Linear and binary search

Michael Nowak

Texas A&M University

Overview

MyArray

Searching for some value in an array

- Linear Search
- Binary Search

Overview

MyArray

```cpp
#ifndef MYARRAY_H
#define MYARRAY_H
struct MyArray {
    int *arr = nullptr;
    unsigned int capacity = 0; // no elements can store
    unsigned int size = 0; // no elements currently held
};
#endif
```

Overview

MyArray

Searching for some value in an array
- Linear Search
- Binary Search

Setting-up

```
#include <iostream>
#include "MyArray.h"
#include "linearSearch.h"
#include "binarySearch.h"

using namespace std;

int main()
{
    MyArray ma1;
    ma1.arr = new int[7];
    ma1.capacity = 7;
    ma1.size = 7;
    for (unsigned int i = 0; i < ma1.size; ++i)
        ma1.arr[i] = i * 10;

    unsigned int idxofvalue = linearSearch(ma1, 20);

    unsigned int idxofvalue2 = binarySearch(ma1, 20);

    delete [] ma1.arr;

    return 0;
}
```

Setting-up

```cpp
#include <iostream>
#include "MyArray.h"
#include "linearSearch.h"
#include "binarySearch.h"

using namespace std;

int main()
{
    MyArray ma1;
    ma1.arr = new int[7];
    ma1.capacity = 7;
    ma1.size = 7;
    for (unsigned int i = 0; i < ma1.size; ++i)
        ma1.arr[i] = i * 10;

    unsigned int idxofvalue = linearSearch(ma1, 20);

    unsigned int idxofvalue2 = binarySearch(ma1, 20);

    delete [] ma1.arr;

    return 0;
}
```

Setting-up

```cpp
#include <iostream>
#include "MyArray.h"
#include "linearSearch.h"
#include "binarySearch.h"

using namespace std;

int main()
{
    MyArray ma1;
    ma1.arr = new int[7];
    ma1.capacity = 7;
    ma1.size = 7;
    for (unsigned int i = 0; i < ma1.size; ++i)
        ma1.arr[i] = i * 10;

    unsigned int idxofvalue = linearSearch(ma1, 20);

    unsigned int idxofvalue2 = binarySearch(ma1, 20);

    delete [] ma1.arr;

    return 0;
}
```

Setting-up

```cpp
#include <iostream>
#include "MyArray.h"
#include "linearSearch.h"
#include "binarySearch.h"

using namespace std;

int main()
{
    MyArray ma1;
    ma1.arr = new int[7];
    ma1.capacity = 7;
    ma1.size = 7;
    for (unsigned int i = 0; i < ma1.size; ++i)
        ma1.arr[i] = i * 10;

    unsigned int idxofvalue = linearSearch(ma1, 20);

    unsigned int idxofvalue2 = binarySearch(ma1, 20);

    delete [] ma1.arr;

    return 0;
}
```

Setting-up

```cpp
#include <iostream>
#include "MyArray.h"
#include "linearSearch.h"
#include "binarySearch.h"

using namespace std;

int main()
{
    MyArray ma1;
    ma1.arr = new int[7];
    ma1.capacity = 7;
    ma1.size = 7;
    for (unsigned int i = 0; i < ma1.size; ++i)
        ma1.arr[i] = i * 10;

    unsigned int idxofvalue = linearSearch(ma1, 20);

    unsigned int idxofvalue2 = binarySearch(ma1, 20);

    delete [] ma1.arr;

    return 0;
}
```

Setting-up

```
#include <iostream>
#include "MyArray.h"
#include "linearSearch.h"
#include "binarySearch.h"

using namespace std;

int main()
{
    MyArray ma1;
    ma1.arr = new int[7];
    ma1.capacity = 7;
    ma1.size = 7;
    for (unsigned int i = 0; i < ma1.size; ++i)
        ma1.arr[i] = i * 10;

    unsigned int idxofvalue = linearSearch(ma1, 20);

    unsigned int idxofvalue2 = binarySearch(ma1, 20);

    delete [] ma1.arr;

    return 0;
}
```

