６．Ｃ言語

Ｃ言語は BASIC に比べてメモリ操作やビット操作等を行えるようにしたものであり，コンピュータのハードウェアにかなり依存したプログラミングが可能であり，BASIC 言語と後で述べるアセンブラ言語の中間的な言語である。

卒論で作成する制御プログラムの中心は後節のアセンブラとＤＳＰである。Ｃプログラムはこれらに対して，画面の初期設定，ＤＳＰボードの設定，データテーブルのローディング（データをディスクからメモリにロードする），ＤＳＰプログラムのローディングおよび制御パラメータの入力等を行う。アセンブラプログラムはＣプログラムのサブプログラムとして実行される。したがって，Ｃ言語はその基本的なプログラミングをマスターすればよい。また，ＣおよびアセンブラプログラムはＤＯＳ上で動作する実行ファイルとするため，BASIC とは異なったいくつかの操作を必要とする。それでは，Ｃ言語の基本的プログラミング法と実行ファイル作成法を順次説明しよう。まず，ファイル管理の準備として，BASIC プログラムをサブディレクトリ「BFILE」中に保存したように，これから作成するＣ言語のソースファイル（Source File：Ｃ言語で記述したファイル）を保存するサブディレクトリ「CFILE」を各自のディスクに作成する。

```
A:\>MD B:\CFILE
```

また，これからの作業によって作られるファイルを格納するたのサブディレクトリ「COBJE」も作成する。

```
A:\>MD B:\COBJE
```

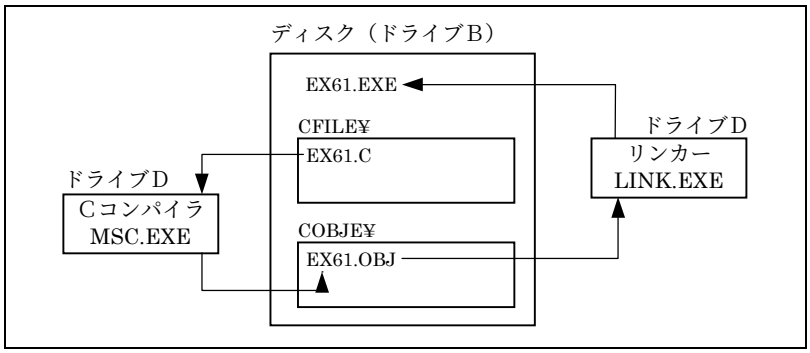
簡単な例として「画面に「POWER ELECTRONICS」という文字を表示する」プログラムを作成する。エディタを起動し，ファイル名を「EX61.C」としてドライブＢサブディレクトリ「CFILE」に作成する（MIFES 初期画面に上窓に「B:¥CFILE¥EX61.C」と入力する）。なお，Ｃソースファイルの拡張子は「C」と決まっている。右のプログラムを入力し，編集を終了する。ただし，Ｃ言語では予約変数を除いて全ての命令は小文字で入力する。１行目はコメントであり「/*」と「*/」で囲まれた範囲は全てコメントとみなされる。２行目は「main 関数」と呼ばれるものであり，Ｃプログラムはこの main 関数から実行を始める。関数の範囲は「{」と「}」で指定する。「printf」は文字列や変数を表示する関数であり，BASIC の「PRINT」に対応する。Ｃ言語では命令等を「関数」と呼ぶが，全ての関数には，その終了を表すために必ず「；（セミコロン）」を付ける決まりがある。なお，main 関数はひとかたまりの関数に対してつけた関数名であるので「；」は付けない。

```
/* EX61.C */
main()
{
printf(“POWER ELECTRONICS”);
}
```

Ｃ言語のソースプログラムは BASIC のようにそのまま実行することはできず，「コン

パイル（Compile）」および「リンク（Link）」という作業を行い，実行可能ファイルを作成しなければならない。BASIC が１行１行機械語レベルに翻訳して実行するのに対して，Ｃ言語では一度に機械語に翻訳して実行を行う。このような実行方法を一般にコンパイルといっている。Ｃ言語のコンパイルを行うものが「MSC.EXE（この他にも付随する幾つかのファイルがある）」ファイルであり，リンクを行うものが「LINK.EXE」である。

本システムではこれらの作業をドライブ「D」のＲＡＭディスク内で行う（必要なコンパイラ，リンカーなどの実行プログラムはあらかじめコピーしておくが，本システムではコンピュータを立ちあげた時にインストールするよう設定している）ので，カレントドライブを「D」として次の操作を行う（下図参照）。



