プログラミング言語変換システムの開発

第1報　PADトランスレータの開発

堀内　征治*・堀内　泰輔**

A Development of a System for Program Transformation

(The first report : A System for the Transformation of Problem Analysis Diagram)

Seiji HORIUCHI, Taisuke HORIUCHI

In the development of software, there are many programing languages. In recent years, structured programing languages — C, structured Basic and so on— have been used for programing instead of earlier languages—Fortran, Basic, assembly language etc. Therefore, it is important to make a transformation between these two programing languages. But it is not so easy to make software for the transformation.

In this paper, an intermediate language is suggested in order to make it easy to develop the transformation software. On the basis of designing the intermediate language, a transformation system for PAD has been developed.

This transformation system has two essential subsystems. In the first subsystem, PAD drawn on the CRT screen is transformed into a source program of the intermediate language. And then in the second one, the source program is transformed into the target language, such as C, Basic, and so on.

1. はじめに

ハードウェア技術の驚異的な発展に追随するかたちで，ソフトウェアの面でも幾多のプログラミング言語の提案が行なわれ，最近では構造化プログラミング指向の言語（たとえばC言語や構造化BASIC）が注目され，着実にユーザ数を増加させている．

一方，これまでに蓄積したプログラム群が足かせとなって，言語の切り替えが困難なユーザが多いことも事実である．このような事情から最近では特定言語間のソースレベルでの変換ユーティリティが市販されるようになった．しかし，このようなユーティリティを開発する場合，変換前後の両言語間の差異のみに注目するため，ほかの2言語間の変換ユーティリティを再開発する際には，それまでに開発したプログラムをほとんど活かすことができず，結果的に多大の労力を強いられることになるだろう．

*　機械工学科　助教授
**　機械工学科　講師
原稿受付　平成元年9月30日

本研究は，これらの状況に鑑み，プログラミング言語間の汎用的な変換システムの開発を目標としている．つまり，文法の差異をデータベースとして記述するだけで，プログラムの変更は一切不要にしたものである．この目的を実現するためのひとつの方法として中間言語を用いることが考えられる．つまり，変換する言語をいったん共通基盤としての中間言語に変換し，次にその中間言語をターゲット言語に変換するという2段階に分割化する方法である．

本報では第一段階として，中間言語のアイデアをもとにしたPADトランスレータについて述べる．PADは，フローチャートに代わる，構造化プログラミング用のツールとして注目されているものであり，今回開発したのは，画面上に描いたPADを自動的に中間言語に変換するものである．さらに，得られた中間言語テキストをC言語などの実際のプログラミング言語に変換するツールも開発した．また，このシステムが特に情報処理教育に有用であることにも言及する．

2. 中間言語の設計

2—1 中間言語について

中間言語の考え方は，例えば，コンパイラを作成するときに用いられる．図1はUNIXシステムにおけるC言語コンパイラの処理過程を示す．この場合，アセンブリ言語が中間言語として位置づけられ，他の言語（例えばFORTRAN77）からの同様な中間言語出力とアセンブリ言語レベルで結合され，最終的に機械語からなる実行可能プログラムに変換される．この仕組みは直接機械語に翻訳するのに比べ手間がかかるが，コンパイラの開発効率，汎用性からみて有用な手法といえる．

本稿で提案する中間言語は，これを高級言語間変換のレベルにまで拡張しようとするもので，図2には従来の中間言語を用いずに直接変換する方法と本方法を対比して示した．本方法を3つ以上の言語間相互変換に適用すると大きなメリットを生ずる．つまり，特定言語を中間言語に変換するためのツールとその逆方向の変換ツールを一度開発しておけば，別の言語への（あるいは，特定言語からの）変換が必要になった場合，この新たな言語に関する同様な2つのツールを開発するだけでよく，あとはこうしてできた4つのツールを組み合わせることで目的が達成できる．これを従来の方法で行なおうとすると，双方の特定言語を意識しつつツール開発を行なう必要があり，また他の言語との変換が必要になったときにせっかく開発したツールがほとんど利用できないことになる．この点中間言語を用いれば，1度開発したツールは何度も再利用できる点で優れている．

