# 電子計算機 **1E**

1E  2003.5.30

## 本日の授業のテーマ

本日の事業のテーマは、以下のとおりです。

(1)　基本演算のプログラミング

本日の授業のゴールは、以下のとおり。

- エディターを使って、プログラムがソースプログラムが書ける。
- コンパイルと実行ができる。
- プログラムの内容がわかる。
    - 変数の宣言の意味
    - キーボード入力の `read(5,*)` の使い方
    - ディスプレイ出力 `write(6,*)` の使い方
    - `stop end` 文などの FORTRAN の約束が分かる。

# 1 プログラム作成・実行手順

課題のプログラムを作成する場合の、手順を図 1 に示します。ただし、これは一例にすぎませんので、この通りにしなくても、プログラムが実行できればよいです。

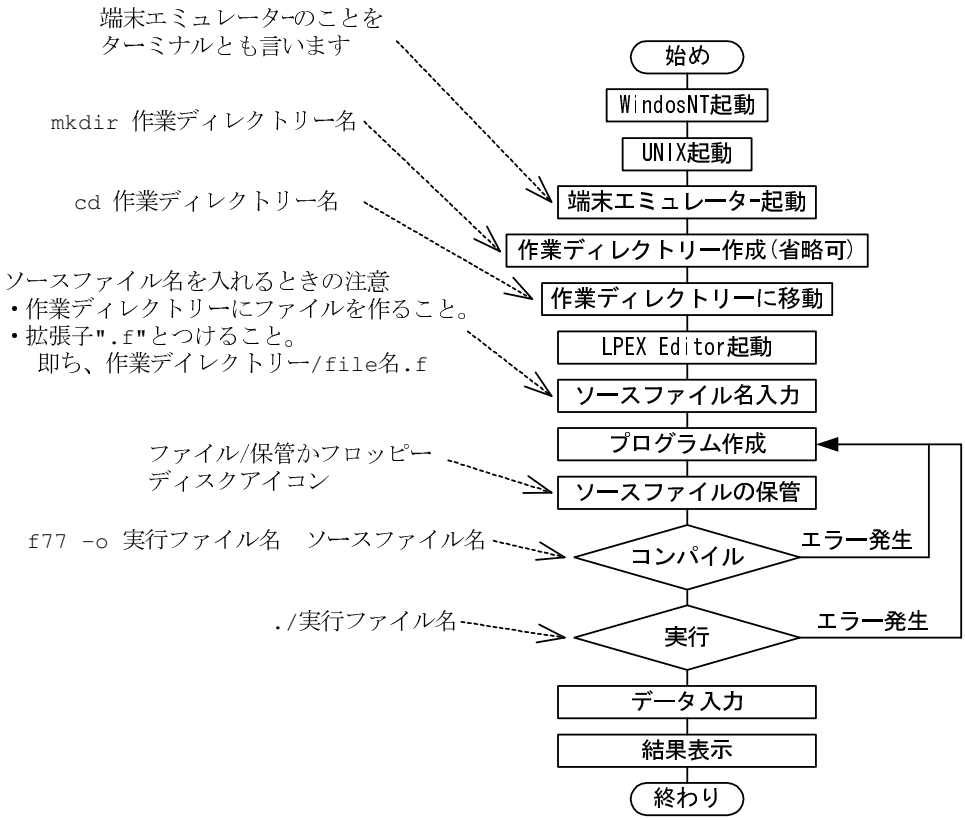


図 1 プログラム作成・実行の流れ図

# 2 各ステップの説明

## 2.1 WindowsNT の起動　(利用の手引き **p. 1～p. 8)**

説明するまでも無いでしょう。パスワードを忘れないでください。忘れた場合は、情報処理センターの担当技官に問い合わせてください。

## 2.2 UNIX の起動　(利用の手引き **p. 12～p. 13)**

これも利用の手引きに書いてあります。パスワードに関しては、WindosNT と同じです。

本校では、FORTRAN の学習は UNIX 環境で行うことになっています。学習に使う計算サーバー(namahage)1 台を、40 人を超える人数で同時に使うことになります。1 台のコンピューターを 40 人程度で使いますが、人数分の UNIX 端末を用意するわけにもいきませんし、交互に 1 人ずつ使うえば学習効率が下がります。そこで、WindosNT を利用します。

