初心者を対象としたC＋＋プログラミング教育について

C＋＋Programming For Beginners

小　林　健一郎
Ken-ichiro KOBAYASHI

（平成10年10月12日受理）

本論文では、プログラミング初心者を対象にしたＣ＋＋プログラミング教育を考える。Ｃ＋＋を選ぶのは、これがひとつの重要な言語であり、実社会からの需要もあるからである。一般にＣ＋＋はプログラミング初心者には難しいと言われているが、「ソフトウエア部品」を使えばＣ＋＋でも比較的容易にプログラミングできる。

本論文では、初心者を対象にしたＧＵＩプログラミング教育を既存のクラスライブラリを使って行うことを提案する。

１．はじめに

「教養としてのプログラミング教育は、大学では不要となった」と言われるようになってすでに久しい。ハードウエア、ソフトウエア双方の急速な進歩の結果、パーソナルコンピュータ（以下、パソコンと略す）が、使いやすくなってきたためである。自分でプログラムを作るより、市販の応用ソフトの使い方を覚える方が、容易かつ実用的でもある。

一方、いまだに「プログラミング教育は必要である」と主張するグループは、大学での教養が必ずしも実用に向けたものでないことを指摘する。「実用的」でないという理由によって否定されるなら、高等数学の教育も大部分不要ではないのか、と。

著者は、両極端のどちらかではなく、その中間に立ってバランスをとりたいと考える。実用一辺倒の教育は、専門学校等に任せるべきであると考える一方、これだけ実用的なパソコンを対象にしながら実用性を無視するのは的外れであると考えるからである。

しかし、「プログラミング教育は不要か」の問いには、「希望する学生には、行うことが望ましい」と答えたい。プログラミングも現実の一部であり、これを一般教育で行うことができれば、大部分の学生にとっての「教養」として、また、その内の一部の学生にとっての「専門への橋渡し」として、望ましいことと考えるのである。

使用する言語にはさまざまな選択肢があるが、著者はＣ＋＋によるＧＵＩ（グラフィカルユーザインターフェース）プログラミングをとりあげたいと考える。Ｃ＋＋は思想的にこれまでのプログラミング言語のひとつの集大成であり、また、実用的でもある。ＵＮＩＸの中心言語であるＣの拡張言語であり、またパソコンＯＳのデファクトスタンダード（事実上の標準）とも言えるウインドウズ上で動作するプログラムの多くもＣ＋＋で書かれている。また、ウインドウズはＧＵＩプログラムのためのＯＳであり、ＵＮＩＸでも今ではＧＵＩが主流である。単純に考えれば、これを教えることに問題はないはずである。

しかし、問題はその「難解さ」にある。C＋＋の習得は容易ではなく、そのようなプログラミング技術を持った教員も、現在のところ、あまりいない。さらにGUIプログラミングは以前のキャラクタベースのプログラミングよりかなり面倒な準備が必要になるのである。

一方、習得の容易なBASICなどで書かれるプログラム、また、C＋＋であってもキャラクタベースのプログラムは、学生には、もはや「実用的ではない」と考えられてしまう傾向にある。(実際にはキャラクタベースのプログラムでも種類によっては十分実用的である。)優れたGUIソフトウエアを目にしている学生の多くはそう思うようであり、BASICやその他のキャラクタベースのプログラミングに興味を持たせることは容易ではないようだ。

もし、「C＋＋によるGUIプログラミング」の習得が容易になれば、これは大きな選択肢となるはずである。著者は本小論で、C＋＋の教育方法（学生に比較的容易に習得させる方法）について考えてみたい。実際、容易ではないが、「難解」と考えるのは誤りであると思う。

なお、本小論の目標は「一般教育でのプログラミング」であり、真に実用的なプログラムの作成までは考えない。しかし、専門家予備軍の初期教育として考えられることは言うまでもない。