2—2 中間言語の設計

まえがきで述べたように，プログラム変換の必要性は従来の比較的低機能な言語（以後，低言語と呼ぶ）から，最近注目されている構造化プログラミングを意識した高機能な言語（以後，高言語と呼ぶ）への変換（これを，低高変換と呼ぶことにする）が主であり，例えばC言語からBASICといった逆方向の変換（高低変換）はあまりニーズがないとみてよい．ただ，高言語間の相互変換は必要であろう．

このような実状を考慮して，中間言語のレベルを高言語，低言語のどちらに置くかを検討

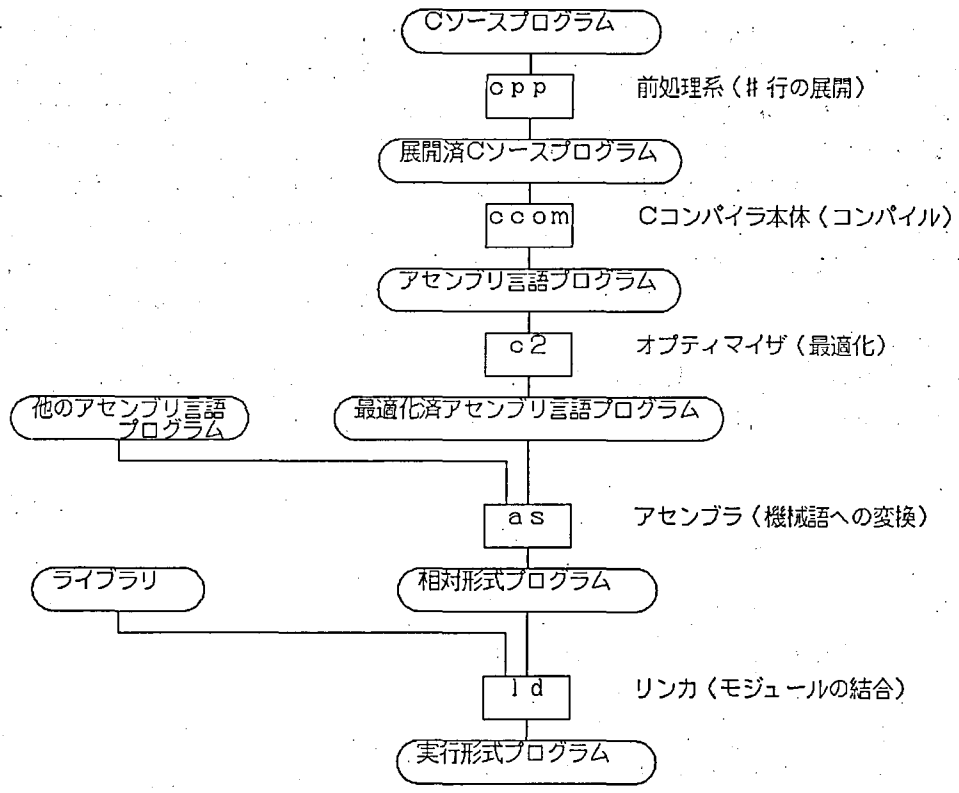
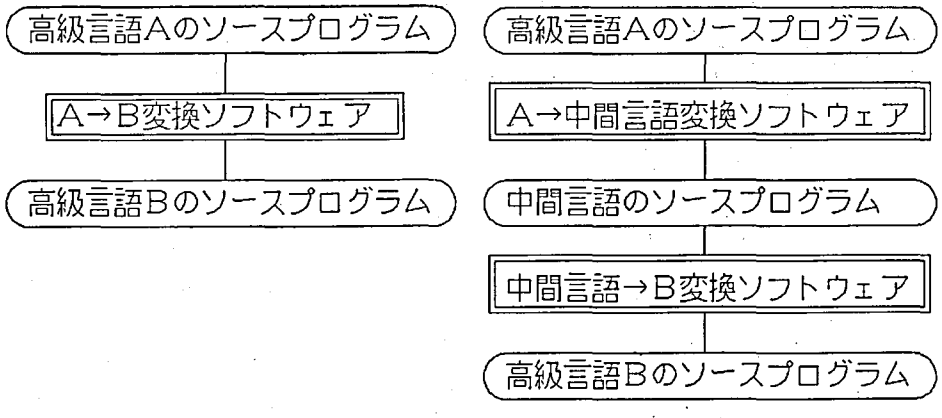


