プログラミング教育の比較研究

―― C言語とCOBOL言語 ――

太田信宏

1．はじめに

インターネット，ウィンドウズ，マルチメディア，GUI(グラフィカルユーザインターフェース）といった言葉が最近の情報社会におけるキーワードになっている。このような社会状況とそれに伴う変化によってプログラミング言語としてのC言語の役割と位置付けがこの数年，非常に注目されてきている。

C言語は非常に適用範囲が広いという特徴を持っており，グラフィックス，アプリケーションソフト，通信，ゲームなど様々な分野で利用されている。「C言語とは」というときに幅広い適用分野のことがすぐに言われるが，実はもう一つの側面として情報処理教育あるいはプログラミング教育を行うのに非常に適した言語の一つであるということが言える。

C言語が注目されている一つの裏付けとして情報処理技術者試験や情報処理活用能力検定（J検）などの各種資格検定において，この数年の間にC言語受験者の数が急増しているという状況がある。このような動向はCプログラミング技術者の需要が高まっているという社会背景があると同時に，広く教育現場においてC言語が使われていることを示すものである。

本論では，Cプログラミングを習得するためにはどのようなことが必要であるかという観点から，C言語教育の目標や到達点，指導内容，留意点などを考察していくものとする。

なお筆者は過去にCOBOL言語を使用してプログラミング教育を行ったことがある。特徴の異なる2つの言語「CとCOBOL」ではプログラミング指導上にも当然いろいろな相違点がある。筆者の場合はすでにCOBOLの知識を持っている学生に対してC言語の指導を行った。このように「すでにCOBOLを習得した学生が第2言語としてC言語を学習する」ケースなども考えられる。このケースでは「COBOLで出来ることと出来ないこと（があること）を理解させる」また「C言語に適した業務とCOBOLに適した業務があることを理解させる」などが重要になる。

以降ではC言語とCOBOL言語の指導上の共通点や相違点などをいくつかの視点から比較検討し，論述することとする。

2．C言語教育の意義と目標

C言語が普及する以前は事務処理システムならばCOBOL，科学技術計算には FORTRANというように，適用する分野によって使用する言語が異なっていた。これがC言語の登場によってさまざまな分野のソフトウェア開発が，そのかなりの部分をC言語のみで行えるようになってきた。

C言語の適用範囲の広さは，ソフトウェア開発の生産性向上という点から見て望ましいことであると同時に，情報処理教育あるいはプログラミング教育という点から見ても，教材（道具）として非常に適したものであると言える。

その理由としてC言語は他の言語仕様ではサポートされていない機能を数多く持ってい

るために，従来の高級言語にはない概念を学習することができるからである。またC言語を学ぶことによって得られた知識は，ハードウェアの構造やOSの仕組みを理解する上で非常に役に立つ。さらには他の高級言語を習得していく上においても大きな手助けになると考えられる。

C言語の指導上のポイントとなるものを列挙するとおおよそ次のようになるであろう。

(1)構造化プログラミング

基本三構造を理解するとともに，適切な制御構造と文を用いてコーディングができること

(2)基本的なデータ型とデータ構造

変数や定数のデータ型の種類，配列およびポインタの概念，構造体や共用体を用いてのさまざまなデータ構造の表現法などをマスターすること

(3)演算子と式

代入演算，算術演算，論理演算，インクリメント・デクリメント，シフト演算，ビット操作など，豊富な演算子や式を正しくかつ効果的に使用できること

(4)モジュール化の概念

関数と引数の使い方，記憶クラス，モジュール分割とインターフェース設計，再帰呼び出し，ライブラリの利用法などをマスターすること

(5)ファイル処理

ファイルの概念，FILE構造体，ファイルに対する入出力関数などをマスターすること

3．カリキュラムの概要比較

「C言語の標準カリキュラム」はCAIT（中央情報教育研究所）や専教振（専修学校教育振興会）などで定めたものが1つの指針になっており，そこには目安となる授業時間数も設定されている。

この標準カリキュラムにおける教育目的や授業時間数を前提におきながら，C言語教育の「指導目標」，「指導内容」，「指導上の留意点」などを考察しておきたい。

専教振の「標準カリキュラム」が設定している授業時間数は，総時間数180時間を最小ラインとしている。これは90分授業で換算するとほぼ通年4コマ分（120コマ）となる。

表1は専教振の「標準カリキュラム」の概要（コマ数は90分換算）である。

比較のために筆者がCOBOLのプログラミング教育を行ったときの年間カリキュラムを表2に示す。COBOLの方は年間コマ数合計が92コマとなり，これは通年3コマ分に相当する。(実際の授業は通年4コマを9ヶ月程度で行った。)

