6 Example : Vector2D

Here we want to develop a class that provides 2 dimensional vector operations. A 2D vector consists of two components

$$\vec{a} = \begin{pmatrix} x \\ y \end{pmatrix}$$

6.1 Operations

- Addition, subtraction
  - For $\vec{a} = \begin{pmatrix} x_1 \\ y_1 \end{pmatrix}$ and $\vec{b} = \begin{pmatrix} x_2 \\ y_2 \end{pmatrix}$, $\vec{a} + \vec{b} = \begin{pmatrix} x_1 + x_2 \\ y_1 + y_2 \end{pmatrix}$ and $\vec{a} - \vec{b} = \begin{pmatrix} x_1 - x_2 \\ y_1 - y_2 \end{pmatrix}$
- Dot Product
  - For $\vec{a} = \begin{pmatrix} x_1 \\ y_1 \end{pmatrix}$ and $\vec{b} = \begin{pmatrix} x_2 \\ y_2 \end{pmatrix}$, $\vec{a} \cdot \vec{b} = x_1 x_2 + y_1 y_2$
- Norm
  - For $\vec{a} = \begin{pmatrix} x_1 \\ y_1 \end{pmatrix}$, $|\vec{a}| = \sqrt{x_1^2 + y_1^2}$
- Scalar operation
  - For $\vec{a} = \begin{pmatrix} x_1 \\ y_1 \end{pmatrix}$, $\alpha\vec{a} = \begin{pmatrix} \alpha x_1 \\ \alpha y_1 \end{pmatrix}$, $\vec{a}\alpha = \begin{pmatrix} \alpha x_1 \\ \alpha y_1 \end{pmatrix}$
  - For $\vec{a} = \begin{pmatrix} x_1 \\ y_1 \end{pmatrix}$, $\vec{a}/\alpha = \begin{pmatrix} x_1/\alpha \\ y_1/\alpha \end{pmatrix}$

6.2 Class Design

- Class holds x and y components as private member variables.

```
class vector2d
{
public:
    // constructor
    vector2d();
    vector2d(double x,double y);
private:
    // private member variables
    double x;
    double y;
};
```

6.2.1 Contractors

- Two constructors
  - Default contractor sets $x = y = 0$
  - Ac constructor takes x and y, and stores them to the private member variables.

```
vector2d::vector2d():x(0),y(0){
}

vector2d::vector2d(double xi,double yi):x(xi),y(yi){
}
```

6.2.2 Mutator, Accessor

```
// Mutator
void set_xy(double xi,double yi);

// accessors
double get_x(void ) const;
double get_y(void ) const;

//output
friend std::ostream& operator<<(std::ostream& strm, const vector2d& p);
//input
friend std::istream& operator>>(std::istream& strm, vector2d& p);
```

6.2.3 Operators

```
/// overloading operators
friend vector2d operator+(const vector2d& p1,const vector2d& p2);// addition
friend vector2d operator-(const vector2d& p1,const vector2d& p2);// subtraction
friend vector2d operator+=(vector2d& p1, const vector2d& p2);//addition & assign
friend vector2d operator-=(vector2d& p1, const vector2d& p2);// subraction & assign
friend double operator*(const vector2d& p1,const vector2d& p2);// dot product
friend vector2d operator/(const vector2d& p1, const double& d);// division
friend vector2d operator*=(vector2d& p1, const double& d);// multiplication & assign
friend vector2d operator/=(vector2d& p1, const double& d);// division & assign
friend vector2d operator*(const vector2d& p1, const double& d);// multiplication
friend vector2d operator*(const double& d, const vector2d& p);// multiplication
```

See **sample37.cpp** for details.

6.3 Third Party C++ Libraries for Linear Algebra

- Boost uBlas `http://www.boost.org/doc/libs/1_60_0/libs/numeric/ublas/doc/index.html`
- Eigen `http://eigen.tuxfamily.org/index.php?title=Main_Page`
- Armadillo `http://arma.sourceforge.net/`