計算機プログラミングⅠ　A クラス 中間試験 2007 総評

出題・採点: 井上純一 (情報科学研究科棟 8-13)　　平成 19 年 6 月 15 日

受験者数: 71, 欠席者数: 4.

6 月 1 日に実施した中間試験の採点をしてみました. 第 7 回講義まで学習した内容: 変数の宣言, 繰り返し, 条件分岐, 関数等 (今回「配列」は含まず) の全てを使って一つのまとまったプログラムを作ってもらう問題でしたが, 出題側の指示を全て満たした完全な正解が全体の約 1 割, 問題の要求を満たしていなくても, ともかく, 3 つの関数全てに正しい出力を与えるプログラムが全体の約 3 割, 3 つの自作数学関数のうちどれかが間違っている, あるいは完成していないものがやはり 3 割. 残りの 3 割の答案がコンパイルの段階で何らかのエラーが出るもので, 極少数ですが, 持ち込み可にもかかわらず, ほとんどプログラムが書けていない答案もありました. そんなわけで, 全体的にみて決して良い出来とは言えない結果でした. 今までは各項目ごとに練習問題を解いてもらっているので, 各項目は理解しているというものの, それらを組み合わせて一つのプログラムを書くという訓練はしていなかったので, そのあたりが現段階では難しいと言えば難しかったのかもしれません.

いずれにしても, ここでの誤りをいかに今後に活かすか, がとても重要ですので, 以下に全てではないですが, 皆さんの答案に散見された間違いを列挙しておきます. 各自が再度自分の答案 (※ 青ペンで添削しておきました. 余談ですが私は採点業務に普段「赤色」ではなく, 「青色」か「緑色」を使います. 「『青』や『緑』なら同じ×でも『目に優しい』」) と照らし合わせて確認し, 今後の学習上の参考にしてください.

皆さんの答案に見られたいくつかの問題点

- scanf() 関数で, 変数 x にキーボード入力する際の

  ```
  scanf("%lf",&x);
  ```

  の & を忘れたもの.

- C 言語で等式は “=” ではなく “==” ですが, これを書き間違えたもの多数. これは紙に書く, キーボード入力するにかかわらず, 頻繁におこす間違いで, 実行結果にも影響を与える場合がありますので, 要注意です.

- 指数関数 Exp(x) は問題の指示に従わず, かつ, math.h に定義された指数関数 exp(x) を使わなくても, 自然対数の底を例えば $e = 2.71$ などと置き, 変数 x の a 乗を与える数学的関数: pow(x,a) を用いて

  ```
  return (pow(2.71,x));
  ```

  のように値を返す関数を作ることもできます (もちろん, わざわざ関数にしなくても, メイン関数内で pow(2.71,x) を直接的に表示させることも可能). しかし, この math.h に定義された関数 pow(x,a) を使うためにはプログラム先頭で

  ```
  #include<math.h>
  ```

  と, このヘッダファイルをインクルードする必要があります. これを忘れると, pow(x,a) が未定義としてコンパイル・エラーが出ます.

- 指数関数 Exp(x) の作り方としてはこの他に

```
double Exp(double x)
{
   int i;
   double y;
   for(i=1,y=2.71; i<=x; i++){
    y = y*2.71;
    }
   return (y);
}
```

としたものが複数ありました. 気持ちはわかりますが, まず, 上の関数は Exp(x) の引数 x が整数の場合しか使えません. 期待通りの出力結果を与えないという意味で, コンパイルは通っても実行時にエラーが出ます.

- Sin(x) の作り方とその使い方に関して

```
double sin(double x, int j)
{
     if(j==1){
     return (x);
     }else{
     return (pow(-1,j-1)*pow(x,2j-1)/fact(2j-1) + sin(x,j-1));
     }
}
```

のように指数関数の展開と sin(x, j) を再帰的に用いて作成しても OK です. ただし, j = 1 の場合の関数値を予め与えておかないとメイン関数の中で使う際に実行時エラー (セグメンテーション違反) が出ますから注意が必要です.

- 同じ正弦関数を作るにも for ループで数え上げをしていく直前での変数の初期化失敗が致命的な間違いを起こしているような答案も複数ありました. 例えば

```
double sin(double x)
{
     int i, n=100;
     double ans=0,term=0,mx2=-x*x;
     for(i=1; i<n; i++){
        term *= mx2/(2*i*(2*i-1));
        ans += term;
        }
     return (ans);
}
```

としてしまうと, term はゼロで初期化されてますから, これ以降, for ループ中の term *= ... で何を何回掛け合わせようが値はゼロです. for ループを用いた繰り返し演算を行う際には**面倒でも具体的にはじめの数項を紙に書き出してチェックしてみる**ことがこの手の実行時エラーを少なくする方法の一つです.

- 関数の名前の付け方に関し, sin(x) など既に math.h に定義されたものを使う場合, コンパイル時に名前の重複があったとしてエラーが出ます. プログラム中に他の数学関数を一切使わないのであれば, math.h のインクルードを外すことで, この種のエラーは無くなりますが, それは現実的選択ではないので, 解答例に示したように先頭の文字を Sin(x) のように大文字にするなど, 名前の付け方を工夫する必要があります. また, 関数名の途中にスペースを入れて Sin func(x) などとすることもできません. コンパイル時にエラーがでます.

