教材用プログラム開発の指導
——C言語使用の試み——

宮　下　英　明*

1　C言語によるプログラム開発の指導

1.1　C言語によるプログラム開発のメリット

C言語で教材のプログラムを実際に開発してみると，C言語の使用もBASIC言語の使用も，負担の程度は変わらないという実感に至るようになる。要するに馴れの問題であり，実際C言語に馴れると，今度は逆にBASIC言語に負担を感じるようになる。C言語を用いることにメリットを感じ，勝手なもので，従来のインタプリタタイプのBASICを使う気がしなくなってくる。

インタプリタタイプのBASICと比較するときに，C言語を用いるメリットとして実感されるものを，ここで，思いつくままに挙げてみる。

(1)　実行プログラムが直接OSの上で動く。
(2)　実行速度が速い。
(3)　自動変数の概念があるために，関数の追加が自在——変数の重複について神経を使わなくて済む。
(4)　プログラムを自在に分割できる。
(5)　構造体が使える。
(6)　再帰処理ができる。
(7)　if文，for文，while文が書きやすい。
(8)　switch文がある。
(9)　ライブラリが充実していて，機能が豊富。
(10)　ハードに直結した細かい処理ができる。
(11)　コマンドラインで引数を渡すことができるように，プログラムをつくれる。
(12)　行番号がないだけ，ソースプログラムがやかましくない。
(13)　BASIC上のエディタを使わないで済む。

1.2　C言語の自学習

ここでのプログラム開発指導は，基本的に，指導される側におけるC言語の自学習を前提して進める。それは，時間的余裕という問題もあるが，言語学習というものが本来自学習的性格のものだからである(教えるよりも，自分でやってもらった方がはやい)。しかも，C言語の場合，自学習のための環境は，市販の書籍に関しては，十分過ぎる程整っている。

1.3　C言語使用のための準備

1.3.1　何かと物入りなC言語指導

教材用プログラムの作成の指導では，BASIC言語の使用が一般的である。この理由は何か。言語への入り易さということが先ず頭に浮かぶが，理由の第一は，これではないように思われる。

BASIC言語を止めてC言語に替えるとき，一番様変わりするものは何かというと，言語の

*宮下　英明　　金沢大学教育学部

様相よりも先ず，プログラム作成のための手続きである。そしてそれを遂行するための新たな環境の整備が必要になる。

要するにインタープリタ言語とコンパイル言語との違いということになるのであるが，具体的に何がどう違ってくるか。これを，必要なものの確認という形で，見てみる。

BASIC言語によるプログラム作成の指導で準備する必要のあるものは，BASIC実行ファイル一本だけである。それだけで指導は成り立つ。(いまは，プログラム作成の指導を，ファイル，ディスク処理のようなパソコンの基本操作の指導から切り離して考える。)

C言語の場合はどうなるか。Cコンパイラソフトだけでは足りない。

先ず，パソコンにRAMボードが装着されていることが必須である(註)。これが無ければ，学習者が気の毒で，とても作業させようという気にはなれない。

プログラムのソースファイル作成のためのエディタが必要である。これは，タグジャンプ機能のあるものでなければならない。

グラフィクスライブラリが必要である。Cコンパイラソフトで標準的に提供されているライブラリには，グラフィクス関連のものがない。何故黒板や印刷物に代えて敢えてパソコンのディスプレイなのかと言えば，ヴィジュアル面の様々な利点からである。そして勿論それは，テクストではなくグラフィクスの方に関する。C言語に因る教材プログラム作成は，グラフィクス用ライブラリが与えられていることを初めから前提している。

また，Cコンパイラソフトに，ソーステクストの中のタブ（特に，インデント）をそのままの形でプリントアウトするプリントユーティリティが貼付されていないときには，自作するにしても，これを用意しなければならなくなる。

(註) “RAMボードが必須”の条件のために，指導で許容できる学習者数が自ずと限定されてくる。パソコン指導がパソコン台数に見合った数の学習者に対してしか成立しないように，C言語によるプログラム作成の指導は，RAMボードが装着されているパソコンの台数に見合った数の学習者に対してしか成立しない。そして，限られた予算の中からRAMボードへの出費を捻出するのは，優先順位の問題から，容易なことではない。

