〈実践報告・資料〉

# 図形を使ったC++教育法

港湾職業能力開発短期大学校横浜校　　内藤光明

An Education of C++ using graphics

Mitsuaki NAITOH

**要約**　オブジェクト指向(OO:Object Orient)が唱えられ、プログラミング言語C++に興味をいだく人が増えてきた。しかし、初心者に対しC++のオブジェクト指向的側面を文法のみで具体的理由付けなしに説明する事は難しい。筆者は図形を利用してC++のオブジェクト指向の概念を従来型プログラミング手法と比較して説明する方法を考案し、実際に能開セミナーで実施した。この方法は学習者の興味をひきやすいと考えられる。

## I　はじめに

ソフトウェア開発は複雑な作業であり多くのマンパワーを要する。ソフトウェア開発は開発者個人の技能に頼った方法から工学的方法へと進化してきた。しかし、従来のプログラミング手法ではソフトウェアの再利用性などが悪く、生産性や保守性は期待されるほど向上しなかった。それに対する解決方法としてオブジェクト指向(OO:Object Orient)が唱えられた。従来のプログラミング手法が手続き中心でプロセス指向であったのに対しオブジェクト指向プログラミング手法はデータ指向である。オブジェクト指向ではデータとそれに対する処理を一体化させたもの(オブジェクト:object)を作り、アプリケーションプログラム全体の処理の流れからデータとそれに対する処理の独立性を高めている。また、オブジェクト間の関連づけなどによりソフトウェアの再利用性の向上や操作仕様の共通性の実現でき、生産性や保守性の向上が期待できる。オブジェクト指向プログラミングを実現できる言語としてC++は、従来のC言語のソフトウェア資産が生かせる、などの理由によりユーザが急速に広がりつつある。オブジェクト指向を学習するならC++でないほうがよいと言う意見があり、オブジェクト指向と言った観点から見ると、C++に不足している部分が多いことは承知している。しかし、広く普及しているC言語のスーパーセットであり、利用者の拡大している現状ではC++は無視できない存在になっている。現状においてC++の機能を十分に利用できるプログラマは少なく、C++がどんな言語かを知っている人もC言語やＣＯＢＯＬを知っている人に比較すると少ない。C++がどんな特徴を持つ言語か知りたいという人々のニーズからオブジェクト指向及びC++の教育の重要性が増してきた。そのため能開セミナーでC++教育を始めた。C++を教えて感じたことは、オブジェクト指向を用いなくても同等な機能を実現するソフトウェアを作成できることが多いため、何故、オブジェクト指向なのか疑問に持つ受講生が多いことである。オブジェクト指向の特徴と利点をわかりやすく伝える方法は無いものかと考え、図形を用いる方法を思いついた。ここでは筆者が考案して実際に能開セミナーで実施したオブジェクト指向の基本的概念を図形を利用して教授する方法を報告する。

## II　プログラミング言語C++の特徴

C++の開発者 Bjarne Stroustrup によれば、プログラミング言語C++は次の三つの特徴を持つように設計されている[1]。

(1)　-be a better C
　　(より良いCである)
(2)　-support data abstraction

論文受付日　H7．1．17

（データの抽象化をサポートする）

(3)　-support object-oriented programing

（オブジェクト指向プログラミングをサポートする）

これらの特徴を受講者に説明する立場で検討する。(1)の better C としての側面とは、厳密な型チェック、使いやすい命令の追加、有用なライブラリの増加、文法上の洗練などである。これらは従来のプログラミングパラダイムに慣れた者にとってソフトウェア作成上の大きな概念の変更を必要としないので理解するのはそれほど困難でない。(2)のデータ抽象とは従来の型の概念を拡張したものであり、ユーザが新しい型を定義できる。新しい型はデータ構造とデータに施す固有の操作を一体化した定義を可能にし、抽象データ型(abstruct data type)と呼ばれ、C++ではクラス(class)と呼ばれている。この型によって実体を生成されたものをオブジェクト(object)と呼び、従来のデータ型にたいする変数がクラスに対するオブジェクトに相当する。これは新しい型を作るという従来あまり行ってなかったことを行うため困難を伴う。(3)のオブジェクト指向プログラミングのサポートとは継承や多態を利用できることを意味する。継承はあるクラスで定義されたデータ構造とそれに対する処理をそのまま引継ぎ、部分的にプロセスやデータの追加や変更を可能にする。多態は複数の異なる処理を同一の名称で行う多重定義など、同一の名称や演算子で異なった振る舞いを可能にする。これらの有効な使用法を理解することは、C言語を習熟した者にとっても、それほど易しくはない。後の二つをいかに分かり易く教えるかが、重要である。(1)の特徴のみを利用するのであれば従来の概念の延長で可能だが、C言語を知っている者がオブジェクト指向を意識してC++言語を学ぶことは、「プログラムは手続きの記述である」という概念を捨て「プログラムとは処理対象となる＜もの＞の記述である」という新しい概念を持たなければならない[2]。

