NDC 007．63、548

フリーソフトウェアの教材化 II
－C言語学習システムの構築－

岸本俊祐*

The Usage of Free-software as a Teaching Aid II
-C System to Computer Practice-

Shunsuke KISHIMOTO*

A software system to study C-programming was assembled for beginners. It consists of several Free-Softwares that are a screen text editor, a full set of C-compiler, an object file linker and some utility programs.

The compiler system conforms to the new standard, known as ANCI C. The dos-shell is modified functionally for the users to operate this system easily. To help the users' operation in the training, this system provides many online manuals and an online help file which can be read directly on the CRT display.

This system is also usefull for developing applications in the C programming language.

1．はじめに

津山高専の情報工学科では、5年間でＢＡＳＩＣ言語をはじめ、アセンブリ言語等数種のプログラム言語を学習する。その中で、中心のプログラミング言語として位置づけられているものはＣ言語である。2年生でその導入教育が行われるが、以後、アルゴリズムやデータ構造等のプログラミングの基本を学ぶ上での記述言語として、また、卒業研究等で用いる実用プログラム開発言語として、上級生まで一貫して学習が行われる。

プログラミングやその言語の学習では、知識とともに技術を学ぶ側面が大きな割合をしめ、コンピュータを用いた“実習”は初期の段階から欠かせない学習形態となっている。本校でもプログラミングやその関連科目では、ほとんど例外なくコンピュータ実習が実施されていて、それをいかに上手に行うかは担当者の腕の見せどころとなっている。ハードウェア上の工夫はもちろんのことであるが、とりわけソフトウェアは工夫の余地や自由度が大きく、努力の成果が顕著に表れる。にもかかわらず、それぞれの実習でそれ専用にカスタマイズされたソフトウェアシステムはほとんど作られていないのが現状である。

我々は、専門の工学実験では実験テーマや目的に合わせて実験装置や方法等に色々工夫をこらすように、コンピュータ実習においてもそうあるべきだと考えて、まずパソコン実習導入教育用の専用ソフトウェアシステムを開発し、それを用いた実習を実施してきた[1]。それに続いて今回は、Ｃ言語によるプログラミング学習をサポートするためのソフトウェアシステムをフリーソフトウェアを中心に構築することにした。フリーソフトウェアを用いるのは、第1報でも示したが、ソフトウェア自身が簡単に手に入ること、ほぼ無償ですばらしいシステムが作れること、著作権等の心配なしにいくらでもコピーがとれること等の理由による。

＊ 情報工学科
平成5年8月31日受理

2．Ｃ言語学習システムに要求されるもの

第1報[1]で報告したように、一人1台のパソコンを用いて、ファイル操作や日本語テキスト処理を含むＤＯＳの基本操作についての実習はすでに実施ずみである。今回の実習システムは操作の連続性から、そのＤＯＳの操作性をそのまま継続させる必要がある。そのため、ＤＯＳシステムのコマンドシェル機能の拡張、ファイルマネージャー、日本語ＦＥＰ等の操作環境は1年生の実習システムとまったく同じにし、そこへＣ言語処理系を組み込むことにした。

プログラミング実習や実用プログラム開発で最も多く、そして長くかかわるソフトウェアツールは“エディタ”である。ソースコードの打ち込み、コンパイル時のエラーメッセージつぶし、実行時の試行錯誤によるディバッグ等々、

実習時におけるパソコン操作時間のほとんどの部分はエディタ操作で費やされているといっても過言ではない。そのため、エディタは高機能よりも操作性の善し悪しが実習の効率を大きく左右する。実習向きエディタは何よりも操作が単純明快で、手軽なことが要求される。基本機能としては、入力、挿入、削除、コピー・カット＆ペースト、検索・置換、ロード・セーブ等最低限のものが備わっていればよい。ブロック化や編集マクロ等、より高度の機能もあれば便利であるが、実習ではほとんど利用する機会がない。機能をしぼった軽くて動作の速いエディタが理想である。