- ニュートン法の収束性チェックの条件文で

```
if(fabs(x-y) < EPS){
return (y);
}
```

などとする際に変数 EPS をマクロ定義しないで用いた答案も見られました. プログラムの先頭で

```
#define EPS 1.0e-10
```

などと定義すべきです. あるいは, 今後頻繁に使うのであれば, ヘッダファイル (constant.h など名づける) を作成し, そこにまとめてこの手の定数を定義して書き込んでおくと良いと思います. その際には既に学んだように

```
#include "constant.h"
```

のように"..."で囲ってインクルードすることを忘れないように.

- こちらの全く想定していなかったもので, よくあったものとして, メイン関数の中で別の関数を定義して用いているような答案がありました. 例えば整数の階乗を表示させるプログラムで書けば

```
#include<stdio.h>
#include<math.h>
double fact1(int n);
main()
{
        int a;
        printf("a=");
        scanf("%d",&a);
/* メイン関数の中で別の関数本体を定義 */
 double fact1(int n)
 {
       int i;
       double ans=1.0;
        for(i=1; i<=n; i++){
        ans *= i;
        }
        return (ans);
}
        printf("%d!=%f (def.1)\n", a,fact1(a));
}
```

のようにメイン関数の中で関数 fact1 を定義し, この値を出力させて使っている答案です. 意外なことにこのプログラムはコンパイルも通り, 正しい答えを出力しますが, プログラム構造上の観点からみると, fact1 を関数にした意味がほとんど無く, 非常に読みにくい (つまり, きれいでない) プログラムになってしまいます.

- 関数を全く使わないでメイン関数の中に全てを書き込んでしまっているような答案もかなりの数ありました (これは今まで私が教えた経験上, 高学年へと進んだ際の「学生実験」等でプログラムを書いてもらう際にも見受けられるので注意してください. おそらく, 気をつけないと, 知らず知らずのうちに全てをメイン関数の中に書き込んでプログラムを作るというのが, ある種, 自分のプログラミングにおける「スタイル」になってしまうのだと思う). 上の例で書けば

```
#include<stdio.h>
#include<math.h>
main()
{
        int a;
        printf("a=");
        scanf("%d",&a);

       double ans=1.0;
        for(i=1; i<=a; i++){
        ans *= i;
        }
        printf("%d!=%f (def.1)\n", a,ans);
}
```

とするようなものです. この例程度のプログラムならば, これでも何も問題ではないのですが, 今後, 様々な関数を用いて複数の処理を行わせるような場合, この手の書き方でメイン関数を大きくしていくと, 見通しが悪く, バグを見つけることが非常に困難なプログラムになってしまいます. おそらく, このようなプログラムを書いた方の多くは関数の作り方と使い方をまだ理解していないものと思われますので, 今のうちに再度確認しておいてください.

- 関数プロトタイプ宣言の場所をメイン関数の直後

```
main(){
double Sin(double x); /* 自作正弦関数のプロトタイプ宣言 */
double Exp(double x); /* 自作指数関数のプロトタイプ宣言 */
double Root(double x); /* 自作平方根を返す関数のプロトタイプ宣言 */
.........
.......
......
```

にしているものあり. コンパイルエラーが出ます.

- 一番多かったプログラム上の問題を含む答案としては, 間違いではないですが, 指示通りに「再帰的関数定義」を使っていないものです. 再帰的関数定義とは自分自身を return 文の中に含むもので, 次のような構造を持っています.

```
double sine(double x,int n)
{
  ...........
  ........
  .....
  return ((何かの式)*sine(x,n-1));
 }
```

こういう再帰構造を使って関数を定義し, それを用いて正弦関数を自作し, その値を出力させることがここでの課題でした. 従って, 厳密な採点基準をとれば次のプログラムは再帰的な関数定義を使っていないので「不正解」です.

```
double Sin(double a)
{
      int i;
      double t,s=0;
      for(i=1;i<=10;i++){
       t = pow(-1,i-1)*pow(a,2*i-1)/fact(2*i-1);
       s += t;
       }
      return (s);
}
```

(ここで, fact() は関数マクロ, あるいは外部関数として定義されていると思ってください). プログラムの実行結果だけを見ると間違いではありませんが, 問題の指示とは明らかに違います. 「再帰的関数」の「再帰的」という意味を上の例でもう一度確認しておいてください.

- せっかく関数自体は書けているのに, return 文を忘れてしまったばっかりに値を返さない関数を書いてしまっているものも見受けられました. メイン関数の中で処理だけを実行するような値を返さない関数も作れますが, その場合には void 型関数として宣言したことを確認しておいてください.
- 再帰的な関数定義は使ってませんが (その意味で完全な正解ではないですが), 次のようにして指数関数を作ることもできます. この手の答案を書いてくださった方も居ました.

```
double Exp(double x,int n)
{
      int i;
      double term, term2,sum;
      for(i=1,sum=1,term=1,term2=1,i<=n; i++){
      term = i*term;
      term2 = x*term2;

      sum = sum + term2/term;
      }
      return (sum);
}
```

この他にも細かくみると printf 関数を問題の指示通りに使っていないもの, $\sqrt{x}$ の値を計算する関数を作る際に指示に従ったニュートン法ではなく, 2 分法で作成したものなどありました. 実際に紙に鉛筆で書いたプログラムでは細かな部分が間違っている場合があり, そうした自分のミスに気づかないこともあると思います. そこで, 各自が返却された自分の答案を Xemacs でファイルに書き込み, コンパイル・実行することで自分の答案の問題点を確認しておいてください.