1.3.2 学習・作業用ディスク

つぎに，“学習・作業用ディスク”として学習者に提供するディスクを，考えてみる。

先ず，Cコンパイラソフトのインストールで，ディスクは2枚になる。——Sモデルのみにしても，以下のものが加えられることによって，どうしても2枚にはなる。

Cコンパイラソフトをインストールしたディスクには，(OSのシステムディスクに含まれるユーティリティソフトの他に) つぎのものを含ませる必要がある：

(1) RAMディスクに関連して，RAMディスクドライバおよび必要最少限のRAMディスクユーティリティ；

(2) グラフィクス用関数が含まれているCライブラリ；

(3) エディタ；

そして，環境設定の指導を省きたいということ

になれば,

(5) CONFIG. SYS：RAMディスクドライバの登録；

(6) AUTOEXEC. BAT：PATH設定等の環境設定や，リセットへの対応(註)を定める。

その他,

(7) MAKEFILEのテンプレートファイル；

(8) つぎの手続きを定めたMAKEバッチファイル：面画出力のテクストファイルへのリダイレクトをともなうMAKE実行，そしてこのときのファイルのエディタによる読み込み。

(註) C言語によるプログラム開発では，実行プログラムのテストでそれが暴走する事態を予想しなければならない。そしてこのために，リセットによって速やかに元の作業環境に至る対応を考えることになる。

1.4 C言語学習での躓きとなる諸概念

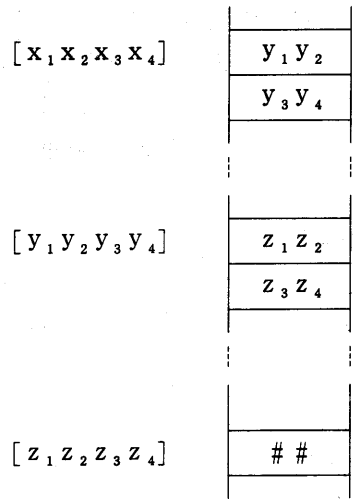
1.4.1 “変数”

ソースプログラムの中の変数は，実行プログラムにおいてメモリアドレス指定のオペランドで置き換わるところのものである（但し，変数の型によってオペコードも決まる）。C言語の実践では，“変数”をこのような形で理解できていることが必須になる。

特に，たポインタ変数は，つぎのような具合に理解される必要がある。例えば，

char ＊＊p；

で宣言されたポインタ変数pでは，ソースプログラム中のp，＊p，＊＊pは，実行プログラムにおいて，それぞれつぎのような関係にあるオペランド[$x_1x_2x_3x_4$]，[$y_1y_2y_3y_4$]，[$z_1z_2z_3z_4$] に置き換わる：



1.4.2 “ヘッダファイル”と“ライブラリ”

ヘッダファイルおよびライブラリがどういったものかを理解させるには，実際にやらせてみるのがはやい。

1.4.2.1 先ず，プリプロセッサコマンド#includeの機能（ファイルのマージ）を理解させる。このためには，例えばつぎのような実験をさせてみる。

(1) ソースファイル test. c：

```
main()
{
    sub()；
}
sub()
{
}
```

を，それぞれmain()，sub()一個でなる二つのファイル——main. cとsub. c——に分割する。

(2) main. cの頭に

#include "sub. c"

の一行を挿入する。この書き直したmain. cをコンパイル，リンクした結果は，test. cをコンパイル・リンクした結果と同じになる。(ファイル比較のユーティリティを用いて確認。)

1.4.2.2 つづいて，ヘッダファイルの意味をつかませるために，つぎのような実験をさせる。

(1) test. cの直接の分割でつくったmain. c, sub. cの二つを，それぞれコンパイルし，そして生成される二つのオブジェクトファイルをリンクする。この結果は，test. cをコンパイル・リンクした結果と同じになる。