図1　UNIX (4.2/3) におけるCコンパイルの処理過程



(a)　直接変換　　(b)　中間言語を仲介とした変換

図2　2つのプログラム変換方法

する必要がある．一般に低高変換は，高言語の低言語以上の機能を犠牲にすれば容易である．逆に高低変換は展開を要するため極めて困難といえる．低言語間，高言語間の変換は，高低変換に比べれば容易である．この考察から，中間言語を低言語にすると，高言語間の変換をする際に高低変換が必要になり不利になる．一方中間言語を高言語にすれば，この問題は生じない．このことから，中間言語の機能を豊富にすることが低高変換においては重要となる．

前述のように，通常は高低変換の必要性はあまりないが，情報処理教育の面から見ると，これも必要になる．つまり，構造化言語を教育したいが，現実にはそのための資源がなく，BASIC 等の非構造化言語で教育せざるを得ないときに，構造化フローチャートとしての PAD や構造的な疑似言語を援用する場合，これら高言語から BASIC 等の低言語への高低変換ツールが必要となる．以上のことから，このような高低変換においては，中間言語のレベルを高言語，低言語のどちらに置いても差異はないといえる．

表1　中間言語の文法

```
<ﾌﾟﾛｸﾞﾗﾑ>::=<ｽﾃｰﾄﾒﾝﾄ>{<ｽﾃｰﾄﾒﾝﾄ>}
  <ｽﾃｰﾄﾒﾝﾄ>::=<代入文>|<入力文>|<出力文>|<dim文>|<制御文>
    <代入文>::=<変数>=<式>|swap <変数>,<変数>
    <入力文>::=in <変数>{,変数}
    <出力文>::=out <変数>|<式>{,<変数>|<式>}
    <dim文>::=dim <変数>(<整定数>){,<変数>(<整定数>)}
    <制御文>::=<ifﾌﾞﾛｯｸ>|<whileﾌﾞﾛｯｸ>|<forﾌﾞﾛｯｸ>|<break文>
      <ifﾌﾞﾛｯｸ>::=if <式><ｽﾃｰﾄﾒﾝﾄ>{<ｽﾃｰﾄﾒﾝﾄ>} endif|
                  if <式><ｽﾃｰﾄﾒﾝﾄ>{<ｽﾃｰﾄﾒﾝﾄ>} else <ｽﾃｰﾄﾒﾝﾄ>{<ｽﾃｰﾄﾒﾝﾄ>} endif
      <whileﾌﾞﾛｯｸ>::=do while <式><ｽﾃｰﾄﾒﾝﾄ>{<ｽﾃｰﾄﾒﾝﾄ>} loop
      <forﾌﾞﾛｯｸ>::=do for <整変数>=<式>,<式> <ｽﾃｰﾄﾒﾝﾄ>{<ｽﾃｰﾄﾒﾝﾄ>} loop
      <break文>::=break
<変数>::=<識別子>|<識別子><型文字>|<配列変数>
  <識別子>::=<英字>{<英字>|<数字>}
    <英字>::=a|b|c|d|e|f|g|h|i|j|k|l|m|n|o|p|q|r|s|t|u|v|w|x|y|z
    <数字>::=1|2|3|4|5|6|7|8|9|0
    <型文字>::=%|!|#|$
    <配列変数>::=<識別子>(<式>)|<識別子><型文字>(<式>)
<整変数>::=<識別子>|<識別子>%
<式>::=<単純式>|<単純式><２項演算子1><単純式>
  <単純式>::=<符号><項>|<符号><項><２項演算子2><項>
    <符号>::={+|-}
    <２項演算子1>::=<|>|<>|<=|<=
    <２項演算子2>::=+|-|or
  <項>::=<因子>|<因子><乗除演算子>
    <乗除演算子>::=*|/|mod|and
    <因子>::=<変数>|<関数名>(<式>{,<式>})|(<式>)|not <因子>|<定数>
      <関数名>::=abs|sqr|sin|cos|exp
      <定数>::=<数値定数>|<文字定数>
        <数値定数>::=<整定数>|<実定数>
          <整定数>::=<符号><数字列>
            <数字列>::=<数字>{<数字>}
          <実定数>::=<符号><数字列>.<数字列>
        <文字定数>::="<文字列>"
          <文字列>::={<英字>|<数字>|<特殊記号>}
            <特殊記号>::=!|#|$|%|&|'|(|)|=|~|||{|}|@|[|]|*|;|*|+|<|
                        >|?|/|.|,|_
```

