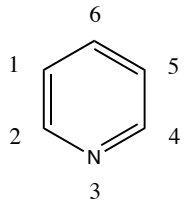
2022年 6月 21 日

分子構造データ

必要なファイルを git clone https://github.com/NGS-maps/PracticeMol からダウンロードしておく。molフォーマットと言うファイル形式ではたとえば右図のようなピリジンの分子構造を以下のように記述する（ここでは水素原子は省略されている）



pyridine.mol

```
------------------------------------------------------------
6 6 0 0 0 0 0 0 0 0999 V2000
-0.6135  0.5570 0.0000 C 0 0 0 0 0 0 0 0 0 0 0 0
-0.6154 -0.7920 0.0000 C 0 0 0 0 0 0 0 0 0 0 0 0
 0.5496 -1.4645 0.0000 N 0 0 0 0 0 0 0 0 0 0 0 0
 1.7175 -0.7909 0.0000 C 0 0 0 0 0 0 0 0 0 0 0 0
 1.7128  0.5628 0.0000 C 0 0 0 0 0 0 0 0 0 0 0 0
 0.5467  1.2298 0.0000 C 0 0 0 0 0 0 0 0 0 0 0 0
1 2 4 0 0 0 0
3 4 4 0 0 0 0
4 5 4 0 0 0 0
2 3 4 0 0 0 0
5 6 4 0 0 0 0
6 1 4 0 0 0 0
------------------------------------------------------------
```

ピリジンはC5N1H5で、右側の図の様な６角形の分子である。このデータでは炭素についた５個の水素原子が省略されている。molファイル一行目の最初の６は原子数、二つ目の６は結合数を表す。３カラム目以降のエントリーは今回の実習では使わないので無視して構わない

その下の６行（原子情報行）が各原子についての情報で、最初の３つの実数は `x, y, z` （図では横軸がx、縦軸がy 面外にz）それぞれの座標、次の `C, N` などの文字が 炭素(C) 窒素（N）酸素(O) 水素(H) などの元素の種類を表す。４つ目以降のエントリーは無視して構わない。（ピリジンは真っ平らなので、全原子がxy平面に収まるためzの値が全て0.0になっている）

続く６行（結合情報行）は結合表で、その一行目の１２４では、１番目と２番目の原子がタイプ４の結合で結ばれていること、二行目以降は、`3-4, 4-5, 2-3, 5-6, 6-1` の原子間の結合が存在することを表している。三番目の数字（ここでは全て４）は結合次数(bond order)であり一重結合が１、二重結合２、三重結合が３で、４はベンゼン環の共役結合を表している。今回は結合字数は特に必要としないので無視しても可。このようなファイルを読み込んで、データを構造体の配列に格納し、計算を行うプログラムを作ることを考える。原子情報（atom_info）の構造体は読み込みデータとして、x,y,zのそれぞれの座標を実数型として持ち、元素記号（`Fe, He` 等二文字までのものがあるが今回は一文字のみとして可）を文字型として持つ。結合情報（bont_info）の構造体は読み込みデータとして、左側の２つのカラム、何番の原子から(from)、何番の原子への（to）結合であるかという２つの整数値を保持する。

STEP 0　ヘッダファイル

上記のような原子情報（元素記号（今日の課題では1文字とみなして良い）とx,y,z 座標）、結合情報（両端の原子のid番号）を記述する為の構造体、atom_infoとbond_infoの定義を含むヘッダファイル、**mol.h**を作成せよ。

STEP 1　分子構造ファイルの読み込み

STEP１のヘッダファイルを使い、molフォーマットのファイルを先に定義した構造体の配列に読み込み、それぞれの元素の数を数えて**組成式**として記すプログラム **mol.c** を作成せよ。ただし、今日の演習では、元素の種類としては `C, H, N, O, P, S` のみを考慮し、この順番に記す。存在しない元素は表記しない。上のファイルの例では、`mol pyridine.mol` と実行したときに標準出力に

