プログラミング基礎

構造体と動的配列

課題③

```
#include <stdio.h>

struct planets{
    char name[64];
    long long diameter;
    long long distance;
};
```

値が大きい可能性あり

構造体の定義

課題③

```
int main(){
    struct planets a[1000];
    int i,j;
    FILE* f;

    f = fopen("planetsdata.dat","r");
    if(f == NULL){
        printf("can't open the file");
        return -1;
    }

    i = 0;
    while(fscanf(f, "%s %lld %lld", &(a[i].name), &(a[i].diameter), &(a[i].distance)) != EOF){
        i++;
    }
    fclose(f);

    for(j=0; j<i; j++){
        printf("%s  %lld  %lld¥n", a[j].name, a[j].diameter,  a[j].distance);
    }
    return 0;
}
```

構造体配列の宣言
planets型の構造体を1000個作る

ファイルを開く
もし，ファイルが開けなければ，強制終了．

EOFが出るまでファイルのデータを読み込んでいく
構造体のメンバには” 構造体．メンバ ”でアクセス

ファイルを閉じる

⑥ 行列とベクトルの掛け算 Ax = y （動的配列の利用例）～ポインタを宣言～

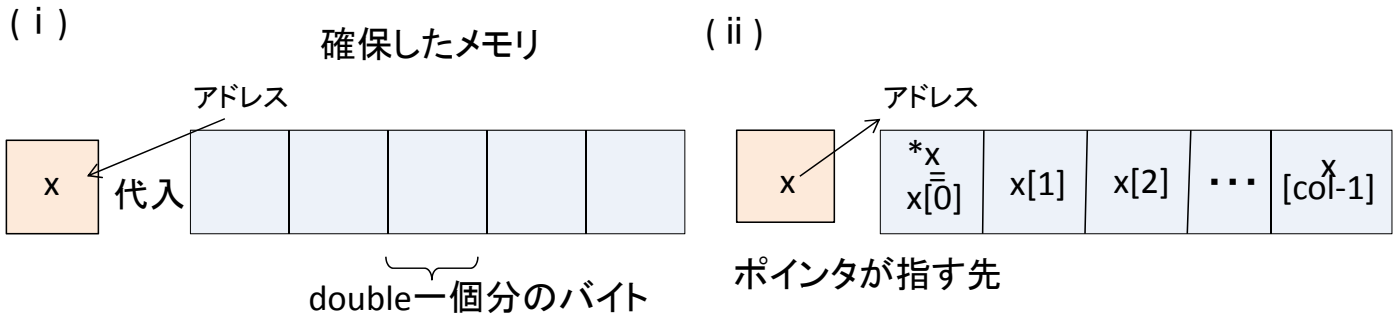
Aの行数列数をコンパイル後に決めたい → 動的配列が必要

```
int row, col;//行，列
double **A, *x, *y;
int i,j;

printf("行数を入力してください¥n");
scanf("%d", &row);

printf("列数を入力してください¥n");
scanf("%d", &col);
```

ポインタを宣言．
2次元配列が取りたければダブルポインタ．

⑥ 行列とベクトルの掛け算 Ax = y（動的配列の利用例）～一次元動的配列～

```
x = (double*)malloc( col *sizeof(double) );
y = (double*)malloc( row*sizeof(double) );
```