2—3 PAD トランスレータにおける中間言語の設計

今回の開発では，PAD (Problem Analysis Diagram：1970年代に日立製作所で開発された構造化図法) から，低言語としての BASIC や高言語であるC言語や BASIC/98 (構造化 BASIC の1つ) へのプログラム変換を対象とした．これは現実に教育面で上記のニーズが高いためである．PAD がきちんとかければ構造化プログラミングを一応習得したことになるし，任意のプログラミング言語での記述への移行もスムーズである．よって，PAD は形態こそ異なるにせよ，一種のプログラミング言語として位置づけられよう．

このように，今回の PAD トランスレータ開発には，高低，低高の両変換を伴うこと，および対象を情報処理教育用としたことを考慮した．そのため中間言語のレベルとしては，初歩的なアルゴリズム教育に最低限必要な制御構造とステートメントを扱う程度とし，低言語に近いものに設定した．この中間言語の仕様を表1に示す．

3. システム概要

図3に PAD トランスレータの概要を示す．システム全体は，メニュープログラム，PAD エディタ，3本の中間言語変換プログラムの5本のプログラムからなる．

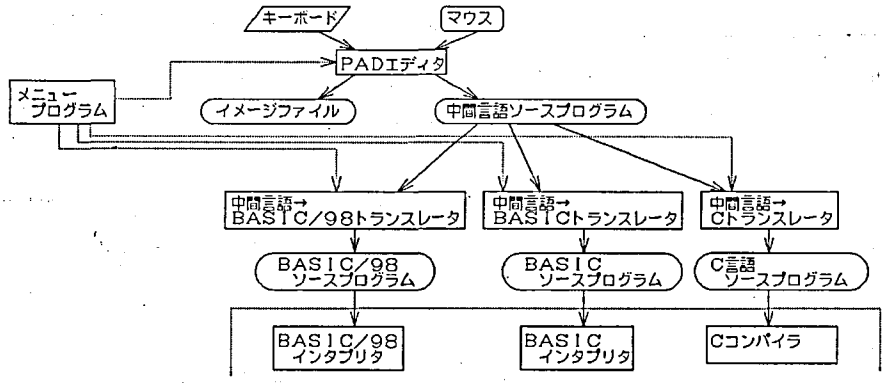


図3　PAD トランスレータシステム概要図

PAD エディタは PAD チャートを画面上に作成するためのソフトウェアである．入力装置としてはマウスを積極的に活用し，キーボードの使用は PAD シンボル内のステートメントやファイル名入力など，最小限に抑えた．さらに，マニュアルなしでも操作できるよう各 PAD シンボルや諸機能をアイコン化し，シンボルの配置・結線から中間言語への変換に至るまでを簡単な操作で行えるよう配慮した．なお，PAD エディタの出力は2つあり，PAD 図の再編集に必要なイメージファイルと，中間言語に変換されたソースファイルである．

中間言語から3つの特定言語への変換は，それぞれの中間言語変換プログラムにより行なわれる．制御構造の弱い BASIC への変換は高低変換，機能の多いC言語と BASIC/98 への変換は低高変換となる．

なお，これら一連の操作を簡単化するために，全体を統括するメニュープログラムも作成した．また，ターゲットマシンは FMR-50（富士通製）であり，ソフトの開発にはC言語(Lattice C) を用いた．

4. PAD エディタ

メニュープログラムからPADエディタに制御が移ると，図4に示す初期画面が現れる．これは，前述のように，新たにPAD図を作成する場合と，すでに保存されているものの再編集を可能とするように設計したもので，次の3群の機能をそれぞれ，画面左端にアイコン化した．

(1) PAD 作図のための各種シンボルボックスの配置と結線のための機能シンボル
(2) 画面のイメージファイルの保存および呼び出しのための機能
(3) PAD エディタからメニュープログラムへの移動用機能

