された領域を再描画する必要性があるとき，その領域を所有しているアプリケーションプログラムのコールバック関数を呼び出して，再描画させるのが一般的である。

コールバック関数は，オペレーティングシステムまたはウィンドシステムの機能として実装され，アプリケーションプログラムは適当なコールバック関数を用意し，登録しなければならない。

オブジェクト指向形言語として注目されているC ++言語では，コールバック関数を，仮想関数によって実装することが可能である。本論文では，エディタへの実装を例として，コールバック関数としてのC ++言語の仮想関数の実装例を述べる。

2．エディタへの実装

エディタへの実装例では，編集テキストを記憶するテキストバッファ部，編集テキストを画面に表示する画面表示部，編集制御部のみ示す。

2．1　C言語による実装

(1)　画面表示部

```
void AfterInsert (Buff *FirstLine) { ... }
void BeforeDelete (Buff *FirstLine) { ... }
```

関数 AfterInsert ( )は，行挿入直後に呼び出されるコールバック関数，関数 BeforeDelete ( )は，行削除直前に呼び出されるコールバック関数である。

(2)　編集制御部

＊　助教授　情報工学科

```
CommandCode GetCommand (void) { .... }
void main (void) {
        Buff *FirstLine;
        char Line[MAX_LINE_CHARS];
        RegAfterInsert(AfterInsert);
        RegBeforeDelete(BeforeDelete);
        switch (GetCommand()) {
        case CmdDeleteLines :
                DeleteLines(FirstLine);
                break;
         case CmdInsertLine :
                 InsertLine(FirstLine, Line);
                 break;
                 }
         }
```

関数 main ( )は，コールバック関数を登録するのに関数 RegAfterInset ( ) と RegBeforeDelete ( )を使う。

(3) テキストバッファ部

```
void (*FuncAfterInsert)(Buff *) = NULL;
void (*FuncBeforeDelete)(Buff *) = NULL;
void RegAfterInsert (void (*Function)(Buff *)) {
        FuncAfterInsert = Function;
        }
void RegBeforeDelete (void (*Function)(Buff *)) {
        FuncBeforeDelete = Function;
        }
void InsertLine (Buff *TargetLine, char *Line) {
        Insert(TargetLine, Line);
        FuncAfterInsert(TargetLine);
        }
void DeleteLines (Buff *FirstLine) {
        FuncBeforeDelete(FirstLine);
        Delete(FirstLine);
        }
```

関数 RegAfterInsert ( ) と RegBeforeDelete ( )は，それぞれ行挿入直後と行削除直前に呼び出すコールバック関数を登録する。

行挿入時に関数 InsertLine ( )は行挿入後，コールバック関数 AfterInsert ( )を呼び出す。

行削除時に関数 DeleteLines ( )は，コールバック関数 BeforeDelete ( )を呼び出した後，行を削除する。

2．2 C++言語による実装

(1) 画面表示部

```
class ScreenClass : public BuffClass {
protected:
```

```
        virtual void AfterInsert(LineClass *FirstLine) { ... };
        virtual void BeforeDelete(LineClass *FirstLine) { ... };
    public:
        ScreenClass (void) : BuffClass() {};
        };
```

関数 AfterInsert ( )と BeforeDelete ( )は，それぞれ行挿入直後と行削除直前に呼び出される仮想関数である。

(2) 編集制御部

```
CommandCode GetCommand (void) { ... }
void main (void) {
    ScreenClass Screen;
    LineClass *FirstLine;
    char Line[MaxLineChars];
    switch (GetCommand()) {
    case CmdDeleteLines :
        Screen.DeleteLines(FirstLine);
        break;
    case CmdInsertLine :
        Screen.InsertLine(FirstLine, Line);
        break;
        }
    }
```

(3) テキストバッファ部

```
class BuffClass {
    void Insert(LineClass *TargetLine, char *Line) {};
    void Delete(LineClass *FirstLine) {};
protected:
    virtual void AfterInsert(LineClass *FirstLine) {};
    virtual void BeforeDelete(LineClass *FirstLine) {};
public:
    BuffClass (void) {};
    void DeleteLines (LineClass *FirstLine){
        BeforeDelete(FirstLine);
        Delete(FirstLine);
        };
    void InsertLine (LineClass *TargetLine, char *Line) {
        Insert(TargetLine, Line);
        AfterInsert(TargetLine);
        };
    };
```

