プログラミング言語としてのAMPLの基本

基礎まとめ

1. AMPL コマンドスクリプト

・AMPLのコマンドの列は *.run ファイル（バッチファイル）に記述される。
・実行はコマンドプロンプトから「ampl *.run」と入力する（*はファイル名）。
・コマンドスクリプトの終わりには「exit;」を記述し、AMPLを終了させる。
・コメントを記入したい場合は、行頭に「#」を入れる。#以降が無視される。
・コマンドを記述したら、行の終わりに「;」（セミコロン）を入れることを忘れてはならない。

2. 表示について

・display *; で*の内容が画面に表示される。
・display * > ファイル名; とすると画面ではなくファイルに書き込まれる。
・display 文は改行が複数行われるため、表示を制御したい場合はprintf 文を使う。
・例：「printf ： "i= %d ¥n", j; 」の意味は、「j」の内容を「i= %d ¥n」という形式で表示させる。
・「%d」は整数型であり、「%f」は実数型を表す。
・記述形式はC 言語に準拠する。
・「¥n」は改行コードを示す。
・まとめると、「printf ： "i= %d ¥n", j; 」は「i=」をテキストとして表示した後に、「j」を形式「%d」で表示して改行するというコマンドである。

3. パラメタ、変数の定義

・一般的にはコマンドスクリプトの中で使われるパラメタ（定数、あるいは式の値など）は最初に定義しなければならない。
・パラメタの定義は次のように行う。

```
param a; 「a というパラメタを定義」
var x; 「x という変数を定義」
```

・AMPL では変数とは、数理計画問題の定式化で用いられる変数（ソルバで最適化されるもの）を表す。
・変数や反復回数などに応じて変わる値はすべてパラメタであることに注意が必要。

5. 代入文

・パラメタや変数に値を代入する場合、単純に「a=a+1;」と記述してはならない。
・正しくは、let 文を「let a:=a+1; 」（コロン＋等号）と記述する。

反復と条件文

1. 反復 for 文

・「a..b」（ただし、a<b は整数値）はaからbまでの整数値の集合を表す。
・例：「1..10」は集合{1,2,3,…,10}を表す。for {i in 1..10} でiが1から10までの反復を表す。
・例：1から10までの整数値を表示

```
for {i in 1..10}
{
        display i;
}
```

・例：2重ループ、以下のように2重ループも記述できる。

```
for {i in 1..5}
{
    for {j in 1..5}
    {

    }
}
```

・注意事項： for 文で使われた for {i in 1..5}などに含まれる反復回数を示すカウンタ「i」は、for 文の中でのみ使われていれば、パラメータとして定義する必要はない。

2. repeat while 文

・例：

```
repeat while(条件１ )
{
      条件1が成立する限りここに示すコマンドを実行する。
}
```

3. if-then-else 文

・ある条件が成立するとき、特定の操作を行うようにする。
・条件に対する算術式としての表現は以下のようなものがある。
=, <>, <=, >, >=, 等がある。等しいかどうかはC言語と異なり、単に「=」と記述。
・例：if(条件1)then

```
{
      条件1が成立するときここに示すコマンドを実行する。
}
else
{
      if-then での条件1が成立しないときにここに示すコマンドを実行する。
}
```

・else 以降は省略してもよい

4. 反復からの脱出

・break; コマンドは反復からの脱出を行う。

乱数の使用

1. 乱数の基礎

・区間(m,n)における一様分布に従う乱数はUniform(m,n)

・正規分布N(μ, σ2)に基づく乱数はNormal(μ, σ2)

・区間$[0,2^{24})$における整数値をとる一様乱数はIrand224( )で与えられる。

例：(0,1)における一様乱数を10個生成

```
option randseed 0; #乱数の初期値を更新 0以外の正整数の場合は同じ系列の乱数
for{i in 1..10}
{
   display Uniform(0,1); #Uniform(0,1)は0以上1以下の乱数
}
```

2. サイコロを振るシミュレーション

・ガウス記号[x] (x以下の最大整数) を表すには関数 floor(x)を用いる。

・x以上の最小整数を表すには関数 ceil(x)を用いる。

例：関数値ceil(Uniform(0,6))で1から6までの整数値をとる一様乱数を作成する。

```
option randseed 1;
for{i in 1..10}
{
   display ceil(Uniform(0,6)); #ceilの場合(0,6) floorの場合(1,7)
}
```