これらの中で，(1)の作図に関するシンボルは図5に示す6種類に限定した．先に述べたように，PAD トランスレータを情報処理教育用の入門に有用とするための配慮から，必要最小限のものに絞ったためである．

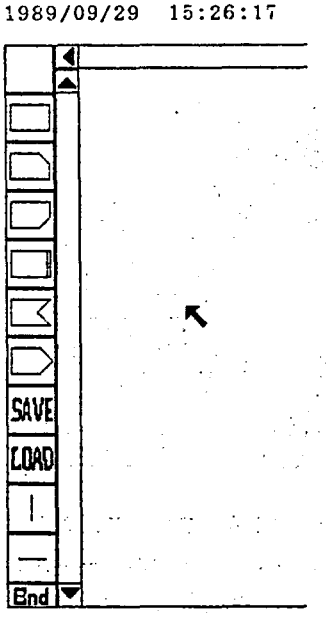


図4 PAD エディタの初期画面（部分）

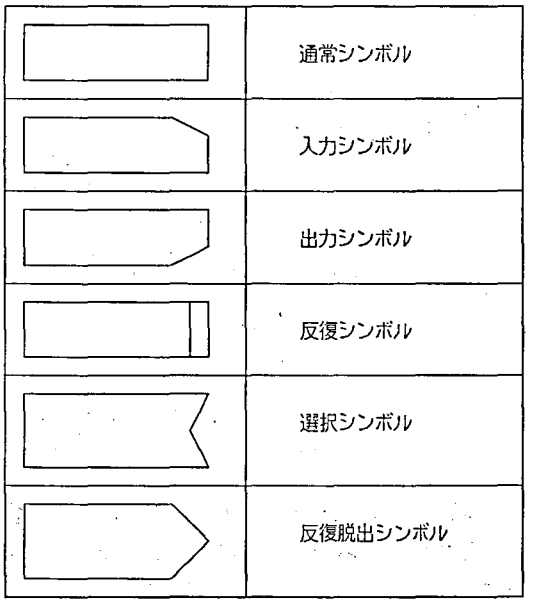


図5 PAD シンボル一覧表

PAD チャートの描画は，シンボルアイコンの選択と配置，そしてシンボル内への記述項目の入力を繰り返した後，各シンボルを結合するという手順をとる．原則としてマウスの左ボタンは選択，右ボタンはキャンセルとした．また，各シンボル結合の手続きでは，不適切なシンボル同士が結ばれないようチェックも施した．なお，画面は開始時の状態に対して上