実習システムの中心となるC言語処理系は標準的な機能を備えていること、つまり、コンパイラ本体はＡＮＳＩ Ｃ規格案に準拠していること、ヘッダーファイルや関数ライブラリ等は標準規格のものがきちんと備わっていること、リンカ等も含めソースコードのコンパイルからネイティブのマシンコード生成までの一連の処理に必要なコマンドモジュールをすべて含んでいること等が採用のための最低条件となる。また、教育上の視点からは、コンパイル処理の過程がよく見えるもの、分かりやすいエラーメッセージをだすもの、分割コンパイルやアセンブラ混在のソースコードを一括して扱うことができるものが望ましい。

その他、実用的なシステムとするためには、ぜひグラフィックス関係の関数ライブラリを含む処理系がほしいところである。しかし、グラフィックス処理はどうしてもハードウェア依存にならざるを得ないため、“標準的な”という条件とは矛盾する。このため、これはオプション機能とせざるを得ない。

以上、実習システムを構成する重要な３つのソフトウェア、つまり、ＯＳ、エディタ、Ｃ言語処理系について言及した。さらに、実習システムとして必要なものは、コンパイラドライバやハードコピー用ユーティリティ等がある。ＯＳを除き、以上述べたソフトウェアはすべてフリーソフトウェアの中から調達することができる。

まず、ＤＯＳ機能拡張ツールとして“ＫＳＨ”を、ファイルマネージャーとして“ＦＤ”を、日本語処理用ＦＥＰには“ＷＸＰ”を選んだ。これらのソフトウェアはすでに第１報で報告した通りである。また、スクリーンエディタとしては“ＳＥ３”を、Ｃ処理系としてはこの分野での唯一のフリーソフトである“ＬＳＩＣ８６（試食版）”を、プリントユーティリティとしては“ＰＲＴ”を選んだ。

実習システムを構築するにあたり、ファイルの圧縮や展開、メモリ常駐の状況チェック等をその都度行う必要があるが、圧縮・展開に“ＤＩＥＴ”や“ＬＨＡ”を、メモリチェックに“ＭＳ”をフリーソフトの中から選んで利用した[2]。

なお、プログラミング教育に伴うコンピュータ実習は、多くの場合、１クラス４０名の学生が一斉に実習を行うという形態をとらざるを得ない。しかも、教科担任が単独で指導を行う場合がほとんどである。他の専門実験・実習に比べて、学生１人当たりの指導者密度は極端に低い。このような形態の実習をサポートするため、システムに関するできるだけ多くの情報をオンラインマニュアルやオンラインヘルプとして準備し、実習者が操作に行き詰まったり、エラーをだした時点で、いつでも適切なメッセージや指示が得られるようになっていることが望ましい。そのため、実習システムを構成する主なソフトウェアにはオンラインマニュアルをつけ、さらに、エディタはオンラインヘルプが利用出来るようにした。

３．組み込みフリーソフトウェアの概要

実習システムに組み込んだり組み込み時に使用したフリーソフトウェアのうち、ＫＳＨ、ＦＤ、ＷＸＰ，ＤＩＥＴ等はすでに第１報で報告した。フルスクリーンエディタのＳＥ３、実習システムの中心となるＣ処理系のＬＳＩＣ８６試食版、プリントユーティリティのＰＲＴ等もこの分野ではよく名の知れたソフトウェアのため、多くの文献や紹介記事が見られる[3]。ここでは、その概要を紹介しておく。

①スクリーンエディタ “ＳＥ３”

情報処理教育のために開発されたフルスクリーンのテキストエディタである。実際に実習を行う教育現場からのニーズによって開発されたため、テキストエディタとしての必要にして十分な機能を備えている。圧縮されているとはいえ、本体の容量は約２０Ｋバイトしかなく、それでいて高速高機能の軽快なエディタである。

