Linear and binary search

Michael Nowak

Texas A&M University

Notes

Overview

MyArray

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

Notes

Overview

MyArray

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

Notes

MyArray

```
#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
```

Notes

Overview

MyArray

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

Notes

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;
}
```

Notes

Setting-up

```
1  #include <iostream>
2  #include "MyArray.h"
3  #include "linearSearch.h"
4  #include "binarySearch.h"
5
6  using namespace std;
7
8  int main()
9  {
10     MyArray ma1;
11     ma1.arr = new int[7];
12     ma1.capacity = 7;
13     ma1.size = 7;
14     for (unsigned int i = 0; i < ma1.size; ++i)
15         ma1.arr[i] = i * 10;
16
17     unsigned int idxofvalue = linearSearch(ma1, 20);
18
19     unsigned int idxofvalue2 = binarySearch(ma1, 20);
20
21     delete [] ma1.arr;
22
23     return 0;
24 }
```

Notes

Setting-up

```
1  #include <iostream>
2  #include "MyArray.h"
3  #include "linearSearch.h"
4  #include "binarySearch.h"
5
6  using namespace std;
7
8  int main()
9  {
10     MyArray ma1;
11     ma1.arr = new int[7];
12     ma1.capacity = 7;
13     ma1.size = 7;
14     for (unsigned int i = 0; i < ma1.size; ++i)
15         ma1.arr[i] = i * 10;
16
17     unsigned int idxofvalue = linearSearch(ma1, 20);
18
19     unsigned int idxofvalue2 = binarySearch(ma1, 20);
20
21     delete [] ma1.arr;
22
23     return 0;
24 }
```

Notes

Setting-up

```
1  #include <iostream>
2  #include "MyArray.h"
3  #include "linearSearch.h"
4  #include "binarySearch.h"
5
6  using namespace std;
7
8  int main()
9  {
10     MyArray ma1;
11     ma1.arr = new int[7];
12     ma1.capacity = 7;
13     ma1.size = 7;
14     for (unsigned int i = 0; i < ma1.size; ++i)
15         ma1.arr[i] = i * 10;
16
17     unsigned int idxofvalue = linearSearch(ma1, 20);
18
19     unsigned int idxofvalue2 = binarySearch(ma1, 20);
20
21     delete [] ma1.arr;
22
23     return 0;
24 }
```

Notes

Setting-up

```
1  #include <iostream>
2  #include "MyArray.h"
3  #include "linearSearch.h"
4  #include "binarySearch.h"
5
6  using namespace std;
7
8  int main()
9  {
10     MyArray ma1;
11     ma1.arr = new int[7];
12     ma1.capacity = 7;
13     ma1.size = 7;
14     for (unsigned int i = 0; i < ma1.size; ++i)
15         ma1.arr[i] = i * 10;
16
17     unsigned int idxofvalue = linearSearch(ma1, 20);
18
19     unsigned int idxofvalue2 = binarySearch(ma1, 20);
20
21     delete [] ma1.arr;
22
23     return 0;
24 }
```

Notes

Setting-up

```
1  #include <iostream>
2  #include "MyArray.h"
3  #include "linearSearch.h"
4  #include "binarySearch.h"
5
6  using namespace std;
7
8  int main()
9  {
10     MyArray ma1;
11     ma1.arr = new int[7];
12     ma1.capacity = 7;
13     ma1.size = 7;
14     for (unsigned int i = 0; i < ma1.size; ++i)
15         ma1.arr[i] = i * 10;
16
17     unsigned int idxofvalue = linearSearch(ma1, 20);
18
19     unsigned int idxofvalue2 = binarySearch(ma1, 20);
20
21     delete [] ma1.arr;
22
23     return 0;
24 }
```

Notes

Setting-up

```
1  #include <iostream>
2  #include "MyArray.h"
3  #include "linearSearch.h"
4  #include "binarySearch.h"
5
6  using namespace std;
7
8  int main()
9  {
10     MyArray ma1;
11     ma1.arr = new int[7];
12     ma1.capacity = 7;
13     ma1.size = 7;
14     for (unsigned int i = 0; i < ma1.size; ++i)
15         ma1.arr[i] = i * 10;
16
17     unsigned int idxofvalue = linearSearch(ma1, 20);
18
19     unsigned int idxofvalue2 = binarySearch(ma1, 20);
20
21     delete [] ma1.arr;
22
23     return 0;
24 }
```

Notes

Overview

MyArray

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

Notes

Calling our linearSearch function

```
1  #include <iostream>
2  #include "MyArray.h"
3  #include "linearSearch.h"
4  #include "binarySearch.h"
5
6  using namespace std;
7
8  int main()
9  {
10     MyArray ma1;
11     ma1.arr = new int[7];
12     ma1.capacity = 7;
13     ma1.size = 7;
14     for (unsigned int i = 0; i < ma1.size; ++i)
15         ma1.arr[i] = i * 10;
16
17     unsigned int idxofvalue = linearSearch(ma1, 20);
18
19     unsigned int idxofvalue2 = binarySearch(ma1, 20);
20
21     delete [] ma1.arr;
22
23     return 0;
24 }
```

Notes

`linearSearch()`

