atomicpp.h

Es un *header* para C++ que está pensado para facilitar el acoplamiento de algoritmos de optmización estructural atómica y molecular con códigos como VASP o Quantum Espresso.

Se declara la clase Atom, un átomo tiene un símbolo atómico (*string*), tiene una posición en el espacio (vector de tamaño 3, una dimensión) y además tiene un radio atómico, de esta forma el código modela al átomo como una esfera y no como un simple punto sin dimensiones, lo que es más realista y necesario para algunas aproximaciones. Atom() es el constructor por defecto. El método read_Atom actualiza los valores del objeto, está pensado para trabajar con el constructor de la clase Atomic_Structure (ver abajo).

```
class Atom
{
   public:
     string Symbol;
     double x[3];
     double R;
     Atom();
     void read_Atom(string, float, float, float);
};
```

Cuando se juntan los átomos se pueden formar estructuras atómicas como moléculas, cúmulos atómicos (clusters) o cristales. Se implementa la clase Atomic_Structure como el caso más general, una estructura atómica tiene un número de átomos que la conforman y un conjunto de átomos, que en la clase se escribe como un puntero para poder usar un arreglo de memoria dinámica en el constructor. Mediante herencia se crean las clases hijas Molecule, Crystal y Cluster.
A continuación se presentan únicamente las definiciones de las clases mencionadas, el resto del código (constructores y métodos implementados) se encuentran en Github: https://github.com/J-Fabila/Librarys/blob/atomic.h/atomicpp.h

```
class Atomic_Structure{
     public:
        int Nat;
        Atom *atom;
        Atomic_Structure(string);          //Constructor desde un archivo *.xyz
        Atomic_Structure();                //Constructor por defecto
        ~Atomic_Structure();               //Destructor de la clase
        void read_xyz(string file);        //Lee la estructura desde archivo *.xyz
```

```
        void read_VASP(string file);       //Lee la estructura desde archivo *.vasp
        void print_xyz(string);            //Crea un archivo con la información
                                          de la estructura en un archivo *.xyz
        void print_VASP(string, string, float, float (*)[3]);
                                           //Crea un archivo con la información
                                             de la estructura en un archivo *.vasp
        double x_min(); double x_max();    //Devuelve el valor mínimo/máximo
                                             en x de la estructura
        double y_min(); double y_max();    //Devuelve el valor mínimo/máximo
                                             en y de la estructura
        double z_min(); double z_max();    //Devuelve el valor mínimo/máximo
                                             en z de la estructura
        bool fit_in(float, float, float, float, float, float);
                                           //Devuelve 'True' o 'False' si la
                                             molécula cabe o no en el rango especificado
};



class Cluster : public Atomic_Structure{
     public:
        string type;
        double quirality;
        Cluster(string);
        Cluster();
        ~Cluster();
        void move(float, float, float);    //Mueve la estructura al punto indicado
        void rotate_Rad(float, float);     //Rota en coords esféricas usuales (radianes)
        void rotate_Deg(float, float);     //Rota en coords esféricas usuales (grados)
        void kick(float);                  //Implementada en la rutina Basin Hopping
        void swap(int);                    //Idem
        void srand_generator(string, int, string, int, float);
                                           //Construye una estructura atómica
                                             aleatoriamente (srand)
        void rand_generator(string, int, string, int);
                                           //Construye una estructura
                                             atómica pseudoaleatoriamente
        void centroid();                   //Mueve toda la estructura al centroide
};



class Crystal : public Atomic_Structure{
     public:
       //Parámetros de celda
       double x[3][3];
```

```
       double factor;
        Crystal(string);
        Crystal();
        ~Crystal();
};


class Molecule : public Atomic_Structure{
    public:
       Molecule(string);
       Molecule();
       ~Molecule();
       void move(float, float, float);    //Mueve la estructura al punto indicado
       void rotate_Rad(float, float);     //Rota en coords esféricas usuales (radianes)
       void rotate_Deg(float, float);     //Rota en coords esféricas usuales (grados)
    void centroid();                      //Mueve toda la estructura al centroide

};
```