## III　図形を用いる意味

図形を利用してC++を教えてみようと強く感じたのは最初のセミナーを行った後である。実習中のようすをみたところソフトウェア開発を職業としていない人にとって概念が分かりにくいようだった。そこであまり専門家でない人にも、それなりに分かる方法はないかと考え、以前C++で図形を描いたときの経験から、C++教育に図形を用いる方法を考えた。グラフィックスを用いて絵を描くことにより、具体的で親しみやすい方法でオブジクト指向の概念を説明して理解してもらう方法である。C++は部分的にシミュレーションを目的として開発された言語Simula67を参考にして作られ、C言語よりモデル化などに向いている。図形はモデル化とその視覚化といった面でオブジェクト指向と親和性がある。図形は直感的で具体的に変化が見えて、興味がわきやすいと考える。例えば（図1）の絵は太陽を sun,ヨットを ship というクラスに定義して描いている。3隻のヨットは ship型オブジェクト yacht1,yacht2,yacht3 としてモデル化されて目に見えているので直感的である。

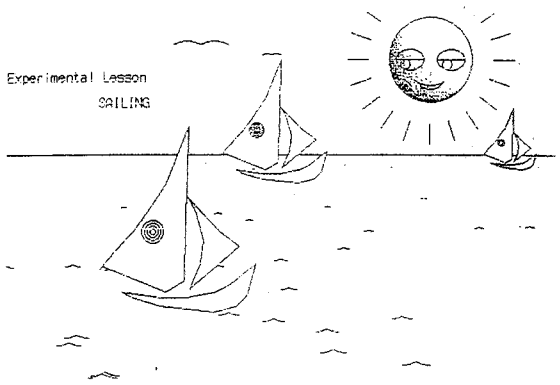


図1　C++を使って描いた図形

## IV　実施した教育方法

図形を使うと言っても全てを図形で説明できるわけではない。ここではオブジェクト指向の初心者を対象にしてオブジェクト指向の三つの基本的こと、つまりデータ抽象(Data abstraction)、継承(Inheritance)、多態(Polymorphism)といったことを分かりやすく教えることに重点をおいた。具体的には従来型の構造化設計(SA/SD)パラダイムとオブジェクト指向型設計パラダイムの比較を心がけながら進めて行った。図形を描くことのできるライブラリを持つコンパイラとして turbo C++ を用いた。コンパイラにはあえてDOS版を用いた。windows版を用いるとwindows自体の説明と関連する事柄の説明が必要になり、講義自体がシンプルでなくなるからである。

### 1．従来の方法での図形描写

最初に通常の単に命令を順に記述する方法で簡単な図形を描く方法を提示した。図形を描く部分はなるべく単純にした。図形を描く方法は機種に依存することが多く、複雑な図形や複雑な命令であると、その理解に時間がかかりすぎるからである。

次に図形を描く部分を従来型のサブルーチン（関数）

にする方法を提示して、それに位置などのパラメータ等を取り入れ、複数の図形を描くようにした。

プログラム例

```
void kao(int,int);
void
main(void)
{
        (省略)
  kao(300,200); /* 座標300,200に顔の絵を描く */
  kao(400,100); /* 座標400,100に顔の絵を描く */
        (省略)
}
void kao(int x,int y)/* 顔の絵を描く関数 */
{
        (省略)
}
```

## 2．クラス(class)、抽象データ型(abstract data type)の説明

従来の方法で図形を描く方法を提示した後、一つ一つの図をオブジェクトとして扱う方法を提示した。そして、色の属性や、大きさの変化を追加して、複数のオブジェクトが独立したデータを保持することを確認させた。またmain部分に変更、追加をしないですむことを確認させた。

プログラム例

```
class person{
        (省略)
};
void
main(void)
{
  person taro1(100,105),jiro1;
        (省略)
       taro1.kao();
}
```

## 3．継承(inheritance)の説明

それまでに作った図を描くクラスを基底クラスとして、それを少し変えた図を描く派生クラスを作成し、表示させた。個々の人の進度に合わせて階層化、多重継承を行わせた。次の例では person class のオブジェクト taro1 と person class の派生クラス woman class の オブジェクト hanako を描いたものである。