WindosNT 上で X サーバー「Exceed」というソフトウェアーを稼動させて、そこから UNIX マシーンを操作します。これらのコンピューター間で通信を行うことによ

り、WindowsNT マシーンから UNIX が操作できるようになっています。

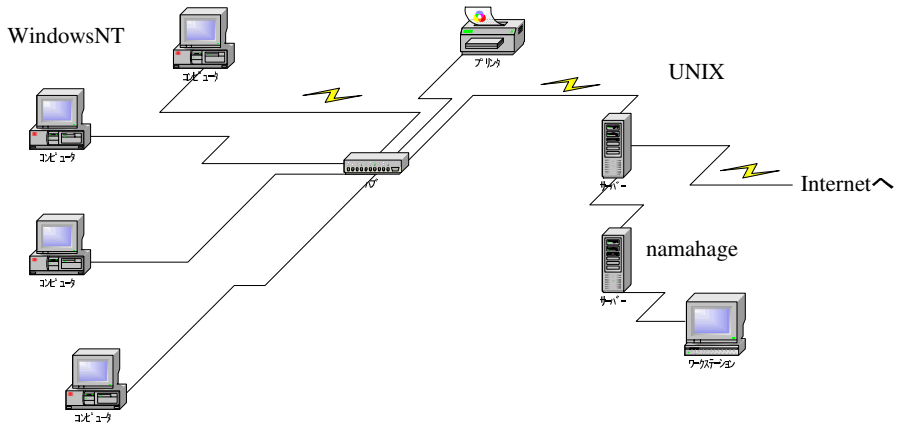


図 1 ネットワークのモデル

Windows で FORTRAN を勉強すればよいではないかという議論もありますが、以下の理由をもって、UNIX で勉強するほうが教育的と私は考えます。

- 各 Windows マシーンに FORTRAN コンパイラーを導入すると、非常にコストがかかります。
- Windows 用のフリーの FORTRAN コンパイラー、例えば g77 などを実行させる場合、UNIX と同じコマンドが使えないと利用できません。ここでも、UNIX とほとんど同じコマンドを覚える必要があります。
- FORTRAN は科学技術計算用の言語です。将来、皆さんが大規模な科学技術計算を行う場合、多分 UNIX を使うことになります。世界一速度の速いコンピューター「地球シミュレーター」も UNIX です。将来役に立つ人もいるはずです。

とはいっても、Windows で FORTRAN を勉強するのも良いと思います。私のノートパソコンには、Windows 上で擬似的な UNIX 環境を構築する Cygwin というソフトウェアーを導入しています。これを使って、FORTRAN や C 言語の学習ができます。なんと言っても、パソコンで動作するので手軽で重宝しています。

他にも、Windows で動作するフリーの FORTRAN コンパイラーがあるかもしれませんが、私は良く知りません。興味がある人は、調べてください。

**2.3** 端末エミュレーター起動　**(利用の手引き p. 13)**

UNIX の場合、ターミナルからコマンドを打ち込んで、コンピューターに指令を出します。この方法を Character User Interface (CUI)と言います。一方、Machintosh や Windows のように、アイコンやマウスを使う方法を Graphical User Interface (GUI)と言います。UNIX も GUI が使えるようになっていますが、CUI を使うほうが多いです。

Windows も実を言うと、CUI が可能です。ためしに、コマンドプロンプト(スタート/プログラム/アクセサリ/コマンドプロンプト)を立ち上げてください。ほとんど、UNIX と同じコマンドが使えます。そればかりでなく、直接実行ファイルを指定してアプリケーションを立ち上げることもできます。

GUI が便利であることに変わりがありませんが、CUI を覚えておくと、いろいろ役

に立つ場合があります。どうしても、CUI でコマンドを入れなくてはならないとき、大体 UNIX のコマンドに似ています。

**2.4** 作業ディレクトリー作成　**(**利用の手引き **p. 15**、プリント **UNIX** コマンド **p.4)**

これから、プログラムを作成するときの作業をする場所を確保します。ここに、ソースファイル[1]や実行ファイル[2]などを保存します。皆さんが、ワープロで文章を作成するときにフォルダーを作成するのと同じです。そうすることにより、データを整理できます。

コマンドは、

```
mkdir ディレクトリー名
```

です。確認は、

```
ls
```

というコマンドでできます。

**2.5** 作業ディレクトリーに移動　**(**利用の手引き **p. 15**、プリント **UNIX** コマンド **p.3)**

先に作った作業ディレクトリー[3](ワーキングディレクトリー、カレントディレクトリーということもある)に移動します。コマンドは、以下の通り。

```
cd ディレクトリー名
```