2．選択

情報教育には選択が付き物である。OSは何にするか、コンパイラはどうするか…。本小論で特に問題になるのは、マイクロソフト社の製品との関わりである。ウインドウズプログラミングの業界デファクトスタンダードと考えられているのはVisualC＋＋（マイクロソフト社のコンパイラ)／MFC(マイクロソフトファウンデーションクラス、マイクロソフト社のC＋＋ライブラリ）であろう。これを使うのか、ウインドウズは使うが別のコンパイラやライブラリを使うのか、あるいは、すべてマイクロソフト社以外のもの、たとえばUNIXなどを使うのか。

著者はマイクロソフト社の支持者でも反対者でもない。マイクロソフト製品を選ぶことが学生への誠意なのか、マイクロソフト社自身が言っているように、デファクトスタンダードは簡単に変わりうるものなのだから、１企業の仕様にこだわらない方がよいのか、難しい問題である。

しかし、OSを選ばないことには、先に進めない。ここで、とりあえずウインドウズを使うことにしよう。では、ウインドウズプログラミングにするのかDOS窓プログラミング（つまりキャラクタベースのプログラミング、コンソールプログラミングと呼ばれるもの）にするのか。これは本論と関係するので後で考えたい。

次に、コンパイラ、また、(付属することが多い)ライブラリの選択である。以下で詳説したいが、ライブラリを選ぶことは非常に重要である。これに関しては、「本小論は方法論を扱う」ということで、ここではMFCかそれに対抗する別のライブラリを選ぶかは決めないことにする。有用と考えられるライブラリには、MFCのほかに、ボーランド社のOWL(オブジェクトウインドウズライブラリ)、標準規格のSTLなどがある。

3．プログラミングの長い道程

我が国では、教育における基礎が非常に強調される。この考え方は、もちろん、一概に否定できない。しかし、現代社会において、この強調はときに誤った方向に向かうように思う。

Ｃ＋＋によるウインドウズプログラミングを目標とした場合、多くは以下のようなコースが推奨される。

0．簡易なプログラミング言語を通してプログラミングの考え方を学ぶ
1．Ｃによるコンソールプログラム（キャラクタベースのプログラム）を学ぶ
2．Ｃ＋＋によるコンソールプログラムを学ぶ
3．Ｃによるウインドウズプログラムを学ぶ
4．Ｃ＋＋によるウインドウズプログラムを学ぶ

第0項目を強調する指導者も多いが、これはさすがに省略される傾向にある。これを省くことにはあまり異論はないだろう。一方、アルゴリズムやハードウエアについては別に学習しなければならない。

これ全部でどの位の時間がかかるだろうか。たとえば、項目3で有名な書籍[1]の著者はここだけで半年間勉強する必要があると書いている。他の項目も同じくらいとすると、プログラムだけ2年間勉強しつづける必要があることになる。

現在プロのプログラマたちはおそらくそのように学習してきたのであろうし、著者自身の経験もそれに近い。しかし、他にも学ぶことがある学生が、大学4年間で項目4まで修了するのはかなり大変だということになるだろう。

また、特に強調したいことだが、時間的に可能でも、それだけ長い間興味を持続させることはかなり困難であると思われる。今の学生がＤＯＳ窓で、「足し算引き算のプログラム」を動かしてもあまり「感動」はないだろうと思うのである。「一応にも格好のついたプログラムが書けるまで、それだけ勉強しても2年かかる」と考えれば、専門家を目指す学生でも気力を維持するのは大変だろう。また、一般教育としてしかプログラミングを学習しない学生には、「時代遅れな科目」にしか見えないかもしれない。最近よく言われる「一般教育におけるプログラミング不要論」にはこのような背景もあると思われる。

著者は以前の論文[2]で項目1を省略し、項目2から入る方法を考えた。その後、いろいろな機会に実践し、その方式が機能することを確かめることができたと考える。この方式は、「専門家を目指す学生」「プログラミングを継続的に学習する学生」には有効であると思う。しかし、本小論では「半年から1年程度のみ学習する学生」にも「継続的に学習する学生」にも使える別のコースを考えてみたいのである。