(2) test. cでsub()をvoid sub()に書き変え，(1)と同じことをすると，エラー("前方参照"によるエラー)になる。このエラーは，main. cの頭に

void sub()；

の一行を加えることで，無くせる。

(3) つぎの手続きは，この"void sub()；"の挿入と同じ結果をもたらす。即ち，test. cを

sub. h

```
void sub()；
```

main. c

```
#include "sub. h"
main()
{
    sub()；
}
```

sub. c：

```
void sub()
{
}
```

のように分解し，main. cとsub. cをコンパイル・リンクする。

1.4.2.3 リンクの意味を分割コンパイルとリンクの実践を通して理解させた上で，汎用的なオブジェクトファイルをライブラリとしてとりまとめるという考え方を知らせる。

1.4.2.4 つぎのことを理解させる：

(1) ライブラリで用意されている関数の使用では，コンパイルでの"前方参照"の問題から，その関数の型を最初に宣言しておかねばならないこと；

(2) そして現実には，関数の型の宣言を，その関数の型を定義したヘッダファイルのインクルードという形で行なっているということ。

1.4.2.5 一度作成した関数の引用には，つぎのそれぞれの形態があり得ることを理解させる。

(1) テクストファイルの形で保存しておいて，エディタでマージする；

(2) テクストファイルの形で保存しておいて，プリプロセッサコマンド#includeを使って，マージする；

(3) オブジェクトファイルの形で保存しておいて，リンクする；

(4) ライブラリの形で保存しておいて，リンクする。

1.4.3 構造体

構造体は，実践的に理解されるものであると言える。ともかく使ってみる方がはやいし，また，一旦構造体を知ると，プログラム作成において構造体は（“それが無いと不便で仕様がない”という意味で）不可欠のものになる。特に，構造体の中に構造体を含めるという形で，変数を階層的に管理することができるようになる。

1.5 デバグ（debug）

デバグを，つぎの3つに分けて考える：

(1) コンパイルエラーを無くすデバグ；

(2) リンクエラーを無くすデバグ；

(3) 実行プログラムが期待通りに動くようにするためのデバグ。

(1)に関しては，つぎのことが指導の内容になる：

・コンパイルエラーとは文法エラーであること；

・エラーがエラーを生むことがあり，エラーメッセージに現われた分だけエラーがあるわけではないということ；

・各エラーメッセージに対応する文法的エラーのタイプを知るよう，努めること。

また特に，“自動変数が初期化されていない”のウォーニングについては，これがどういう意味でウォーニングになるのかを，熟知させておく。

初心者の場合，リンクのエラーメッセージは，“外部変数の参照に対しその変数が見つからない”，“呼び出している関数が見つからない”，“同一の関数名で異なる関数が定義されている”が殆どである。そしてはじめの二つは，スペルミスに因っている。なお，ここでは，GREPユーティリティの使用を指導しておいてもよいであろう。

本来のデバグは(3)のタイプのものである。そして一般的な対処として示唆できることは，つぎのようなものである：

・プログラムの暴走の事態は，配列，ポインタ変数の扱いのエラーに因ることが多い。

・暴走は見掛けで実際は無限ループ，という可能性もある。

・つぎのようなデバグ方法——簡単ではあるが，通常はこれで十分間に合う：プログラムの実行の中で，疑問のある変数の値を画面に表示させ，値を確認しては，キー入力でプログラムをつぎの段階に進める。

```
        ⋮
    printf("%d¥n", x);
    getch();
        ⋮
```

・外部変数の値が期待通りにならない原因の一つに，これの記号と同じものが自動変数に使われているということがある。これは結構気づきにくい。

・アルゴリズムの仔細なチェック。

2 プログラムの位置付け——非自習教材

ここで考える“教材プログラム”は，子どもの自習用プログラムではない。黒板，チョーク，定規，マーカー等を使って教えることと併行して，パソコンのディスプレイを用いようとするのである。特に，子どもに操作をわたすことはしない。

またこのような位置付けから，文字の表示ははじめから無用である。画面には文字を一切現わさない。

3　教具としてのパソコンプログラムのメリット

教具としてのパソコンプログラムのメリットとして，ここでは以下のことを挙げておく：

(1) 軽量——特に，保存が問題にならない。

(2) 複製の譲渡とかソースプログラムの公開といった方法で，同じものを皆が持てるようになる。実際，これを目的に考えるのでなければ，プログラム作成に多くの時間を費す意味がない。