両者を比較すると，CとCOBOLの年間授業コマ数の比は約4：3となり，Cの方がより多くの授業時間数を必要とすることがわかる。

ただしプログラミングの授業というのは「机上」で説明して知識習得をさせる部分と，実際にコンピュータを使用して自分が作成したプログラムの動作を確認する「実習」の両面から成り立っている。一般には実習時間を十分にとり，様々な課題をより多くこなすことでプログラミングの力は定着していく。したがって実習のためにどの程度の時間数を当てるかで，年間コマ数はかなり変わってくる。その意味では先ほどのコマ数比4：3というのはあくまで一つの目安である。ただCの方が言語仕様の機能が多い分，授業時間数も多く必要であろう。

（表１）Ｃ言語標準カリキュラムの概要

NO.	項　　目	内　　容	コマ数
1	Ｃの概要	Ｃ言語とは，Ｃの特徴， Ｃプログラムの構造	2
2	データ型	データ型の分類，変数の宣言と定義， 変数の初期化　等	2
3	標準入出力	書式付き入出力， エスケープシーケンス	4
4	演 算 子	演算子の種類と使用法 （代入，算術，論理，関係， インクリメント・デクリメント，シフト）等	8
5	制御構文	if～else，switch～case，for，while， do～while，break，continue，goto文　等	16
6	配　　列	配列の宣言と定義，初期化， 一次元配列，二次元配列	12
7	ポインタ	ポインタ変数，ポインタとアドレス， 配列とポインタ，ポインタ配列	12
8	関　　数	関数の構造と宣言， 引数と戻り値，再帰呼び出し 関数へのポインタ	19
9	プリプロセッサ	プリプロセッサの機能，マクロ定義	4
10	データ型と記憶クラス	記憶クラス，外部変数， 分割コンパイル	4
11	ライブラリ	標準ライブラリ，ヘッダ，　文字検査， 文字列関数	5
12	構造体と共用体	構造体の宣言，構造体変数， 構造体メンバ参照と入出力， 構造体配列，構造体ポインタ， 構造体と関数， 共用体の宣言，共用体変数， 共用体メンバ参照と入出力， ビットフィールド， 線形リスト，二分木構造	25
13	ファイル処理	ファイル処理の概要，FILE型構造体， 入出力関数	7
		（合　計）	120

（表２）COBOL年間カリキュラムの概要

NO.	項　　目	内　　容	コマ数
1	COBOLの概要	COBOLとは， プログラムの特徴と構造	2
2	基本命令	データ部の定義と数値の編集， 分岐およびループ処理の基本構造， COBOL基本命令， ファイルおよびレコードの入出力	10
3	明細表作成	明細行の編集，出力行数の制御， 改ページと見出しの処理， グループインジケーション	10
4	グループ コントロール	グループごとの集計や改ページの制御， キーのブレークと保存， 多段階のグループコントロール	8
5	表操作	表の定義と基本操作，添字と指標， １次元の表 ～ ｎ次元の表， 内部分類，テーブル集計， 表探索（線形探索と二分探索）	16
6	マッチング	１：１のマッチング， １：ｎのマッチング， ファイルの照合，併合，保守	16
7	データチェック	数字チェック，範囲チェック，順序チェック など各種データチェックを含む処理	8
8	整　　列	COBOLの整列機能（SORT文）， USING/GIVINGのソート， RELEASE/RETURN のソート	8
9	索引ファイル 相対ファイル	ファイルの定義とキーの指定， 順アクセスと乱アクセス， レコードの検索と更新	10
10	その他	プログラム間連絡機能を用いた処理， 条件名条件 など	4
		（合　計）	92