いま、自分が居るデイレクトリーは、以下のコマンドで分かります。

```
pwd
```

**2.6 LPEX Editor** 起動　**(**利用の手引き **p. 16**～**17)**

ソースプログラムは、文字だけのテキストファイルでなくてはなりません。テキストファイルを編集するプログラムを、エディターと呼びます。本校の namahage には、プログラムを編集するエディターとして、LPEX Editor が使用できます。これは、[Tab] キーを押すと、自動的にカーソルが 7 桁目に移動して、そこからプログラムの本文を書くことができ便利です。

エディターは、どのような OS にもインストールされています。例えば、Windows のワードパッドがそれです。昔の Machintosh は、SimpleText だったかなー

テキストファイルは、便利で、どのようなコンピューターでも、そのファイルを読むことができます。MS Word のファイルは、windows 専用です。UNIX で読むことはできませんし、Machintosh でもきちんと読めないでしょう。

**2.7** ソースファイル名入力 **(**利用の手引き **p. 16)**

LPEX Editor を起動すると、これから作成するテキストファイル名を聞いてきます。ここで、ソースプログラムのファイル名を入力します。ソースプログラムのファイ

---

[1] FORTRAN のプログラムが書かれたテキストファイル。

[2] ソースファイルをコンパイルしてできたマシン語のファイル。ファイル名の後にアスタリスク'*'がつく

[3] ワーキングディレクトリー、カレントディレクトリーということもある。

ル名は、*****.f でなくてはなりません。要するに、拡張子の'.f'が必要ということです。

LPEX Editor が立ち上がると、それはホームディレクトリー[4]で立ち上がります。したがって、先ほど作成した作業ディレクトリーにソースファイルを作る場合、そのパスを書く必要があります。以下の通りです。

作業ディレクトリー/*****.f

ディレクトリーの区切りは、/です。Windows の場合は、\です。

**2.8** プログラム作成　**(利用の手引き p. 16、教科書、課題プリント)**

課題に沿ったプログラムを作成して、その内容を書いていきます。FORTRAN の場合、本文は 7 桁目から書きます。[Tab]キーを押せば、カーソルが自動的に 7 桁目に移動します。

**2.9** ソースファイルの保管　**(利用の手引き p. 17)**

プログラムの作成が完了したら、保管[5]します。保管方法は、

①「ファイル」メニューの「保管」を選択
②ツールボックスのフロッピーディスクのアイコンをクリック

のいずれかです。保管後、ls コマンドか、ファイルマネージャーで見るとファイルが作成されていることが分かります。

**2.10** コンパイル　**(利用の手引き p. 18〜19)**

コンパイルの方法は、ターミナルで、

f77 -o　実行ファイル名　ソースファイル名

とコマンドを入力します。エラーが無ければ、実行ファイルができるはずです。実行ファイルは、ls コマンドで確認できます。指定した実行ファイル名の後に、アスタリスク*がついたファイルがあるはずです。エラーがあるときは、ソースプログラムを編集しなおしてください。

ソースプログラムは、人間が理解しやすい FORTRAN 言語でかかれています。これは、コンピューター、正確には CPU は理解できません。そこで、CPU が分かる言語、正確には CPU そのものの動作を記述する言語であるマシン語[6]に翻訳する必要があります。この翻訳の作業をコンパイルと言います。

実際、f77 コマンドは、コンパイル以外にもいろいろな作業を実施します(図 2)。これは、覚える必要はありませんが、ソースファイルをコンパイルしてマシン語の実行ファイルが出来上がることは、理解してください。

---

[4] プリント UNIX コマンドの P2 を見よ。
[5] 保管のことを、保存、あるいはセーブ(save)ということもあります。
[6] マシン語は、0 と 1 の数字の羅列です。3 年生で、学習します。