4．ウインドウズかDOS窓か

ウインドウズ全盛の時代にＤＯＳ窓を使うコンソールプログラムにこだわることはできないだろう。学生も当然、ウインドウズプログラミングを期待している。

ここで障害になるのは、イベント駆動型というＧＵＩ特有のプログラム構造である。一般に、プログラムには少なくとも2つの部分があると考えられる。

A　ユーザインターフェース
B　実際の仕事をする部分

である。実際には、インターフェースの制御や両部分の中間もあるがここでは簡単に上のように分類することにする。ウインドウズプログラムでは「インターフェース」の比重が非常に大きく、また複雑である。ここを理解し、ある程度まで自由にプログラムできるようになるまでの時間が、第3節で引用した「ここだけで半年間勉強する必要がある」の「半年」なのである。コンソールのCを熟知したプログラマがGUIでCを使いこなすのに半年ということである。

一方、コンソールプログラムでは「インターフェース」は非常に簡単で、ほとんど意識されることすらない。そのため、「実際に仕事をする部分」に専念でき、アルゴリズムの実装練習にも向いている。このため、教員側にコンソールプログラムを好む傾向があるのは無理からぬものでもある。

ここにあるのは「アルゴリズムの理解と実装方法を学ぶことが重要である」と考える教員側と「ウインドウズ上で動くプログラムを作りたい」と思う学生の意識のギャップである。学生に「ユーザインターフェースなど本質ではない」と強調しても、数値計算等の専門家を目指しているのでもなければ、多くの学生は納得しないだろう。そもそも、ソフトウエアの見た目ばかりに気を取られる学生が間違っていて、アルゴリズム教育に熱心な教員が正しいのであろうか？

著者は、多くの場合、初心者の学生は「実際に仕事をする部分」の重要性を理解していないが、同様に、古いプログラマである教員側は「インターフェース」の重要性を理解していないと考える。現代情報社会において、ユーザインターフェースはいくら強調してもしすぎないほど重要なのである。それを理解できない、あるいは理解しようとしない教員は、もはや時代に取り残されていると言わざるを得ない。

初心者を対象とする市販の教科書の中には、まず、簡単にコンソールプログラムを説明し、すぐにウインドウズプログラムに移行するものが、最近多く見られるようになってきた。これは出版社が社会のニーズがウインドウズプログラムにあることを認めた結果でもある。

いずれにせよ、OSとしてウインドウズを採用した時点で、プログラムはウインドウズプログラムであるべきなのである。著者はアルゴリズムの重要性を過小に評価するものではないが、学生にはまずウインドウズプログラム、特に「インターフェース」部分を教えて良いと思う。「基礎として」はじめにコンソールプログラムから入る必要もないと思う。（もちろん、時間があり学生に動機があればそれでもよい。）つまり、著者の主張は、第3節の項目0、1に続いて項目2「C++によるコンソールプログラムを学ぶ」も省略可能であるということである。

GUIの「インターフェース」部分の習得が難しいということは事実だが、コンソールプログラムに慣れた教員がGUIに感じる困難と、そもそもコンソールプログラムを知らない初心者に見える困難を取り違えてはならない。また、最初に学習させるべきことは「インターフェースのすべて」ではなく「インターフェースのエッセンス」である。これは教員側にGUIに対する理解が十分あれば、必ずしも、大変な困難ではない。

著者は、初心者が、ウインドウズプログラムでのインターフェースの作り方と制御の基本を学ぶのに、週1回の演習でも、半年で可能であると思う。(もちろん、いわゆる「教養」としてのプログラミング教育、もしくは、「入門」としてである。専門家を育てるためには、さらに多くの時間が必要であることは言うまでもない。)実際には、もう半年あれば良いと考えるが、そうであれば、インターフェースのテクニック習得を深めるか、アルゴリズムの学習をするか選べば良いだろう。もし、アルゴリズムを選ぶならここでコンソールプログラムに行くのも良い。ＧＵＩの基本を理解した学生は、「きちんとしたアルゴリズムの知識がなければ良いプログラムは書けない」ということを実感として理解できるはずであり、この段階でならコンソールプログラムにも強い興味を持つだろうからである。