(3) 手直し，改良がしやすい。あるいは，手直し，改良する気になれる。

(4) 画面に表示される操作シミュレーションを，実際に手作業で行なうことは，準備も含めて，大変。

但し，パソコンのディスプレイ上でのデモンストレーションには，画面が小さく一斉指導には向かないという難点がある。——パソコン用のOHPでは，画面がモノクロになるので，ここでは使えない。また，画質の問題から，テレビの画面も使いたくない。

4　プログラム作成の実際

4.1　整数比構造のデモンストレーションプログラム

4.1.1　プログラム作成の理由

整数倍，整数比の概念をユークッドの互除法のデモンストレーションを通して理解させることを，主題化しよう。このとき，二本のリボンを素材として，ユークッドの互除法をこれの間で行なうとする。なお，一回で整除される場合が整数倍ということになり，整数比の特殊になる。

このデモンストレーションを，黒板の上で実現するとなると，結構大仕事になる。

素材のリボンとしては，マグネット色板でつくったものが扱いやすい。このときには，剰余を示すリボンも用意することになる。そして互除操作の経緯がその都度読めるようにするために，これは組み込まれる回数分必要になる。また，剰余の分も最初から用意しているというと，授業がいかにも予定調和的で白々しいものになるので，これはその都度つくっていくということになる。

このような手間暇の問題から，色々な場合をやって，とことん定着を図るということができない。指導は，一つか二つの例で理解させてしまおうとするものにならざるをえない。

ここから，パソコンのディスプレイ上で，互除法のデモンストレーションをやってしまおうという発想が出て来る。

4.1.2　“整数倍”モードと“整数比”モード

“ユークッドの互除法”において，“整数倍”は“整数比”の特殊になるから，“整数倍”用と“整数比”用というように二本のプログラムをつくる必要はない。“整数倍”モード，“整数比”モードを，選択できるようにすればよい。

“整数倍”モードは，ここでは“非整数倍整数比”モードとしてつくる。“整数比”モード／“非整数倍整数比”モードの選択は，コマンドラインでスイッチ指定する方法で行なえるようにする。また，プログラム実行中でも，特殊キー(ここではスペースキー)の押下で，モードを切り替えられるようにしておく。

指導では，整数比の特殊としての整数倍——1対nの形の整数比——は，プログラムで

定めたモード変更の手続きによって，教師の裁量で適宜加えることとする。

なおこのときには，“整数倍”，“(非整数倍)整数比”のどちらのモードにいまなっているかを，何かの表示で，教師がわかるようにしておかねばならない。しかしまた，その表示は、子どもの意にとまらないようなものでなければならない。ここでは，画面の右すみにピリオドを表示して，これが青なら“整数倍”モード，赤なら“整数比”モードというようにしておく。

4.1.3 プログラムの実行内容

プログラムの実行内容を，(非整数倍)整数比モードの場合で，述べておく。

(1) メインルーチン

メインルーチンは，課題設定 (random())，表示データの計算 (calc_())，画面表示 (disp_()) の無限ループである。一周の終了 (中断) 時でのキー入力待ち (cont_()) でエスケープキーを押すことによって，この無限ループを抜け，プログラム終了となる。

(2) 課題設定

課題は，15以下の自然数m，nに対する整数比m：nである。mとnはランダムに発生させる。m：nが整数倍になるときは，新たにm，nを発生させる。

(3) 表示データの計算

表示に必要なデータを，予め計算して求めておく。

(4) 画面表示

互除操作の対象になる二本のリボンを，上下に水色で表示する。このとき，課題変更をスペースキーで受けけ付けられるようになっている。スペースキーの押下で新たな課題設定へと移行し，これ以外のキー入力で互除操作の表示に移行する。