例：サイコロを1000回振って各目が出る回数をカウントせよ。

```
param ransu;
param count{i in 1..6};
for{i in 1..6}
{
        let count[i]:=0; #カウント値のリセット
}
option randseed 1;
for{i in 1..100}
{
        let ransu:=ceil(Uniform(0,6)); #乱数値をransuに与える
        for{j in 1..6}
        {
             if(ransu=j)  then     #乱数値がjとなれば
             {
                   let count[j]:=count[j]+1; #jが出る回数に1加える
              }
        }
}
display count;
```

例題1

1. 割算の基礎

割算 20÷6=3 あまり 2 の計算を行う。単に 20/6 を表示すると 3.3333 が答となる。商の 3 を表示するには、20÷6 の整数部分を表すことが必要である。ガウス記号[x] (x 以下の最大整数) を表すには関数 floor(x)を用いる。

```
param a;
param b;
let a:=20;
let b:=6;
display a/b;
display floor(a/b);
display a-b*floor(a/b); #floor は切り捨て関数 ceil は切り上げ
printf:"%d 割る % d は %d あまり %d ¥n",a,b,floor(a/b),a-b*floor(a/b);
```

2. 約数を求める

パラメータ c=20 の約数を求める。20 を 1 から 20 までの整数 i で割って、余りが 0 ならば i は c=20 の約数であることに注意して、以下の空白を埋めよ。

```
param c;
let c:=20;
for{空白 }
{
        if( 空白                        ) then #c を i で割ってあまりが 0 ならば
        {
        printf:"%d は %d の約数である ¥n",i,c; #i は c の約数→約数 i を表示
        }
}
```

3. 素因数分解

パラメータ a=88 を素因数分解すると、88=2*2*2*11 となる。88 を 2 から 88 までの数で割っていき、割り切れる場合は、88=2*44 なので、次は a=44 を 2 から 44 までの数で割っていく。

88=2*44, 44=2*22, 22=2*11, 11=1*11

終了するのは 11 を 11 で割った商が 1 となる場合である。終了判定条件を含めて次の空白を埋めよ。

```
param a; #a というパラメータを定義
let a:=88;
display a;
repeat while( 空白 )
{
        for{i in 空白 } #a を 2 から a までの数で割っている
        {
                if(空白 ) then #a を i で割って余りが 0 ならば
                {
                 let a:=  空白;  #a を更新
                 display i;
                 空白;  #ここで脱出すると repeat に戻る break;
                }
        }
}
```

例題2：ソーティング

1. 並べ替えの基本

次の5個のデータを小さい順に並べ替える。

```
param a{i in 1..5};
let a[1]:=8; let a[2]:=3; let a[3]:=6; let a[4]:=5; let a[5]:=4;
a[1], a[2], a[3], a[4], a[5]
```

8, 3, 6, 5, 4 初期値
3, 8, 6, 5, 4 1と2を比較して入れ替え
3, 6, 8, 5, 4 2と3を比較して入れ替え
3, 6, 5, 8, 4 3と4を比較して入れ替え
3, 6, 5, 4, 8 4と5を比較して入れ替え、最大の8がa[5]に入る
3, 6, 5, 4, 8 1と2を比較
3, 5, 6, 4, 8 2と3を比較して入れ替え
3, 5, 4, 6, 8 3と4を比較して入れ替え、8を除いて最大の6がa[4]に入る
3, 5, 4, 6, 8 1と2を比較
3, 4, 5, 6, 8 2と3を比較して入れ替え、6,8を除いて最大の5がa[3]に入る
3, 4, 5, 6, 8 1と2を比較、5,6,8を除いて最大の2がa[2]に入り、最小の3がa[1]

2. プログラミング

```
param a{i in 1..5};
let a[1]:=8;
let a[2]:=3;
let a[3]:=6;
let a[4]:=5;
let a[5]:=4;
param aa; #入れ替えのために用いるパラメータ
display a;
for{k in 0..3}
{
        for{j in 空白 }
        {
                if(条件 a[j] と a[j+1] の比較)then
                {
                 a[j] と a[j+1]の入れ替えを行う;
                }
        }
}
display a;
```