基本操作はオーバーラップ形ウィンドウ内のメニューをカーソルキーで選択する方式であるため、初心者でもほとんどマニュアル無しで容易に行える。また、随時オンラインヘルプが使用でき、操作に行き詰まったら簡易マニュアルを画面で参照可能である。

②フルセットＣコンパイラシステム “ＬＳＩＣ８６ 試食版”[4]

現在、パソコンのＤＯＳ上で動作している言語プロセッサは大部分が外国で開発されたものを日本向けにカスタム化したものであるが、このコンパイラは純国産である。ＡＮＳＩ Ｃ規格案（Ｘ３Ｊ１１）に準拠したフルセットＣコンパイラであり、機械依存のコードは生成しないので、ほとんどのＤＯＳマシン上で動作する。

“試食版”と称して製品版の評価用に公開されているシステムは、フリーソフトウェアであるが故の機能的な制限はなく、製品版とまったく同じということである。ただし、付属している関数ライブラリが、コード、データともに６４Ｋバイトのメモリ範囲に制限される“Ｓ”モデル[5]のみである。この唯一の制限は、大規模なソフトウェアを開発する場合は問題となるが、分割コンパイルの手法等も利用できるので、実習や卒業研究等教育で用いるレベルのプ

ログラム処理ではほとんど問題にならない。

このコンパイラシステムは、本体の他にプリプロセッサ、アセンブラ、リンケージエディタなどから成り、ソースコードをコンパイルし、ネイティブのマシンコードを生成するまでに必要な実行モジュールをすべて含んでいる。また、分割コンパイルやアセンブラコード、オブジェクトコードのファイルを一括して扱うことのできるコンパイラドライバ、メイクやプロファイラ等のプログラム開発やディバッグ時に有用なユーティリティもソースコード付きで収録されている。

③テキストファイル印刷ツール　“ＰＲＴ”

テキストファイルを整形してプリンタに印刷するためのソフトウェアツールである。様々な印刷様式をメニューから選択でき、通常の印刷の他に、縮小、縦書き、２段組み、袋綴じ印刷が可能である。印刷するテキストファイルは、ファイラと同じ操作でディレクトリをたどりながら選択することができるので大変便利である。実習成果としてのプログラムや実行結果のハードコピーはもちろんのこと、かなり本格的な報告書でも、わざわざワープロソフトを立ち上げることなく、エディタとこのツールを組み合わせれば作成可能である。

４．Ｃ処理系の実装

パソコン通信やコンピュータ雑誌の付録フロッピーディスク[6]で入手できるＬＳＩＣ８６システムは、コンパイラ等すべてのソフトウェアが約４２０Ｋバイトの自己展開形式のアーカイブファイルにまとめて圧縮された形で供給される。これを実際使用できる形式に展開し、一連のシステムに組み上げていくインストール作業は以下のようになる。

①アーカイブファイルの展開

作業の第１段階は、アーカイブファイルを展開し、それぞれのファイル単位に分解することである。展開後のファイル容量は元の倍以上になるので、空き容量が１Ｍバイト以上あるフロッピーディスクかハードディスクが作業用に必要である。それを展開先のディスクに指定してアーカイブファイルを自己展開させると、作業ディスクに実行ファイル、ライブラリファイル、ヘッダーファイル、ドキュメントファイル等がきちんと分類され、それぞれのディレクトリのもとにコピーされる。展開後の全ファイル数は約７０本、合計容量は約１Ｍバイトである。以後これらのファイルを素材ファイルと呼ぶことにする。

②ＯＳへの組み込み

インストール作業の第２段階は、素材ファイルの中から実習用コンパイラシステムを構成するために必要なファイルを選び出し、ＤＯＳシステムディスクの中に組み込むことである。アーカイブファイルをパソコン通信で入手した場合はもちろんのこと、雑誌の付録等で得た場合でも、コンパイラシステムの内容についての説明はほとんどなく、せいぜいアーカイブファイルの展開の仕方が書いてある程度である。詳細な情報はすべて展開後に得られるオンラインマニュアルやドキュメントによらねばならない。オンラインマニュアルをエディタ等で読むと、コンパイラシステムの標準的な構成方法の説明があり、とりあえずそれに従って素材ファイルを組み立てていった。そのディレクトリ構造を図１に示す。[LSIC86] を親ディレクトリとして、そのもとに [BIN] にはコマンドモジュールが、[LIB¥S] には Ｓモデルのライブラリが、[INCLUDE] と [INCLUDE¥SYS]にはヘッダーファイルが納められている。

