```
 //
//  BinaryChop_unitttest.cpp
//  CppTests
//
//  Created by Stewart Bracken on 12/9/13.
//  Copyright (c) 2013 Stewart Bracken. All rights reserved.
//

#include <iomanip>
#include <iostream>
#include <vector>
#include <gtest/gtest.h>

#include <time.h>

double get_seconds(clock_t ct)
{
    return ((double)ct)/CLOCKS_PER_SEC;
}

#include "BinaryChop.h"

typedef int (*chop_ptr)(int, const std::vector<int>&);

TEST(BinaryChop, Chop_iterative){
    clock_t start = clock();

    //Use point because it's easier to copy paste test data
    chop_ptr chop = &(BinaryChop::chop1);
    for(int i=0;i<1000;++i){
    std::vector<int> data = {  };
    ASSERT_TRUE( NOT_FOUND == chop(3, data));
    data = {1};
    ASSERT_TRUE(NOT_FOUND == chop(3, data));
    ASSERT_TRUE(0 == chop(1, data));

    data = {1, 3, 5};
    ASSERT_TRUE(0 == chop(1, data));
    ASSERT_TRUE(1 == chop(3, data));
    ASSERT_TRUE(2 == chop(5, data));
    ASSERT_TRUE(NOT_FOUND == chop(0, data));
    ASSERT_TRUE(NOT_FOUND == chop(2, data));
    ASSERT_TRUE(NOT_FOUND == chop(4, data));
    ASSERT_TRUE(NOT_FOUND == chop(6, data));
```

```
    data = {1, 3, 5, 7};
    ASSERT_TRUE(0 == chop(1, data));
    ASSERT_TRUE(1 == chop(3, data));
    ASSERT_TRUE(2 == chop(5, data));
    ASSERT_TRUE(3 == chop(7, data));
    ASSERT_TRUE(NOT_FOUND == chop(0, data));
    ASSERT_TRUE(NOT_FOUND == chop(2, data));
    ASSERT_TRUE(NOT_FOUND == chop(4, data));
    ASSERT_TRUE(NOT_FOUND == chop(6, data));
    ASSERT_TRUE(NOT_FOUND == chop(8, data));

    for(int i = 0; i < 500; ++i){
        data.push_back(i*2 + 9);
    }
    ASSERT_TRUE(250 == chop(501, data));
    ASSERT_TRUE(500 == chop(1001, data));
    ASSERT_TRUE(NOT_FOUND == chop(1000000, data));
    }
    clock_t end = clock();
    std::cout<<std::setprecision(10);
    std::cout<<"("<<get_seconds(end-start)<<" seconds)"<<std::endl;
}



TEST(BinaryChop, Chop_recursive){
    clock_t start = clock();
    chop_ptr chop = &BinaryChop::chop2;
    for(int i=0;i<1000;++i){
    std::vector<int> data = {  };
    ASSERT_TRUE( NOT_FOUND == chop(3, data));
    data = {1};
    ASSERT_TRUE(NOT_FOUND == chop(3, data));
    ASSERT_TRUE(0 == chop(1, data));

    data = {1, 3, 5};
    ASSERT_TRUE(0 == chop(1, data));
    ASSERT_TRUE(1 == chop(3, data));
    ASSERT_TRUE(2 == chop(5, data));
    ASSERT_TRUE(NOT_FOUND == chop(0, data));
    ASSERT_TRUE(NOT_FOUND == chop(2, data));
    ASSERT_TRUE(NOT_FOUND == chop(4, data));
    ASSERT_TRUE(NOT_FOUND == chop(6, data));
```

```
        data = {1, 3, 5, 7};
        ASSERT_TRUE(0 == chop(1, data));
        ASSERT_TRUE(1 == chop(3, data));
        ASSERT_TRUE(2 == chop(5, data));
        ASSERT_TRUE(3 == chop(7, data));
        ASSERT_TRUE(NOT_FOUND == chop(0, data));
        ASSERT_TRUE(NOT_FOUND == chop(2, data));
        ASSERT_TRUE(NOT_FOUND == chop(4, data));
        ASSERT_TRUE(NOT_FOUND == chop(6, data));
        ASSERT_TRUE(NOT_FOUND == chop(8, data));

        for(int i = 0; i < 500; ++i){
            data.push_back(i*2 + 9);
        }
        ASSERT_TRUE(250 == chop(501, data));
        ASSERT_TRUE(500 == chop(1001, data));
        ASSERT_TRUE(NOT_FOUND == chop(1000000, data));
