目 次

第 1 章 電卓としての利用 ............................... 5
  1.1 キー操作と用語の復習 .............................. 5
  1.2 Cfep/Asir の起動法と電卓的な使い方 ............... 6
  1.3 エラーメッセージ ................................... 11

第 2 章 変数とプログラム ................................ 15
  2.1 変数 .............................................. 15
  2.2 くりかえし ........................................ 18
  2.3 実行の中止 ........................................ 20
  2.4 エンジン再起動 .................................... 21
  2.5 ヘルブの利用 ...................................... 22

第 3 章 グラフィック ..................................... 25
  3.1 ライブラリの読み込み ............................... 25
  3.2 線を引く関数 ....................................... 25
  3.3 円を描く関数を作ってみよう ......................... 28

第 4 章 For 文による数列の計算 ........................ 31
  4.1 超入門. 第 2 の関門: 渐化式でできる数列の計算 ....... 31
  4.2 円を描く数列 ...................................... 32

第 5 章 cfep 上級編 ................................. 35
  5.1 TeX によるタイプセット (実験的) ................... 35
  5.2 選択範囲のみの実行 ................................ 35
  5.3 エンジンを起動しない ......................... 36
  5.4 OpenGL インタプリタ .............................. 37
第1章 電卓としての利用

神戸大学の教育用計算機環境が MacOS X に変更されるのに伴い、筆者が教材として利用していた Windows で動作する 10 進 Basic が利用できなくなつた。Cfep/asir はその代替として、2006 年初頭から開発を進めているシステムである。10 進 Basic の優れている点の一つは、丁寧な入門解説が付属していることである。“Cfep/asir 超入門”はこの解説にすこしでも近付こうと努力してみた。Asir の入門テキストに “Asir ドリル”があるが、この超入門では “Asir ドリル”の一章およびその先の入門的内容を丁寧に（少々ぐどく）説明した。

この節では MacOS X での cfep/asir の起動法、電卓風、Basic 風の使い方を説明する。ファイルの保存等 MacOS X の共通の操作方法にはほとんどふれていないが、cfep/asir は MacOS X 標準のファイルの保存等を用いているので、このような部分では他のソフトウェアと利用方法は同一である。初心者の人は適当な本やガイドを参照されたい。

1.1 キー操作と用語の復習

キーボード、マウスの操作の用語。

1. **Command** キーや **ALT** キーや **SHIFT** キーや **CTRL** キーは他のキーと一緒に押すことで始めて機能するキーである。これらだけを単独に押してもなんにも起きない。以後 **SHIFT** キーをおしながら他のキーを押す操作を **SHIFT** と書くことにする。command キー、alt キー、ctrl キーについても同様である。

2. **SHIFT + a** すると大文字の A を入力できる。

3. **BS** とか **DEL** と書いてあるキー押すと一文字前を消去できる。

4. 日本語キーボードの場合 ング（バックスラッシュ）は **ALT + 仮** で入力できる。

5. **SPACE** キーは空白を入力するキーである。計算機の内部では文字は数字に変換されて格納および処理される。文字に対応する数字を文字コードと呼ぶ。文字コードにはいろいろな種類のものがあるが、一番基礎的なのはアスキーコード系であり、アルファベットや数字、キーボードに現れる記号などをカバーしている。漢字はアスキーコード系では表現できない。**A** のアスキーコードは 65 番である。以下 **B** が 66、**C** が 67、と続く。空白のアスキーコードは 32 番である。日本語入力の状態で入力される空白は “全角空白”と呼ばれており、アスキーコード 32 番の空白 (半角空白) とは別の文字である。全角空白がプログラムに混じっているとエラーを起こす。asir ではメッセージやコメント等に日本語が利用可能であるが、慣れるまでは英字モードのみを利用することをお勧めする。右上の言語表示が
となっている状態で cfep/asir に入力しよう。

6. 「」（シングルクオート）と「」（バッククオート）は別の文字である。プログラムを読む時に注意。また、プログラムを読む時は 0（ゼロ）と 0（零）の違いにも注意。

7. マウスの操作には次の三種類がある。

   (a) クリック：選択するとき、文字を入力する位置（キャレットの位置）の移動に用いる。マウスのボタンをちゃんとおとす。

   (b) ドラッグ：移動、サイズの変更、範囲の指定、コピーのときなどに用いる。マウスのボタンを押しながら動かす。

   (c) ダブルクリック：プログラムの実行。open（ファイルを開く）をするために用いる。マウスのボタンを２回つけてちゃんとおとす。ダブルクリックをしたアイコンは白くなったり形状が変わることがあり、ダブルクリックしたらしばらく待つ。計算機が忙しいときは起動に時間がかかることもあり、むやみにダブルクリックを繰り返すとその回数だけ起動されてお騒がなる。

1.2 Cfep/Asir の起動法と電卓的な使い方

cfep のアイコン (いのぶた君)