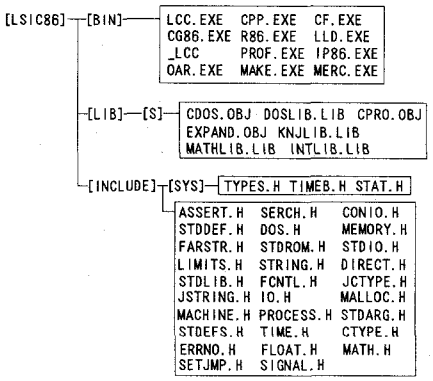

図１　コンパイラシステムのディレクトリ構造

③環境設定

インストール作業の第３段階は、ＤＯＳの環境変数の設定やコンパイラなどの実行モジュールがあるディレクトリへのパスの設定である。このコンパイラシステムは特別な環境変数を使用しないので、通常の FILES、COMSPEC、PASS等の環境変数を設定すればよい。ユーザープログラムで時間関数を使用するときは、実行時に時間関数が参照する環境変数 TZ にローカル時間と世界標準時との時間差を設定しておかねばならない[7]。

以上の作業がすんだ段階で、コマンドラインからソースファイル名を指定して、コンパイラやリンカ等を直接起動することにより、一通りのコンパイル作業を行うことのできるシステムとなっている。しかし、まだこの段階では、ヘッダーファイルやライブラリのパス指定、エラーや警告メッセージのレベル指定、コードサイズを小さくするか実行スピードを優先させるかの最適化指定等の多くのオプションパラメータをその都度入力しなければならないので、不便である。

5．Cシステムのカスタム化

ＬＳＩＣ８６システムには、定型的なコンパイル作業を行う場合、便利に使用できる“コンパイラドライバ”がソースコード付きで付属している。これを用いれば、コンパイラ、アセンブラ、リンカ等を順次呼び出して、Cのソースコードをコンパイルし、マシンコードを得るまでの一連の作業を自動的に行ってくれる。さらに、コンパイラやリンカに与えるオプションパラメータを“コンフィギュレーションファイル”という特殊ファイルに記述しておくことにより、コンパイラドライバが必要なパラメータを読み取ってくれるようになっている。そこで、あらかじめ実習向きの環境に合わせたコンフィギュレーションファイルを作っておけば、コンパイラドライバを起動するだけで一連のコンパイル作業はすべて自動的に行われるようになる。実習用に書き直したコンフィギュレーションファイルの１例を図２に示す。

```
# LSI C-86 compiler's configuration file

-DLSI_C
-XC:¥LSIC86¥BIN -LC:¥LSIC86¥LIB
-IC:¥LSIC86¥INCLUDE -T -O
-jl -v2 -acdos.obj $LSICOPTS
&
-lmathlib -lknjlib -ldoslib
```

図２ コンパイラドライバの
コンフィギュレーションファイル

また、１Ｍバイト以上の増設メモリが使用できるパソコンを用いる場合は、それをＲＡＭディスクとして用い、一連のコンパイル作業がすべてこのＲＡＭディスク上で行えるようにシステムを設定するバッチファイルも用意した。

6．完成システムと試用結果

完成した学習システムの総ファイル容量は、可能な限り圧縮をほどこしても、約１５００Ｋバイト強となる。そのうちＣ処理系が４１０Ｋバイト、ＦＤ等のコマンドファイルやデバイスドライバ等のシステムファイルが２６０Ｋバイト、日本語の辞書ファイルが３８０Ｋバイト、オンラインマニュアル類が５２０Ｋバイトとなっている。２ＨＤのフロッピーディスクでも全体は収納しきれないので、オンラインマニュアル類のみを別にして、他のファイルすべてを１枚のディスケットに納め、主システムとした。