C5N

と出力すれば良い。

（注：実際にはピリジンは表記されていない５個の水素を持つのでC5H5Nだが本演習ではC5Nで良い）
ヒント１：原子数、結合数は一行目に記述されているので、一行目を読み込んだ後で、atom_info, bond_infoの構造体について配列をmallocする。（今回のプログラムではrewindは必要ない）
ヒント２： pyridine.mol以外の幾つかの .molファイルのサンプルを置くので、**作成したプログラムがすべてのmolファイルで動作す**ることを確かめることこのステップでは結合情報や原子の座標情報は用いないが、後のステップのためにファイルを読み込むプロセスではすべての情報を格納しておく

STEP 2　関数

STEP1の出力に加えて、それぞれの結合情報について、結合している二つの原子間の距離を計算し、出力する（ピリジンなら６本の結合距離）プログラム **mol2.c** を作成せよ。
原子間距離を計算する部分をメイン関数から分離した関数として、原子情報を指すポインタと、距離を計算する二つの原子の番号を渡せば、距離を計算して返す関数 calc_distance を作り、これを用いたプログラムとせよ。
二つの原子のx,y,z座標をそれぞれ、x1,y1,z1、x2,y2,z2とすると、
原子間の距離は sqrt((x1-x2)^2 + (y1-y2)^2 + (z1-z2)^2)
で求められる。molフォーマットの結合情報の行の右側に、距離を加えた物を出力ファイルとする。
出力例

```
mol pyridine.mol
Number of atoms is 6
Number of bonds is 6
C5N
1. 1 2 1.3487
2. 3 4 1.3483
3. 4 5 1.3537
4. 2 3 1.3454
5. 5 6 1.3434
6. 6 1 1.3412
```

STEP 3 分割コンパイル

先に定義した原子間距離の計算をするファンクション　calc_distance()ををmain関数を含むファイル(mol3.c)とは別のファイル（calc_distance.c）に記述し、別々にコンパイルを行ったのちに、リンクするという手順でプログラムを完成する。Cのソースコードを含む.cファイルを **-c** オプションをつけてコンパイルすると、.o のsuffixを持つオブジェクトファイルが生成する。これら複数の.oファイルを引数としてgccを実行すると、これらのファイルのコードとライブラリファンクションをまとめた実行形式ファイルが作成される。この手続きをリンクと呼ぶ。

距離計算をおこなう関数を分けるときには、main関数の前、include文の後に　以下のような距離計算関数を定義する。**日本語太字で記述した部分はcのコードに直すこと**

```
-----------------------------------------------------------------
float calc_distance(int from, int to, atom_info *atoms)
{
float diff_x, diff_y, diff_z, distance;

diff_x = from番目の原子のx座標とto番目の原子のx座標の差を計算;
diff_y = from番目の原子のy座標とto番目の原子のy座標の差を計算;
diff_z = from番目の原子のz座標とto番目の原子のz座標の差を計算;
distance = sqrt(diff_x * diff_x + diff_y * diff_y + diff_z * diff_z);

return distance;
}
-----------------------------------------------------------------
```

sqrtを用いているので、#include <math.h>を加え、コンパイル時に -lm オプションを最後につける。

main関数の末尾に、距離のリストを出力する以下のような部分を加える、

```
-----------------------------------------------------------------
float distance;
for(i=0;i<結合の数;i++)
 {
 distance = calc_distance(添字のリスト);
 printf("%3d %10.4f¥n",i+1,distance);
 }
-----------------------------------------------------------------
```

（注１）結合距離は、1.0~1.6程度の値で、2.0を超えるような数値が出力される場合はどこかが間違っているので修正すること．ただしpyridine.molのデータでは結合距離が0.8前後と短く出る（データファイルが正しい座標になっていないため）