互除操作の表示では，除算がなされる二つの剰余を緑色で表示し，一方の剰余を他方の剰余の傍らに並べていくという形で除算を表現する。このとき，割る側の剰余からそれのコピーが発生し，割られる側の剰余に向かって移動するというようにする。

表示は，《描画を裏VRAMで行ない，描き終わったら表VRAMと切り替える；そしてこの繰り返し》のやり方で行なう。このとき，移動リボンの描画と消去が絵の他の部分に影響する段階では，全部——即ち，課題の二本のリボンと，これにこれまでに貼付してきた剰余のリボン——を描き直し，そうでないところでは，移動リボンのみの描画・消去とする。これによって，リボンの移動は，《“離着陸”でゆっくり，その他で速く》というようになり，ちょうど良い具合になる。なお，ファンクションコール06によるキーセンスを間に挟み，リターンキーで一時停止，エスケープキーで目下の課題の中止が，それぞれできるようにする。

4.2 n進命数法のデモンストレーションプログラム

4.2.1 プログラム作成の理由

n進命数法のデモンストレーションのために，装置を実際に製作する（しかも必要な台数だけつくる）ことは，結構大仕事である。しかも，単元が済めば，それの保存が問題になる。保管するスペースが無ければ，折角つくったものを廃棄することになる——そしてつぎの年度にまた新たに製作することになる。

4.2.2　n進命数法のデモンストレーション

自然数は，“つぎ”を重ねることで際限無く生成される。そして，この無際限の数を或る個数の記号だけを用いて表現しようとするとき，n進の命数・記数法のアイデアが出てくる。

n進位取り記数法のデモンストレーションは，つぎのようになる（簡単のために普通言っている“十進数”の場合で述べる）。

表示が記号

X_0（“無”記号），X_1，X_2，……，X_9

で一周するところのカウンタを考える。（“カウンタ”と言っても，ここでは何かを数えるわけではない。あくまでも，数詞を生成するための装置である。）このカウンタを複数台用意して，横に一列に並べる。列の序列をカウンタの〈位〉の序列と読み直す。

数記号生成の作業は，つぎの規則で行なう：

(1) 作業を開始する前に，すべてのカウンタの表示をX_0にしておく。

(2) 作業の最初のステップは，最低位のカウンタを一つ進めることである。この最初のステップが，数詞の生成の第1回目である。

(3) 作業中，各カウンタの下に，“0”，“1”，“2”，……，“9”をつぎの規則で記す：一巡以上している“無”記号X_0に対しては“0”，記号X_1に対しては“1”，……，記号X_9に対しては“9”。

(4) “つぎ”を導入する毎に，最低位のカウンタの表示を一つ進める。

(5) 各カウンタについて，それの表示が“無”記号に戻るとき，一つ上位のカウンタの表示を一つ進める。

作業開始以降，“0”，“1”，……“9”でなる列が，カウンタの下につぎつぎと表示される。これを数詞の生成と見なす。

4.2.3　“数字”モードと“図”モード

カウンタの表示記号は，数字に限る必要はない。互いに区別できる記号でありさえすればよい。ここでは，表示記号が数字であるもの（“数字”モード）に二種類（〈窓〉と〈ダイアル〉），図であるもの（“図”モード）に四種類（〈円〉と〈リボン〉の二種類に，それぞれ〈線分〉と〈領域〉の二種類），それぞれ用意する。

但し，ここで用意した“図”モードは，“n進単位システム”のデモンストレーションと混同され易いので，取り上げる場合には注意を要する。即ち，〈図＝記号〉の違いを，実際に，単位の数（かず）の違いにしているので，図が《一まとまりで一個の記号》のようにではなく，《単位いくつ》のように受け取られ易い。

4.2.4　n進単位システムのデモンストレーションとの区別

“n進命数法のデモンストレーション”は，もともと，“n進単位システムのデモンストレーション”と混同されるものではない。実際，後者では，各単位がそれの上位の単位と異なることになるからである。

二つのデモンストレーションが混同されるのは，一つの位だけ——“……→X_0→X_1→……X_{n-1}→X_0→……のローテーション”——を見ているときである。

しかしこの場合にも，《記号X_{n-1}から“無”記号X_0への移行がつぎの何れで意味付けられているか》という形で，“n進命数法のデモンストレーション”と“n進単倍システムのデモンストレーション”の区別を考えることができる。