        }
        clock_t end = clock();
        std::cout<<std::setprecision(10);
        std::cout<<"("<<get_seconds(end-start)<<" seconds)"<<std::endl;
    }



    TEST(BinaryChop, Chop_functional_vector){
        clock_t start = clock();
        chop_ptr chop = &BinaryChop::chop3;
        for(int i=0;i<1000;++i){
        std::vector<int> data = {  };
        ASSERT_TRUE( NOT_FOUND == chop(3, data));
        data = {1};
        ASSERT_TRUE(NOT_FOUND == chop(3, data));
        ASSERT_TRUE(0 == chop(1, data));

        data = {1, 3, 5};
        ASSERT_TRUE(0 == chop(1, data));
        ASSERT_TRUE(1 == chop(3, data));
        ASSERT_TRUE(2 == chop(5, data));
        ASSERT_TRUE(NOT_FOUND == chop(0, data));
        ASSERT_TRUE(NOT_FOUND == chop(2, data));
        ASSERT_TRUE(NOT_FOUND == chop(4, data));
        ASSERT_TRUE(NOT_FOUND == chop(6, data));

        data = {1, 3, 5, 7};
```

```
        ASSERT_TRUE(0 == chop(1, data));
        ASSERT_TRUE(1 == chop(3, data));
        ASSERT_TRUE(2 == chop(5, data));
        ASSERT_TRUE(3 == chop(7, data));
        ASSERT_TRUE(NOT_FOUND == chop(0, data));
        ASSERT_TRUE(NOT_FOUND == chop(2, data));
        ASSERT_TRUE(NOT_FOUND == chop(4, data));
        ASSERT_TRUE(NOT_FOUND == chop(6, data));
        ASSERT_TRUE(NOT_FOUND == chop(8, data));

        for(int i = 0; i < 500; ++i){
            data.push_back(i*2 + 9);
        }
        ASSERT_TRUE(250 == chop(501, data));
        ASSERT_TRUE(500 == chop(1001, data));
        ASSERT_TRUE(NOT_FOUND == chop(1000000, data));
        }
        clock_t end = clock();
        std::cout<<std::setprecision(10);
        std::cout<<"("<<get_seconds(end-start)<<" seconds)"<<std::endl;
    }


    TEST(BinaryChop, Chop_concurrent){
        clock_t start = clock();
        chop_ptr chop = &BinaryChop::chop4;
        for(int i=0;i<1000;++i){
        std::vector<int> data = {  };
        ASSERT_TRUE( NOT_FOUND == chop(3, data));
        data = {1};
        ASSERT_TRUE(NOT_FOUND == chop(3, data));
        ASSERT_TRUE(0 == chop(1, data));

        data = {1, 3, 5};
        ASSERT_TRUE(0 == chop(1, data));
        ASSERT_TRUE(1 == chop(3, data));
        ASSERT_TRUE(2 == chop(5, data));
        ASSERT_TRUE(NOT_FOUND == chop(0, data));
        ASSERT_TRUE(NOT_FOUND == chop(2, data));
        ASSERT_TRUE(NOT_FOUND == chop(4, data));
        ASSERT_TRUE(NOT_FOUND == chop(6, data));

        data = {1, 3, 5, 7};
        ASSERT_TRUE(0 == chop(1, data));
        ASSERT_TRUE(1 == chop(3, data));
        ASSERT_TRUE(2 == chop(5, data));
        ASSERT_TRUE(3 == chop(7, data));
        ASSERT_TRUE(NOT_FOUND == chop(0, data));
        ASSERT_TRUE(NOT_FOUND == chop(2, data));
        ASSERT_TRUE(NOT_FOUND == chop(4, data));
        ASSERT_TRUE(NOT_FOUND == chop(6, data));
        ASSERT_TRUE(NOT_FOUND == chop(8, data));

        for(int i = 0; i < 500; ++i){
            data.push_back(i*2 + 9);
        }
        ASSERT_TRUE(250 == chop(501, data));
        ASSERT_TRUE(500 == chop(1001, data));
        ASSERT_TRUE(NOT_FOUND == chop(1000000, data));
        }
        clock_t end = clock();
        std::cout<<std::setprecision(10);