関数定義部分を、別のファイル、calc_distance.cに移し、main**関数を含む、mol2.cの中のcalc_distance関数の定義は削除する**

関数定義ファイル　calc_distance.c

```
-----------------------------------------------------------------
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include "mol.h"

float calc_distance(int from, int to, atom_info *atoms)
{
float diff_x, diff_y, diff_z, distance;

diff_x = from番目の原子のx座標とto番目の原子のx座標の差を計算;
diff_y = from番目の原子のy座標とto番目の原子のy座標の差を計算;
diff_z = from番目の原子のz座標とto番目の原子のz座標の差を計算;
distance = sqrt(diff_x * diff_x + diff_y * diff_y + diff_z * diff_z);

return distance;
}
-----------------------------------------------------------------
```

定義した関数 calc_distanceをmain関数の中で用いるためには、以下のようなプロトタイプ宣言が必要となる。

```
-----------------------------------------------------------------
float calc_distance(int from, int to, atom_info *atoms);
-----------------------------------------------------------------
```

molread.c の中の main関数の前にこの一文を置いても良いが、より一般的には、**ヘッダファイル、molread.hの末尾にこの一文を加える**

STEP 4 make

molread.c, molread.h, calc_distance.c　の準備ができたら、makefileという名前のファイルを作成する。ソースファイルの数が多くなると、-cオプションによるコンパイルやリンクを実行しなければならない回数が多くなるため、煩雑なコンパイル手続きを簡便化するためにmakeを用いる。.molファイルが、幾つか準備されているので、これらのファイルをコピーして来て、すべてのデータでの実行を確認すること。

makefile

```
-----------------------------------------------------------------
molread : molread.o calc_distance.o
          gcc -o molread molread.o calc_distance.o -lm

 molread.o : molread.c mol.h
          gcc -c molread.c

 calc_distance.o : calc_distance.c mol.h
          gcc -c calc_distance.c

clean :
          /bin/rm -f core molread *.o
-----------------------------------------------------------------
```

gcc, /bin/rm の前の余白、影を付けた部分　　4カ所はスペースではなく <TAB> が一つはいっているので注意

以上の準備ができたら、**make** とタイプすればすべてのソースコードのコンパイルとリンクが一度に行われる。

makeが見つからない場合には

```
sudo apt update
sudo apt install make
```

makefile の記述は、それぞれのファイルよりも、指定したファイルの方が新しければ、二行目のTab以下の文を実行するという形式をとっている。
たとえば、

```
molread.o : molread.c mol.h
         gcc -c molread.c
```

では、molread.oが、存在しないか、molread.c または mol.hより古ければ、gcc -c molread.c により、新しいmolread.oを作成する、
また、

```
molread : molread.o calc_distance.o
          gcc -o molread molread.o calc_distance.o -lm
```

では、molreadが存在しないか、molread.oまたはcalc_distance.oより古ければ、gcc -o molread molread.o calc_distance.o -lm　を実行して、新しい molread を作成する。

最後の、cleanを定義することにより、”make clean” を実行すると、coreファイルなどの不要なファイルと、先に作成して実行形式ファイルmolread, および全てのオブジェクト(molread.o)が消去され、その後に”make”を実行することで、全てのコンパイル出力が現在あるソースに依存しているということを明示的に行うことができる。

プログラムの動作を確認して、**012yamada11mol3**（自分の学生番号下３けたと名前で置き換え）と言う名のディレクトリに、**mol3.c, mol.h, calc_distance.c, makefile の４つ**を入れ、tar,gzipにより圧縮し、**012yamada11mol3.tar.gz** という圧縮ファイルを添付して山田**11**というタイトルをつけてメールで提出する。
ここまでが基本課題（提出期限・日曜日まで）

応用課題(来週の水曜まで)
分子構造中２本の結合に挟まれた角度（結合角）を全て列挙し、３つの原子の番号で定義できるそれぞれの角度を0~180度の範囲で計算するcalc_angleを定義し、計算した結合角を出力するプログラムを作成しなさい。ピリジンの場合1-2-3,2-3-4,3-4-5,4-5-6,5-6-1,6-1-2の６つの角度（それぞれおよそ120度）を出力する。他のmolファイルでも動作を確認し **012yamada11angle** というディレクトリ名でtar, gzをして添付する