パターン認識・課題0

パターン認識演習に必要なCプログラミングのために，簡単な演習問題を提示する．
ヒントはあるが，基本的には自分で調べること．
困ったときはこのサイト (http://www.google.co.jp/)が非常に役立つ．
なお，今回はヒントがたくさんあるが，今後はヒントはあまり出てこないので自分で調べられるようになっておくこと．

今回の課題は特に提出の必要はないが，いずれ演習で利用するものばかりである．

課題0.1 main関数への引数

実行時に引数を渡すプログラムを作成し，その引数を表示せよ．
実行結果は以下のようになる．

実行例

```
$ gcc -o pattern01 pattern01.c
$ ./pattern01 test
test
$
```

ヒント

```
#include<stdio.h>

int main(int argc, char* argv[]){
  /*
        argcには引数の数
        argvには引数が入る
  */
  char *value = ○○○○;
  printf("%s\n", value);
}
```

課題0.2 ファイルの読み込み

以下に示すような内容のファイルをnumber1.datという名前で作成し，
ファイルを指定して数字を読み込んで，縦に並べて表示するプログラムを作成せよ．

```
5 4 6 -8 1 2 9 31 -45 0
```

実行例

Yasutomo KAWANISHI

```
$ gcc -o pattern02 pattern02.c
$ ./pattern02 number1.dat
5
4
6
-8
1
2
9
31
-45
0
```

ヒント

```
#include<stdio.h>

int main(int argc, char* argv[]){
    char *fileName = /*最初の引数を代入する*/;

    FILE *fp = /*fopenを使って読み込みファイルポインタを取得*/;

    int num;
    int i;
    for(i = 0; i < 10; i++){
        /*変数numにファイルから読み込んだ数値を代入．fscanfを利用すると良い*/
        printf("%d\n", num);
    }
    fclose(fp); /* fopenしたファイルはfcloseすること */
}
```

課題0.3 別ファイルの読み込み

以下のような内容のファイルをnumber2.datという名前で作成し，先ほど作ったプログラムpattern02で読み込め．
ただし，コンパイルしてはいけない．

```
0 1 2 3 4 5 6 7 8 9
```

実行例

```
$ ./pattern02 number2.dat
0
1
2
3
4
5
6
7
8
9
```

課題0.4 配列への代入

number1.datを読み込んで，配列に格納せよ．
その後，配列の中から最大値と最小値を取り出して表示せよ．

実行例

```
$ gcc -o pattern04 pattern04.c
$ ./pattern04 number1.dat
max=31
min=-45
```

ヒント

```
  int num;
  for(i = 0; i < 10; i++){
        /*変数numにファイルから読み込んだ数値を代入．fscanfを利用する*/
        array[i] = num;
  }

  int max = /*適切な初期値*/;
  int min = /*適切な初期値*/;
  for(i = 0; i < 10; i++){
        /*最大値をmaxに代入*/;
        /*最小値をminに代入*/;
  }
  printf("max=%d\n", max);
  printf("min=%d\n", min);
```

ヒント2

とても大きな定数，小さな定数は**limits.h**で定義されています．

課題0.5 配列のかけ算

number1.datから読み込んだ配列をp_1, number2.datから読み込んだ配列をp_2としたとき，
二つの配列の積，すなわち

$$p_1 \cdot p_2^t$$

を求めるプログラムを作成せよ．

実行例

```
$ gcc -o pattern05 pattern05.c
$ ./pattern05 number1.dat number2.dat
-83
$
```

課題0.6 多次元配列の読み込み

ここから第2回演習の内容 以下に示すようなファイルを作成し，matrix.datという名前で保存せよ．
その後，このファイルを読み込み多次元配列に格納し，出力するプログラムを作成せよ．
ただし，matrix.datは，5x3の配列であり，その中身が次の行からの値であることを意味している．
この課題以降で，5x3以外のデータが入力されることもあるので，それに対応できるようにしておくこと．

```
5 3
1 0 0 0 1
0 1 0 1 0
1 1 1 1 1
```

ヒント: 可変長多重配列の作り方

5x3の配列を作成するには以下のようにする.

```
    int **data;
    int w = ?; // 5 をファイルから読み込み
    int h = ?; // 3 をファイルから読み込み

    data = (int**)malloc(h * sizeof(int*)); /* ポインタ配列を確保 */

    /* 配列の要素それぞれにつき、メモリ領域を確保 */
    for(i = 0; i < h; i++){
        data[i] = (int*)malloc(w * sizeof(int));
    }

    /* 値の読み込み */
    /* 例えば… */
    data[0][0] = 1; // 2次元配列の(0, 0)に1を代入
    data[4][2] = 1; // 死にます. 何故でしょう？

    /* 値の表示 */

    /* 使い終わったら必ずメモリは開放しましょう */
    for(i = 0; i < h; i++){
        free(data[i]);
    }
    free(data);
}
```

注意点

一旦ヒープに確保したメモリ領域は, 必ず解放すること.

