C 演習 2017-05-22

練習問題

1. 2つの変数の値の入れ替え
   例: int 型の変数 x と y の値を入れ替える
   x = y ; y = x; ではできない
   それにはもう一つの変数を必要とする=> もう一つの変数を z とすると
   z = x; x = y; y = z;
   なぜこれでできるか、なぜこれでなければならないか、説明せよ

2. 以下に出てくるmax/min プログラムの変種
   配列と配列の要素数を引数とし、最大値となる要素の「インデックス」を返す関数を書け。
   例： 配列の要素が { 3, 5, 8 -1, 2}ならば、 2 を返す
   （3 番目の要素が最大で、インデックスとしては2だから）。

(1) MIN/MAX パタン

```
型 関数名(引数の並び) {
    // 引数には、配列とその要素数(int)
  // 変数宣言:答えを記憶する変数、繰り返しパラメタ
  // 答えを記憶する変数に初期値を与える、大抵は配列の先頭の要素
    // 繰り返し処理
    // return 答え;
}
```

例:
```
int max(int ar[], int n) {
  // 配列の型と関数の型は一致
    // n は配列の要素数なので int になる
   int ans, i; // ans は答えの候補、i は繰り返しのパラメタ
  ans = ar[0]; // ans に初期値を入れる
  for (i = 1; i < n; i++) {   // 配列の要素を繰り返しで調べる
      if ( ans < ar[i]) {      // 配列の要素と答え候補を比較
          ans = ar[i];      // 答えの候補を更新
      }
  } // ここまで繰り返し処理の内容
  return ans;   // 答えを返す
}
```

問題 1. 配列の2番目に大きな要素を求める second
ヒント: 答えの候補と、最大の要素の二つを記憶する必要がある
この2つの変数に適切に初期値を与えること
繰り返しにおいては、この二つを更新する
答えを返すのは、2番目に大きな要素だけ

問題 2. 目標値(引数で与えられる)に最も近い配列の要素を求める nearest
ヒント: 配列の要素と答えの候補の比較には、「目標値との差の自乗」を用いる
その値を記憶する変数 dif を用意し、また答えの候補が変わるたびにこの変数も更新するとよい

問題 3. 条件が複数ある例:配列の要素のうち、奇数で最大のものを求める oddMax
ヒント: 答えの候補の変数に適切な値が入っているかどうかを表す変数(フラグ、という)を用意し、
その初期値を True(C では「1」の値)としておく。
繰り返しの中では、配列の要素が条件を満たす(この問題では奇数)かどうか調べ、

フラグが True ならそれを答えの候補にし、フラグを False(C では「0」)にする、
フラグが False ならば、その要素と答えの候補と比較して、必要なら答えの候補の値を更新
ここで返す値に注意。「そのようなものがない」という場合もあることを考慮すること

プログラムの全貌

```
#include <stdio.h>
#define NUM 1000
#define FNAME "data.txt"

// second, nearest, oddMax 関数などをここにいれる

int max(int ar[], int n) {        //  配列の型と関数の型は一致
   // n  は配列の要素数なので  int  になる
   int ans, i; // ans は答えの候補、i は繰り返しのパラメタ
   ans = ar[0]; // ans に初期値を入れる
   for (i = 1; i < n; i++) {    //  配列の要素を繰り返しで調べる
         if ( ans < ar[i]) {        //  配列の要素と答え候補を比較
                  ans = ar[i];        //  答えの候補を更新
         }
   } //  ここまで繰り返し処理の内容
   return ans;     //  答えを返す
}

int main(void) {
      int n=0, array[NUM];
      FILE *fp;

      fp = fopen(FNAME, "r");
      while (fscanf(fp,"%d", &array[n]) != EOF) {
         printf("%d ",array[n]);
         n++;
      }
      fclose(fp);

      printf("\n------\n");
      printf("max = %d, second = %d\n", max(array, n), second(array, n));

      return 0;
}
```

(2) SUM パタン

```
型 関数名(引数の並び) {
    // 引数には、配列とその要素数(int)
  // 変数宣言:答えを記憶する変数、繰り返しパラメタ
  // 答えを記憶する変数に初期値を与える(大抵の場合 0)
    // 繰り返し処理
  // return 答え;
}
```

答えを記憶する変数(ans としておく)に、配列の要素から求まる値を繰り返しを用いて足していく

例: int sum(int ar[], int n) {
// 配列の型と関数の型は一致しないことがあるので注意
// n は配列の要素数なので int になる
int ans, i; // ans は答えの候補、i は繰り返しのパラメタ
ans =0; // ans に初期値を入れる
for (i = 0; i < n; i++) { // 配列の要素を繰り返しにより取りあげる
ans += ar[i];
} // ここまで繰り返し処理の内容
return ans; // 答えを返す
}

問題 4. 平均を求める関数 mean(今までにも議論した)
sum を利用する

問題 5. 標本分散を求める関数 var(今までにも議論した)
ヒント: 配列の要素の平均 mean を sum を元に求める。ans の初期値を 0 とする。
配列の要素と mean との差の自乗を ans に足していく
ans の値を要素数で割った値を返す(float, もしくは double とする)

問題 6. 標準偏差を求める関数 std(今までにも議論した)
var を利用する

問題 7. 区分求積法 integral
関数 f(x)の区間[a,b]の積分の近似値 S を求める

$$S = \int_a^b f(x)dx \approx \sum_{i=0}^{n-1} f(a + h * i) * h$$

ここでは、[a, b]を *n* 等分するとし、*h*=(b-a) /*n* とした
しかし、実際のプログラムでは、次に示すように *h* は刻み幅として、x を a から b まで変化させた繰り返しを行う方が良いだろう:

```
s = 0.0 ; h = 1.0/n;
for(x = a; x < b; x += h) {
        s += f(x)*h;      // x の値が a から h 刻みに b まで変化する
}
```

プログラムの全貌

```
#include <stdio.h>
#define NUM 1000
#define FNAME "data.txt"

// sum 関数などをここにいれる

int sum(int ar[], int n) {
   //  配列の型と関数の型は一致しないことがある
   // n  は配列の要素数なので  int  になる
   int ans, i; // ans は答えの候補、i は繰り返しのパラメタ
   ans =0; // ans に初期値を入れる
   for (i = 0; i < n; i++) {    //  配列の要素を繰り返しにより取りあげる
         ans += ar[i];
   } //  ここまで繰り返し処理の内容
   return ans;    //  答えを返す
}

int main(void) {
      int n=0, array[NUM], temp;
      FILE *fp;

      fp = fopen(FNAME, "r");
      while (fscanf(fp,"%d", &array[n]) != EOF) {
          printf("%d ",array[n]);
          n++;
      }
      fclose(fp);

      printf("\n------\n");
      temp = sum(array, n); //  総和
      printf("sum = %d, mean = %8.2f\n", temp, (float)temp/n );

      return 0;
}
```