```
#include "Ninja.h"
#include <iostream>
#include <math.h>
using namespace std;

/*
 * ----------------------------------------------------------------------------
 * "THE BEER-WARE LICENSE" (Revision 42):
 * Antoine BOULANGÃM-^I & Pierre_Elliot CABRERA wrote this file. As long as you
retain this notice you
 * can do whatever you want with this stuff. If we meet some day, and you think
 * this stuff is worth it, you can buy me a beer in return
 * ----------------------------------------------------------------------------
 */

//=====================================
// Constructeur
//=====================================
Ninja::Ninja (int x, int y, int w, int h, int mvtX, int mvtY)
    :MovableElement(x, y, w, h, mvtX, mvtY)
    , _status{WALKING}
    , _time{0}
    , _forward{true}
    , _toStopX{false}
    , _toStopY{false}
    , _toChangeDirection{UNCHANGED}
    , _xBeginJmp{0}
    , _yBeginJmp{0}
    , _reduction{false}
{}

//=====================================
// DÃ©termine le dÃ©placement du ninja
// en fonction de son statut
//=====================================
void Ninja::move() {
    switch (_status){
        case WALKING:
            if(_x+_mvtX >= 0 && _x+_mvtX <= SCREEN_WIDTH){
                if(_x+_mvtX <= 0)
                    _x = 0;
                else if(_x+_w+_mvtX >= SCREEN_WIDTH)
                    _x = SCREEN_WIDTH - _w;
                else
                    _x += _mvtX;
            }
            break;

        case JUMPING:
            jump();
            break;

        case SOARING:
        case TRANSITIONNING:
            soar();
            break;
    }

}

//=====================================
// Determine la parabole servant au saut
// du ninja grÃ¢ce aux lois de Newton
//=====================================
void Ninja::jump(){
    float theta;

    if(_forward)
        theta = PI / 3;
    else
        theta = 2 * PI / 3;

    int newX = _xBeginJmp + ((fabs(_mvtX) * cos(theta) * _time) * 0.04);

    if(_reduction)
        _x = (newX) - ((newX - _x) * 0.93);
    else
        _x = newX;

    _y = (_yBeginJmp - _h/2) - ((((-1) * GRAVITY * (_time * _time) / 2000) + (_m
vtY * sin(theta) * _time) + (_h / 2)) * 0.05);

    _time += 75;

    if(_x < 0 || _x > SCREEN_WIDTH - _w){
        if(_x < 0)
            _x = 0;
        else if(_x > SCREEN_WIDTH - _w)
            _x = SCREEN_WIDTH - _w;
    }

    // RÃ©initialise les variables nÃ©cessaires Ã  la fin du saut
    if((_y + _h) > GROUND_HEIGHT){
        _y = GROUND_HEIGHT - _h;
        _time = 0;
        _reduction = false;
        if(_status != SOARING)
            _status = WALKING;
        if(_toChangeDirection == LEFT)
            _forward = false;
        else if(_toChangeDirection == RIGHT)
            _forward = true;
    }
}

//=====================================
// DÃ©termine les mouvements du ninja
// lorsqu'il est sous l'effet du bonus 'VOL'
//=====================================
void Ninja::soar(){
    MovableElement::move();

    if(_x < 0)
        _x = 0;
    else if(_x > SCREEN_WIDTH - _w)
        _x = SCREEN_WIDTH - _w;

    if(_y < 0)
        _y = 0;
    else if(_y > GROUND_HEIGHT - _h){
        _y = GROUND_HEIGHT - _h;
        if(_status == TRANSITIONNING){
            _status = WALKING;
            _time = 0;
        }
    }
}

//=====================================
// Mutateurs
//=====================================
void Ninja::setStatus(NINJA_STATUS value){ _status = value; }

void Ninja::setDirection(bool forward){ _forward = forward; }

void Ninja::setToStopX(bool toStop){ _toStopX = toStop; }

void Ninja::setToStopY(bool toStop){ _toStopY = toStop; }
```

```
void Ninja::setXBeginJmp(int value){ _xBeginJmp = value; }

void Ninja::setYBeginJmp(int value){ _yBeginJmp = value; }

void Ninja::setToChangeDirection(NINJA_CHANGE_DIRECTION newDirection){ _toChange
Direction = newDirection; }

void Ninja::setReduction(int toReduce){ _reduction = toReduce; }

NINJA_STATUS Ninja::getStatus() const{ return _status; }

bool Ninja::getDirection() const{ return _forward; }

//=====================================
// Accesseurs
//=====================================
bool Ninja::getToStopX() const{ return _toStopX; }

bool Ninja::getToStopY() const{ return _toStopY; }

NINJA_CHANGE_DIRECTION Ninja::getToChangeDirection() const{ return _toChangeDire
ction; }
```