```h
#include <iostream>
#include <cmath>

using namespace std;


// The use of templates poses an annoying problem...
// Definition and declaration of template functions and classes
// and their methods HAVE TO BE in the same file, the header file
// conventionally.

// Also NOTE that the friend function inside the class declaration
// HAS TO be declared with a different template parameter (U in this
// case).


template <typename T>
class v2d {
private:
    T x;
    T y;
public:
    v2d(T , T);
    v2d(const v2d &);
    ~v2d(void);

    v2d<T> & operator=(const v2d<T> &);
    v2d<T> & operator+(const v2d<T> &);
    v2d<T> & operator*(double k);
    double operator*(const v2d<T> &);
    double length(void);

    // NB template parameter must be different
    template <typename U>
    friend std::ostream& operator<<(std::ostream & os, const v2d<U> & V);
};


////////////////////////////////////////////////////////////////////////


////////////////////////////////////////////////////////////////////////
//                        FRIEND FUNCTION                             //
////////////////////////////////////////////////////////////////////////

template <typename U>
std::ostream& operator<<(std::ostream & os, const v2d<U> & V) {
    os << "[" << V.x << " , " << V.y << "]";
    return os;
}
```

```
59
60  ////////////////////////////////////////////////////////////////////////
61  //                              METHODS                               //
62  ////////////////////////////////////////////////////////////////////////
63
64  template <typename T>
65  v2d<T>::v2d(T a , T b) {
66      x = a;
67      y = b;
68  }
69
70  template <typename T>
71  v2d<T>::v2d(const v2d<T> & v) {
72      this->x = v.x;
73      this->y = v.y;
74  }
75
76  template <typename T>
77  v2d<T>::~v2d() { }
78
79  template <typename T>
80  v2d<T>& v2d<T>::operator=(const v2d<T> & v) {
81      this->x = v.x;
82      this->y = v.y;
83      return *this;
84  }
85
86  template <typename T>
87  v2d<T>& v2d<T>::operator+(const v2d<T> & v) {
88      this->x = this->x + v.x;
89      this->y = this->y + v.y;
90      return *this;
91  }
92
93  template <typename T>
94  double v2d<T>::operator*(const v2d<T> & v) {
95      return (this->x * v.x + this->y * v.y);
96  }
97
98  template <typename T>
99  v2d<T>& v2d<T>::operator*(double k) {
100     x *= k;
101     y *= k;
102     return *this;
103 }
104
105 template <typename T>
106 double v2d<T>::length() {
107     return (sqrt(pow(x, 2) + pow(y, 2)));
108 }
```