Setting-up

```cpp
#include <iostream>
#include "MyArray.h"
#include "linearSearch.h"
#include "binarySearch.h"

using namespace std;

int main()
{
    MyArray ma1;
    ma1.arr = new int[7];
    ma1.capacity = 7;
    ma1.size = 7;
    for (unsigned int i = 0; i < ma1.size; ++i)
        ma1.arr[i] = i * 10;

    unsigned int idxofvalue = linearSearch(ma1, 20);

    unsigned int idxofvalue2 = binarySearch(ma1, 20);

    delete [] ma1.arr;

    return 0;
}
```

Overview

Calling our linearSearch function

```cpp
#include <iostream>
#include "MyArray.h"
#include "linearSearch.h"
#include "binarySearch.h"

using namespace std;

int main()
{
    MyArray ma1;
    ma1.arr = new int[7];
    ma1.capacity = 7;
    ma1.size = 7;
    for (unsigned int i = 0; i < ma1.size; ++i)
        ma1.arr[i] = i * 10;

    unsigned int idxofvalue = linearSearch(ma1, 20);

    unsigned int idxofvalue2 = binarySearch(ma1, 20);

    delete [] ma1.arr;

    return 0;
}
```

linearSearch()

```cpp
#include "linearSearch.h"

int linearSearch(MyArray const &mya, int valSearchFor)
{
    for (unsigned int i = 0; i < mya.size; ++i) {
        if (mya.arr[i] == valSearchFor)
            return i;
    }
    return -1;
}
```

linearSearch()

```cpp
#include "linearSearch.h"

int linearSearch(MyArray const &mya, int valSearchFor)
{
    for (unsigned int i = 0; i < mya.size; ++i) {
        if (mya.arr[i] == valSearchFor)
            return i;
    }
    return -1;
}
```

linearSearch()

```cpp
#include "linearSearch.h"

int linearSearch(MyArray const &mya, int valSearchFor)
{
    for (unsigned int i = 0; i < mya.size; ++i) {
        if (mya.arr[i] == valSearchFor)
            return i;
    }
    return -1;
}
```

linearSearch()

```cpp
#include "linearSearch.h"

int linearSearch(MyArray const &mya, int valSearchFor)
{
    for (unsigned int i = 0; i < mya.size; ++i) {
        if (mya.arr[i] == valSearchFor)
            return i;
    }
    return -1;
}
```

linearSearch()

```cpp
#include "linearSearch.h"

int linearSearch(MyArray const &mya, int valSearchFor)
{
    for (unsigned int i = 0; i < mya.size; ++i) {
        if (mya.arr[i] == valSearchFor)
            return i;
    }
    return -1;
}
```

linearSearch()

```cpp
#include "linearSearch.h"

int linearSearch(MyArray const &mya, int valSearchFor)
{
    for (unsigned int i = 0; i < mya.size; ++i) {
        if (mya.arr[i] == valSearchFor)
            return i;
    }
    return -1;
}
```

linearSearch()

```cpp
#include "linearSearch.h"

int linearSearch(MyArray const &mya, int valSearchFor)
{
    for (unsigned int i = 0; i < mya.size; ++i) {
        if (mya.arr[i] == valSearchFor)
            return i;
    }
    return -1;
}
```

linearSearch()

```
#include "linearSearch.h"

int linearSearch(MyArray const &mya, int valSearchFor)
{
    for (unsigned int i = 0; i < mya.size; ++i) {
        if (mya.arr[i] == valSearchFor)
            return i;
    }
    return -1;
}
```

linearSearch()

```cpp
#include "linearSearch.h"

int linearSearch(MyArray const &mya, int valSearchFor)
{
    for (unsigned int i = 0; i < mya.size; ++i) {
        if (mya.arr[i] == valSearchFor)
            return i;
    }
    return -1;
}
```

Returned to `main`

```cpp
#include <iostream>
#include "MyArray.h"
#include "linearSearch.h"
#include "binarySearch.h"

using namespace std;

int main()
{
    MyArray ma1;
    ma1.arr = new int[7];
    ma1.capacity = 7;
    ma1.size = 7;
    for (unsigned int i = 0; i < ma1.size; ++i)
        ma1.arr[i] = i * 10;

    unsigned int idxofvalue = linearSearch(ma1, 20);

    unsigned int idxofvalue2 = binarySearch(ma1, 20);

    delete [] ma1.arr;

    return 0;
}
```

Overview

Binary Search

- Requirement: data is sorted!