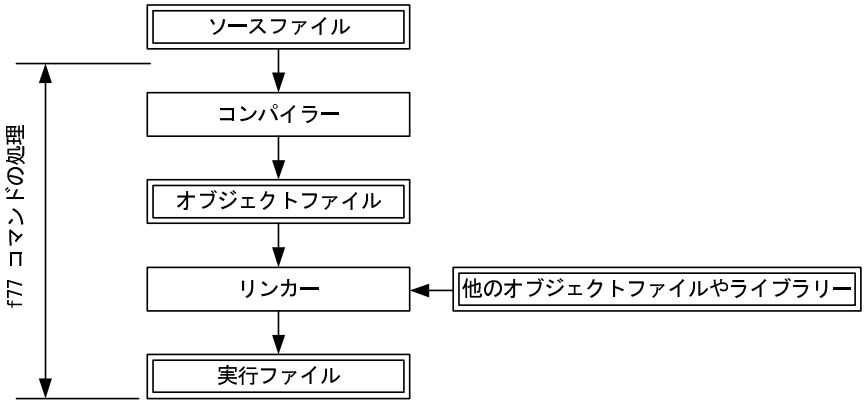


図 2 実行ファイル作成の流れ

**2.11** 実行　**(利用の手引き p. 20～21)**

実行ファイルの名前をターミナルから入力すると、プログラムは実行されます。`ls` コマンドでファイル名をディスプレイに出したときに表示されるファイル名の後にあるアスタリスク"*"は、不要です。

**2.12** データ入力　**(利用の手引き p. 20)**

キーボードから、データを入れます。

**2.13** 結果表示　**(利用の手引き p. 20)**

計算結果が、ディスプレイに表示されるはずです。

## 3 プログラムの説明

最初の課題のプログラムの説明をします。もちろん、正解はこれに限らず、無数にあります。与えられた課題の計算ができれば、良いです。回答のプログラムについて説明します。

**3.1** 注釈文(コメント文)　**(教科書　p.7, p.9)**

これが無くても動きますが、プログラムの内容をわかりやすくするために書きます。第 1 桁目が、*, c ならば、その行は注釈文です。

```
c ======================================================
c =                                                    =
c =                                                    =
c =                  written by Masashi Yamamoto       =
c =                   e-mail yamamot@akita-nct.jp      =
c =                          2003.05.20                =
c ======================================================
```

**3.2** プログラム文　**(教科書　p.7)**

この文がなくても、プログラムは実行できます。プログラムに名前を付けています。

```
program arealength
```

**3.3** 変数宣言　**(教科書　p.8, p.13～14)**

プログラムの主な動作はデータの加工です。その場合、データを入れる入れ物みたいなものが必要です。その入れ物は、型と名前(変数名)を宣言してから、使わなくて名なりません。なかなか理解できないと思いますが、4.1 節にもう少し詳しく説明しています。

変数名は、英文字で始まる 6 文字以内の英数字と決まっていますので、ギリシャ文字のπが使えないので、代わりに `pi` としています。

このプログラムでは、変数は実数を使いますので、入れ物の型として、`real` と宣言しています。

```
real pi, x
real S1, S2, S3
real L1, L2, L3
```

**3.4** 代入　**(教科書　p.8)**

変数宣言で用意した pi に、`3.14159265` を代入しています。

```
pi = 3.14159265
```

**3.5** データの読み込み　**(教科書　p.8, p.12)**

`read` 文でデータを読み込みます。括弧内の最初の数字は、読み込む場所を指定します。5 番は、キーボードと約束されています。ほかの媒体からデータを読み込みたい場合、例えばハードディスク、5 とは異なる番号を使います。ハードディスクを指定する場合の文法は、教科書の p.207 以降に書いています。

括弧の 2 番目、ここでは`*`、これは書式の文番号を記入するものです。`*`と記入した場合、デフォルトで適当にデータを読み込みます。初心者は、とりあえず、ここは、`*`を使います。プログラマの指定通りに読み込む方法は、教科書の p.329 以降に書いてあります。

括弧に引き続き、読み込んだデータを保存する場所、すなわち変数名を書きます。ここで、書かれた変数名の順序に従い、読み込まれたデータが格納されます。ここでは、データが 1 個しかないので、順序はありません。

```
read(5,*)x
```

**3.6** 基本演算　**(教科書　p.8, p.19)**

変数を用いた演算を実施し、その結果を、変数に代入しています。当然、この変数の中身は、数値データです。数学のように乗算(かけ算)の記号`*`を省略できません。変数の名前の区切りがわからなくなります。

```
S1 = x**2
S2 = pi*(x/2)**2
S3 = S1-s2

L1 = 4*x
```

```
L2 = pi*x
L3 = L1 - L2
```