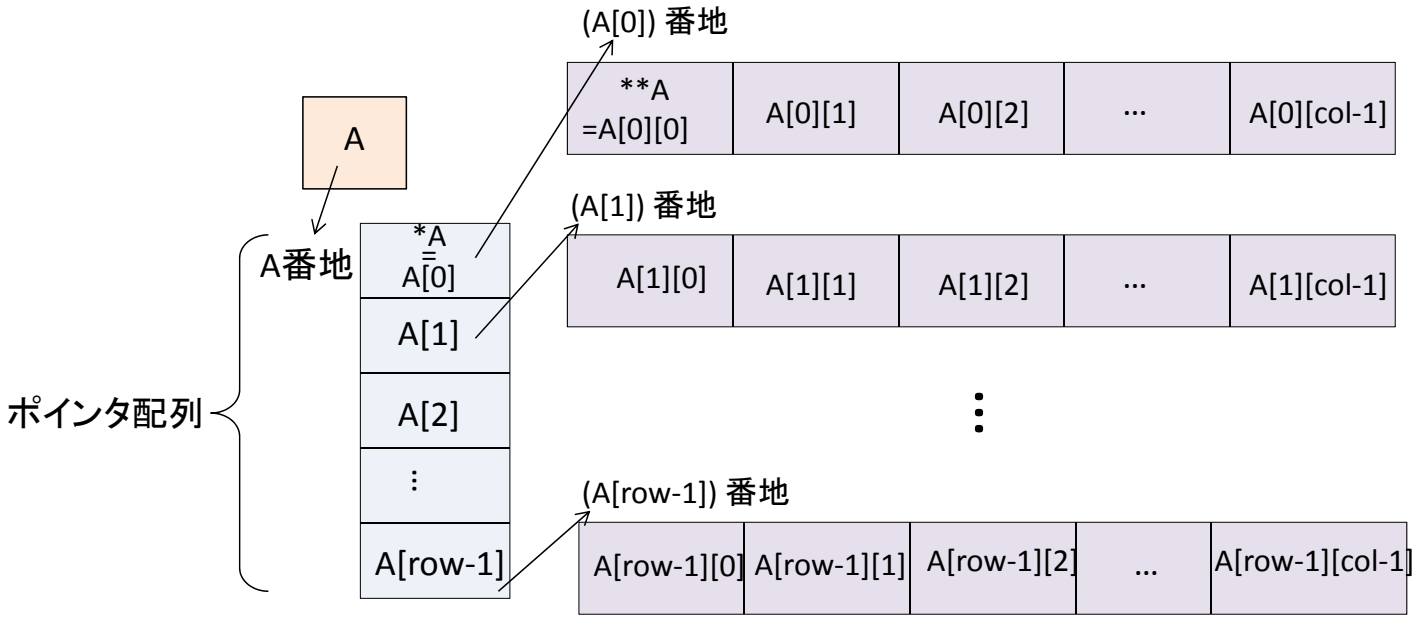
ｉ） x用のメモリを，（行数）×（double一個に必要なバイト数） 確保し，そのメモリブロックの先頭へのポインタをx に代入．

ii） xが指すのはx[0]のアドレスで，確保したメモリが連続であることから，x[0]，x[1]，… ,x[col-1] とアクセスすることができる．
(*x = x[0])

⑥ 行列とベクトルの掛け算 Ax = y（動的配列の利用例）～二次元動的配列～

```
A = (double**) malloc( row * sizeof(double*) );
for(i=0; i<row; i++)
    A[i] = (double*) malloc( col * sizeof(double) );
```

（行数）×（double へのポインタ一個に必要なバイト数）のメモリを確保．その先頭アドレスをAに代入



⑥ 行列とベクトルの掛け算 Ax = y（動的配列の利用例）～初期化と計算～

配列の初期化

```
for(i=0; i<col; i++)
    x[i] = 0.0;

for(i=0; i<row; i++)
    y[i] = 0.0;

for(i=0; i<row; i++)
    for(j=0; j<col; j++)
        A[i][j] =0.0;
```

要素の入力

```
printf("要素を入力してください¥n");
for(i=0; i<row; i++)
    for(j=0; j<col; j++)
        scanf("%lf", &A[i][j]);

printf("ベクトルを入力してください¥n");
for(i=0; i<col; i++)
    scanf("%lf",&x[i]);
```

行列の計算

```
for(i=0; i<row; i++)
    for(j=0; j<col; j++)
        y[i] += A[i][j] * x[j];
```

⑥ 行列とベクトルの掛け算 Ax = y（動的配列の利用例）～メモリ解放～

```
for(i=0; i<row; i++)
    printf("%lf¥n",y[i]);
```

結果の出力

```
free(x);
free(y);

for(i=0; i<row; i++)
    free(A[i]);

free(A);
```

メモリの解放

free(ポインタ) で解放.
多次元の場合は，後で確保した領域から解放.

⑥ 行列とベクトルの掛け算 Ax = y（動的配列の利用例）

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

int main(){
        int row, col;
        double **A, *x, *y;
        int i,j;

        printf("行数を入力してください¥n");
        scanf("%d", &row);

        printf("列数を入力してください¥n");
        scanf("%d", &col);

        x = (double*)malloc( col * sizeof(double) );
        y = (double*)malloc( row * sizeof(double) );

        A = (double**) malloc( row * sizeof(double*) );
        for(i=0; i<row; i++)
                A[i] = (double*) malloc( col * sizeof(double) );
```

```
for(i=0; i<col; i++)
        x[i] = 0.0;

for(i=0; i<row; i++)
        y[i] = 0.0;

for(i=0; i<row; i++)
        for(j=0; j<col; j++)
                A[i][j] =0.0;

printf("要素を入力してください¥n");
for(i=0; i<row; i++)
        for(j=0; j<col; j++)
                scanf("%lf", &A[i][j]);

printf("ベクトルを入力してください¥n");
for(i=0; i<col; i++)
        scanf("%lf",&x[i]);
```

```
        for(i=0; i<row; i++)
                for(j=0; j<col; j++)
                        y[i] += A[i][j] * x[j];

        for(i=0; i<row; i++)
                printf("%lf¥n",y[i]);

        free(x);
        free(y);

        for(i=0; i<row; i++)
                free(A[i]);

        free(A);

        return 0;
}
```