Auto-MM

Auto-MM (Automatic Molecular Modeling) 是一个用来快速构建分子模型的小软件。

1 编译和运行

```
cd auto-mm
cmake CMakeLists.txt
make
./auto-mm < examples/cnt.build
```

在bin文件夹中已存放两个静态编译的版本

```
auto-mm        Linux版本
auto-mm.exe    Windows版本
```

可直接运行

- Linux bash

```
cd auto-mm
cp bin/auto-mm ./
chmod +x ./auto-mm
./auto-mm < examples/cnt.build
```

- Windows powershell

```
cd auto-mm
cp .\bin\auto-mm.exe .\
Get-Content .\examples\cnt.build | .\auto-mm.exe
```

- Windows cmd

```
cd auto-mm
cp .\bin\auto-mm.exe .\
.\auto-mm.exe < .\examples\cnt.build
```

2 输入文件

在 `examples` 文件夹中提供了部分输入文件，下面说明输入文件的构成。

2.1 程序逻辑

在程序中存在两个抽象的三维空间，*fragment* 空间和 *modeling* 空间。

- 当输入建模指令时，*fragment* 空间会被清空，并在其中按要求构建出相应的结构。例如，当输入 `nanotube(0, 6, 50.0)` 时，*fragment* 空间会被清空，并会在其中构建出一根手性指数为 $n=0$ 、$m=6$ ，并且长度为 $50.0A$ 的碳纳米管。
- 接下来可以同构位置控制命令改变 *fragment* 空间中的结构。例如 `move` 和 `mirror` 等命令。
- 调整好 *fragment* 空间中的结构的位置后，可以使用 `add` 命令将 *fragment* 空间中的结构添加到 *modeling* 空间中。
- 参考以上的操作，可在 *modeling* 空间中多次添加不同的结构。
- 当建模完成，*modeling* 空间中的结构即为所需模型。此时可通过 `create` 命令将 *modeling* 空间中的结构输出为结构文件。例如，输入 `create cnt.data` 就可将 *modeling* 空间中的结构写入文件名为 `cnt.data` 的文件中。

2.2 输入格式

2.2.1 定义变量

在此程序的输入中可以定义变量，在输入的任何位置都可以定义变量。完成定义之后，直接使用变量名就可以使用变量。变量有整数型变量 `intg` 和实数型变量 `real` 的分别。变量名可以以字母或下划线开头，后面可以加字母数字下划线若干。

- 整数型变量定义以 `intg` 标识。例如，`intg a = 10` 定义了一个变量名为 `a` 的整数型变量，变量值为 10。
- 实数型变量定义以 `real` 标识。例如，`real a = 5.0` 定义了一个变量名为 `a` 的整数型变量，变量值为 5.0。

2.2.2 调用函数

- 此程序提供若干随机函数，用于生成随机数。例如，`randintg(1, 100)` 就产生一个在 $[1,100]$ 的随机数。
- 此程序提供了指数运算函数 `powintg` 和 `powreal`。例如，`powreal(10, 2)` 将返回一个数值为 100.0 的实数。

2.2.3 定义宏

此程序可定义宏，宏的名称以 `macro` 标识，后面加若干字母或数字，或者字母数字的组合，区分大小写。完成定义之后，直接使用宏名称就可以使用宏。宏与变量有本质的区别，变量时一个数值，而宏是一个字符串，使用宏的时候就是做了一次字符串替换。例如

- `intg randnum = andintg(1, 100)` 定义了一个变量，变量的值为一个随机数。虽然定义时 `randnum` 的取值为一个随机数，但是完成定义后，`randnum` 的数值确定，因此每次使用这个变量时，得到相同的值。
- `macro randnum = randintg(1, 100)` 定义了一个宏，宏的值为一个字符串 `randintg(1, 100)`。每次使用 `randnum` 这个宏时，就等价于在此位置放置了一个 `randintg(1, 100)` 的字符串，于是，每次使用 `randnum` 这个宏时，都会得到一个随机数，数值与其他使用此宏的位置不同。

2.2.4 注释的书写

任何一行，从字符 `!` 以后的所有内容将被视为注释，对程序执行的结果没有任何影响。

2.3 建模命令

使用建模命令可以在 $fragment$ 空间构建相应的模型结构。

2.3.1 碳纳米管

使用命令 `nantube` 可以构建一个碳纳米管。需要向此命令提供三个参数，手性指数 `n` 和 `m`，以及长度 `l`。例如，输入 `nanotube(0, 6, 50.0)` 时，$fragment$ 空间会被清空，并会在其中构建出一根手性指数为 $n=0$ 、$m=6$ ，并且长度为 $50.0A$ 的碳纳米管。

使用 `nanotube` 命令构建的碳纳米管，其初始管轴沿 z 轴，且位于 z 轴正半轴，碳管最低处位于 xy 平面。

2.4 位置调整命令

位置调整命令用于调整 $fragment$ 空间中结构的位置。

2.4.1 `move`

`move` 命令用于平移 $fragment$ 空间中的结构。需要三个参数 `Vx` 、`Vy` 和 `Vz` ，分别是平移矢量在三个坐标轴上的分量。例如，输入 `move(0, 0, 10.0)` 命令将会使 $fragment$ 空间中的结构向 z 轴正方向平移 $10.0A$。

2.4.2 `mirror`

`mirror` 命令用于将 $fragment$ 空间中的结构做镜面对称。需要四个参数 `a` 、`b` 、`c` 和 `d`。输入 `mirror(a, b, c, d)` 命令，$fragment$ 空间中的结构会关于平面 $ax+by+cz+d=0$ 做镜面对称。

2.5 `add` 命令

使用 `add` 命令可将 $fragment$ 空间中存在的结构添加到 $modeling$ 空间中。此命令不需要任何参数。

2.6 `create` 命令

使用 `create` 命令可将 $modeling$ 空间中的结构写到文件中，需要向此命令提供文件名参数。例如，输入 `create cnt.data`，$modeling$ 空间中的结构将会被写入到 `cnt.data` 文件中。