4．C言語教育の指導内容と指導上の留意点

標準カリキュラムの各項目について，C言語の指導内容および指導を通じて感じたポイントや留意点などを以下に示す。

(1)Cの概要

①C言語とは

・C言語開発の歴史と現在のソフトウェア業界における位置付けなどについて説明する。

・高級言語でありながら，アセンブラのような機械語レベルの処理が可能であるなど，アセンブラ言語と対比させた説明も行う。

②Cの特徴

・プログラムの記述性，移植性，生産性，保守性，適用分野，および他のプログラム言語との相違点などC言語の特徴を理解させる。

・書式やコーディング規約について説明する。

③Cプログラムの構造

・main関数を含む1つ以上の関数の集合体で構成されていることを説明する。

・main関数がプログラムの実行開始位置であることを理解させる。

・プログラムは完全なブロック制御構造であることを理解させる。

・注釈行（コメント）のつけかたを説明する。

(2)データ型

①データ型の分類

・定数と変数およびそのデータ型について説明する。

・記憶領域の大きさとメモリ上のアドレスの概念をデータ型と関連させて理解させる。

・それぞれのデータ型の長さと取りうる値の範囲，符号の意味，内部的な表現方法などを説明する。

②変数の宣言と定義，初期化

・すべての変数は使用する前に宣言する必要があることを説明する。

・変数の宣言と同時に初期化する方法について説明する。

(3)標準入出力

①書式付き入出力

・printf関数，scanf関数について説明する。

・書式部分（変換仕様）とデータの対応関係を理解させる。

・printf関数とscanf関数の引数の違い（変数とポインタの違い）を認識させる。

②エスケープシーケンス

・改行やヌル文字などの制御文字について説明する。

・文字として表示できないものをエスケープシーケンスという形で表現する意味を理解させる。

(4)演算子

①演算子の種類と使用法

・代表的な演算子である代入演算子，算術演算子，論理演算子，関係演算子，インクリメント・デクリメント，シフト演算子について説明する。

・代入の(=)と関係の(==)の違いを強調する。(特にCOBOLを知っている場合は間違えやすい。)

・異なる型同士の演算等で起こる型変換やキャスト式の意味を理解させる。

・演算子に優先順位があることや，式の実行（解析）順序などを説明する。

(5)制御構文

①if～else文

・条件式の書き方（関係演算子や論理演算子の使い方）を説明する。

・条件の判定は「真」が0以外，「偽」が0であることを説明する。

・if文を使ったいろいろな表現法を身につけさせる。(単文と複文，else ifを使った場合分けなど)

②switch～case文

・switch文の使い方を説明する。

・break文やdefault文について，これらがある場合とない場合の違いを理解させる。

・caseラベルを連続して置くことでOR条件が表現方法できることを説明する。

・①のelse if文やCOBOLのEVALUATE文などと比較しての説明も行う。

③ループ構造

・前判定型のfor文，while文について説明し，相違点などを理解させる。

・ブロック化によるループ処理の範囲を理解させる。

・COBOLのPERFORM　VARYING UNTIL文と比較しての説明も行う。

・後判定型のdo文と前判定型との違いを理解させる。

・break文，continue文，goto文の動作を説明する。

・多重ループ（ネスト）を使ったプログラム事例を学習させる。

(6)配列

①配列の宣言と初期化

・配列の宣言のしかたを説明する。

・添字は0から（要素数－1）までであることを強調する。

・配列に初期値を設定する方法を説明する。

・配列名がその配列の先頭アドレスを表していることを認識させる。

②二次元配列

・二次元配列の宣言と初期化のしかたを説明する。

・要素の入出力方法をいろいろな事例で説明する。

・char型では文字列の一次元配列として処理できることを理解させる。

(7)ポインタ

①ポインタ変数とアドレス

・ポインタ変数とはどのようなものであるかを通常の変数と比較して説明する。

・アドレス演算子とポインタによるオブジェクトの間接参照について説明する。

②配列とポインタ

・ポインタへの加減算は，ポインタ宣言の型によって値が変わることを理解させる。

・配列とポインタは密接な関係があることを説明すると同時に，その相違点も明確にしておく。

③ポインタ配列

・ポインタ配列を使って文字列の配列を表現する方法を説明する。

・ポインタのポインタを使って文字列の配列を表現する方法を説明する。

(8)関数

①関数の構造と宣言

・プログラムがmain関数を含む1つ以上の関数で構成されていることを説明する。

・関数の宣言，呼び出し方，return文について説明する。

②引数と戻り値

・仮引数と実引数について説明する。

・戻り値が1つの場合(return文)と2つ以上の場合（ポインタ使用）の違いを理解させる。

・call by valueとcall by referenceについて説明する。

・main関数のコマンドライン引数の使い方を理解させる。

③関数のプロトタイプ宣言

・プロトタイプ宣言の方法とその目的を把握させる。

④再帰呼び出し

・再帰呼び出しの事例や利点，欠点について説明する。

⑤関数へのポインタ

・関数名自体がアドレスを持ち，これを引数とすることができることを説明する。

(9)プリプロセッサ

（筆者が行った授業ではインタプリタ型のRUN/Cを使ったため，プリプロセッサについての十分な実習は行えなかった。ただし本来指導すべき内容としては以下のものがあげられる。）