Calling our binary search function

```
#include <iostream>
#include "MyArray.h"
#include "linearSearch.h"
#include "binarySearch.h"

using namespace std;

int main()
{
    MyArray ma1;
    ma1.arr = new int[7];
    ma1.capacity = 7;
    ma1.size = 7;
    for (unsigned int i = 0; i < ma1.size; ++i)
        ma1.arr[i] = i * 10;

    unsigned int idxofvalue = linearSearch(ma1, 20);

    unsigned int idxofvalue2 = binarySearch(ma1, 20);

    delete [] ma1.arr;

    return 0;
}
```

binarySearch()

```
#include "binarySearch.h"
#include <iostream>

int binarySearch(MyArray const &mya, int valSearchFor)
{
    /* return (any) position if valSearchFor is in sorted
        mya.arr[0..size-1] or -1 if valSearchFor is not
        present */
    unsigned int lowerBound = 0;
    unsigned int upperBound = mya.size - 1;
    while (lowerBound <= upperBound) {
        unsigned int midpt = (lowerBound + upperBound) / 2;
        if (mya.arr[midpt] < valSearchFor)
            lowerBound = midpt + 1;
        else if (mya.arr[midpt] == valSearchFor)
            return midpt;
        else /* mya.arr[midpt] > valSearchFor */
            upperBound = midpt - 1;
    }
    return -1;
}
```

binarySearch()

```
#include "binarySearch.h"
#include <iostream>

int binarySearch(MyArray const &mya, int valSearchFor)
{
    /* return (any) position if valSearchFor is in sorted
        mya.arr[0..size-1] or -1 if valSearchFor is not
        present */
    unsigned int lowerBound = 0;
    unsigned int upperBound = mya.size - 1;
    while (lowerBound <= upperBound) {
        unsigned int midpt = (lowerBound + upperBound) / 2;
        if (mya.arr[midpt] < valSearchFor)
            lowerBound = midpt + 1;
        else if (mya.arr[midpt] == valSearchFor)
            return midpt;
        else /* mya.arr[midpt] > valSearchFor */
            upperBound = midpt - 1;
    }
    return -1;
}
```

binarySearch()

```cpp
#include "binarySearch.h"
#include <iostream>

int binarySearch(MyArray const &mya, int valSearchFor)
{
    /* return (any) position if valSearchFor is in sorted
        mya.arr[0..size-1] or -1 if valSearchFor is not
        present */
    unsigned int lowerBound = 0;
    unsigned int upperBound = mya.size - 1;
    while (lowerBound <= upperBound) {
        unsigned int midpt = (lowerBound + upperBound) / 2;
        if (mya.arr[midpt] < valSearchFor)
            lowerBound = midpt + 1;
        else if (mya.arr[midpt] == valSearchFor)
            return midpt;
        else /* mya.arr[midpt] > valSearchFor */
            upperBound = midpt - 1;
    }
    return -1;
}
```

binarySearch()

```
#include "binarySearch.h"
#include <iostream>

int binarySearch(MyArray const &mya, int valSearchFor)
{
    /* return (any) position if valSearchFor is in sorted
        mya.arr[0..size-1] or -1 if valSearchFor is not
        present */
    unsigned int lowerBound = 0;
    unsigned int upperBound = mya.size - 1;
    while (lowerBound <= upperBound) {
        unsigned int midpt = (lowerBound + upperBound) / 2;
        if (mya.arr[midpt] < valSearchFor)
            lowerBound = midpt + 1;
        else if (mya.arr[midpt] == valSearchFor)
            return midpt;
        else /* mya.arr[midpt] > valSearchFor */
            upperBound = midpt - 1;
    }
    return -1;
}
```

binarySearch()

```cpp
#include "binarySearch.h"
#include <iostream>

int binarySearch(MyArray const &mya, int valSearchFor)
{
    /* return (any) position if valSearchFor is in sorted
        mya.arr[0..size-1] or -1 if valSearchFor is not
        present */
    unsigned int lowerBound = 0;
    unsigned int upperBound = mya.size - 1;
    while (lowerBound <= upperBound) {
        unsigned int midpt = (lowerBound + upperBound) / 2;
        if (mya.arr[midpt] < valSearchFor)
            lowerBound = midpt + 1;
        else if (mya.arr[midpt] == valSearchFor)
            return midpt;
        else /* mya.arr[midpt] > valSearchFor */
            upperBound = midpt - 1;
    }
    return -1;
}
```

