```
//
//  EULERODE.cpp
//  ODEsolver
//
//  Created by Ben Stager on 5/3/21.
//

#include "EULERODE.hpp"
#include <iomanip>

EulerODE::EulerODE():EulerODE(1){
}
EulerODE::EulerODE(int
size):vectorSize(size),elapsedTime(0.0),timeStep(defaultTimeStep){
    // allocate space
    vectorY = new double[vectorSize];
    vectorYdot = new double[vectorSize];
    // initialize Y(0)=0
    std::fill(vectorY,vectorY+vectorSize,0.0);
}
// copy constructor
EulerODE::EulerODE(const EulerODE &p):EulerODE(p.getSize()){
    for (int i = 0 ; i < vectorSize ; i++){
        vectorY[i] = p.getComponent(i);
    }
}
//  destructor
EulerODE::~EulerODE(){
    delete [] vectorY;
    delete [] vectorYdot;
}

// Integrate for a time step
void EulerODE::incrementTime(){
    // compute Y'(t)
    computeY_dot();

    // Y(t+dt) = Y(t) + dt * Y'(t)
    for (int i = 0 ; i < vectorSize ; i++){
        vectorY[i] += vectorYdot[i] * timeStep;
    }
    elapsedTime += timeStep;
}

// compute time derivatives
void EulerODE::computeY_dot(){
    // Y' = 1
    for (int i = 0 ; i < vectorSize ; i++){
        vectorYdot[i] = 1.0;
    }
}

int EulerODE::getSize() const{
    return vectorSize;
}

double EulerODE::getComponent(int i) const{
    return vectorY[i];
}
```

```
double EulerODE::getDotComponent(int i) const{
    return vectorYdot[i];
}

double EulerODE::getTimeStep() const{
    return timeStep;
}

void EulerODE::setTimeStep(double dt){
    timeStep = dt;
}

double EulerODE::getElapsedTime() const{
    return elapsedTime;
}

void EulerODE::setComponent(int i,double v) {
    vectorY[i] = v;
}

// set a derivative of component
void EulerODE::setDotComponent(int i,double vp) {
    vectorYdot[i] = vp;
}

std::ostream& operator << (std::ostream& str, const EulerODE &p){
    str << std::setprecision(5) << std::fixed;
    str << p.elapsedTime;
    for (int i = 0 ; i < p.vectorSize ; i++){
        str << " " << p.vectorY[i];
    }
    return str;
}
```