Ｃコンパイラを次の手順で行う。

```
D:\>MSC
Microsoft (R) C Compiler  Version 4.00
Copyright (C) Microsoft Corp 1984, 1985, 1986. All rights reserved.

Source filename[.C]: B:\CFILE\EX61
Object filename[EX61.OBJ]: B:\COBJE\
Source listing [NUL.LST]:
Object listing [NUL.COD]:

D:\>
```

Ｃ言語における「コンパイラ（Compiler）」とは「編集者」，BASIC 言語の「インタープリタ（Interpreter）」とは「解釈，解説者，通訳」の意味である。いずれにしても最終的に機械語に翻訳されるわけであるが、実際に同じ様な処理を行うプログラムを実行させた場合には、その実行速度（演算を行う時間）が異なる。一般に、人間にとって分かりやすい言語ほど実行速度は遅くなると考えてよい。特に制御用のプログラムを組むためにはこの実行時間が重要になり、できるだけ実行時間の速いプログラムを作成する必要がある。そのためには「アセンブラ言語」を使用しなければならない。「アセンブラ言語」は「機械語」に直接対応した言語であり、ＣＰＵそのものの動作を表した言語である。したがって、コンピュータのハードウエアに直接関係するので、ハードウエアの構造自体を知ってないとどうしようもない。「ＢＡＳＩＣ」や「ＦＯＲＴＲＡＮ」などの高級言語になるほどハードウエアの知識は必要でなくなってくる。言い換えれば、このような言語は制御用としては向いていないということである。「Ｃ言語」は両者の中間的な言語であるが、これでも制御用としては実行時間が遅すぎる。

「Source filename」はＣプログラムのソースファイル名をディレクトリを含めて指定する。なお，拡張子「.C」は省略可能である。
「Object filename」はソースファイル名と同一名で作成するので，ここでは保存するディレクトリ名を指定する。指定せずにリターンキーを押すと現在のカレントに作成される。なお，オブジェクトファイルの拡張子は「.OBJ」と決まっている。
「Source listing」および「Object listing」のリスティングファイルは通常作成しないので，そのままリターンキーを押す。なお，「NUL（ヌル）」は何も作成しないことを表す。

これでコンパイル作業が始まり，指定ディレクトリに「EX61.OBJ」を作成し，プロンプトを表示して入力待ちとなる（目的のファイルが作成されたか確認する）。ソースプログラムにエラーがある場合には「エラーメッセージ（Error Message）」を表示するので，エディタで修正し再びコンパイルを行う。オブジェクトファイルはまだ，ＣＰＵが直接解読できるファイルではなく，「LINK」に対する種々の情報や他のオブジェクトファイルとの結合のための情報をもつファイルである。「MSC」では，全てのパラメータを指定して起動することもでき，次のように入力する。

```
D:\>MSC B:\CFILE\EX61,B:\COBJE\;
```

パラメータを「，」で区切って指定する。最後の「；」はそれ以降のパラメータを省略（デフォルト）することを表す。

次に，リンカーを次の手順で起動する。

```
D:\>LINK
Microsoft (R) Overlay Linker  Version 3.51
Copyright (C) Microsoft Corp 1983, 1984, 1985, 1986. All rights reserved.

Object Modules [.OBJ]: B:\COBJE\EX61
Run File [D:EX61.OBJ]: B:
List File [NUL.MAP]:
Libraries [.LIB]:

D:\>
```

「Object Modules」はオブジェクトファイル名をディレクトリを含めて指定する。なお，拡張子「.OBJ」は省略可能である。「Modules」とは「加群，加法群」の意味であり，リンカーによって幾つかのオブジェクトファイルを結合する時は，ここで指定する。
「Run File」はカレントドライブ上にオブジェクトファイル名（複数の時は最初のファイル名）と同一名で作成するので，ここでは保存するディレクトリ名を指定する。なお，Run ファイルの拡張子は「.EXE」であり，これが実行可能ファイルとなる。
「List File」は通常作成せず，また「Libraries」のライブラリは用いないので，そのままリターンキーを押す。

以上で，リンク作業が始まりエラーがなければ（この場合は１つのオブジェクトファイルから実行ファイルを作成するのでエラーは発生しなしが，複数をリンクしてエラーを生じた場合にはそれぞれのソースファイルを見直す必要がある）プロンプトを表示する。「MSC」の場合と同様に「LINK」でも起動時に全てのパラメータ設定ができ，次のように入力する。

```
D:\>LINK B:\COBJE\EX61,B:;
```

ドライブＢの親ディレクトリに「EX61.EXE」があることを確認し，実行する。

カレントドライブＡの時	カレントドライブＢの時
A:\>B:EX61	B:\>EX61

「POWER ELECTRONICS」を１行に表示する。

このように文字を表示させるための実行ファイルを作成するために，かなりの操作を行わなければならない（ソースプログラムは簡単であるが）。こうした手順はソースプログラムを変更すれば毎回必要となり面倒であるので, 実際はこれら一連の操作を順次行う「バッチファイル（Batch File）」を作成しているが，これについては後節で簡単に説明する。