簡単に言うと、

「コンソールプログラミング」－＞「ＧＵＩプログラミング」

「ＧＵＩプログラミング」－＞「コンソールプログラミング」

という2つのコースのうち、従来型の前者ではなく後者も可能であるということ、一般の学生が対象なら、後者の方がより良いとする主張であることに注意されたい。(もちろん、数値計算等の専門家を目指す学生には、前者の方が良いかもしれない。)

最後の問題は、第3節項目3「Ｃによるウインドウズプログラムを学ぶ」の扱いである。目標がＣであるならば、当然ここを学ぶべきである。しかし、Ｃを目標とするのでないなら、この部分も省略は可能である。前述した「インターフェースの基本は半年で十分」というのは、ＣによるＧＵＩを省略した場合である。

表題にあるようにＣ＋＋の教育を考えているので、第3節項目3は省略する場合をとることにしよう。ただ、「ＣによるＧＵＩ」には詳細な参考書が多く、そのため、Ｃ＋＋プログラマにとっても重要なので、やや本格的に学習することを望む学生には、省略しないという選択も有り得る。これはＣの知識の問題ではなく、純粋に文献の問題である。残念なことにＣ＋＋によるウインドウズプログラミングに、[1]のような詳細な教科書は著者の知る限り見当たらない。ただ、ここではこれ以上議論しない。

まとめると、プログラムのまったくの初心者への入門コースで、いきなり第3節項目4「Ｃ＋＋によるウインドウズプログラムを学ぶ」から入ることができるという主張を述べたことになる。これは、以前なら「暴挙」と考えられるコースであるが、著者は、可能であり、また、ひとつの良いコースであると確信する。その方法について、次節で説明したい。

5．ライブラリを使う

Ｃ＋＋は「クラスを作って使う」言語である。クラスとはプログラマによる定義型であり、これによりプログラムの部品化が行われる。このため、他のプログラマのコード（やコンパイル後のバイナリ）を容易に利用できるわけだが、これがＣ＋＋の「ライブラリ」である。

一般に、Ｃ＋＋の教科書で説明されるのは、クラスの作り方であり、クラスの使い方を記述しているものは、特別なライブラリに対するものを除くと数少ない。この特別なライブラリがマイクロソフト社のＭＦＣなのである。

まず、ＭＦＣであれ何であれ、著者は初心者には「クラスの作り方」より「クラスの使

い方」を先に教えることを提案したい。これが本小論の主題である。

教員も含めて現在活躍しているプログラマの多くは、プログラミング技術の進歩とともに段階的に学習してきている場合が多い。そのようなバックグラウンドでは、「まずクラスの使い方から」という方針に抵抗を覚えるかもしれない。

また、使うライブラリの選択でも困難を感じてしまうものである。大きなシェアを持ち、参考書数も圧倒的なＭＦＣを選ぶべきか、他のライブラリを選ぶべきかという問題である。ウインドウズ用のライブラリとしては、ＭＦＣの他にボーランド社のＯＷＬなどが有名であるが他にもある。いずれにせよ私企業のライブラリである。数ある企業の製品のうちひとつを選んで講義に使うことに問題はないだろうか。