プログラム例

```
class person{
        (省略)
};
class woman:public person{
        (省略)
};
void
main(void)
{
  person taro1(110,105);
  woman  hanako(400,320);
        (省略)
}
```

## 4．多態性(polymorphism) の説明

多態性は関数や演算子に同じ名称や記号で異なる処理をさせるオーバーロード(Over Load)などで実現させるが、ここではポインタを使い、別の Class のオブジェクトを操作することによる例を図形で説明した(図2)。

ポインタを使用したソースプログラムのメイン部分が同じであるのに関わらず、異なった振る舞いをすることを異なる図形を描く方法で示した。C言語でもコンパイル時のスイッチ #ifdef などで、場合分けさせるが、C++ではコンパイルが完了した後でも実行時にそのことが可能であることを合わせて説明した。

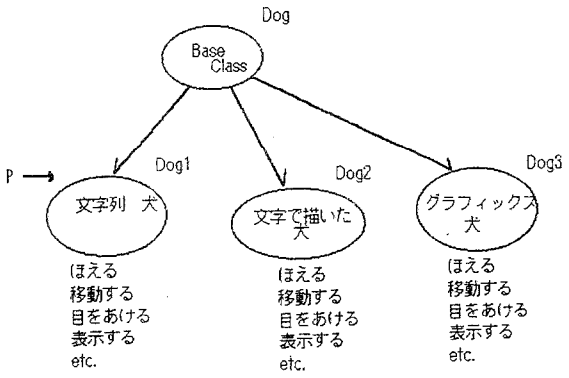


図2　多態性の説明

## 5．動的リンク

今回は実施をしなかったが、実行時にモジュールを結合する動的リンクを具体的に説明するためには次の

方法が有効ではないかと考える。
new でオブジェクトを生成して絵を描き deleteで解放する方法である。プログラムではnewで生成された人の顔が順に描かれ、deleteでdestructorを呼び出し、背景色で描き、消えたように見せている。ここでmoveはperson class 内で定義された関数（メンバ関数）であり、画面上での顔の座標データを変更する。

プログラム例

```
        省略
void
main(void)
{
  person  *personPtr;
     (省略)
  personPtr=new person[3];
  personPtr -> move(200,300);
  personPtr -> kao();
  personPtr ++;
  personPtr -> move(100,219);
  personPtr -> kao();
  getch();
  personPtr--;
  delete personPtr++;
  getch();
  delete personPtr;
        (省略)
}
```

## V　セミナーアンケート結果

1993年４月、私がポリテクカレッジ横浜港に来てから1994年12月までに、能開セミナーとしてC++は３回行った。最初は図形を使わずに文字ベースで講義による説明と実習を伝統的方法で行った。２回目、３回目はパソコンで図形を描く実習を交えながら行った。受講者の反応を知る目安として、簡単なアンケート結果を載せる。当校では当初、能開セミナーでアンケートをとる習慣はなかったので、１回目のC++セミナーの時は個人的に今後の参考として意見をもらう程度に考えてアンケートをとった。２回目以後は学務課でアンケートの書式を用意してくれたので、そのアンケートを行った。最初から統計的結果を得ようと思っていたわけでなかったため図形を利用しなかった１回目と図形を利用した２回目、３回目では形式が不統一となり単純に比較できなくなっている。またサンプル数も少ないため、このアンケートから結論は得られず単なる目安にすぎない。１回目のときは特に選択肢はもうけないで、時間配分、教材、実習環境、わかりやすさ、その他の項目について受講者に記述してもらった。（表１）はその記述内容から判断して分類したものである。どちらかというとわかりやすくないと言う人が多く、わかり易いという人と２つに分かれいる。受講者はソフトウェア開発技術者から、C言語自体全く知らない人まで広く集まった。

表１　（図形を使わない）第１回C++セミナーのアンケート結果

| | | | | | |
|---|---|---|---|---|---|
| 時間配分<br>人数 | よい<br>4 | ややよい<br>1 | ふつう<br>3 | 少し悪い<br>3 | 悪い<br>2 |
| 教材 | よい<br>6 | ややよい<br>1 | ふつう<br>3 | やや良くない<br>3 | 良くない |
| 実習環境 | よい<br>4 | ややよい<br>1 | ふつう<br>3 | 少し悪い<br>3 | 悪い<br>2 |
| 解りやすさ<br>(無回答１) | よい<br>3 | ややよい | ふつう<br>3 | やや良くない<br>5 | 良くない<br>1 |