課題：上のデータを大きい順に並べよ。

課題：10個のデータを乱数により作成し、小さい順に並べよ。

例題3：ニュートン法

1. ニュートン法

非線形方程式 $f(x)=0$ を解く Newton 法について示す。関数 $y=f(x)$ をある x^0 において、直線（接線）で近似する。接線の方程式は以下のようになる。

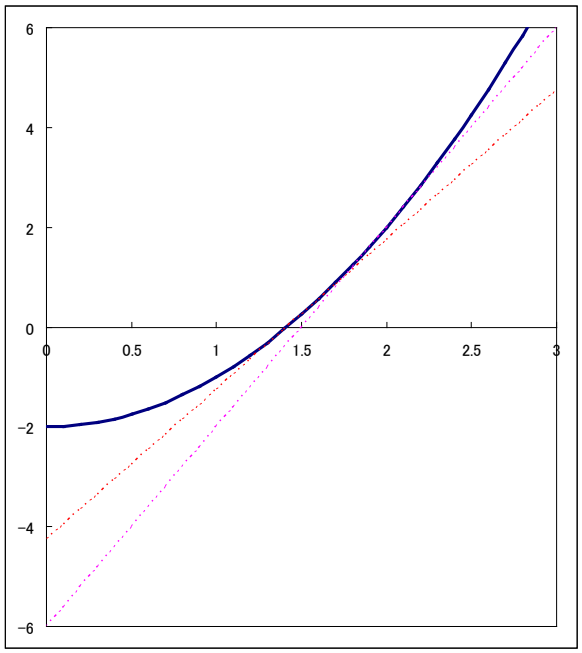
$y-f(x^0)=f'(x^0)(x-x^0)$

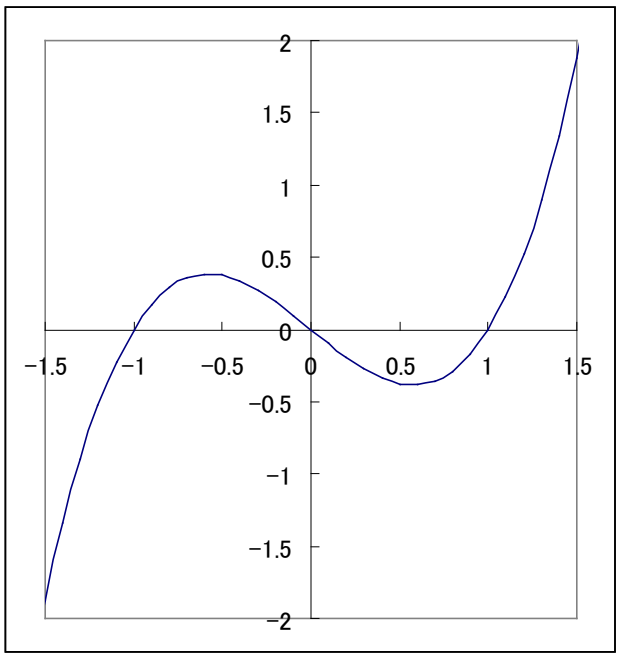
この接線が x 軸と交わる点を x^1 とすると、$f(x^0)$ よりも $f(x^1)$ の方が 0 に近いと考えられる。よって、接線が x 軸と交わる x^1 を求める。

$x^1=x^0-f(x^0)/f'(x^0)$

これより、以下の更新公式が導かれる。

$x^{n+1}=x^n-f(x^n)/f'(x^n)$





課題: $f(x)=x^2-2=0$ を満たす解を求める。

```
param x0;
param x1;
param i;
param n;
let i:=1; #反復回数
let n:=10; #反復回数の上限
let x0:=10; #初期値
display i, x0;
repeat while(i<n)
{
   let x1:=      空白;
   let x0:=x1; #得られた x1 を新たな初期値に
   let i:=i+1; #反復回数の更新
   display i, x0;
}
exit;
```

課題: f(x)= x(x−1)(x+1)=0 の解を 初期値-1.2, −0.3, 0.4, 0.8 の値について求めよ。