をダブルクリックすると図 1.1 のように cfep/asir が起動する。以下 cfep/asir を単に asir とよぶ。

図 1.1 の入力窓に計算したい式やプログラムを入力して“始め”ボタン

をおすと実行を開始する。式の計算やプログラムの実行が終了すると、新しいウィンドウ OutputView が聞き結果がそのウィンドウに表示される。“始め”ボタンをおして実行を開始することを計算機用語では“入力の評価を始める”という。
出力小窓にはシステムからのいろいろな情報が出力されるが、内容は中上級者向けのものが多い。
ファイルメニュー
1.2. Cfep/Asir の起動法と電卓的な使い方

図 1.1: cfep/asir の起動画面

から“保存”や“別名で保存”を実行すると入力窓の内容をファイルとして保存できる。出力小窓の内容や OutputView の内容は保存されないので注意してほしい。

cfep/asir を完全に終了するには cfep メニュー
の “cfep を終了” を実行する。

さて図 1.1 では $3 	imes 4 + 1$ の計算をしている。

Asir における計算式は普通の数式と似ていて、足し算は +、引き算は – と書く。かけ算と割算は × や ÷ がキーボードにないという歴史的理由もあり、それぞれ * と / で表現する。積乗 $P^N$ は $P^N$ のように * 記号を用いて表す。

式の末を処理系 (asir) に教える（示す）のに ;（セミコロン）を書かないといけない。文末の “;” のような役割を果たす。またかけ算の記号 * の省略はできない。

例 1.1 以下の左の計算式を asir では右のようにあらわす。

<table>
<thead>
<tr>
<th>式</th>
<th>asir の表示</th>
</tr>
</thead>
<tbody>
<tr>
<td>$2 \times (3 + 5^4)$</td>
<td>2*(3+5^4);</td>
</tr>
<tr>
<td>${(2 + \frac{2}{3}) \times 4 + \frac{1}{5}} \times 2 + 5$</td>
<td>((2+2/3)*4+1/3)*2+5;</td>
</tr>
<tr>
<td>$AX + B$</td>
<td>A*X+B;</td>
</tr>
<tr>
<td>$AX^2 + BX + C$</td>
<td>A<em>X^2+B</em>X+C;</td>
</tr>
<tr>
<td>$\frac{1}{X-1}$</td>
<td>1/(X-1);</td>
</tr>
</tbody>
</table>

計算の順序は括弧も含めて普通の数式の計算と同じである。ただし数学ではかっことして、[,],[,] などがつかえるが asir では（）のみ。[,] や {,} は別の意味をもつ。上の例のように（）を重ねにもつかってよい。この場合括弧の対応関係がわかりにくい。括弧の対応を調べたい範囲をマウスでドラッグして選択し。

ボタンをおすごすことにより括弧の対応を調べることができる。図 1.2 の例では (1+2*(3+4)) と書くべきところを (1+2*(3+4) と書いておりエラーが表示されている。
1.2. Cfep/Asir の起動法と電卓的な使い方

図 1.2: 括弧の対応

質問“Basic 風の使い方を説明する”と書いてありましたが、Basic って何ですか？

答えコンピュータに仕事をさせるには最終的にはプログラム言語（計算機への仕事の手順を指示するための人工言語）を用いる。ウェブもプログラム言語で記述されている。Basic は最も古いプログラマ言語の一つであり、初心者や初歩から、また計算機の仕組みやプログラム言語の理解にも有用である。Basic は高校の数学の教科書に紹介される。著者はいままで“10 進 BASIC”を初心者向け教材として活用していたが、“10 進 BASIC”が MacOS X で動作しないため Cfep を開発した。Asir言語もプログラム言語であり Basic とよく似ているが、C 言語にもっと近い。

質問 MacOS X って何ですか？

答えまだ書いてない。

Asir は数の処理のみならず、\( \sqrt{x} \)や三角関数の近似計算、多項式の計算もできる。左の数学的な式は asir では右のように表す。

<table>
<thead>
<tr>
<th>( \pi ) (円周率)</th>
<th>@pi</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \cos x )</td>
<td>( \cos(x) )</td>
</tr>
<tr>
<td>( \sin x )</td>
<td>( \sin(x) )</td>
</tr>
<tr>
<td>( \tan x )</td>
<td>( \tan(x) )</td>
</tr>
<tr>
<td>( \sqrt{x} )</td>
<td>( x^{1/2} )</td>
</tr>
</tbody>
</table>

三角関数の角度にあたる部分の \( x \) はラジアンという単位を用いて表す。高校低学年の数学では角度を度 (degree) という単位を用いて表すが、数学 3 以上では角度はラジアンという単位で表す。

90 度 (直角) が \( \pi/2 \) ラジアン、180 度が \( \pi \) ラジアン。一般に \( d \) 度は \( \frac{d}{180} \pi \) ラジアンである。

単位ラジアンをもちいると微分法の公式が簡単になる。たとえば \( x \) がラジアンであると \( \sin x \) の微分は \( \cos x \) である。

\( \sin(x) \)や \( \cos(x) \)の近似値を求めるにはたとえば

\[
\text{deval}({\sin(3.14)})
\]