２回目以後は学務課の用意した書式のアンケートである。質問項目は次のものである。

---

[１]．今回受講したセミナーについてお聞きします。
(1)セミナー開催はどの様な方法で知りましたか？
(2)受講の目的（動機）をお聞かせ下さい。
(3)講義（講座）内容はどうでしたか？
[２]．セミナーの開催日程等についてお聞きします
(1)受講するのに都合の良い曜日、時間帯は？
(2)あなたにとって無理なく受講できる講習時間は？
[３]．今後希望するセミナーの分野についてお尋ねします。
[４]．今回のセミナーに対する感想、また今後のセミナーへのご意見、ご希望等についてお書き下さい。

---

アンケート項目のうち講義内容の難易度に関する項目の質問結果を（表２）に載せる。質問内容は「講義（講座）内容はどうでしたか？」というもので、５段

表２　（図形を使った）第２、３回C++セミナーの学務科アンケート結果

質問項目＜講義（講座）の内容はどうでしたか？＞

| | やさしすぎた | 少し<br>易しかった | ちょうど<br>よかった | 少し<br>難しすぎた | 難しすぎた |
|---|---|---|---|---|---|
| 第２回 | | ３人 | ６人 | ２人 | |
| 第３回 | | | ４人 | ２人 | １人 |

階の選択肢を用意している。
また、3回目のときは私が独自にC++で図形を用いて説明をしたことへのアンケートをとった。質問内容は「C++の説明で図形を使ったことへの感想を記入して下さい」で項目として「基本的概念の把握」、「言語の学習」、「その他」をあげ、それぞれ5段階に分け記入してもらった。また、それらの理由を尋ねるため、「理由を上げて下さい」という項目も用意した。その

表3　「C++の説明で図形を使った事への感想」アンケート結果

| | | | | | |
|---|---|---|---|---|---|
| 基本的概念の把握 | よい 6 | ややよい 1 | ふつう 3 | やや良くない 3 | 良くない |
| 言語の学習 | よい 4 | ややよい 1 | ふつう 3 | 少し悪い 3 | 悪い 2 |

結果を（表3）に示す。
「基本的概念の把握」においても「言語の学習」においても否定的回答はなかった。「その他」に記入されたものとしては次のようなものがあった。

・自分で追加しようとした場合
－＞　やや良くない
・実務での使用方法としては
－＞　やや良くない
・実習について
－＞　良い

各項目に対する回答理由をあげてもらったところ次のようなものであった。

・視覚的にはわかりよかったが数字の計算がつらかった。
（項目1ふつう、項目2ふつう）
・業務で使わない関数を覚えることができたので、とてもよかった。
（項目1ややよい、項目2ややよい）
・直接目で結果を見ることができ良いが座標がすぐにわからなく、追加がやりづらかった。
（項目1よい、項目2よい、項目3やや良くない）
・業務システム内ではどのように使用していくのかがわからない
（項目1ややよい、項目2ふつう、項目3ややよくない）
・図形は有効、しかし直感的すぎて時に絵を描くことに夢中になる恐れがあるのでは
（項目1ややよい、項目2ややよい）
・図形を使うことにより結果が出るからよかった
（項目1ふつう、項目2ふつう、項目3よい）
・目でみて分かる
（項目1よい、項目2ややよい）

## VI　まとめ

図形を利用したC++教育を行った。図形はプログラムの構造や流れに大きな影響を与えずに個人の好みにより変えることが可能であるため裁量範囲が広い。そして変更がすぐ結果として目に見えることから、学習者の興味をひきやすいと考えられる。その反面、図形以外への応用法がわかりにくくなる。具体的応用面への指導と補足が必要である。また、限られた時間で教育を行う場合に今回の図形を使用した例題と図形以外の例題とどちらがよいか今後さらに検討が必要である。

**【参考文献】**

(1) Bjarne Stroustrup,"the C++ programing language 2nd edition",Addison Wesley,1991,p13
(2) 深澤良彰・粟野俊一、オブジェクト指向プログラミング入門　－CからC++へ，共立出版，1993
(3) B.R.ﾗｵ，C++とOOPパラダイム，マグロウヒル，1994
(4) P.ｺｰﾄﾞ/E.ﾖｰﾄﾞﾝ，オブジェクト指向分析（OOA）第2版，トッパン，1993
(5) Borland International, "TURBO C++ programmer's guide",Borland International Inc.,1992