        std::cout<<"("<<get_seconds(end-start)<<" seconds)"<<std::endl;
    }

    TEST(BinaryChop, Chop_deferred_equality){
        clock_t start = clock();
        chop_ptr chop = &BinaryChop::chop5;
        for(int i=0;i<1000;++i){
        std::vector<int> data = {  };
        ASSERT_TRUE( NOT_FOUND == chop(3, data));
        data = {1};
        ASSERT_TRUE(NOT_FOUND == chop(3, data));
        ASSERT_TRUE(0 == chop(1, data));

        data = {1, 3, 5};
        ASSERT_TRUE(0 == chop(1, data));
        ASSERT_TRUE(1 == chop(3, data));
        ASSERT_TRUE(2 == chop(5, data));
        ASSERT_TRUE(NOT_FOUND == chop(0, data));
        ASSERT_TRUE(NOT_FOUND == chop(2, data));
        ASSERT_TRUE(NOT_FOUND == chop(4, data));
        ASSERT_TRUE(NOT_FOUND == chop(6, data));

        data = {1, 3, 5, 7};
        ASSERT_TRUE(0 == chop(1, data));
        ASSERT_TRUE(1 == chop(3, data));
        ASSERT_TRUE(2 == chop(5, data));
        ASSERT_TRUE(3 == chop(7, data));
        ASSERT_TRUE(NOT_FOUND == chop(0, data));
```

```
        ASSERT_TRUE(NOT_FOUND == chop(2, data));
        ASSERT_TRUE(NOT_FOUND == chop(4, data));
        ASSERT_TRUE(NOT_FOUND == chop(6, data));
        ASSERT_TRUE(NOT_FOUND == chop(8, data));

        for(int i = 0; i < 500; ++i){
            data.push_back(i*2 + 9);
        }
        ASSERT_TRUE(250 == chop(501, data));
        ASSERT_TRUE(500 == chop(1001, data));
        ASSERT_TRUE(NOT_FOUND == chop(1000000, data));
        }
        clock_t end = clock();
        std::cout<<std::setprecision(10);
        std::cout<<"("<<get_seconds(end-start)<<" seconds)"<<std::endl;
    }

    TEST(BinaryChop, recursive_speedup){
        clock_t start = clock();
        chop_ptr chop = &BinaryChop::chop6;
        for(int i=0;i<1000;++i){
        std::vector<int> data = {  };
        ASSERT_TRUE( NOT_FOUND == chop(3, data));
        data = {1};
        ASSERT_TRUE(NOT_FOUND == chop(3, data));
        ASSERT_TRUE(0 == chop(1, data));

        data = {1, 3, 5};
        ASSERT_TRUE(0 == chop(1, data));
        ASSERT_TRUE(1 == chop(3, data));
        ASSERT_TRUE(2 == chop(5, data));
        ASSERT_TRUE(NOT_FOUND == chop(0, data));
        ASSERT_TRUE(NOT_FOUND == chop(2, data));
        ASSERT_TRUE(NOT_FOUND == chop(4, data));
        ASSERT_TRUE(NOT_FOUND == chop(6, data));

        data = {1, 3, 5, 7};
        ASSERT_TRUE(0 == chop(1, data));
        ASSERT_TRUE(1 == chop(3, data));
        ASSERT_TRUE(2 == chop(5, data));
        ASSERT_TRUE(3 == chop(7, data));
        ASSERT_TRUE(NOT_FOUND == chop(0, data));
        ASSERT_TRUE(NOT_FOUND == chop(2, data));
        ASSERT_TRUE(NOT_FOUND == chop(4, data));
        ASSERT_TRUE(NOT_FOUND == chop(6, data));
        ASSERT_TRUE(NOT_FOUND == chop(8, data));

        for(int i = 0; i < 500; ++i){
            data.push_back(i*2 + 9);
        }
        ASSERT_TRUE(250 == chop(501, data));
        ASSERT_TRUE(500 == chop(1001, data));
        ASSERT_TRUE(NOT_FOUND == chop(1000000, data));
        }
        clock_t end = clock();
        std::cout<<std::setprecision(10);
        std::cout<<"("<<get_seconds(end-start)<<" seconds)"<<std::endl;
    }
```