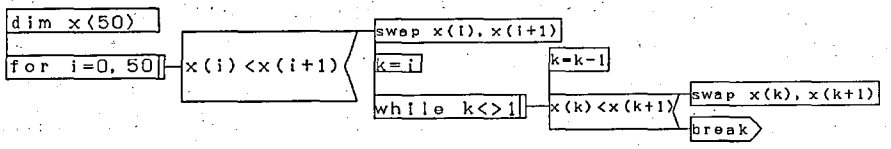


図6 PAD の例

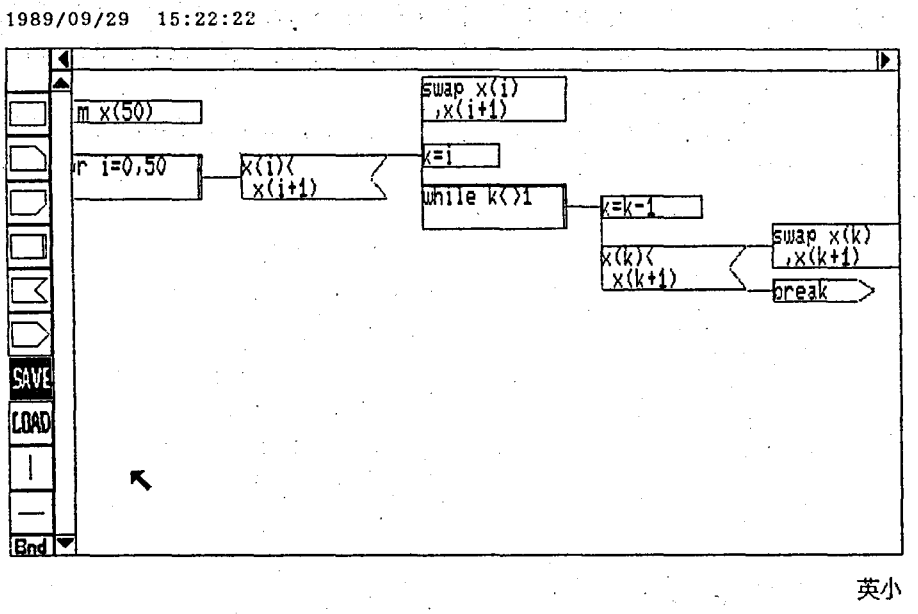


図7 PAD エディタの使用例

下左右にスクロールでき，可視画面の9倍までの作図が可能である．図6に示す PAD の簡単な例を，PAD エディタに展開したものが図7である．

前述のように，PAD エディタの出力対象はイメージファイルと中間言語ファイルの2つであり，SAVE アイコンをクリック後，システムが発生するメッセージに答える形で選択できる．「トランスレート」を指定すると，後述するような形で中間言語ファイルがカレントドライブに保存される．このときのファイル名拡張子は「.PCD」とした．

中間言語へのトランスレートのアルゴリズムを図8に示す．

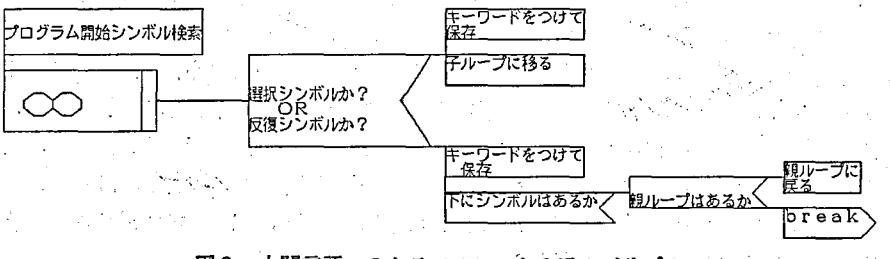


図8 中間言語へのトランスレートのアルゴリズム

PAD チャートにおける開始シンボルの検索は，作図されたパターンを解析することによって行われ，これによりツリーウォークの開始点が定まる．開始点が確定されると，そのシンボルの種類に応じたキーワードと内容，次のシンボルの番号を与え，以下このループを繰り返す．シンボルの種類が選択シンボルや反復シンボルのときは，そのシンボルの右側のシンボルに制御が移る．このときの右側のシンボルを子シンボル，元のものを親シンボルと称した．これらの制御ブロックが閉じるときには，ループの終わりを示すキーワード（“loop” “endif”）が与えられ，再び，親シンボルの下に制御が移ることになる．このような検索をしながら，個々の PAD シンボルに対し，一対一の中間言語が生成される．図9に図7のトランスレート結果を示す．

なお，PAD エディタからメニュープログラムへの移行は，END アイコンの選択により，自動的に行われる．

```
dim x(50)
do for i=0,50
if x(i) < x(i+1)
swap x(i),x(i+1)
k=i
do while k<>1
k=k-1
if x(k) < x(k+1)
swap x(k),x(k+1)
else
break
endif
loop
endif
loop
```

図9　中間言語への変換例

```
100 DIM X(50)
110 FOR I=0 TO 50
120   IF X(I) < X(I+1) THEN
130     SWAP X(I),X(I+1)
140     K=I
150     DO WHILE K<>1
160       K=K-1
170       IF X(K) < X(K+1) THEN
180         SWAP X(K),X(K+1)
190       ELSE
200         EXIT FOR
210       END IF
220     LOOP
230   END IF
240 NEXT I
```

図10　BASIC/98 への変換例

5. 高級言語へのトランスレート

今回開発した PAD トランスレータでは，前述のように BASIC（N88BASIC），構造化 BASIC（BASIC/98）およびC言語への変換を可能にした．これらの選択は前節同様メニュープログラムでなされ，それぞれの変換ソフトウェアに制御が移る．

