2014 年度 プログラミング入門期末試験

2014 年 8 月 1 日

試験時間　13:15～14:35 （80 分）

【注意事項】

- 解答開始の指示があるまで問題文を見てはいけません。
- 携帯電話やスマートフォンは必ず電源を切ること。音が鳴った場合は不正行為とみなします。
- 試験に使用できないハンドアウト、ノート、全ての電子機器類は鞄にしまうこと。
- 問題用紙は表紙を含む 4 ページ、解答用紙は表裏 1 枚です。
- 解答用紙に必ず学籍番号と氏名を記入すること。
- 解答開始後 45 分（14:00）までは退室してはいけません。
- 問題用紙は持ち帰ること。
- 不正行為をしてはいけません。確認された場合、今学期の全科目の成績がFになります。
- 解答用紙は 8 月 4 日（月）に返却される予定です。詳細は採点終了後メールにてお知らせします。

問題 1 ((1),(2) 各 1 点, (3),(4) 各 2 点, 計 35 点)

(1) 以下の説明文について、正しいものには「○」を、間違っているものには「×」を記入しなさい。

(a) コンピュータが理解できる言語をコンパイラ言語という。×
(b) 明示的な型変換をキャストと呼ぶ。○ (int)x みたいなの
(c) double 型変数 a に入力するには scanf("%lf",a); とすればよい。× %d %lf &のつけ忘れに注意
(d) for 文は主に見張り方式のループに利用される。× ある条件を満たしているか見張るのはwhile
(e) int data[5][20]; の要素数は 100 である。○ 5*20
(f) プログラムの処理手順を図的に表す方法として、フローチャートがある。○
(g) プログラムを見やすくするために、ループ内などで字下げすることをインデントという。○
(h) scanf の戻り値は、正しく読み込めた個数を返す。○ %dや%lfの数, ファイルの最後はEOF(-1)
(i) 関数内の自動変数はプログラムが終了するまで保持される。× ローカル変数は、関数終了時に破棄される。
(j) 関数呼出の引数と関数定義の仮引数は同じ名前にする必要がある。× 関数が違えれば、名前が同じでも違くてもよい。

(2) 以下の説明文について、空欄に当てはまる語句を括弧内の選択肢から選び、番号を記入しなさい。

(a) 浮動小数点型の値を整数型の変数に代入すると、小数点以下の値は[　　]。 2
[1:四捨五入される, 2:切り捨てられる, 3: 場合により不定]

(b) printf の書式指定で小数点以下の表示桁数を指定した場合、次の桁以降の値は[　　]。3 そのため四捨五入は演算により求める。
[1:四捨五入される, 2:切り捨てられる, 3:場合により不定]

(c) C 言語のプログラムソースコード中にコメントを含む場合、[　　]をつける。2
[1:行頭に「#」, 2:コメントの前後に「/*」と「*/」]

(d) 変数名として使えないものは[　　]である。
[1:data.out, 2:dATA, 3:_2data] 1

(e) C 言語の条件式で偽となる値は[　　]である。
[1:−1, 2:1, 3:0] 3 0以外は真として扱われる。

(f) 二次元配列の初期化は[　　]ような順序で行われる。
[1:まず 1 列を埋める, 2:まず 1 行を埋める] 1 二重ループの中のループ(j)で代入をする。

(g) exit 関数を使用する際に #include で宣言すべきなのは[　　]である。
[1:math.h, 2:stdlib.h] 2

(h) C 言語の三角関数では角度は[　　]で取り扱われる。
[1:度(°), 2:ラジアン] 2 PI(3.14…)を使った浮動小数点型で表す。

(i) gcc によるコンパイルの際に, 出力される実行ファイルを a.out 以外の名前にしたい場合はオプション[　　]を使う。
[1:−lm, 2:−o] 2 -lm は、math.hを使うときに使う。

(3) 以下の各処理に相当するプログラムコードを書きなさい。問題文中の変数は既に宣言されているものとしてよい。また、ループのカウンタ変数は int 型変数 i とする。

(a) int 型変数 a,b において a の b による剰余を出力する。 printf("%d", a%d);
(b) int 型変数 a の逆数を小数点以下 3 桁で出力する。printf("%.3f", 1.0/(double)a);
(c) 3x3 の int 型 2 次元配列 array の対角成分をループで出力する。
(d) double 型変数 x と y をキーボードから入力する。

(c) for(int i=0; i < 3; i++) printf("%d", a[i][i]);
(d) scanf("%lf%lf",&x, &y);
(b)キャストを忘れない。 (c) 対角成分は、行・列ともに動く。
(d) 改行(\n)を付けない。 &つけ忘れに注意！