と入力する。これは \( \sin(3.14) \) の近似値を計算する。\( \sin \pi = 0 \) なので 0 に近い値が出力されるはずである。実際 0.00159265 を出力する。deval (evaluate and get a result in double number precision の略) は 64 bit の浮動小数点数により近似値計算する。64 bit の浮動小数点数とは何かの説明は超入門の範囲外であるが、計算機は有限の記憶領域 (メモリ) しか持たないので、小数も有限桁しか扱えないと覚えておこう。64 bit は扱える桁数を表している。詳しくは“asir ドリル”を参照して欲しい。
例 1.2 $\sqrt{2}$, $\sqrt{3}$ の近似値を計算しなさい。

入力

```
print(deval(2^(1/2)));
print(deval(3^(1/2)));
```

出力は図 1.3 をみよ。

上の例のように、セミコロン；で区切られた一連の命令のあつまりはもっとも単純な asir プログラムの例である。一連の命令は始めから順番に実行される。print（式等）は“式等”の値を計算して値を画面に表示する。

さて出力の 1.41421（ひとよ ひとよに ひとみごろ）は $\sqrt{2}$ の近似値なので、print（deval（2^-（1/2）））；の実行結果である。さて出力の 1.73205（ひとみに おごやや）は $\sqrt{3}$ の近似値なので、print（deval（3^-（1/2）））；の実行結果である。最後の 0 はなんなのであるか？実はこれは最後の print 文の戻している値である。むつかしい？別の例で説明しよう。

入力

```
1+2;
2+3;
3+4;
```

この時出力は（OutputView への表示は）

```
7
```

となる。cfe/asir ではとくに print 文をかかない限り最後の文の計算結果（評価結果）しか出力しない。いまの場合は 3 + 4 の結果 7 を出力している。
問題 1.1 1. \(2^8, 2^9, 2^{10}\) の値を計算して答えを表示するプログラムを書きなさい。

2. 2 の累乗はパソコンの性能説明によく登場する。たとえば検索システム google にキーワード “512 メモリ 搭載” を入力したところ “ビデオメモリを 256M から 512M に倍増させ” など、数多くの記事がヒットする。このような記事を (意味がわからなくても)10 件あつめてみよう。512以外の 2 の累乗でも同じことを試してみよう。

3. (中級) 2 の累乗がパソコンの性能説明によく登場する理由を論じなさい。

![図 1.4: 関数のグラフ](image)

発展学習 X11 環境が動いているれば plot(f); 命令で \(f\) の関数 \(f\) のグラフを描ける。\(x\) の範囲を指定したいときはたとえば plot(f,[x,0,10]) と入力すると, \(x\) は 0 から 10 まで変化する。

入力例

```
plot(sin(x));
plot(sin(2*x)+0.5*sin(3*x),[x,-10,10]);
```

問題 1.2 いろいろな関数のグラフを描いてみせてみよう。数学の知識を総動員して計算機の描く形がどうしてそうなのか説明を試みてみよう。

1.3 エラーメッセージ

入力にエラーがあると、エラーメッセージが表示される。
図 1.5: 文法エラー

図 1.5 では 2+4= と入力している。最後に = を書く表現は asir の文法では許されていないので、
“文法エラー” と指摘されている。

大体これでわからしてくれないじゃない。ところがおもしろいもってもプログラム言語は一切読
がきかない。

なお

error([[41,4294967295,parse error,[asir_where,[[toplevel,1]]]]])

の部分は上級者向けの情報なのでとりあえず無視してもかまわない。

図 1.6: エラー行

図 1.6 では

```asir
print( 2^7);
print( 2^8);
print( deval(2^(1/2)));
print( deval(3^(1/2)));
```

と入力している。3 行目は右括弧がひとつ足りなくて print( deval(2^(1/2))); が正しい入力であ
る。エラー行の 3 行目にキャレットが自動的に移動しているはずである。なおこの例では
ボタンをもってもすぐエラーの場所がわかる。
注意: 表示された行はエラーの発生位置であるが、エラーの原因はその前の方の行にあることも多い。たとえば

```
1+2
2+3;
```

と入力するとエラー行は 2 行目であるが、原因は 1 行目で；を書き忘れたことである。

エラー行が複数表示された場合はそれらの中のどこかにエラーがある。複数あるエラー行に順番にジャンプしていくには、実行メニューから次のエラー行へを選択する。

問題 1.3 エラーを生じる式またはプログラムを 5 つ作れ。
第2章 変数とプログラム

2.1 変数

変数に数値等を記憶しておける。変数名は大文字で始まる。なお後述するように asir では多項式計算ができるが小文字で始まる文字列は多項式の変数名として利用される。

2 の累乗を表示する次のプログラムを考えよう。

```plaintext
print( 2^1 );
print( 2^2 );
print( 2^3 );
print( 2^4 );
print( 2^5 );
print( 2^6 );
print( 2^7 );
print( 2^8 );
```

このプログラムは変数 X を用いて次のように書いておけば 2 の累乗だけでなく 3 の累乗を表示するのに再利用できる（図 2.1）。

```plaintext
X = 2;
print( X^1 );
print( X^2 );
print( X^3 );
print( X^4 );
print( X^5 );
print( X^6 );
print( X^7 );
print( X^8 );
```