主システムのディレクトリ構造の主要部分を図３に示す。ただし、Ｃコンパイラシステムのディレクトリ構造は図１に示したので、ここでは省略してある。このシステムは日本語処理も含めて、Cのソースコードを入力し、コンパイルし、マシンコードを生成するまでに必要なすべてのソフトウェアが含まれていて、約１Ｍバイト容量となっている。空き容量は２００Ｋバイト程度あり、ここを作業領域として、ほとんどの場合このフロッピーディスク１枚で実習を行うことが可能である。

オンラインマニュアル類はもう１枚のディスケットに格納し、それを副システムとした。Ｃ処理系についてのマニュアルは３５０Ｋバイトほどもあり、システム構成についてはもちろんのこと、標準ライブラリの中の１つ１つの関数仕様についての説明まで、ありとあらゆる情報がつまっている。これらのオンラインマニュアルはリードオンリーモードで格納し、エディタで読みだしたとき、不用意に書き換えたり、消去したりできないようにした。副システムの未使用部分は約７００Ｋバイトあり、例題プログラムの格納や実習作業領域として利用できる。

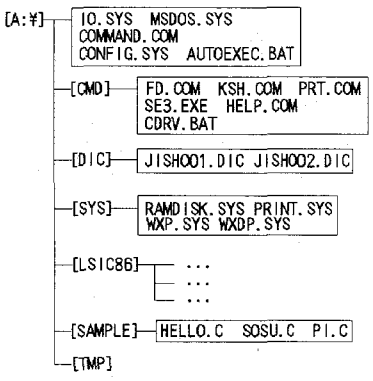

図３ 主システムのディレクトリ構造

この実習システムの使い勝手を調べるため、プログラミング実習で出題される課題のうち、応用問題の中から少し大きめなＣプログラムを処理してみた。この例題はマチンの公式[8]による級数展開で、円周率πの値を多数桁求めるものである。図４が解答例の１つで、プリントユーティリティのＰＲＴを用いて、行番号付きで、出力したものである。小数以下１万桁まで正しい結果が得られる完全なソースコードで、分かりやすいアルゴリズムにすることを最優先にプログラミングがなされており、実行速度に関する最適化はほとんど行なっていない。このプログラムを実習システムで処理した時のコンパイルメッセージと実行結果の一部を図５に示す。

コンパイルメッセージは、できるだけ多く出力されるようコンパイラのコンフィギュレーションファイルを設定しているので、プリプロセッサ、パーサ、コードジェネレータ等の処理の過程で、図に示す様なメッセージがリアルタイムで次々と出力される。

