Complex Object Collections in C++

C++ Example

In C++, vectors act as dynamic arrays, with the ability to resize themselves as elements are inserted or deleted.

To make them available:

```
#include <vector>
```

Most vector examples show simple data elements, e.g., to create a vector of int values:

```
// Initialize
vector<int> numberList;

// Add new elements to the vector:
vector.push_back(1);
vector.push_back(2);
vector.push_back(3);
```

But, how to add complex objects, when you need something like a dataset? It's actually pretty straightforward. This example will use planetary data.

First, create a class to hold an individual data element:

```
class PlanetData {
public:
  string planetName;
  double tp_PeriodOrbit;
  double long_LongitudeEpoch;
  double peri_LongitudePerihelion;
  double ecc_EccentricityOrbit;
  double axis_AxisOrbit;
  double incl_OrbitalInclination;
  double node_LongitudeAscendingNode;
  double theta0_AngularDiameter;
  double v0_VisualMagnitude;

  PlanetData() {}

  PlanetData(string planetName, double tp_PeriodOrbit,
             double long_LongitudeEpoch, double peri_LongitudePerihelion,
             double ecc_EccentricityOrbit, double axis_AxisOrbit,
             double incl_OrbitalInclination, double
node_LongitudeAscendingNode,
             double theta0_AngularDiameter, double v0_VisualMagnitude) {
    this->planetName = planetName;
```

```
    this->tp_PeriodOrbit = tp_PeriodOrbit;
    this->long_LongitudeEpoch = long_LongitudeEpoch;
    this->peri_LongitudePerihelion = peri_LongitudePerihelion;
    this->ecc_EccentricityOrbit = ecc_EccentricityOrbit;
    this->axis_AxisOrbit = axis_AxisOrbit;
    this->incl_OrbitalInclination = incl_OrbitalInclination;
    this->node_LongitudeAscendingNode = node_LongitudeAscendingNode;
    this->theta0_AngularDiameter = theta0_AngularDiameter;
    this->v0_VisualMagnitude = v0_VisualMagnitude;
  }
};
```

Create a vector object, using the class as the vector type:

```
vector<PlanetData> planetData;
```

Add elements to the vector using instances of the class:

```
planetData.push_back(PlanetData("Mercury", 0.24085, 75.5671, 77.612,
0.205627, 0.387098, 7.0051, 48.449, 6.74, -0.42));
```

Then, you can loop through the vector, looking for an individual element:

```
// This example assumes it's being called inside a
// function, and returning an instance of the
// found element.

for (int i = 0; i < planetData.size(); i++)
  if (planetData[i].planetName == planetName)
    return planetData[i];
```

Complete Example

example.cpp

```
#include <iostream>
#include <vector>

using namespace std;

class PlanetData {
public:
  string planetName;
  double tp_PeriodOrbit;
  double long_LongitudeEpoch;
  double peri_LongitudePerihelion;
  double ecc_EccentricityOrbit;
  double axis_AxisOrbit;
```

```
  double incl_OrbitalInclination;
  double node_LongitudeAscendingNode;
  double theta0_AngularDiameter;
  double v0_VisualMagnitude;

  PlanetData() {}

  PlanetData(string planetName, double tp_PeriodOrbit,
  double long_LongitudeEpoch, double peri_LongitudePerihelion,
  double ecc_EccentricityOrbit, double axis_AxisOrbit,
  double incl_OrbitalInclination, double node_LongitudeAscendingNode,
  double theta0_AngularDiameter, double v0_VisualMagnitude) {
  this->planetName = planetName;
  this->tp_PeriodOrbit = tp_PeriodOrbit;
  this->long_LongitudeEpoch = long_LongitudeEpoch;
  this->peri_LongitudePerihelion = peri_LongitudePerihelion;
  this->ecc_EccentricityOrbit = ecc_EccentricityOrbit;
  this->axis_AxisOrbit = axis_AxisOrbit;
  this->incl_OrbitalInclination = incl_OrbitalInclination;
  this->node_LongitudeAscendingNode = node_LongitudeAscendingNode;
  this->theta0_AngularDiameter = theta0_AngularDiameter;
  this->v0_VisualMagnitude = v0_VisualMagnitude;
  }
};

PlanetData planetLookup(string planetName) {
  vector<PlanetData> planetData;

  planetData.push_back(PlanetData("Mercury", 0.24085, 75.5671, 77.612,
0.205627, 0.387098, 7.0051, 48.449, 6.74, -0.42));
  planetData.push_back(PlanetData("Venus", 0.615207, 272.30044, 131.54,
0.006812, 0.723329, 3.3947, 76.769, 16.92, -4.4));
  planetData.push_back(PlanetData("Earth", 0.999996, 99.556772,
103.2055, 0.016671, 0.999985, -99.0, -99.0, -99.0, -99.0));
  planetData.push_back(PlanetData("Mars", 1.880765, 109.09646, 336.217,
0.093348, 1.523689, 1.8497, 49.632, 9.36, -1.52));
  planetData.push_back(PlanetData("Jupiter", 11.857911, 337.917132,
14.6633, 0.048907, 5.20278, 1.3035, 100.595, 196.74, -9.4));
  planetData.push_back(PlanetData("Saturn", 29.310579, 172.398316,
89.567, 0.053853, 9.51134, 2.4873, 113.752, 165.6, -8.88));
  planetData.push_back(PlanetData("Uranus", 84.039492, 356.135400,
172.884833, 0.046321, 19.21814, 0.773059, 73.926961, 65.8, -7.19));
  planetData.push_back(PlanetData("Neptune", 165.845392, 326.895127,
23.07, 0.010483, 30.1985, 1.7673, 131.879, 62.2, -6.87));

  for (int i = 0; i < planetData.size(); i++)
    if (planetData[i].planetName == planetName)
      return planetData[i];

  return PlanetData("NoMatch", 0, 0, 0, 0, 0, 0, 0, 0, 0);
}
```

```
void printDetails(PlanetData planetDetails) {
  cout << planetDetails.planetName << " " <<
planetDetails.tp_PeriodOrbit << " "
    << planetDetails.long_LongitudeEpoch << " "
    << planetDetails.peri_LongitudePerihelion << " "
    << planetDetails.ecc_EccentricityOrbit << " "
    << planetDetails.axis_AxisOrbit << " "
    << planetDetails.incl_OrbitalInclination << " "
    << planetDetails.node_LongitudeAscendingNode << " "
    << planetDetails.theta0_AngularDiameter << " "
    << planetDetails.v0_VisualMagnitude << endl;
}

int main() {
  printDetails(planetLookup("Mercury"));
  printDetails(planetLookup("Venus"));
  printDetails(planetLookup("Earth"));
  printDetails(planetLookup("Mars"));
  printDetails(planetLookup("Jupiter"));
  printDetails(planetLookup("Saturn"));
  printDetails(planetLookup("Uranus"));
  printDetails(planetLookup("Neptune"));
  printDetails(planetLookup("Pluto")); // won't be found -> not in the
dataset

  return (0);
}
```

c and cpp