C++言語では，仮想関数を定義するだけで，基底クラスから最も遠い派生クラスの仮想関数が呼び出される。

この例では，行挿入時に関数 InsertLine ( )は行挿入後，関数 AfterInsert ( )を呼び出すが，呼び出されるのは ScreenClass の AfterInsert ( )で

ある。

ここで，基底クラスは派生クラスの知識がなくても仮想関数を呼び出し可能なことが重要である。この例では，1個の派生クラスだけだが，派生が多層になっても，基底クラスの内容を変更する必要はない。基底クラス BuffClass から2層のクラスを挟んでクラス ScreenClass を派生する例を示す。

```
class BuffClass {
protected:
        virtual void AfterInsert(LineClass *FirstLine) {};
public:
        void InsertLine (LineClass *TargetLine, char *Line) {
                Insert(TargetLine, Line);
                AfterInsert(TargetLine);
                };
        };
class DerivedClass1 : public BuffClass {
        };
class DerivedClass2 : public DerivedClass1 {
protected:
        virtual void AfterInsert(LineClass *FirstLine) {
                ... ;
                DerivedClass1::AfterInsert(FirstLine);
                };
        };
class ScreenClass : public DerivedClass2 {
protected:
        virtual void AfterInsert(LineClass *FirstLine) {
                ... ;
```

この例では，行挿入時に関数 InsertLine ( )のからは，基底クラスから最も遠い派生クラス ScreenClass の AfterInsert ( )が呼び出される。ScreenClass の AfterInsert ( )は，挿入時の画面描画後，一つ前のクラス DerivedClass2 の AfterInsert ( )を呼び出す。同様に，DerivedClass2 の AfterInsert ( )は，挿入操作後，一つ前のクラス DerivedClass1 の AfterInsert ( )を呼び出す。こうすることで，各階層の派生クラスで必要な操作を順次行なうことができる。また，ある階層の派生クラスで不必要なら仮想関数を定義しなくてもよい。この場合，その前のクラスの仮想関数が呼び出される。例えば，DerivedClass1 には AfterInsert ( )が未定義なので，DerivedClass2 が呼び出す AfterInsert ( )は，実際には BaseClass の AfterInsert ( )である。

C 言語でも，最も遠い派生クラスの概念に相当する最も外側の関数から，内側の関数を順次呼び出すことはできる。しかし，コールバック関数を管理するための機能を自分で実装しなければならないのに加えて，各階層間でコールバック関数の登録順番には十分に注意しなければならない。

3. まとめ

C ++言語の仮想関数をコールバック関数として使うことができることを示した。この場合の利点をまとめると次のようになる。

(1) コールバック関数を登録する必要はない。したがって，登録順番の管理なども不要である。

(2) 一つ前の基本クラスの仮想関数を呼び出すことで，最も遠い派生クラスから，各階層の派生クラスで必要な操作を順次実行できる。

(3) 仮想関数を定義しなければ自動的に，一つ前のクラスの仮想関数が呼び出される。

(4) システム毎に異なっているコールバック関数の管理が不要になる事が最も重要である。仮想関数は C ++言語の一部として実装されており，

オペレーティングシステムまたはウィンドウシステムに独立な，移植性の高いプログラムを作成することができる。

C++言語の仮想関数のコールバック関数的一面を示したが，C++言語はこれだけでなく数々のオブジェクト指向的な機能を持っている。C++言語は現在でも仕様の拡張が続いており，ソフトウェアの生産性向上に寄与することは間違いない。

4．参考文献

1） Bjarne Stroustrup：C++ Programming Language, Addison-Wesley, 1986

2） Margaret A. Ellis and Bjarne Stroustrup, The Annotated C++ Reference Manual, Addison-Wesley, 1992
邦訳：足立高徳，小山裕司共訳，注解C++リファレンスマニュアル，トッパン，1992

3） Microsoft Visual C++ Development System for Windows Version 1.0ランゲージリファレンス，マイクロソフト株式会社，1993

（平成6年11月29日受理）