①プリプロセッサの機能

・ソースプログラムの翻訳の前処理を行う機能で，＃という文字で始まる特別な行であることを説明する。

・分割コンパイルにおいて，共通部分をヘッダにすることによる開発効率や保守性について理解させる。

②マクロ定義

・マクロ定義の活用によるプログラムの見やすさや保守性の向上について理解させる。

・関数形式のマクロと通常の関数の使い分けを身につけさせる。

・マクロ定義を利用したデバッグ方法について説明する。

(10)データ型と記憶クラス

①変数と記憶クラス

・変数と記憶クラスの関係を説明する。

・変数の記憶期間と初期化のタイミングについて理解させる。

・記憶クラスの利用のしかたがメモリの有効活用につながることを理解させる。

・関数の記憶クラスと分割コンパイルおよび外部結合，内部結合について説明する。

(11)ライブラリ

①標準ライブラリとヘッダ

・汎用的に利用する関数やマクロはライブラリとして整理すること（あるいは整理されていること）を説明する。

・ヘッダの取り込み方法について理解させる。

②文字検査，文字列関数

・標準ライブラリにある関数について説明し，よく使われる文字検査や文字列関数などを習得させる。

(12)構造体と共用体

①構造体の宣言と構造体変数

・構造体とは複数の型を１つにまとめて作り出した新しいデータ型であることを理解させる。

・構造体タグ名と構造体変数，typedefによる宣言について説明する。

・さまざまなデータ構造を構造体で表現する能力を身につけさせる。

②構造体メンバの参照と構造体ポインタ

・構造体メンバの参照方法と構造体ポインタについて説明する。

・メンバ参照の演算子であるピリオド（.）と（->）の違いを理解させる。

③共用体の宣言と共用体変数

・共用体とは複数個のデータを同じアドレスに割り当てるデータ型であることを理解させる。

・型の定義や共用体変数の使い方は構造体と同様であることを説明する。

④共用体メンバの参照と共用体ポインタ

・共用体メンバの参照方法と共用体ポイン

タについて説明する。
・メンバ参照の演算子であるピリオド（.）と（->）の違いを理解させる。
⑤構造体や共用体のプログラム事例
・さまざまなプログラム事例をもとに構造体や共用体の使い方を理解させる。
（線形リスト，二分木構造，ビットフィールドなど）

⑬ファイル処理

①ファイル処理の概要
・C言語におけるファイルの考え方を説明する。
・標準入出力(stdin，stdout，stderr)について説明する。
②FILE型構造体
・入出力を行うために必要なFILE型ポインタについて理解させる。
③入出力関数
・標準入出力関数および低水準の入出力関数について説明する。
・レコード単位の入出力，定義したレコード以外（より細かい単位）での入出力およびランダムアクセスなど，いろいろなアクセス方法を身につけさせる。

5．C言語教育が目指すもの

前項では，C言語標準カリキュラムの項目にしたがって授業内容を分析してきた。次に「標準カリキュラムが目指しているC言語教育」と「Cプログラミング学習を通して身につけてほしいもの（特にCOBOLプログラミングと比較して）」などをまとめておきたい。

(1)標準カリキュラムが目指すC言語教育

ここ数年，高等教育機関におけるプログラミング教育・言語教育の中心にC言語がおかれている。「なぜ今Cなのか」に対する明快な答えの一つとして，産業界の需要という社会的背景があげられる。

さまざまなアプリケーションに対応できるC言語は，適用分野の広さや開発効率などの点から現在のコンピュータ業界においてメジャーな存在となっていることは間違いなく，そのような現場で必要とされる技術者の育成は重要な課題である。

ただ，「産業界がCの人材を求めているから，Cを教えている」というだけでは，C言語教育の目的の一面をとらえているにすぎない。

C言語を教えるもう一つの側面としては，Cを勉強していく過程の中で身についていくと考えられる「アドレスやメモリに対する感覚」，「構造化やモジュール化の概念」，「他の高級言語ではサポートされていない機能や，さまざまなプログラミング技法」などの知識習得があげられる。そしてこれらを学習することがさらに別の知識を習得していく上での大きな助けとなるのである。それらはたとえば次のような項目であろう。

・ハードウェア（CPUやメモリ）に関する知識習得
・OSの概念や構成，仕組みなどの知識習得
・他のプログラミング言語についての知識習得

すなわち，「なぜ今C言語教育が必要なのか」に対する答えは次の2つの点に要約することができる。

①産業界から求められている人材を育成する必要がある。
②コンピュータ教育の題材の1つとしてC言語は非常に効果的である。

(2)Cプログラミングの基本習得のために

まず第一に高級言語の一つとして，Cの基本的な文法を理解し，制御構文にしたがった

構造化プログラミングができることが必要である。

アルゴリズムの組み立て方や基礎的なプログラミング技法というのは言語の種類に関わらず必要になる知識である。その意味で他の言語（ここではCOBOLを例にとる）のプログラミングを知っていればそれだけ有利である。その場合COBOLにおける表現と比較しながら指導することが効果的であろう。