3 の累乗を表示するには X=2 の行を X=3 に変更すればいいだけである。

アルファベットの大文字 ではじまる英数字の列が asir の変数である。つまり、X, Y, Z はもちろんのこと、Sum とか Kazu とか X1 など 2 文字以上の英数字の列の組み合わせが変数名として許される。

変数を含んだ式をプログラム中で自由につかうこともできる。たとえば

```plaintext
X = 2;
A = 1;
print( 2*X^2 - A );
```

を実行すると 7 が表示される。
図 2.1: 変数の利用

このような例をみると、変数の機能は中学数学で学ぶ文字式と似ていると思うだろう。超入門としてはこれでほぼ正しい理解であるが、よりステップアップしていくには、次のことを強く覚えておこう。

変数とは計算機に数値等を保存しておくメモリ上の場所の名前である。

さて、超入門 第一の関門である。

\[
\begin{align*}
\text{= 記号は次のような形式でつかう:} \\
\text{変数名 = 式;}
\end{align*}
\]

これはまず右辺の式を計算しそのあとその計算結果を左辺の変数に代入せよという意味。= 記号は右辺を計算してその結果を左辺へ代入せよという命令であるだろうと想像したい。

たとえば、\(X=1\) は \(X\) が 1 に等しいという意味ではなく、1 を変数 \(X\) に代入せよという意味である。

ここで注意したいことは、

\[
\begin{align*}
\text{= 記号の意味が数学での意味と違うよ！}
\end{align*}
\]

ということである。これで混乱する入門者も多いのでプログラム言語によっては“2 を変数 \(X\) に代入せよ”を \(X:=2\) と書く場合もある（たとえばプログラム言語 Pascal）。

次のプログラムは \(2, 2^2, 2^4, 2^8\) を計算して表示する。
2.1. 変数

出力が図2.2のようになる理由を説明しよう。まず1行目で変数Xに2が代入される。次に3行目ではま

```python
X=2;
print(X);
X = X*X;
print(X);
X = X*X;
print(X);
X = X*X;
print(X);
```

出力が図2.2になる理由を説明しよう。まず1行目で変数Xに2が代入される。次に3行目ではま

図2.2: 変数の利用

ず右辺の式を計算する。この場合Xの値は2であるので、2×2で結果は4である。この計算が終った後
結果の4が変数Xに代入される。5行目では右辺の式は4×4なので、その計算結果の16が左辺の
変数Xに代入される。

発展学習 Asirは多項式計算でもできる。実はAsirは計算機で記号的に数式を処理するための数式処
理システムでもある。

1. 小文字ではじまる記号は多項式変数である。数学とちがって変数の名前は一文字とはかぎら
   ない。たとえばrateと書くと、rateという名前の多項式変数となる。たとえばx2と書く
   と、x2という名前の多項式変数となる。xかける2はx*2と書く。

2. fctr(poly)はpolyを有理数係数の範囲で因数分解する。fctrはfactorの短縮表現である。
   図2.3のfctrの出力の最初は\(x^2 + 2xy + y^2\)が1×(x+y)^2と因数分解されることを意味して
   いる。図2.3のfctrの出力の2番目は\(x^2 - 1\)が
   \[1^1 \times (x - 1)^1 \times (x + 1)^1\]
   と因数分解されることを意味している。
2.2 くりかえし

くりかえしや判断をおこなうための文が asir には用意されている。この文をもちいると複雑なことを実行できる。まず一番の基礎であるくりかえしの機能をたててみよう。

例 2.1 図 2.1 のプログラムは次のように繰り返し機能 — for 文 — を用いて簡潔に書ける。

```plaintext
X = 2;
for (I=1; I<=8; I++) {
    print( X^I );
}
```

実行結果は図 2.4 をみよ。

繰り返し関連の表現の意味を箇条書きにしてまとめておこう。

1. for (K=初期値; K<=終りの値; K++) {ループの中で実行するコマンド}; はある程度、一度も繰り返したい時に用いる。for ループと呼ばれる。“K<=N” は、“K ≤ N か” という意味である。した表現に、“K>N” があるが、これは “K ≥ N か” という意味である。= のない “K<N” は、“K < N か” という意味である。
2.2. くりかえし

2. \( ++K \) あるいは \( K++ \) は \( K \) を 1 増やすという意味である。\( K = K+1 \) と書いてもよい。同じく、
\( --K \) や \( K-- \) は \( K \) を 1 減らせという意味である。

3. \( ++K \) あるいは \( K++ \) は \( K \) を 1 増やすという意味である。\( K = K+1 \) と書いてもよい。同じく、
\( --K \) や \( K-- \) は \( K \) を 1 減らせという意味である。

\[
\begin{array}{c}
x = 2; \\
\text{for (I=1; I<=8; I++)}
\end{array}
\]
\[
\begin{array}{c}
\text{print("2の"+rtostr(I)+"乗は ",0);}
\text{print( }X^I\text{ );}
\end{array}
\]

\[
\begin{array}{c}
\text{Outp}
\end{array}
\]

図 2.5: for 文

for のあとの { , } の中には複数の文 (命令) を書ける。

\[
\begin{array}{c}
X = 2; \\
\text{for (I=1; I<=8; I++)}
\end{array}
\]
\[
\begin{array}{c}
\text{print("2の"+rtostr(I)+"乗は ",0);}
\text{print( }X^I\text{ );}
\end{array}
\]