以下，Ｃ言語の命令，関数について，現在使用しているものについて説明するが，詳細および他の関数等はマニュアルを参照されたい。

先の「EX61.C」プログラムを右のように変更し実行すると，同じ表示となる。このプログラムでは「hyouji」という関数を定義し，それを main 関数から呼び出すものである。このように関数を自分で作成し main 関数とは別に作成できる。その関数は main 関数と同様に「{」と「}」で囲まれる。また, main 関数は必ずしもプログラムの最初に記述する必要はない。ただし，自分で作成する関数名は予約語以外とする。

```
/* EX62.C */
hyouji()
{
printf(“POWER ELECTRONICS”);
}
main()
{
hyouji();
}
```

次はプログラム中を

「printf(“POWER\nELCTRONICS”);」

に変更して実行すると，２つの単語が２行に分割される。すなわち「\n」は改行を意味している。「\」はコントロールキャラクタと呼ばれ，それに続く文字によって種々の機能がある。

［変数の入出力］

「EX63.C」はキーボードから２つの値を入力し，その和を表示するものである。プログラム中で使用する変数は全て定義しなければならない。定義の意味は次のとおりであるが，ビット数は使用するＣＰＵに依存する。

```
/* EX63.C */
main()
{
int a,b,c;
printf("a=");scanf("%d",&a);
printf("b=");scanf("%d",&b);
c=a+b;
printf("c=%d\n",c);
}
```

int：１６ビット正負整数表現
char：８ビット正負整数表現
unsigned int：１６ビット正の整数表現
unsigned char：８ビット正の整数表現
float：実数表現（単精度）
double：実数表現（倍精度）

これらは演算における変数形式の変換にも使用される。「scanf」はキーボードからの入力を行う関数であり，BASIC の「INPUT」に対応する。変数名には「&」を付け，そのアドレスを指示する。「“%d”」は入力形式を定義するものであり，これは「printf」でも使用できる。形式内容は次のとおりである。

%d：１０進数の整数表現
%f：１０進数の実数表現

［外部入出力］

コンピュータの外部装置とのデータのやり取りを行う関数である。

inp(ｱﾄﾞﾚｽ)：指定したｱﾄﾞﾚｽからﾃﾞｰﾀ（８ﾋﾞｯﾄ）を取り込む
outp(ｱﾄﾞﾚｽ,ﾃﾞｰﾀ)：指定したｱﾄﾞﾚｽへﾃﾞｰﾀ（８ﾋﾞｯﾄ）を出力する

Ｃ言語では数値は１０進数を示すが，１６進数表現で与える場合には数値の前に「0x（ゼロとｘ）」を付ける。なお，上記関数を使用するには環境変数を定義した「ヘッダーファイル」と呼ばれる「インクルードファイル」でプログラムの最初に次のように定義する。

「#include <io.h>」

［ディスクからのデータ読み込み］

「EX64.C」は４章で作成した「EX43.DAT」（１から１０までの２乗値）からそれらのデータを読み込み表示するものである。

```
/* EX64.C */
#include <stdio.h>
main()
{
FILE *fp;
int i,dd;
fp=fopen("B:\\BFILE\\EX43.DAT","r");
for(i=1;i<=10;i++)
{
 fscanf(fp,"%d",&dd);
 printf("dd=%d\n",dd);
}
fclose(fp);
}
```

「stdio.h」のインクルードファイルと「FILE *fp」（ファイルポインタ）は「fopen」関数に関するものである。「fopen」関数はディスクファイルをオープンするものであり，オープンするファイル名（「\」記号はコントロールキャラクタとみなされるので，ディレクトリの区切り記号とする時には「\\」のように記述する）を指定し，読み込みモード「“r”」とする。「fscanf」関数はファイルから形式に従ってデータを読み込む関数である。オープンされたファイルは「fclose」関数によって閉じられる。また，「for」文は BASIC のそれに対応し，「（ ）」内で示された条件に従って続く「｛｝」内の関数を繰り返すものである。例では，変数「i」を１から１０まで１づつ増加して繰り返すことを行う。「i++」は「i=i+1」と同じ事であるが，厳密には前者は内容を「＋１」する，すなわちメモリ内容をそのまま「＋１」（インクリメント（increment）という）するのに対し，後者は「１」を記憶した別メモリとの加算を行う。「－１」する時は「i--」である。