COBOLでもCでも，どちらでも表現できるアルゴリズムについてはそれぞれの文法にしたがって記述したものを並記して示し，これを比較することで共通点や相違点を説明することができる。

逆にCOBOLでは表現しにくい，あるいは処理することが不可能なアルゴリズムやデータの表現方法(例えばポインタや構造体など)については，かなり時間をかけた指導が必要となる。

これらはC言語特有のいわゆる「Cらしい」部分であり，教える立場から言えば最も強調し，また力点を置いて指導をしたい部分である。ただ学生にとっては未知のわかりにくい概念であること，さらに授業時間数という物理的な制約などもあって，聞く側が本当に理解できるまで指導するというのは難しい面もある。

ポインタや構造体というのはC言語特有の概念である。これらの本当にCらしい特徴やこれを利用した「プログラミング技法」といったものをどこまで指導していくべきか，そして学生はそれをどこまで理解することができるのか，この見極めがC言語教育の大きな課題の1つであると考える。

6．まとめ

以上，C言語教育の目標やカリキュラムの内容・目的などを述べてきた。最後にまとめとしてC言語教育のあり方や果たすべき役割を述べておきたい。

①C言語を通してコンピュータ・アーキテクチャを理解させること

前述のポインタを学習することはメモリやアドレスの概念を理解する上で役に立つ。また，データ型の種類や特徴を知ることで記憶領域内のデータの表現形式を理解することができる。さらにビットの論理演算やシフト演算から2進数表記やシフトの意味，符号と数値の関係などを知ることができる。

本来これらはC言語を学習する上での前提となるべき知識である。

しかしこの中には抽象的な概念もあり机上の学習だけではなかなか身につきにくい点や，理解はできていてもなかなか実感できないという側面も持っている。したがってこれらコンピュータ・アーキテクチャに関する知識をC言語の授業を通して復習することによって，より確実に自分の知識として習得することができるのである。

②C言語の位置付けや適用分野を理解させること

COBOLのプログラミング教育の場合は，事務処理システムをコンピュータ化していく場合にどのようにプログラミングすればよいかという点を主として学習する。COBOLを知っている学生が第2言語としてC言語を学習する場合はC言語の習得が終了した時点で「Cではどのようにしてプログラミングを行い，どのようなシステムで利用できるのか」ということをCOBOLシステムと比較して理解できていなければならない。

複数の言語を学ぶこと，それぞれの言語の特徴を理解することによって，要求されたシ

ステムの中で何が最適な言語であるかを次第に判断できるようになる。

システムの適用範囲が大変に広く，いろいろなアプリケーションで利用されるというＣ言語の特徴は標準カリキュラムの第一単元にも書かれていることである。しかしこれを言葉だけで説明してみてもなかなか実感するのは難しいであろう。だからといってこれらの様々なアプリケーションプログラムの１つ１つを授業の中で実際に作成していくことはほとんど不可能である。

１つの方法としてＣ言語で作成されているアプリケーション（例えばワープロソフトなど）の実物を用意し，その中の一部分の処理をそれと対応するソースコードとともに見せて（もちろん学生が読みとれるレベルでないとあまり意味がないが),そこで行われているプログラム技法などを紹介し，説明する。

これをグラフィックソフトのプログラムでも，あるいはマルチウィンドウシステムのプログラムでも，あるいはロールプレイングゲームのソフトでもというように，いろいろな分野のソフト（の一部）を少しずつでもいいから紹介し，学生になるべく多くのＣアプリケーションソフトに触れさせる機会を授業の中で持つような手法も考えられるであろう。

こうすることで，たとえ実習時間はあまり多く取れなかったとしても，学生の頭の中で「Ｃ」でできることのイメージがより広がっていくのではないかと思えるのである。

授業の現場を考えた場合いろいろ難しい点もあるが，以上はＣ言語とCOBOL言語の両者を指導して感じたものであり，反省をこめた１つの結論でもある。これを今後の機会への課題としたい。

〈参考文献〉

「Ｃテキスト30講」　三村　坦　著
「Ｃ言語」指導手引き書
財団法人　専修学校教育振興会
「構造化プログラミング入門　COBOL」
衛藤　敦・太田　信宏　著