この例では日本語を含むので前の節で述べたように日本語空白をプログラム本体にいれないようにして、注意深くプログラムを入力してもらいたい。実行結果は図 2.5 をみよ。print("2の"+rtostr(I)+"乗は ",0); の部分を簡単に説明しておこう。まず最後の 0 は出力のあと改行しない、つまり次の print 文の出力をそのまま続けよという意味。" でかこまれた部分は文字列と呼ばれている。これはこのまま表示される。rtostr(I) は数字 I を文字列表現に変換しない、という意味 (超入門として難しい?)。" と文字列に対して + を適用すると文字列が結合される。

【雑談】江戸時代の数学の本にあった問題の改題
殿様: このたびの働きはあっさりであった。褒美はなにがよいか？
家来: 今日は一円、明日は二円、明後日は四円で、前日の 2 倍づつ、これを 4 週間続けてくださるだけで結構でございます。
殿様: なんともささやかな褒美じゃのう。よしよし。
さて、家来はいくら褒賞金をもらえるだろう？これもまた 2 の累乗の計算である。Cfep/asir で計算してみよう。

例 2.2 for による繰り返しを用いて \( \sqrt{N} \) の数表をつくろう。
for (I=0; I<2; I = I+0.2) {
    print(I,0); print(" : ",0);
    print(deval(I^(1/2)));
}

print(A)
2.4 エンジン再起動

Cfep/asir では次のように 3 つのプロセスが互いに通信しながら動作している。

\[ \text{cfep} \Leftrightarrow \text{コントローラ (ox_texmacs)} \Leftrightarrow \text{計算エンジン (ox_asir)} \]

計算エンジン（計算サーバ）を再起動したり別のものにとりかえたりできる。

エンジン再起動ボタン

をクリックすると、現在利用している計算エンジンを停止し、新しい計算エンジンをスタートする。選択範囲のみを実行するモードでないかぎり利用上で中止との違いはあまりないが、再起動のときのメッセージ

にもあるように、別の計算エンジンを起動することも可能である。この例では unix shell も起動できる。

また、「実行」メニューから“エンジンを自動スタートしない”モードを選んでいる場合に計算エンジンを手動でスタートするには、このボタンを用いる。

图 2.6: 実行の中止

[control] control function_id is 1030
[control] control_reset_connection.
Sending the SIGUSR1 signal to 1226: Result = 0
In ox103_reset: Done.
515
Done

このような開発者専用のメッセージも出力されるが、とりあえずこのようなメッセージがでたら中止が成功したということである。
2.5 ヘルプの利用

Cfep/asir での “関数” は数学の関数のように引数を与えると計算して値を返し、かつある仕事（表示等）をする手続きの集まりである。例えば print, deval, sin, fctr 等は関数である。関数を自分で定義することも可能である。これについては後の説明および “asir ドキュメント” を参照。

あらかじめ定義済みの関数を“組み込み関数” とよぶ。組み込み関数の詳しい説明を調べるには“cfep のヘルプ” から

### ヘルプ

1. 使用説明書のフィールドをfinderで開く

検索したい言葉をスポットライトの窓へ入力。検索のヒント: PDF文書のみに検索したい場合は“検索語 kind:pdf”と入力。

2. cfep の操作説明（まだ書いてない）
3. Risa/Asir マニュアル
4. Risa/Asir実装の関数マニュアル
5. Asir マニュアル Asir.complain 実装的関数マニュアルも含むより詳しい関数一覧

の “Risa/Asir マニュアル” を選び、“Risa/Asir マニュアル” の最初のページの関数一覧から調べたい関数を探す。たとえば fctr (因数分解用の関数) はこの一覧の中にある。

```c
for (i=0; i<oglCommSize; i++) {
    gc = [oglComm objectAtIndex: i];
    [self execute: gc];
}
```
検索には spotlight の活用も有益である。索引

1. 使用説明書のフォルダを finder で開く

2. cep の操作説明 (まだ書いてない)
3. Rind/Apr のマニュアル
4. Rind/Apr の実験的関係マニュアル
5. Asir のマニュアル, Asir-contr. 実験的関数マニュアルも含むより詳しい関数一覧 (は)

この "使用説明書のフォルダを finder で開く" を選ぶと使用説明書のフォルダが開くので、ここを spotlight で検索するような発見があるであろう。ちなみに、この超入門や asir ソリューションはこのフォルダの pdf フォルダの中にある。（なおここからの spotlight 検索は何故か遅いので、メニューの spotlight からの検索の方がいかかもしれない。）
第3章 グラフィック

3.1 ライブラリの読み込み

Asir 言語で書かれている関数定義の集合がライブラリである。ライブラリを読み込むには import コマンドまたは load コマンドを用いる。マニュアルに記述されている関数でライブラリの読み込みが前提となっているものも多い。たとえば、線を引くコマンド glib_line(0,0,100,100); を実行しても、“glib_line が定義されていません”というエラーが表示される。グラフィックコマンドのライブラリ読み込むコマンド

```as
import("glib3.rr");
```