### 3.7 データの書き出し　(教科書　p.8, p.13)

write 文でデータを書き出します。括弧内の最初の数字は、書き出す場所を指定します。6 番は、ディスプレイと約束されています。ほかの媒体にデータを書き出したい場合、例えばハードディスク、6 とは異なる番号を使います。ハードディスクを指定する場合の文法は、教科書の p.207 以降に書いています。

括弧の 2 番目、ここでは`*`、これは書式の文番号を記入するものです。`*`と記入した場合、デフォルトで適当にデータを書きます。初心者は、とりあえず、ここは、`*`を使います。プログラマの指定通りに書き出す方法は、教科書の p.329 以降に書いてあります。

括弧に引き続き、書き出したいデータ、すなわち変数名を書きます。ここで、書かれた変数の順序に従い、書き出されます。

```
write(6,*)S1, S2, S3
write(6,*)L1, L2, L3
```

### 3.8 終了　(教科書　p.9)

stop でプログラムの実行を止めます。end でプログラムの終わりを示します。今のところ、おまじないと思って書いてください。すくなくとも、コンパイルエラーが出ないという御利益はあります。

```
stop
end
```

## 4 その他雑多なこと

この辺は少し難しいので、ひまなときにでも読んでおいてください。

### 4.1 変数の宣言が必要な理由　(教科書　p.14〜15)

初心者がプログラミング言語を学び始めると、変数の宣言に戸惑います。数学を勉強するとき、変数に宣言など使ったことがないからです。また、宣言がなくても、アルゴリズムに不都合がないからです。変数の宣言が無くても、問題がないように思ってしまいます。

変数の宣言がなくても、アルゴリズム上、問題はありません。しかし、コンパイラーにとっては、大きな問題です。コンパイラーがソースプログラムをマシン語に翻訳する場合、その変数を表現するための領域の大きさ(バイト数)と表現方法を決める必要があります。この辺については、2 年生で学習します。

できるだけ大きな数字が表現できる型、例えば FORTRAN の 4 倍精度実数型に統一する方法もあります。しかし、そうすると以下のような不都合が生じます。

- 整数どうしの加算や減算、乗算が整数にならない場合が生じる。桁落ちの問題。
- 大きなメモリー領域が必要になり、資源の無駄になる場合が多い。

これらを、コンパイラー言語では、型宣言によって避けます。効率的なマシン語を作成するために、変数の宣言は絶対に必要となるのです。

参考のために、表 1 に FORTRAN の代表的な宣言を示します。1 バイトで、1 と 0 が 8 個です。したがって、INTEGER と宣言した場合、コンピュータ内部では、32 個の 1 と 0 を使って整数を表します。コンピューター内部では、命令もデータもすべて 1 と 0 で表現しています。

一方、インタープリンター方式の言語の場合、型の宣言が無い場合があります。例えば、BASIC や Perl のような場合[7]、型宣言は不要です。あらかじめ、データ領域を確保する必要が無く、必要になったら、適当に領域を確保しているためだと思います。詳細は、良くわかりません。このような言語では効率的なマシン語の作成は不可能なので、処理速度はコンパイラー言語に比べて格段に劣ります。ただ、手軽に実行ができるため、近頃はやっています。

表 1 FORTRAN の場合の変数の型表現。

| 型 | 宣言 | バイト数 | 表現方式 |
|---|---|---|---|
| 整数 | `INTEGER` | 4 | 固定小数点 |
| 実数 | `REAL` | 4 | 浮動小数点 |
| 倍精度実数 | `REAL*8` | 8 | 浮動小数点 |
| 4 倍精度実数 | `REAL*16` | 16 | 浮動小数点 |

**4.2** 実行ファイル名を指定しての実行について

Machintosh や Windows の GUI 環境に慣れた人にとって、実行ファイル名を指定して、プログラムを起動することに戸惑を感じるかもしれません。しかし、windows でも、UNIX とそっくりのことができます。

時間があるときに、Windows で以下を試してください。

① スタートのプログラムのアクセサリのコマンドプロンプトを起動
② `cd Program Files` と打ち込んで、ディレクトリーの移動
③ `cd Microsoft Office`
④ `cd Office`
⑤ `WINWORD.EXE` と打ち込む。実は、これで `UNIX` と同じように実行ファイル名で `Word` を起動できます。

[7] しかし、近頃、型宣言を行うようにもなってきています。昔は、不要でした。