もちろん、他のソフトウエアでも同じことは言える。たとえば、ワープロソフトや表計算ソフトを使う講義でどの企業の製品を使うべきか、簡単な問題ではない。

ただ、ワープロソフトや表計算ソフトでは、考え方が各社ほとんど共通であるため、ある製品を覚えた後に、本人の意思で別の製品に変えることは多いに有り得るだろう。しかし、Ｃ＋＋プログラムのライブラリは、言語の一部のように機能するため、考え方までその企業の製品に合わせる必要が出てくる。もちろん、ひとつのライブラリの本質を理解できれば、他のライブラリを使いやすいかもしれないが、ワープロソフト等より遙かに困難であることが予想されるのである。

著者がこれまで「ライブラリを使った初等ウインドウズプログラミングコース」を主張できなかったのは、まさにこのためである。「ウインドウズ＋ＭＦＣ」を教えた学生が、将来ＵＮＩＸでＣ＋＋プログラムをする立場になったらどうするのか、と。

しかし、世の中は止まらず、動き続けている。ここで、ライブラリを選ぶしかない、と判断せざるを得ないのである。ウインドウズを使いながら、「ウインドウを持たないプログラム」しか教えられないのは奇妙であるし、また、ライブラリを使わなければ、初心者にＣ＋＋ウインドウズプログラムを教えることはほとんど不可能なのである。

もちろん、ライブラリを受け入れれば、初心者でも簡単にウインドウズのユーザインターフェースが作れることは一言述べておきたい。

私企業のライブラリをどういう基準で選ぶかは本小論の範囲を超えるので議論しないが、いずれを選んでも、なるべく基本的な考え方を捉え、他のライブラリへの移行もできるように教育することが重要であろう。いかなるライブラリも将来の保守管理が保証されているわけではないからである。

なお、ウインドウズライブラリの使用に際して、多くのコンパイラに付属している「定型のコードを自動生成するツール」(「ウイザード」とか「エキスパート」などと呼ばれる。本小論では「ウイザード」と呼ぶことにする）も使うことを考える。

「ウイザードを最初から使うとＧＵＩプログラムの本質がわからなくなるので良くない」という意見がソフトウエア業界にもある。私の方針は、「階段を一段ずつ上り、決して後には戻らない」というものではない。「ウイザード」による初期教育の後、先に進みたい学生は適宜「ウイザードが生成するコードの意味」を学ぶという方針であり、「ＧＵＩプログラムの本質」にも触れることになる。

この方針は一見遠回りなようだが、強い動機付けが得られやすいため、結局、一番の近道になると考えるのである。

また、「ウイザード」を使わせたために本質を見失うというのは、指導者がいない場合であると思う。指導者がいて本質を見失うのは、指導者の責任であろう。

6．ひとつのモデル

前節までの考察をもとにひとつのモデルを提案したい。これはあくまでひとつのモデルであるので、たとえ上記の議論に従うとしても、いくつかある選択肢のひとつでしかない。

半期の演習講義1

内容：ライブラリを使ったC++ウインドウズプログラミング
　　コンパイラ付属の「ウイザード」（コードの自動生成ツール）を使用する。

対象：プログラミング経験のない初心者、ただし、ウインドウズの使用経験は仮定する。

第1週　イントロダクション
　C++やウインドウズプログラムを簡単に紹介し、関数やクラスの意味等を説明する。
　デモンストレーションも良いだろう。

第2週　「ウイザード」
　「ウイザード」の簡単な使い方と生成されるコードの簡単な説明。
　この説明は以後何度も繰り返し深めていく必要がある。

第3週　ＧＵＩ
　メッセージボックスの実習、イベント駆動の説明。WM_LBUTTONDOWN、
　WM_LBUTTONUP、WM_MOUSEMOVEのハンドラを書く。

第4週　ＧＵＩの続き
　前週の演習。

第5週　デバイスコンテキスト
　デバイスコンテキストを使った文字の表示について学ぶ。

第6週　図形の表示
　デバイスコンテキストを使った直線や円などの表示を学ぶ。
　極めて簡単であるが、ウインドウズプログラミングの基本を終えたことになる。

第7週　ダイアログボックス
　ダイアログボックスの演習。

第8週　クラスの自作
　文字列クラスや今まで習った構文を使って簡単なクラスを自作する。
　ここまでクラスは作っていないことに注意。
　if文、for文を説明し、「動きのある」サンプルプルプログラムを考えさせたい。