binarySearch()

```
#include "binarySearch.h"
#include <iostream>

int binarySearch(MyArray const &mya, int valSearchFor)
{
    /* return (any) position if valSearchFor is in sorted
        mya.arr[0..size-1] or -1 if valSearchFor is not
        present */
    unsigned int lowerBound = 0;
    unsigned int upperBound = mya.size - 1;
    while (lowerBound <= upperBound) {
        unsigned int midpt = (lowerBound + upperBound) / 2;
        if (mya.arr[midpt] < valSearchFor)
            lowerBound = midpt + 1;
        else if (mya.arr[midpt] == valSearchFor)
            return midpt;
        else /* mya.arr[midpt] > valSearchFor */
            upperBound = midpt - 1;
    }
    return -1;
}
```

binarySearch()

```
#include "binarySearch.h"
#include <iostream>

int binarySearch(MyArray const &mya, int valSearchFor)
{
    /* return (any) position if valSearchFor is in sorted
        mya.arr[0..size-1] or -1 if valSearchFor is not
        present */
    unsigned int lowerBound = 0;
    unsigned int upperBound = mya.size - 1;
    while (lowerBound <= upperBound) {
        unsigned int midpt = (lowerBound + upperBound) / 2;
        if (mya.arr[midpt] < valSearchFor)
            lowerBound = midpt + 1;
        else if (mya.arr[midpt] == valSearchFor)
            return midpt;
        else /* mya.arr[midpt] > valSearchFor */
            upperBound = midpt - 1;
    }
    return -1;
}
```

binarySearch()

```
#include "binarySearch.h"
#include <iostream>

int binarySearch(MyArray const &mya, int valSearchFor)
{
    /* return (any) position if valSearchFor is in sorted
        mya.arr[0..size-1] or -1 if valSearchFor is not
        present */
    unsigned int lowerBound = 0;
    unsigned int upperBound = mya.size - 1;
    while (lowerBound <= upperBound) {
        unsigned int midpt = (lowerBound + upperBound) / 2;
        if (mya.arr[midpt] < valSearchFor)
            lowerBound = midpt + 1;
        else if (mya.arr[midpt] == valSearchFor)
            return midpt;
        else /* mya.arr[midpt] > valSearchFor */
            upperBound = midpt - 1;
    }
    return -1;
}
```

binarySearch()

```
#include "binarySearch.h"
#include <iostream>

int binarySearch(MyArray const &mya, int valSearchFor)
{
    /* return (any) position if valSearchFor is in sorted
        mya.arr[0..size-1] or -1 if valSearchFor is not
        present */
    unsigned int lowerBound = 0;
    unsigned int upperBound = mya.size - 1;
    while (lowerBound <= upperBound) {
        unsigned int midpt = (lowerBound + upperBound) / 2;
        if (mya.arr[midpt] < valSearchFor)
            lowerBound = midpt + 1;
        else if (mya.arr[midpt] == valSearchFor)
            return midpt;
        else /* mya.arr[midpt] > valSearchFor */
            upperBound = midpt - 1;
    }
    return -1;
}
```

binarySearch()

```
#include "binarySearch.h"
#include <iostream>

int binarySearch(MyArray const &mya, int valSearchFor)
{
    /* return (any) position if valSearchFor is in sorted
        mya.arr[0..size-1] or -1 if valSearchFor is not
        present */
    unsigned int lowerBound = 0;
    unsigned int upperBound = mya.size - 1;
    while (lowerBound <= upperBound) {
        unsigned int midpt = (lowerBound + upperBound) / 2;
        if (mya.arr[midpt] < valSearchFor)
            lowerBound = midpt + 1;
        else if (mya.arr[midpt] == valSearchFor)
            return midpt;
        else /* mya.arr[midpt] > valSearchFor */
            upperBound = midpt - 1;
    }
    return -1;
}
```