(4) 以下は成績の評価を出力するプログラムの一部を二通りの書き方で示したものである。int 型変数 n が 100 では「満点」、80 以上 99 以下では「優」と表示し、79 以下では何も表示しないとする。正しく動作するように(a)〜(d)の空欄を埋めなさい。

【方法 1】
```
if(n [ (a) ] 100) printf("満点");
[ (b) ] if(n [ (c) ] 80) printf("優");
```
(a) ==　(b) else　(c) >=

【方法 2】
```
if(n [ (a) ] 100) printf("満点");
if(n >=80 [ (d) ] n<=99) printf("優");
```
(a) ==　(d) &&

方法1: 満点以外の80点以上の人は優という考え。
方法2: 満点の人はいるか？優の人はいるか？ と2回条件判断するという考え。

問題 2 ((1),(2),(3)各 2 点, (4)10 点, 計 30 点)

(1) 次の(a)〜(c)の空欄に当てはまる内容を書きなさい。
以下のプログラムコードを実行すると、int 型変数 n の値が 1 のとき [(a)]、2 のとき [(b)]、3 のとき [(c)] と表示される。
(a) C　(b) B　(c) AB

breakがないので、case 2:も動いてしまう。

```
switch(n) {
  case 3: printf("A");
  case 2: printf("B"); break;
  default: printf("C"); break;
}
```

(2) 以下のプログラムの一部は、無限ループを使い、入力された整数値が 0〜100 の範囲であればループを抜けて次の処理に移り、そうでない場合は再度入力を求めるものである。正しく動作するように(a)〜(c)の空欄を埋めなさい。

```
int a;
while(1) {
  printf("0 以上 100 以下の数字を入れてください:");
  [ (a) ];
  if( [ (b) ] ) [ (c) ];
  printf("範囲外の値%d が入力されました\n", a);
}
```
(a) scanf("%d", &a)　(b) 0 <= a && a <= 100　(c) break
数学のように 0 <= a <= 100 とは書けない。

(3) 次のプログラムは入力された N 個の整数のうち奇数がいくつあるかを数えるものである。ただし、0 が入力されたらそこで入力をやめ、そのときの奇数の個数を表示するものとする。実行例のように、正しく動作するように(a)〜(d)の空欄を埋めなさい。

```
#include <stdio.h>
#define N 10

main() {
  int i=0, data=0;
  int count=0;
  printf("整数を%d 個入力してください\n", N);
  for(i = 0; i < N; i++) {
    scanf("%d", &data);
    if(data == 0) [ (a) ];
    if(data%2 == 0) [ (b) ];
    [ (c) ]++;
  }
  printf("入力個数%d のうち奇数は%d 個\n", [ (d) ]);
}
```
(a) break　(b) continue　(c) count　(d) i, count
偶数のときはループの始めに戻る。
break: ループを抜ける。
continue: 以降の処理を飛ばす。

【次頁に続く】

【実行例】

```
% ./a.out
整数を 10 個入力してください
1 2 3 4 5 6 7 8 9 10
入力個数 10 のうち奇数は 5 個
% ./a.out
整数を 10 個入力してください
1 2 3 4 5 0
入力個数 5 のうち奇数は 3 個
%
```

(4) 下記は 10 個の整数をキーボードから int 型 1 次元配列 data に入力し、data の要素の最大値と最小値、および、それぞれの値が何番目に入力されたか、を表示するプログラムである。空欄を埋めてプログラムを完成させなさい。

```
#include <stdio.h>
#define N 10

main() {
  int i, max, min, max_index, min_index;
  int data[N];

  printf("整数を%d 個入力してください\n", N);
  for(i = 0; i < N; i++){
    scanf("%d", &data[i]);
  }
  max = data[0];
  min = data[0];
  for(i=1; i<N; i++){
    if(max < data[i]){
      max = data[i];
      max_index = i;
    }
    if(min > data[i]){
      min = data[i];
      min_index = i;
    }
  }
  printf("最大値は%d で、%d 番目に入力された数です\n",
          max, max_index+1);
  printf("最小値は%d で、%d 番目に入力された数です\n",
          min, min_index+1);
}
```

どれが最大・最小値かわからないので、最初の要素を最大・最小と仮決め。

残りの要素を見ていって、それまでの最大より大きい値、それまでの最小より小さい値が出てきたら最大・最小をそれぞれ更新していく。

【実行例】

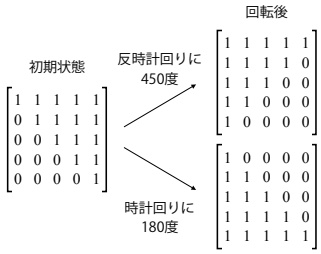
```
% ./a.out
整数を 10 個入力してください
4 8 -1 23 5 37 3 -15 1 21
最大値は 37 で、6 番目に入力された数です
最小値は-15 で、8 番目に入力された数です
%
```