第3節で述べたように，トランスレータにおける中間言語はかなり BASIC に近い構成となった．もちろん，PAD 自体が構造化手法を意識した表現形態であるため，中間言語は結果的には構造化 BASIC の性格に似たものとなっている．したがって，中間言語から BASIC/98への変換はほぼ一対一の対応で処理できる．すなわち，読み込んだ中間言語のリストは先ずキーワード（特定言語の予約語に相当するもの）とそれ以外の情報とに分解される．キーワードを判定するとターゲット言語である BASIC/98 の予約語への変換が行われ，その際に前段で分解された種々の有意情報が当該位置に埋め込まれることになる．図10に，変換された BASIC/98 のプログラムを示す．これからも明らかなように，プログラムの美化を考慮してのオートインデント機能も付加した．

N88BASIC のような非構造化 BASIC への展開は，ことに判定の制御構造に配慮が必要である．今回は条件節として中間言語での条件を論理否定した表現を用い，自動的に適当な位置にラベルを配して，疑似的に構造化を図って対処した．変換例は図11のとおりである．

```
100 DIM X(50)
110 FOR I=0 TO 50
120   IF NOT(X(I) < X(I+1)) THEN *EL1
130     SWAP X(I),X(I+1)
140     K=I
150     WHILE K<>1
160       K=K-1
170       IF NOT(X(K) < X(K+1)) THEN *EL2
180         SWAP X(K),X(K+1)
190         GOTO *ENIF2
200       *EL2
210         GOTO *LOUT1
220       *ENIF2
230     WEND
240     *LOUT1
250     GOTO *ENIF1
260   *EL1
270   *ENIF1
280 NEXT I
```

図11　N88BASIC への変換例

```
static float x[51];
static float i;
static float k;
void main()
{
  float pad_swp;
  for(i=0;i <= 50;i++){
    if (x[(int)i] < x[(int)i+1]) {
      pad_swp=x[(int)i];
      x[(int)i]=x[(int)i+1];
      x[(int)i+1]=pad_swp;
      k=i;
      while (k != 1){
        k=k-1;
        if (x[(int)k] < x[(int)k+1]){
          pad_swp=x[(int)k];
          x[(int)k]=x[(int)k+1];
          x[(int)k+1]=pad_swp;
        }else{
          break;
        }
      }
    }
  }
```

図12　C言語への変換例

高言語であるCへの変換を試みたものが図12である．BASIC 系の言語からの変換で最も重要な部分のひとつは変数の型宣言に関わる部分であり，上述のキーワードの判定の際に，構文解析して変数をピックアップする必要がある．しかるのち，上述のキーワードから予約語部への変換がなされるように設計した．また，この例にみられるように，中間言語で swap と記述された命令を3つの代入文に変換することも必要である．このようなキーワードは数は少ないが入門教育の折りにも必要となることから，本システムではこれらについてもサポートすることとした．

6. 結　　び

本報では，プログラミング言語間の変換の必要性を強調し，この変換が，中間言語を媒介とした汎用的なシステムとして行われるべきであることを提言した．そして，その第一段階として，PAD チャートから3種類の高級言語へのトランスレータの開発を行い，その結果について述べた．

構造化プログラミングの教育においては，PAD からアプローチすることが大きな効果をもたらすことはすでに報告されているが，さらに，今回開発したトランスレータを初心者の入門教育に用いたことにより，構造的な言語の教育にきわめてスムースに移行できることが

明らかになった．また，非構造的であるとされている従来からの BASIC に変換することも容易となり，あわせて，疑似構造化の啓蒙にも貢献できた．さらに低言語から高言語の変換の一例として，C言語のトランスレートも実現できた．

これらの変換において設計した中間言語が，このような多言語のトランスレートを必要とするシステムにとって，実に有用であることも実証された．しかし，さらに多くの言語間の変換を目指したり，あるいは，目的を教育用から実務用に発展させることを考えると，この中間言語の設計はさらに緻密に行うことが必要となろう．

PAD エディタについても，当初の目的を達することはできたが，スクロールの高速化やアイコンの適切な位置配置などについての改善も必要と思われる．これらの改良については第2段階としての言語間トランスレートの開発の中で達成していきたい．

おわりに，本システムの開発の一部は，平成元年度機械工学科の卒業研究のテーマとして与えたものであり，とくに，情報研究室学生岡本寛史君の功績は多大である．ここに深く感謝の意を表する．

参 考 文 献

(1) 古川康一他：プログラム変換，共立出版 (1987)
(2) 金敷準一：PAD 入門，サイエンス社 (1988)