課題0.7 構造体

課題0.6で作成したmatrix.datを読み込み，構造体に格納し, 出力するプログラムを作成せよ.

構造体の作り方

```
/*ここで構造体を作成*/
typedef struct {
  int **data; /*文字データそのもの*/
  int width; /*文字データの幅*/
  int height; /*文字データの高さ*/
} MojiData;

/*文字データを画面に出力する関数のプロトタイプ宣言*/
void printMojiData(MojiData *mojiData);

int main(int argc, char* argv[]){
  MojiData mojiData;


  mojiData.h = ?;
  mojiData.w = ?;

  /* 領域 mojiData.data の確保処理 */

  /*ここでデータをファイルから読み込み*/

  /* 画面に表示 */
  printMojiData(&mojiData);

  /* 領域 mojiData.data の解放処理 */

}

/*文字データ（ここでは行列）を画面に出力する関数*/
/* 注意：関数をここで宣言する場合は，main関数の前にプロトタイプ宣言が必要です．*/
/* プロトタイプ宣言をする場所はよく考えましょう．  */
void printMojiData(MojiData *mojiData){
  int i, j;
  for(i = 0; i < mojiData->height; i++){
        for(j = 0; j < mojiData->width; j++){
          printf("%d ", mojiData->data[i][j]);
        }
        printf("\n");
  }
}
```

課題0.8 構造体

課題0.7で作成した構造体について，領域確保，解放を行う処理を関数化せよ．

ヒント：関数の作り方

課題0.7の「文字データを画面に出力する関数」を参考に関数を作成する．
構造体の変数のように，中身が大きな変数は，値渡しをするとコピーに時間がかかる．
そのため，ポインタ渡しによって変数を関数に渡す．
このとき，アドレス演算子&（変数からポインタを得る）と間接演算子*（ポインタから変数を得る）の使い方に注意．

ヒント：関数の定義例

```
void data_alloc(MojiData *mdata){
        /* ここで mdata->data のメモリを確保 */

}

void data_free(MojiData *mdata){
        /* ここで mdata->data のメモリを解放 */

}
```

ヒント:関数の利用例

```
MojiData moji1;

/* wとhの読み込み */
moji1.w = 5;
moji1.h = 3;
data_alloc(&moji1);

/* dataの読み込み */


/* 解放 */
data_free(&moji);
```

課題0.9 足し算

2つの多次元配列の足し算を行う関数を作成せよ.
ただし,演算を行う多次元配列の大きさは同じであるとする.
また,各多次元配列のメモリは確保済みであるとする.

2つのデータmatrix1.dat, matrix2.datを読み込み,結果を足し算して表示せよ.

ヒント:足し算

作成する関数は,c=a+bの演算であり,a, b, cを引数にとるものとする.

```
void data_add(MojiData *mdata_1, MojiData *mdata_2, MojiData *mdata_out){
    int i, j;
        for(i = 0; i < mdata_1->h; i++){
                for(j = 0; j < mdata_1->w; j++){
            /* mdata_1の要素とmdata_2の要素を足してmdata_outに格納する */

        }
    }
}

int main(int argc, char *argv[]){
    MojiData x1, x2, x3;

    /* メモリの確保 */
    x1.h = ? // matri1.dat から読み込む
    x1.w = ? // matri1.dat から読み込む
    x2.h = ? // matri2.dat から読み込む
    x2.w = ? // matri2.dat から読み込む
    x3.h = x1.h;
    x3.w = x1.w;
    data_alloc(&x1);
    data_alloc(&x2);
    data_alloc(&x3);

    /* 値の読み込み */


    /* 値の表示 */
    printMojiData(&x1);
    printMojiData(&x2);

    /* 足し算 */
    data_add(&x1, &x2, &x3);

    /* 値の表示 */
    printMojiData(&x3);

    /* メモリの解放 */
    data_free(&x1);
    data_free(&x2);
    data_free(&x3);

    return 0;
}
```

戻る