正の場合は4で割った余りを出すことで、 0,1,2,3,4,5,6,7,8… が 、0,1,2,3,0,1,2,3,0…に変換される。

負の場合は、(-x%4)で正に戻したあとに、4で割ることで正の状態と同じ状態にする。(0,1,2,3,0,1,2,3…)
4から引くことで、反時計回りを表す。(4,3,2,1,4,3,2,1…)
最後に4で割ることにより、4→0にする。(0,3,2,1,0,3,2,1…)
負の場合が、ややこしいが円の図などを描いて落ち着いて求めよう。

問題 3 （各 5 点，計 15 点）

下図のように5x5の上三角行列の要素を回転させるように並び替えて出力するプログラムを作成したい。



入力された整数値に 90 度をかけた角度で回転させるものとする。その際、正の数であれば反時計回りに、負の数であれば時計回りに回転させる。例えば、5 が入力された場合は反時計回りに 450 度回転させ、-2 が入力された場合は時計回りに 180 度回転させる。以下の仕様と実行例を満たすように、プログラムの(1)～(3)の空欄を埋めなさい。

【仕様】

- int 型 2 次元配列 matrix は、図に示されている上三角行列の初期状態のデータを格納するものとし、宣言と併せて初期化を行う。
- 配列の宣言やループの制御にマクロを使用すること。
- 行列要素の出力パターンは全部で 4 通り。出力パターンの番号を示す変数を n とし、
  n = 0 → 回転処理なし
  n = 1 → 反時計回りに 90 度回転
  n = 2 → 反時計回りに 180 度回転
  n = 3 → 反時計回りに 270 度回転
  として出力させる。
- n の値は入力される整数値 x を使って算出するものとする。x の値が負の数の場合にも対応させること。

【実行例】

```
% ./a.out
整数を 1 個入力してください:5
11111
11110
11100
11000
10000
%
```

n={-8,-4,0,4,8…}
n={-11,-7,3,3,7,11…}
n={-9,-5,1,5,9,13,…}
n={-10,-6,-2,2,6,10…}

【プログラム】

```
#include <stdio.h>
#define N 5

main() {
  int i, j, x, n;
  (1) int 型 2 次元配列 matrix の宣言と初期化
   int matrix[N][N]={{1,1,1,1,1},{0,1,1,1,1},{0,0,1,1,1}
   {0,0,0,1,1},{0,0,0,0,1}};
  printf("整数を 1 個入力してください:");
  scanf("%d", &x);
  (2) 出力パターン番号 n の計算   if(x >= 0) n= x%4;
                                  else n = (4-(-x%4))%4;
```

行ごとに括弧で分けると間違えが減る。
最後のセミコロンや括弧対応に注意。

【次頁に続く】

以下に、(i, j) の形式で行列の座標を表示している。
無回転行列(case 0)を基準に、回転した場合、i, jが
それぞれどのような値を辿ればいいか意識するように。
行と列の辿り方が違う場合(case 1, case 3) iとjのループが
逆転する(必ずしも、そうする必要はないがしたほうが良い)
ことにも注意すること。覚えるのではなく、考えること。

```
switch(n) {
  case 0:
    for(i = 0; i < N; i++) {
      for(j = 0; j < N; j++) {
        printf("%d", matrix[i][j]);
      }
      printf("\n");
    }
    break;
  case 1:
    for(j = N-1; j >= 0; j--) {
      for(i = 0; i < N; i++) {
        printf("%d", matrix[i][j]);
      }
      printf("\n");
    }
    break;
  case 2:
    for(i = N-1; i >= 0; i--) {
      for(j = N-1; j >= 0; j--) {
        printf("%d", matrix[i][j]);
      }
      printf("\n");
    }
    break;
  case 3:
```

(0, 0) (0, 1) (0, 2) (0, 3) (0, 4)
(1, 0) (1, 1) (1, 2) (1, 3) (1, 4)
(2, 0) (2, 1) (2, 2) (2, 3) (2, 4)
(3, 0) (3, 1) (3, 2) (3, 3) (3, 4)
(4, 0) (4, 1) (4, 2) (4, 3) (4, 4)

(0, 4) (1, 4) (2, 4) (3, 4) (4, 4)
(0, 3) (1, 3) (2, 3) (3, 3) (4, 3)
(0, 2) (1, 2) (2, 2) (3, 2) (4, 2)
(0, 1) (1, 1) (2, 1) (3, 1) (4, 1)
(0, 0) (1, 0) (2, 0) (3, 0) (4, 0)

(3) n=3 の場合の行列要素の出力

```
for(j=0;j<N;j++){
  for(i=N-1;i>=0;i--){
    printf("%d",matrix[i][j]);
  }
  printf("\n");
}
```