即ち，

“……→X_0→X_1→……→X_{n-1}→X_0……のローテーションの中にある移行の一つ”

が前者であり，

“単位のn－1個分としてのX_{n-1}に単位が一つ加わり，一旦単位がn個の対象がつくられる；そしてつぎにこれが解消される”

が後者である。n進単位システムのデモンストレーションプログラムでは，X_{n-1}からX_0への移行が，単位が一つ加わることを契機とするものとして描かれることになるわけである。

例えば，つぎのようなやり方で，“単位の位上がり”を視覚化して見せることができる。即ち，単位n個分になった状態と，それに置き換わる上位の単位一つとを同時に点滅させ，点滅の終了とともに，前者を“無”に変え，後者を確定する。

4.2.5　プログラムの実行内容

(1)　メインルーチン

メインルーチンは，課題設定——何進法何桁までかの入力(input_())，表示データの計算(calc_())，画面表示(disp_())の無限ループである。一周の終了（中断）時でのキー入力待ち(cont_())でエスケープキーを押すことによって，この無限ループを抜け，プログラム終了となる。

また，表示データの配列を上限でとると，使用メモリが大きくなり，また実行プログラムのサイズも大きくなるので，《n進法m桁まで》の入力の度毎にメモリの割当てを行ない，プロセスが終了すればこのメモリを解放するというやり方をとる。またそもそも，《n進法m桁まで》の指定では，現実問題として，大きいnに対してはmを小さく，小さいnに対してはmを大きくとることになるので，配列を上限でとるやり方は，ますますナンセンスということになるのである。

(2)　課題設定

《n進法m桁まで》を，n，mのインプットで決める。ここでは，$2 \leqq n \leqq 16$，$2 \leqq m \leqq 20$とする。教具としての使用を考えて，入力のプロンプトには，文字を使わない。また，入力文字のエコーもしない。——n，mの入力に対するプロンプトは，目立たなくするために，それぞれ青，赤色のカーソルのブリンクとする。

(3)　表示データの計算

表示に必要なデータを，予め計算して求めておく。カウンタの大きさは，《n進法m桁まで》のn，mの大きさに応じて決められる。

(4)　画面表示

《n進法m桁まで》の指定で，m個のカウンタが横一列に表示され，n進数命数法のデモンストレーションが開始される。

“位は左にいくにしたがって大きくなる”の既成概念を壊す目的で，ここでは敢えて“右に行くに従って大きくなる”よう位を定める。

数生成の表示中，ファンクションコール06のキーセンスでキー入力を受け付ける。

プログラムを起動した状態では，“数字”〈窓〉モードになっているが，“数”モードと“図”モードの切り替えがタブキーで行なえるようにする。“図”モードは，初めは〈円〉モードとする。

カーソル移動キーによって，“数字”モードでは〈窓〉と〈ダイアル〉が切り替わり，“図”モードでは〈円〉と〈リボン〉が切り替わる。

また，“図”モードでは，スペースキーによって〈円〉，〈リボン〉のそれぞれにおいて，〈線分〉と〈領域〉の二つの表示形態が切り替わる。

また，リターンキーで一時停止，エスケープキーで目下の課題の中止が，それぞれできる。

数字は，画面の任意に表示させる必要から，グラフィック画面に描く。そこで，《メモリ上にフォントをおき，呼出しに応じてこれを画面に表示する》機能の関数を使うことになるが，これのために実行プログラムのサイズはかなり大きくなる。

表示は，《描画を裏 VRAM で行ない，描き終わったら表 VRAM と切り替える；そしてこの繰り返し》のやり方で行なう。このとき，既に描かれている絵をすべてクリアし，すべてのカウンタを描き直す。これは，一つの数からつぎの数に移行する時間をほぼ一定にすることに効き，また実際，この時間を短くすることは実用の点で意味がない。

カウンタを描くルーチンでは，再帰の技法を用いる。但しこの場合，再帰の使用はむしろ必然である。