ポインタと配列その２（数学講読）

実習で利用するサンプルプログラム

prog6-1.c (メモリの動的確保)

```
/* メモリの動的確保について
   ポインタは単なるアドレスだけを格納する変数なので,
    実数の値を格納するメモリは別に存在しなければならない.
    また, 実数の配列を使う場合, その要素がいくつ必要か
    というのはプログラム中で決定することも多々ある.
     そのような場合に対応するのがメモリの動的確保である.
*/
#include <stdio.h>
#include <stdlib.h>
int main( int argc, char *argv[] )
{
  int i;
  int n;
  double *pa;
/*
   pa は実数変数へのポインタ (アドレス).
   この時点では配列の要素数は明確でないとする.
*/
/* ここで配列の要素数が決定された */
  n = 10;
/*
     malloc 関数は memory allocation の意味,
     double の大きさのデータを n 個分確保して,
     配列としその先頭アドレスを返す.
*/
```

```
  pa = (double*)malloc( n*sizeof(double) );
/*
     メモリ確保に失敗したらNULLという結果が
     帰ってくるので，以後の処理はできず強制終了させる．
*/
  if( pa == NULL )    exit(1);
/* 一旦メモリを確保したら，あとは通常の配列として使用できる */
  for(i=0;i<n;i++){
    pa[i] = (double)i;
    printf( "pa[%d] = %lf\n", i, pa[i] );
  }
/*自分でmallocを使って確保した関数はいらなくなったらfreeで処分する*/
  free(pa);
  exit(0);
}
```

prog6-2.c （ポインタの配列と二重配列）

```
/* 二重配列とポインタの配列 */
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[] )
{
  int i,j;
/* 実数の二重配列 */
  double a[5][5];
/* 実数へのポインタの配列 */
  double *b[5];
/* 実数へのポインタのポインタ（ダブルポインタ） */
  double **c;
/* とりあえず二重配列 a[5][5] のデータの初期化*/
  for(i=0;i<5;i++){
    for(j=0;j<5;j++){
      a[i][j]=(double)(i+1)*(j+1);
    }
  }
```

```
/*
   実数へのポインタの配列 *b[5] の各要素 b[i]
   それぞれが実数へのポインタ．また二重配列の
   a[i] と書けばこれもポインタ．よって次の記法が可能
*/
  for(i=0;i<5;i++){
    b[i] = a[i];
  }
/* この時 b[0] は a[0][0] から a[0][4] までの値を参照するために使え
る */
  for(j=0;j<5;j++){
    printf( "b[0][%d]=%lf\n", j, b[0][j] );
  }
/* ポインタの通常の演算も使える */
  for(j=0;j<5;j++){
    printf( "b[1][%d]=%lf\n", j, *(b[1]+j) );
  }
/*
   ポインタの配列 b[10] の b はポインタのポインタを意味するので
   以下のような代入が可能
*/
  c = b;
  for(i=0;i<5;i++){
    for(j=0;j<5;j++){
      printf( "c[%d][%d]=%lf\n", i, j, c[i][j] );
    }
  }
  exit(0);
}
```

prog6-3.c （文字列の二重配列）

```
/* main 関数の*argv[]：
   ポインタの配列を理解すると main 関数に現れる
   *argv[] の意味がわかる
*/
#include <stdio.h>
```

```
int main( int argc, char *argv[] )
{
  int i;
/* char の配列＝文字列, char の二重配列 ＝ 文字列の配列 */
  char list[12][15] = {"January", "February", "March",
                       "April", "May", "June", "July",
                       "August", "September", "October",
                        "November", "December" };
/* char のポインタの配列 = 文字列 (へのポインタ) の配列 */
  char *season[12];
/* ポインタ配列の各成分 season[i] は文字列の先頭アドレスになる */
  for(i=0;i<12;i++){
    season[i] = list[i];
  }
  for(i=0;i<12;i++){
    printf( "%d 月は%s です。\n", i+1, season[i] );
  }
/*
   これを理解すると main 関数の *argv[] は文字列配列を表す
   ことがわかる．コマンドラインに入力される文字を格納する
   ために用いるものである
*/
  printf( "コマンドラインに入力された文字数は%d 文字\n", argc );
  for(i=0;i<argc;i++){
    printf( "%d 番目は%s\n", i, argv[i] );
  }
  exit(0);
}
```