第9章　自作クラスの使用
　前週に作ったクラスを使うプログラムを考える。

第10週　乱数
　乱数の発生を教える。乱数は初心者に絶対的に必要な項目ではないが、ゲーム等学生
　が興味を持つプログラムの作成には必須である。

第11週　プログラムの自作
　プログラムを自作する。

第12週　プログラムの自作続き
　前週の続き

以上にコメントを付けたい。

1．「細かいことはいずれわかるので最初からこだわらないように」と強調する。
　まず文法を教え、次に使い方を教え…という方針が常に良いとは限らない。上のようなコースでは必然的に「必要なことは必要になったときに教える」という方針になるだろう。そのため、何を教え、何を教えないか、教えるものはいつ教えるのか、よく考えなければならない。もちろん、それはどの分野でも言えることだが、上のコースは専門家を目指すものが毎日勉強して数年かかることの中から、重要なことのみを取捨選択しているのである。上のモデルで、文法としては、特にif文、for文のみを記したが、もちろんほかにもあるし、タイミングのとり方もいろいろあるだろう。
　乱数は学生の気持ちを引き付けるのには良い素材のようである。上記のモデルよりはやく登場させ、たとえば、ランダムに図形を描かせるのも、題材としておもしろいかもしれない。
2．「昔はこういう技法だったが、」といった懐古趣味的コメントを、余談としてすることはよいが、本題と混同してはいけない。
　自分が過去に覚えたことに特別な意義を感じるのは人の常であるが、その知識が明日を生きる若者にも本当に必要かどうかよくよく吟味する必要がある。
　特によくあるのがCとC＋＋の比較である。市販の教科書にそのような記述が多くあるのは、読者をCプログラマと想定しているからである。C＋＋をはじめから学ぶ学生にわざわざCの欠点を説明しても何の得にもならない。
3．教員は教える内容以上にＧＵＩプログラミングについて知っていなければならない。
　おそらくいろいろな学生から多様な質問が出るだろう。「それは後で勉強するように」と言うにしても、指導者本人がよくわかっていなければ間違いにつながるだろう。
　大抵の大学教員はその分野での業績によって教員になっている。しかし、ＧＵＩプログラミングは多くの教員にとって教員になった後に出てきた（盛んになった）ものである。したがって、これらを今学習することは恥ずかしいことではない。
　著者は、教員に最低限必要なＧＵＩプログラミングへの理解は、「ウイザード」のコードの理解であると考える。
4．第11、12週はプログラムの自作としたが、学生の意欲等によって、自作まで可能になっているかどうかわかれると思われる。実際の演習講義では、学生に応じて内容を考えたい。

扱っている内容は非常に限られたものであるということは否めない。しかし、かつてのBASICによる半期演習でもそれほど多くのことは教えていなかったと思う。

以上のコースを修了することで、学生は、アルゴリズムについてまじめに学びたいと考えるかもしれないし、ユーザインターフェースの作り方をより深く学びたいと考えるかもしれない。もし、引き続き半年の演習があるのなら、そこでより多くのことを教えること

ができるだろう。

謝辞

著者は、インターネット上に「Ｃ＋＋入門講座」のホームページを公開した。公開中、数万人を越える「訪問者」があり、数百通のＧＵＩ関連メールを受け取った。本小論の内容はホームページの内容と一般読者からのコメントを元にしている。

静岡産業大学の教員を含む、建設的なご意見をくださったみなさんに感謝したい。

文献

[１] Ｃ．ペゾルト『プログラミングWindows95』1997、アスキー出版局

[２] 小林健一郎 「Ｃ＋＋言語のミニマル・サブセットとオブジェクト指向性」
静岡学園短期大学研究報告1996第9号309