を実行しておくと図 3.1 のように線を描画する。

Asir-contrib プロジェクトにより集積されたライブラリの集合体が asir-contrib である。Asir-contrib を読み込んでしまうと、ほとんどの関数について import が必要かどうか気にする必要はなくなるが、大量のライブラリを読み込むために時間がかかるのが欠点である。asir-contrib は 実行 メニューから読み込む。

---

3.2 線を引く関数

例 3.1

```as
import("glib3.rr");
glib_line(0,0,100,100);
glib_flush();
```

図 3.1 が描画結果である。y 座標は画面が下へいくほど大きくなる。図 3.2 を参照。左上の座標は (0,0), 右下の座標が (400,400). glib_line で (0,0) から (100,100) へ線を描画。glib_flush は画面を更新するためである。flush しないと、描画結果が画面での表示に反映しない場合がある。
glib3.rr をロードすることにより、次のように使えるようになる。

<table>
<thead>
<tr>
<th>関数名</th>
<th>説明</th>
</tr>
</thead>
</table>
| glib_window(X0,Y0,X1,Y1) | window のサイズを決める。画面左上の座標が (X0,Y0)、画面右下の座標が (X1,Y1) であるような座標系で以下描画せよ。ただし x 座標は、右にいくに従いおおきくなり、y 座標は下にいくに従い大きくなる (図 3.2)。
| glib_clear() | 全ての OpenGL オブジェクトを消去し、描画画面をクリアする。
| glib_putpixel(X,Y) | 座標 (X,Y) に点を打つ。
| glib_set_pixel_size(S) | 点の大きさの指定。1.0 が 1 ピクセル分の大きさ。
| glib_line(X,Y,P,Q) | 座標 (X,Y) から座標 (P,Q) へ直線を引く。
| glib_remove_last() | 一つ前の OpenGL オブジェクトを消す。

図 3.1: ライブラリのロード

glib3.rr をロードすることにより、次の関数が使えるようになる。

図 3.2: 座標系

色を変更したいときは、記号で区切ったオプショナル引数 color を使う。たとえば、
3.2. 線を引く関数

```c
import("glib3.rr");
Glib_math_coordinate=1;
glib_window(-2,-2, 2,2);
glib_line(-2,0,2,0 | color=0x0000ff);
glib_line(0,-2,0,2 | color=0x0000ff);
for (X=-2.0; X< 2.0; X = X+0.1) {
    Y = X^2-1;
    X1 = X+0.1;
    Y1 = X1^2-1;
    glib_line(X,Y, X1,Y1);
}
glib_flush();
```

実行結果は図 3.3. ——プログラムの解説はまだ書いてない。
3.3 円を描く関数を作ってみよう

```c
import("glib3.rr");
Glib_math_coordinate=1;
glib_window(-1,-1,1,1);
glib_clear();
E = 0.2; X = 0; Y = 0; R = 0.5;
for (T=0; T<=deval(2*pi*2); T = T+E) {
    Px = X+deval(R*cos(T));
    Py = Y+deval(R*sin(T));
    Qx = X+deval(R*cos(T+E));
    Qy = Y+deval(R*sin(T+E));
    glib_line(Px,Py,Qx,Qy);
    glib_flush();
}
```

プログラムの解説はまだ書いてない。

上のプログラムでは \( \cos, \sin \) を用いて円を描いている。中心、半径を変更したり、色を変更したりしながらたくさんの円を描くには、どのようにすればよいであろうか？“関数”を用いるとそれが容易にできる。

あるひととまとまりのプログラムは関数（function）としてまとめておくとよい。計算機言語における関数は数学でいう関数と似て非なるものである。関数手続き（procedure）とかサブルーチン（subroutine）とかよく言語もある。関数を用いる最大の利点は、関数を一旦書いてしまえば、中身をブラックボックスとして扱えることがある。大规模なプログラムを書くときは複雑な処理をいくつかの関数に分割してまず各関数を十分テストし上げる。それからそれらの関数を組み合わせていくこ
3.3. 円を描く関数を作ってみよう

3.4:

さて円を描く例にもとろう。以下のように関数 circle(X,Y,R,Color) を定義 (def) する。この関数を \( R \) や Color を変化させながら呼びることにより、図 3.4 のような同心円の図を描くことが可能となる。関数について詳しくは "asir ドリル" を参照してほしい。

```python
import("glib3.rr");

def circle(X,Y,R,Color) {
    E = 0.2;
    for (T=0; T<deval(2*@pi); T = T+E) {
        Px = X+deval(R*cos(T));
        Py = Y+deval(R*sin(T));
        Qx = X+deval(R*cos(T+E));
        Qy = Y+deval(R*sin(T+E));
        glib_line(Px,Py,Qx,Qy | color=Color);
    }
    glib_flush();
}

Glib_math_coordinate=1;
glib_window(-1,-1,1,1);
glib_clear();
CC = 0xff0000;
for (P = 0.4; P<0.5; P = P+0.01) {
    circle(0,0,P,CC);
    CC = random()%0x1000000;
}
```