［画面制御］

Ｃ言語では BASIC のようなテキスト画面やグラフィック画面を直接操作する関数がな

いため，これらの関数を定義した次のインクルードファイルを作成している。

「crt.h」　　：テキスト画面用関数
　　locate,color,clstxt,home,beep,console,cdisp,cerase など
「glib.c」　　：グラフィック画面用関数
　　ginit,screen,view,window,cls,pset,line,circle,paint など

なお，これらの関数についてはファイル内をエディタで見て，引き数などを確認されたい。ここでは，実際に制御プログラムで使用している「グラフィックで線を描く」プログラムを示す。右に示す「EX65.C」は左上に横１００ドットの白線を描くプログラムである。これをコンパイルするにはあらかじめ各自のディスク（ドライブBのサブディレクトリ「CFILE」内）に上記のインクルードファイルをコピーしておく必要がある。「ginit()」はグラフィック描画を行うための初期設定関数，「screen()」はグラフィック画面の設定，「cls()」は画面消去を行う。また，「line(X1,Y1,X2,Y2,C,T)」は直線描画であり，スクリーン上の(X1,Y1)ドットから(X2,Y2)ドット（実数で定義）まで直線を引き，「C」は色（整数），「T」は線のタイプ（整数）を指定する。

```
/* EX65.C */
#include "b:\cfile\crt.h"
#include "b:\cfile\glib.c"
main()
{
ginit(); screen(3,0,0,1); cls(3);
line(0.0,0.0,100.0,0.0,7,0);
}
```

［ディスクデータのメモリロード］

「EX66.C」はディスク内に作成したデータをメモリにロードするプログラムであり，その例として，４章の BASIC で行ったようなＶＲＡＭへ直接書き込みするものである。この方法は後述するＤＳＰのプログラムロードやデータテーブルのローディングに使用している。なお，これを実行する際には，あらかじめドライブＢのサブディレクトリ「BFILE」内に「EX66.OJ」のデータファイル（アスキー形式）を作成しておく。このようなデータ形式のファイルは BASIC の「EX47.BAS」を参考にして作成できるであろう（この程度のデータ数であればエディタで直接作成してもよいが，データ数が膨大になると無理である）。

```
/* EX66.C */
#include <process.h>
#include "b:\cfile\crt.h"
#include "b:\cfile\glib.c"
main()
{
ginit(); screen(3,0,0,1); cls(3);
spawnlp(P_WAIT,"d:ld25","d:ld25",
      "b:\\bfile\\ex66.oj","-m","a800",NULL);
}
```

EX66.OJ

```
0000FFFFFFFFFFFFFFFFFFFFFFFF
```

このプログラムはＤＳＰプログラムロード用の実行プログラム「LD25.EXE」を利用し，これを「EX66.EXE」の「子プロセス（Sub Process）」として実行を行うものである。「子プロセス」とは，あるプログラムを実行中に，それとは無関係のプログラムを実行し，終了後に再び元のプログラムに実行を戻すことをいう。「spawnlp」は子プロセスを行う関数であり，その形式は次のようである。

spawnlp(P_WAIT,"ﾌﾟﾛｾｽ名","ﾌﾟﾛｾｽ名","ﾃﾞｰﾀﾌｧｲﾙ名","-m","ｾｸﾞﾒﾝﾄ",NULL)

プロセス名は「LD25.EXE」であり，ドライブ名まで指定する（拡張子は EXE であるので省略）。データファイル名はドライブおよびディレクトリまで指定し，そのファイルの拡張子は「.OJ」とする。これは，「LD25.EXE」がＤＳＰプログラムローディングのための実行ファイルであるためである（実際，後述するようにＤＳＰのプログラムは最終的に「.OJ」拡張子となる）。以降は「LD25.EXE」が必要とするパラメータである。ユーザーが設定するのは「セグメント」である。なお，「spawnlp」関数の詳細はマニュアルを参照されたい。

「LD25.EXE」は指定されたセグメントのメモリに目的のデータファイルをロードするが，その時の「オフセット」はデータファイルの先頭４文字（例では「0000」，１６進数とみなす）となる。それに続くデータ（１６進数）を４桁，すなわち２バイト（１ワード）単位で順次メモリへロードする。例では，データがロードされる先頭の物理アドレスは「a8000H」となり，これは３章で説明したグラフィックＶＲＡＭの GVRAM0（青）の先頭アドレスである。ロードするデータは全て「FFFFH」であり，これが６つあるので，すべてのビットに「１」をロードし，したがって，画面の１２桁すなわち９６ドットの青線が左上に描画される。

なお, 例のプログラムでは左につめて各命令を記述したが, ソースプログラム上では「ブランク」は無視されるので，各自，プログラムを見易くするために適当なブランクを入れて，各行の配置を工夫しよう。