プログラムは関数のプロトタイプ宣言や本体定義で仮引数を（ ）の中で宣言してしまうなどのＡＮＳＩ規格の新しい仕様で記述されているが、問題なくコンパイルできる。

実行結果は全部を掲載するには長すぎるので、小数以下

```
 1: /***  pi 10001 keta ***
 2:       1992/10/26
 3: */
 4:
 5: #include <stdio.h>
 6: #define KETA 10000
 7: #define N KETA/5+5
 8:
 9: long pi[N],a[N],b[N];
10: void add(long *,long *,int);
11: void sub(long *,long *,int);
12: void div(long *,int,long *,int *);
13:
14: void main(void)
15: {
16:     int n,p;
17:
18:     for(n=0;n<N;n++) a[n]=b[n]=pi[n]=0l;
19:     a[0]=16*5l;
20:     n=1;
21:     p=0;
22:     while(p<N-1) {
23:         div(a,25,a,&p); div(a,n,b,&p);
24:         add(pi,b,p);
25:         n+=2;
26:         div(a,25,a,&p); div(a,n,b,&p);
27:         sub(pi,b,p);
28:         n+=2;
29:     }
30:     for(n=0;n<N;n++) a[n]=b[n]=0l;
31:     a[0]=4*239l;
32:     n=1;
33:     p=0;
34:     while(p<N-1) {
35:         div(a,239,a,&p); div(a,239,a,&p);
36:         div(a,n,b,&p)  ; sub(pi,b,p);
37:         n+=2;
38:         div(a,239,a,&p); div(a,239,a,&p);
39:         div(a,n,b,&p)  ; add(pi,b,p);
40:         n+=2;
41:     }
42:     printf("\n\n*** pi %d keta ***\n%11d.",
43:                                 KETA+1,pi[0]);
44:     for(n=1;n<N-4;n++) {
45:         printf("%05ld%05ld",pi[n],p[n+1]);
46:         n++;
47:         if((n%10)==0) printf("\n  ");
48:     }
49:     printf("\n");
50: }
51:
52: void add(long *a,long *b,int p)
53: {
54:     long ca=0l;
55:     int i;
56:     for(i=N-1;i>=p;i--) {
57:         *(a+i)+=*(b+i)+ca;
58:         ca=0l;
59:         if(*(a+i)>=1000001) {
60:             *(a+i)-=1000001;
61:             ca=1l;
62:         }
63:     }
64: }
65:
66: void sub(long *a,long *b,int p)
67: {
68:     long bo=0l;
69:     int i;
70:     for(i=N-1;i>=p;i--) {
71:         *(a+i)=*(a+i)-(*(b+i))-bo;
72:         bo=0l;
73:         if(*(a+i)<0l) {
74:             *(a+i)+=1000001;
75:             bo=1l;
76:         }
77:     }
78: }
79:
80: void div(long *a,int n,long *b,int *p)
81: {
82:     long x,r=0l;
83:     int i,sw=1;
84:     for(i=*p;i<N;i++) {
85:         x=*(a+i)+r*100000l;
86:         *(b+i)=x/n;
87:         r=x%n;
88:         if((*(b+i)==0l) && sw ) *p=i;
89:         else sw=0;
90:     }
91: }
```

図４　マチンの公式によりπの値を
小数以下１万桁求めるプログラムのソースコード

最初の５００桁と最後の３００桁を残し途中を省略した。最後の行の最後の桁がちょうど小数以下１万桁目で、正しい結果が得られていることがわかる。

```
C:¥>lcc pi
cpp -DLSI_C -IC:¥LSIC86¥INCLUDE -j -o 1.$$$ pi.c
cf -chct -j 1.$$$ 2.$$$ 3.$$$
cg86 2.$$$ 3.$$$
main_ __...:::;
add_ __...:::;
sub_ __...::::;
div_ __...:::_...::::;
rg86 -o pi.obj -m pi 3.$$$
lld link.i

C:¥>pi

*** pi 10001 keta ***
3.1415926535 8979323846 2643383279 5028841971 6939937510
  5820974944 5923078164 0628620899 8628034825 3421170679
  8214808651 3282306647 0938446095 5058223172 5359408128
  4811174502 8410270193 8521105559 6446229489 5493038196
  4428810975 6659334461 2847564823 3786783165 2712019091
  4564856692 3460348610 4543266482 1339360726 0249141273
  7245870066 0631558817 4881520920 9628292540 9171536436
  7892590360 0113305305 4882046652 1384146951 9415116094
  3305727036 5759591953 0921861173 8193261179 3105118548
  0744623799 6274956735 1885752724 8912279381 8301194912
         .          .          .          .
         .          .          .          .
         .          .          .          .
  3084076118 3013052793 2054274628 6540360367 4532865105
  7065874882 2569815793 6789766974 2205750596 8344086973
  5020141020 6723585020 0724522563 2651341055 9240190274
  2162484391 4035998953 5394590944 0704691209 1409387001
  2645600162 3742880210 9276457931 0657922955 2498872758
  4610126483 6999892256 9596881592 0560010165 5256375678
```

