数理解析学演習・補足

[1-i] 次のプログラムを作成し，実行結果をファイルに保存せよ.

〈 address.c 〉

```
#include <stdio.h>

main()
{
     float a[5][5];
     int i, j;

     for(i=1;i<=5;i++){
          for(j=1;j<=5;j++){
               printf("&a[%d][%d] = %d\n", i, j, &a[i-1][j-1]);
          }
          printf("\n");
     }

     return 0;
}
```

実行型ファイル名を address とすると，実行結果をファイル (ファイル名 add_rslt.txt) に保存するには,

```
~$ ./address > add_rslt.txt
```

とする (入門 UNIX」5.1.1 参照. printf の標準出力 (モニター) が，ファイル add_rslt.txt にリダイレクションされる).

また，出力されたファイル add_rslt.txt の中身を more コマンドで閲覧し，配列のアドレスが 4 (float 型変数の格納区画の必要バイト数) ずつ増えていることを確認せよ (入門 UNIX」5.2.1 参照).

[2-i] 次のプログラムを作成・実行し，構造体の内部変数の入出力を理解せよ.

〈 kouzou.c 〉

```
#include <stdio.h>

struct Kouzou{
     float x, y;
     int m, n;
     char a;
};
```

```
main()
{
     struct Kouzou k;

     k.x = 3.5;
     k.y = 4.02;
     k.m = 34;
     k.n = -8;
     k.a = 'A';

     printf("k.x = %f, k.y = %f, k.m = %d, k.n = %d, k.a = %c\n", k.x, k.y, k.m,
                                                                    k.n, k.a);

     return 0;
}
```

構造体の内部変数を操作するには,

構造体の変数名.構造体の定義の内部変数名　(例：k.x)

と言う“リモコン”を使用する..

[2-ii] 行列を扱う構造体の入力・出力を理解する為に, 次のプログラムを作成・実行せよ.

⟨ mtx_str.c ⟩

```
#include <stdio.h>

struct Matrix{
     float a[10][10];
     int   dim;
     char  m_name, e_name;
};

main()
{
     struct Matrix x;
     int i, j;

     x.dim = 3;
     x.m_name = 'A';
     x.e_name = 'a';

     for(i=1;i<=x.dim;i++){
```

```
        for(j=1;j<=x.dim;j++){
            printf("%c[%d][%d] = ", x.e_name, i, j);
            scanf("%f", &x.a[i-1][j-1]);
        }
    }

    printf("%c = \n", x.m_name);
    for(i=1;i<=x.dim;i++){
        for(j=1;j<=x.dim;j++){
            printf("%f", x.a[i-1][j-1]);
            if(j != x.dim){
                printf("\t");
            }
            else{
                printf("\n");
            }
        }
    }

    return 0;
}
```