binarySearch()

```cpp
#include "binarySearch.h"
#include <iostream>

int binarySearch(MyArray const &mya, int valSearchFor)
{
    /* return (any) position if valSearchFor is in sorted
        mya.arr[0..size-1] or -1 if valSearchFor is not
        present */
    unsigned int lowerBound = 0;
    unsigned int upperBound = mya.size - 1;
    while (lowerBound <= upperBound) {
        unsigned int midpt = (lowerBound + upperBound) / 2;
        if (mya.arr[midpt] < valSearchFor)
            lowerBound = midpt + 1;
        else if (mya.arr[midpt] == valSearchFor)
            return midpt;
        else /* mya.arr[midpt] > valSearchFor */
            upperBound = midpt - 1;
    }
    return -1;
}
```

binarySearch()

```
#include "binarySearch.h"
#include <iostream>

int binarySearch(MyArray const &mya, int valSearchFor)
{
    /* return (any) position if valSearchFor is in sorted
        mya.arr[0..size-1] or -1 if valSearchFor is not
        present */
    unsigned int lowerBound = 0;
    unsigned int upperBound = mya.size - 1;
    while (lowerBound <= upperBound) {
        unsigned int midpt = (lowerBound + upperBound) / 2;
        if (mya.arr[midpt] < valSearchFor)
            lowerBound = midpt + 1;
        else if (mya.arr[midpt] == valSearchFor)
            return midpt;
        else /* mya.arr[midpt] > valSearchFor */
            upperBound = midpt - 1;
    }
    return -1;
}
```

binarySearch()

```
#include "binarySearch.h"
#include <iostream>

int binarySearch(MyArray const &mya, int valSearchFor)
{
    /* return (any) position if valSearchFor is in sorted
        mya.arr[0..size-1] or -1 if valSearchFor is not
        present */
    unsigned int lowerBound = 0;
    unsigned int upperBound = mya.size - 1;
    while (lowerBound <= upperBound) {
        unsigned int midpt = (lowerBound + upperBound) / 2;
        if (mya.arr[midpt] < valSearchFor)
            lowerBound = midpt + 1;
        else if (mya.arr[midpt] == valSearchFor)
            return midpt;
        else /* mya.arr[midpt] > valSearchFor */
            upperBound = midpt - 1;
    }
    return -1;
}
```

binarySearch()

```
#include "binarySearch.h"
#include <iostream>

int binarySearch(MyArray const &mya, int valSearchFor)
{
    /* return (any) position if valSearchFor is in sorted
        mya.arr[0..size-1] or -1 if valSearchFor is not
        present */
    unsigned int lowerBound = 0;
    unsigned int upperBound = mya.size - 1;
    while (lowerBound <= upperBound) {
        unsigned int midpt = (lowerBound + upperBound) / 2;
        if (mya.arr[midpt] < valSearchFor)
            lowerBound = midpt + 1;
        else if (mya.arr[midpt] == valSearchFor)
            return midpt;
        else /* mya.arr[midpt] > valSearchFor */
            upperBound = midpt - 1;
    }
    return -1;
}
```

binarySearch()

```
#include "binarySearch.h"
#include <iostream>

int binarySearch(MyArray const &mya, int valSearchFor)
{
    /* return (any) position if valSearchFor is in sorted
        mya.arr[0..size-1] or -1 if valSearchFor is not
        present */
    unsigned int lowerBound = 0;
    unsigned int upperBound = mya.size - 1;
    while (lowerBound <= upperBound) {
        unsigned int midpt = (lowerBound + upperBound) / 2;
        if (mya.arr[midpt] < valSearchFor)
            lowerBound = midpt + 1;
        else if (mya.arr[midpt] == valSearchFor)
            return midpt;
        else /* mya.arr[midpt] > valSearchFor */
            upperBound = midpt - 1;
    }
    return -1;
}
```

binarySearch()

```
#include "binarySearch.h"
#include <iostream>

int binarySearch(MyArray const &mya, int valSearchFor)
{
    /* return (any) position if valSearchFor is in sorted
        mya.arr[0..size-1] or -1 if valSearchFor is not
        present */
    unsigned int lowerBound = 0;
    unsigned int upperBound = mya.size - 1;
    while (lowerBound <= upperBound) {
        unsigned int midpt = (lowerBound + upperBound) / 2;
        if (mya.arr[midpt] < valSearchFor)
            lowerBound = midpt + 1;
        else if (mya.arr[midpt] == valSearchFor)
            return midpt;
        else /* mya.arr[midpt] > valSearchFor */
            upperBound = midpt - 1;
    }
    return -1;
}
```