図５　コンパイルメッセージと実行結果（一部）

図６はソースコードの３１行目と４０行目にわざとバグを入れてコンパイルしたときのエラーメッセージの出力例である。エラーメッセージは英文と和文のどちらかを選ぶことができる。最初の段階は和文に設定してある。

```
C:¥>lcc pi
cpp -DLSI_C -IC:¥LSIC86¥INCLUDE -j -o 1.$$$ pi.c
cf -chct -j 1.$$$ 2.$$$ 3.$$$
pi.c  31: ポインタでないものに * や -> が適用された
pi.c  40: 2aa は不正な数

C:¥>
```

図６　エラーメッセジの出力例

なお、図４のソースコードには、main() 関数の他３つの

関数が用いられている。これらの関数を独立のファイルにして分割コンパイルしたり、あらかじめコンパイルしておいて、オブジェクトコードになったモジュールをリンクさせる等、教育上いろいろなバリエーションが考えられるが、これらの処理は、備え付けのコンパイラドライバですべて試みることができ便利である。

7．まとめ

情報処理教育の一環として行われているC言語によるプログラミング実習や言語学習をサポートするための教材として、フリーソフトウェアを中心にまとめて、パソコン用Cプログラム開発システムを構築した。

このシステムの特徴は

①Cのソースコードを入力し、コンパイルし、マシンコードを生成し、それを実行して結果を得るための一連の処理に必要なソフトウェアがすべて1枚のディスケットに納められている。

②エディタやコンパイラ、プリントユーティリティ等の主な実行コマンドについてはすべてオンラインマニュアルを付け、それをいつでも画面上で読めるようにした。

③DOSのコマンドインタプリタ機能が拡張してある。コマンド入力のヒストリ機能やコマンドラインの編集機能が利用できるので、実習時のような、試行錯誤で何回も同じ手順をくりかえさねばならないときは、コマンド入力を大幅に削減でき、操作性が格段によくなっている。

④実習パソコンにRAMディスクが装備されていれば、そこへCシステムを展開してRAMディスクのシステムとして、装備されていなければ主ディスケットのみで、フロッピーベースのシステムとして使い分けができる。

⑤OSを除き、主要なソフトウェアはすべてフリーソフトウェアを利用している。

以上の特徴により、当初目標にしていた“自学独習のための実習用ソフトウェアシステム”がほぼ構築できたものと確信している。しかし、プログラミング実習は、“「何をすべきか」という仕様はあっても「どう作ればよいか」は指定されない場合が多く、通常の実験とは異なった指導が必要である”との指摘[9]には答えきれていない。この指摘は、それ以前の段階、「実習のための言語処理系がきちんと準備・整備されていること」は当然という前提でなされている。今回は、その前提をやっとクリアした段階だと思っている。その上でこの指摘にどう答えていくかは、今後の課題である。

終わりに、フリーソフトウェアの入手にパソコン通信を駆使して御協力いただいた、本校電子制御工学科中尾三徳技官に感謝いたします。

文　献

1) 岸本俊祐：津山高専紀要、No.30 (1992) 43.
2) 大池浩一：98ノート版MS-DOS定番フリーソフトウェア集(アスキー出版、1992) 38.
3) 大池浩一：秀作フリーソフトウェア100選 (アスキー出版、1992) 49.
4) 服部昌博：フリーウェアとC言語 (工学図書、1992).
SOFTBANK：C MAGAZINE Vol.5, No.4 (1993) 45.
5) 河西朝雄：TURBOC 初級プログラミング下 (技術評論社、1988) 326.
6) SOFTBANK：C MAGAZINE No.31 (1992).
7) ｴﾑ･ｴｽ･ｱｲｼﾞｬﾊﾟﾝ：LSI C-86 Ver.3.30 試食版ユーザーズマニュアル(1991) 3.
8) 森口繁一、他：数学公式Ⅱ (岩波書店、1970) 49.
9) 都倉信樹：情報処理、Vol.32、No.10 (1991) 1101.