図 3.4: 収数による同心円の描画
問題 3.1 1. 分度器を描くプログラムを作れ。

2. (発展課題) この分度器、綾、おもり、わりばし、板、cfep/asir によるプログラム等を用いて、木や
ビルの高さを測定する機械とソフトウェアシステムを開発せよ。

問題 3.2 (これは発展課題) cfep には OpenGL インタプリターが組み込んである。OpenGL は 3
次元グラフィックスを用いるソフトウェア作成のために用いられる約 150 種類のコマンドから構成さ
れているパッケージで 3 次元グラフィックスの標準規格のひとつでもある。cfep 1.1 ではその中の 10
弱のコマンドを利用できる。

この OpenGL インタプリターを用い、多面体 (polygon) を材料にし、cfep 上級編、OpenGL のプロ
グラムを参考に“家”を書いてみよう。
第4章  For 文による数列の計算

4.1 超入門、第2の関門：漸化式できる数列の計算

例 4.1  $a$ を正の数とするとき,

$$x_{n+1} = \frac{x_n + \frac{a}{x_n}}{2},$$

$$x_0 = a$$

できる数列 $x_0, x_1, x_2, \ldots$ は $\sqrt{n}$ にどんどん近付くこと (収束すること) が知られている。$a = 2$ の時, $x_1, x_2, \ldots, x_4$ を計算するプログラムを書いてみよう。

```c
A = 2.0;
X = A;
for (I=0; I<5; I++) {
    Y = (X+A/X)/2;
    print(Y);
    X = Y;
}
```

このプログラムの実行結果は図 4.1。

![実行結果のスクリーンショット]

図 4.1: $\sqrt{2}$ に収束する数列

超入門での関門は
\[
Y = (X + A / X) / 2;
X = Y;
\]
の意味を完全に理解すること

である。変数の章で説明したように，

変数名＝式：

はまず右辺の式を計算しそのあとその計算結果を左辺の変数に代入せよという意味である。したがって，

\[
Y = (X + A / X) / 2;
\]

は現在の X と A に格納された数字をもとに \((X + A / X) / 2\) の値を計算し，

その結果を変数 Y へ代入せよ，という意味である。また

\[
X = Y \text{ は } X \text{ が } Y \text{ に等しいという意味ではなく，変数 } Y \text{ に格納された数字を 変数 } X \text{ に代入せよという意味である。}
\]

このように考えれば，上のプログラムが \(x_1, x_2, x_3, x_4\) の値を順番に計算して print している理由が理解できるであろう。自分が計算機になったつもりで，変数中の数値がどのように変化していくのか，

書きながら理解して頂きたい。これがはっきり理解でき，応用問題が自由に解けるようになった，超入門卒業である。

4.2 円を描く数列

数列の計算を用いると，\(\cos\) や \(\sin\) の計算をやらずに円を描くことができる。

import("glib3.rr");
Glib_math_coordinate=1;
glib_window(-2,-2, 2,2);
glib_clear();
E = 0.1;
C1 = 1.0; C2=1.0;
S1 = 0.0; S2=E;
for (T=0; T<=deval(2*@pi); T = T+E) {
    C3 = 2*C2-C1-E*E*C2;
    S3 = 2*S2-S1-E*E*S2;
    glib_line(C1,S1, C2,S2);
    C1=C2; S1=S2;
    C2=C3; S2=S3;
    glib_flush();
}

このプログラムの実行結果は図 4.2.

——プログラムの解説まだ書いてない。

この話題は，数列の計算と差分方程式によるシミュレーションに続く。これについてはまた稿をあらためて書いてみたい。

以上で超入門は終了である。続きは "Asir ドリル" を読んでね。
円を描く数列

図 4.2: cos, sin を使わずに円を描く
第5章 cfep 上級編

5.1 TeXによるタイプセット（実験的）

出力をTeXでタイプセットするには“実行”メニューから“出力をTeXでタイプセット”を選択する。finkを用いて、latex, dvips, pstoimgがインストールされていないと動作しない。TeXを用いた仕上がり例は図5.1を示す。なお、TeXでタイプセットする場合ホームの下にOpenXM_tmpなる作業用のフォルダが作成される。タイプセットは実験機能のため、このフォルダの中の作業用ファイルは自動では消去されてない。時々手動で作業ファイルを消去されたい。

5.2 選択範囲のみの実行

画面上の“選択範囲のみを実行”をチェックすると、“始め”ボタンを押したとき、選択範囲のみが評価される。選択範囲がない場合はキャレット位置の行が自動選択されて実行される。COMMAND+Enterと組み合わせてこの機能を使うと、ターミナルからasirを利用するのにちょっと似てくる。図5.1はこのような実行をしている例である。

図5.1: ターミナル風

質問 cfepのインタフェースでデバッグをしながらプログラムを開発するにはどのようにやるとよいか？
図 5.2: “選択範囲のみを実行” の活用

答え: cfep は初心者向きのインタフェースなので、大规模なプログラム開発を想定していないが、私は次のようにライブラリの開発をしている。

1. 必要な関数を書く。下の例では sum1.