(4, 4) (4, 3) (4, 2) (4, 1) (4, 0)
(3, 4) (3, 3) (3, 2) (3, 1) (3, 0)
(2, 4) (2, 3) (2, 2) (2, 1) (2, 0)
(1, 4) (1, 3) (1, 2) (1, 1) (1, 0)
(0, 4) (0, 3) (0, 2) (0, 1) (0, 0)

```
    break; }
  default:
    printf("n の計算が間違っています\n");
  }
}
```

(4, 0) (3, 0) (2, 0) (1, 0) (0, 0)
(4, 1) (3, 1) (2, 1) (1, 1) (0, 1)
(4, 2) (3, 2) (2, 2) (1, 2) (0, 2)
(4, 3) (3, 3) (2, 3) (1, 3) (0, 3)
(4, 4) (3, 4) (2, 4) (1, 4) (0, 4)

問題 4 **((1),(2)各 5 点，(3),(4)各 10 点，計 30 点)**

月・日を表す数字が 2 組入力されると、それらの日付が年のはじめから何日目かと、2 つの日付の差が何日かを計算して出力するプログラムを作成したい。以下の仕様と実行例を満たすように、プログラムの(1)～(4)の空欄を埋めなさい。

【仕様】

- 1 月 1 日を 1 日目と数え、うるう年は考えない。
- 日付 1 組の入力毎に正しい日付なのかを chkdate 関数でチェックする。年始からの日数の計算は dayyear 関数を用いる。日付の入力や処理結果の出力は main で行なう。
- chkdate 関数は、月と日の値を引数にとり、正しい日付かどうか(月が 1～12 の範囲内かと、日がその月に存在する日か)を判定する。月を表す引数が正しくない場合は"正しくない月です"と、月は問題ないが日の引数がおかしい場合は"正しくない日です"と標準エラー出力に出力し、いずれも exit(1);で強制終了する。月日とも問題ない場合は何もせず、関数を抜ける。
- dayyear 関数は、月と日の値を引数にとり、その日付が年のはじめから何日目か計算して、それを戻り値にする。
- chkdate 関数と dayyear 関数の中では、各月の日数の情報を格納している配列 nday[]を使用すること。switch-case 文や if 文を使って、月毎に処理を場合分けするのは不可とする。

【実行例】

```
% ./a.out
月と日(1 つめ)を入力してください: 1 1
月と日(2 つめ)を入力してください: 2 10
1 月 1 日は 1 日目, 2 月 10 日は 41 日目, 差は 40 日
% ./a.out
月と日(1 つめ)を入力してください: 2 30
正しくない日です
% ./a.out
月と日(1 つめ)を入力してください: 12 1
月と日(2 つめ)を入力してください: 10 30
12 月 1 日は 335 日目, 10 月 30 日は 303 日目, 差は -32 日
%
```

【プログラム】

```
#include <stdio.h>
#include <stdlib.h>

void chkdate(int,int);
int dayyear(int,int);

main(){
  int m1,d1,m2,d2;  /* 入力日付用の変数 */
  int dy1,dy2;      /* 何日目かを調べた結果 */
```

(1) 入力を促し、入力されたら関数を使って日付をチェック

```
printf("月と日(1つめ)を入力してください:");
scanf("%d%d, &m1, &d1);
chkdata(m1,d1); printf("月と日(2つめ)略);
scanf("%d%d, &m2, &d2);
chkdata(m2,d2);
```

(2) 関数を使った日数計算と結果表示

```
  dy1 = dayyear(m1, d1);
  dy1 = dayyear(m2, d2);
  printf(略);
}
```

それぞれの関数の役割を[仕様]から読み取ること。

```
void chkdate(int mon, int day){
  int nday[12] =
    {31,28,31,30,31,30,31,31,30,31,30,31};
```

(3)

```
if(mon < 1 || mon > 12){
  fprintf(stderr, "正しくない月です\n");
  exit(1);
} else if(day < 1 || day > nday[mon-1]){
  fprintf(stderr, "正しくない日です\n");
  exit(1);
} else {
  return;
}
```

標準エラーは、 printf → fprintf, 第一引数に stderr を渡すだけなので、必ず覚えておくこと。printfとの差分で覚える。

```
}
int dayyear(int mon, int day){
  int nday[12] =
    {31,28,31,30,31,30,31,31,30,31,30,31};
```

(4)

```
int i;
for(i=1;i<mon;i++){
  day += nday[i-1];
}
return day;
```

or (好みは→)

```
int i, n = 0;
for(i=1;i<mon;i++){
  n += nday[i-1];
}
n += day;
return n;
```

```
}
```

月ごとの日の最大(最終日)は、ndayで与えられているので、1つ前の月まで求めて、その月の分は、与えられた日(day)を足すだけ。