binarySearch()

```
#include "binarySearch.h"
#include <iostream>

int binarySearch(MyArray const &mya, int valSearchFor)
{
    /* return (any) position if valSearchFor is in sorted
        mya.arr[0..size-1] or -1 if valSearchFor is not
        present */
    unsigned int lowerBound = 0;
    unsigned int upperBound = mya.size - 1;
    while (lowerBound <= upperBound) {
        unsigned int midpt = (lowerBound + upperBound) / 2;
        if (mya.arr[midpt] < valSearchFor)
            lowerBound = midpt + 1;
        else if (mya.arr[midpt] == valSearchFor)
            return midpt;
        else /* mya.arr[midpt] > valSearchFor */
            upperBound = midpt - 1;
    }
    return -1;
}
```

binarySearch()

```cpp
#include "binarySearch.h"
#include <iostream>

int binarySearch(MyArray const &mya, int valSearchFor)
{
    /* return (any) position if valSearchFor is in sorted
        mya.arr[0..size-1] or -1 if valSearchFor is not
        present */
    unsigned int lowerBound = 0;
    unsigned int upperBound = mya.size - 1;
    while (lowerBound <= upperBound) {
        unsigned int midpt = (lowerBound + upperBound) / 2;
        if (mya.arr[midpt] < valSearchFor)
            lowerBound = midpt + 1;
        else if (mya.arr[midpt] == valSearchFor)
            return midpt;
        else /* mya.arr[midpt] > valSearchFor */
            upperBound = midpt - 1;
    }
    return -1;
}
```

binarySearch()

```cpp
#include "binarySearch.h"
#include <iostream>

int binarySearch(MyArray const &mya, int valSearchFor)
{
    /* return (any) position if valSearchFor is in sorted
        mya.arr[0..size-1] or -1 if valSearchFor is not
        present */
    unsigned int lowerBound = 0;
    unsigned int upperBound = mya.size - 1;
    while (lowerBound <= upperBound) {
        unsigned int midpt = (lowerBound + upperBound) / 2;
        if (mya.arr[midpt] < valSearchFor)
            lowerBound = midpt + 1;
        else if (mya.arr[midpt] == valSearchFor)
            return midpt;
        else /* mya.arr[midpt] > valSearchFor */
            upperBound = midpt - 1;
    }
    return -1;
}
```

Returned to `main`

```
#include <iostream>
#include "MyArray.h"
#include "linearSearch.h"
#include "binarySearch.h"

using namespace std;

int main()
{
    MyArray ma1;
    ma1.arr = new int[7];
    ma1.capacity = 7;
    ma1.size = 7;
    for (unsigned int i = 0; i < ma1.size; ++i)
        ma1.arr[i] = i * 10;

    unsigned int idxofvalue = linearSearch(ma1, 20);

    unsigned int idxofvalue2 = binarySearch(ma1, 20);

    delete [] ma1.arr;

    return 0;
}
```

Returned to `main`

```cpp
#include <iostream>
#include "MyArray.h"
#include "linearSearch.h"
#include "binarySearch.h"

using namespace std;

int main()
{
    MyArray ma1;
    ma1.arr = new int[7];
    ma1.capacity = 7;
    ma1.size = 7;
    for (unsigned int i = 0; i < ma1.size; ++i)
        ma1.arr[i] = i * 10;

    unsigned int idxofvalue = linearSearch(ma1, 20);

    unsigned int idxofvalue2 = binarySearch(ma1, 20);

    delete [] ma1.arr;

    return 0;
}
```

Returned to `main`

```cpp
#include <iostream>
#include "MyArray.h"
#include "linearSearch.h"
#include "binarySearch.h"

using namespace std;

int main()
{
    MyArray ma1;
    ma1.arr = new int[7];
    ma1.capacity = 7;
    ma1.size = 7;
    for (unsigned int i = 0; i < ma1.size; ++i)
        ma1.arr[i] = i * 10;

    unsigned int idxofvalue = linearSearch(ma1, 20);

    unsigned int idxofvalue2 = binarySearch(ma1, 20);

    delete [] ma1.arr;

    return 0;
}
```