2. 関数をテストする入力をコメントの形でその関数の近くに書いておく。下の例ではコメントにある sum1(10,1); 等。

```python
/*
testinput: sum1(10,1);
testinput: sum1(10,2);
*/
def sum1(N,M) {
    S = 0; i=1;
    for (I=1; I<N; I++) {S = S+I^M; }
    return S;
}
```

1. “始め” ボタンで関数定義をロード。この時点で文法エラーなどがあればメッセージにしたがって修正。

2. その後 “選択範囲のみを実行” のモードに変更してコメント内の testinput を実行。

3. 実行時のエラーの行番号への移動は “選択範囲のみを実行” のモードを解除してから行う。

5.3 エンジンを起動しない

質問: テスト編集器またはテキストの閲覧だけで計算をするつもりはありませんが。

答え: “実行” メニューで “エンジンを自動起動しない” を選択。
あとでエンジンを起動したい場合は“再起”ボタンをおしてエンジンを起動する。

5.4 OpenGL インタプリタ

Cfep には OpenGL インタプリターが組み込まれている。OpenGL は 3 次元グラフィックスを用いるソフトウェア作成のために用いられる約 150 種類のコマンドから構成されているパッケージで 3 次元グラフィックスの標準規格のひとつでもある。cfep 1.1 ではその中の 10 個のコマンドを利用できる。詳しくは cfep.app/OpenXM/lib/asir-contrib/cfep-opengl.rr を参照。

OpenGL ではまず OpenGL グラフィックオブジェクトを配置し、それから視点の位置から見た画像を描画する方法を用いる。したがって、システムは常に OpenGL グラフィックオブジェクトの集合を保持している。glib_remove_last() 命令はその最後の要素を削除する命令である。cfep-opengl.rr ライブラリでは, opengl.metaRemoveLast() 関数で最後の要素を削除できる。
import("cfep-opengl.rr");
opengl.metaRemoveAll();
opengl.init();
opengl.glib3DefaultScene(0);
opengl.redraw();
opengl.glColor4f(0.0,0.0,1.0,0.3);
opengl.glBegin(GL_POLYGON); Y=0.1;
opengl.glVertex3f(-1.0, Y, 0.5);
opengl.glVertex3f(-1.0, Y, -0.5);
opengl.glVertex3f(1.0, Y, -0.5);
opengl.glVertex3f(1.0, Y, 0.5);
opengl.glEnd();
opengl.glColor4f(1.0,0.0,0.0,0.5);
opengl.glBegin(GL_POLYGON);
opengl.glVertex3f(0.0, 0.5, 0.0);
opengl.glVertex3f(0.0, 0.5, -0.4);
opengl.glVertex3f(0.5, -0.2, -0.4);
opengl.glVertex3f(0.5, -0.2, 0.0);
opengl.glEnd();
opengl.glFlush();
opengl.metaShowListofOpenGLCommands();

このプログラムでは2枚の長方形を描いている。このプログラムの出力は図5.3。詳しく説明はまだ。
OpenGL の画面には普通の数学のように \((x, y)\) 座標がはいっており、画面から手前側が \(z\) 座標が正の方向、画面の向こう側が \(z\) 座標が負の方向である。 "目" から原点方向を見た画像が図 5.3 にあるように 3 つのスライダーを用いて目の位置を動かせるので、OpenGL オブジェクトをいろいろな角度からみることが可能である。下のスライダーが目の \(x\) 座標、右の二つのスライダーがそれぞれ目の \(y, z\) 座標である。目の動きに慣れるには、次の二つのデモ画面をたまえと面白そうだ。

```python
import("cfep-opengl.rr");
opengl.glib3DefaultScene("mesa demo/ray");
```

```python
import("cfep-opengl.rr");
opengl.glib3DefaultScene("dfep demo/icosahedron");
```
索引

; , 6
=, 16
<=, 18
2 の累乗, 11, 15, 19
3 次元グラフィックス, 30
asir-contrib, 25
cfep, 22
deval, 9
fctr, 17, 23
for, 18
for 文, 18
glib, 25
help, 22
import, 25
interrupt, 20
load, 25
OpenGL, 30, 37
opengl, 37
OpenGL グラフィックオブジェクト, 37
OutputView, 6
plot, 11
print, 19, 20
restart, 21
rtostr, 19
spotlight, 23
TeX, 35
X11, 11
余り, 20
因数分解, 17
エラー, 11, 36
エラーメッセージ, 11
エンジン, 21
オプショナル引数, 26
括弧, 13
関数, 22, 28
組み込み関数, 22
くりかえし, 18
繰り返し, 18
クリック, 6
グラフ, 27
計算エンジン, 21
計算サーバ, 21
再起動, 21
出力結果, 10
出力小窓, 6
数式処理, 17
選択範囲のみの実行, 35
多項式, 17
多項式の変数名, 15
多項式変数, 17
代入, 16, 32
代入記号=, 16
ダブルクリック, 6
中止, 20
次のエラー行へ, 13
入力窓, 6
評価, 6
文法エラー, 12
プログラム, 10
ヘルプ, 22
変数, 15, 17
変数名, 15
文字列の結合, 19
文字列表現, 19
ライブラリ, 25
ラジアン, 9