```
1  #include "linearSearch.h"
2
3  int linearSearch(MyArray const &mya, int valSearchFor)
4  {
5      for (unsigned int i = 0; i < mya.size; ++i) {
6          if (mya.arr[i] == valSearchFor)
7              return i;
8      }
9      return -1;
10 }
```

Notes

linearSearch()

```
1  #include "linearSearch.h"
2
3  int linearSearch(MyArray const &mya, int valSearchFor)
4  {
5      for (unsigned int i = 0; i < mya.size; ++i) {
6          if (mya.arr[i] == valSearchFor)
7              return i;
8      }
9      return -1;
10 }
```

Notes

linearSearch()

```
1  #include "linearSearch.h"
2
3  int linearSearch(MyArray const &mya, int valSearchFor)
4  {
5      for (unsigned int i = 0; i < mya.size; ++i) {
6          if (mya.arr[i] == valSearchFor)
7              return i;
8      }
9      return -1;
10 }
```

Notes

linearSearch()

```
1  #include "linearSearch.h"
2
3  int linearSearch(MyArray const &mya, int valSearchFor)
4  {
5      for (unsigned int i = 0; i < mya.size; ++i) {
6          if (mya.arr[i] == valSearchFor)
7              return i;
8      }
9      return -1;
10 }
```

Notes

linearSearch()

```
1  #include "linearSearch.h"
2
3  int linearSearch(MyArray const &mya, int valSearchFor)
4  {
5      for (unsigned int i = 0; i < mya.size; ++i) {
6          if (mya.arr[i] == valSearchFor)
7              return i;
8      }
9      return -1;
10 }
```

Notes

linearSearch()


```
1  #include "linearSearch.h"
2
3  int linearSearch(MyArray const &mya, int valSearchFor)
4  {
5      for (unsigned int i = 0; i < mya.size; ++i) {
6          if (mya.arr[i] == valSearchFor)
7              return i;
8      }
9      return -1;
10 }
```


Notes

linearSearch()


```
1  #include "linearSearch.h"
2
3  int linearSearch(MyArray const &mya, int valSearchFor)
4  {
5      for (unsigned int i = 0; i < mya.size; ++i) {
6          if (mya.arr[i] == valSearchFor)
7              return i;
8      }
9      return -1;
10 }
```


Notes

linearSearch()

```
1  #include "linearSearch.h"
2
3  int linearSearch(MyArray const &mya, int valSearchFor)
4  {
5      for (unsigned int i = 0; i < mya.size; ++i) {
6          if (mya.arr[i] == valSearchFor)
7              return i;
8      }
9      return -1;
10 }
```

Notes

linearSearch()

```
1  #include "linearSearch.h"
2
3  int linearSearch(MyArray const &mya, int valSearchFor)
4  {
5      for (unsigned int i = 0; i < mya.size; ++i) {
6          if (mya.arr[i] == valSearchFor)
7              return i;
8      }
9      return -1;
10 }
```

Notes

Returned to `main`

```
1  #include <iostream>
2  #include "MyArray.h"
3  #include "linearSearch.h"
4  #include "binarySearch.h"
5
6  using namespace std;
7
8  int main()
9  {
10     MyArray ma1;
11     ma1.arr = new int[7];
12     ma1.capacity = 7;
13     ma1.size = 7;
14     for (unsigned int i = 0; i < ma1.size; ++i)
15         ma1.arr[i] = i * 10;
16
17     unsigned int idxofvalue = linearSearch(ma1, 20);
18
19     unsigned int idxofvalue2 = binarySearch(ma1, 20);
20
21     delete [] ma1.arr;
22
23     return 0;
24 }
```

Notes

Overview

MyArray

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

Notes

Binary Search

- Requirement: data is sorted!

Notes

Calling our binary search function

```
1  #include <iostream>
2  #include "MyArray.h"
3  #include "linearSearch.h"
4  #include "binarySearch.h"
5
6  using namespace std;
7
8  int main()
9  {
10     MyArray ma1;
11     ma1.arr = new int[7];
12     ma1.capacity = 7;
13     ma1.size = 7;
14     for (unsigned int i = 0; i < ma1.size; ++i)
15         ma1.arr[i] = i * 10;
16
17     unsigned int idxofvalue = linearSearch(ma1, 20);
18
19     unsigned int idxofvalue2 = binarySearch(ma1, 20);
20
21     delete [] ma1.arr;
22
23     return 0;
24 }
```

Notes

binarySearch()

```
1  #include "binarySearch.h"
2  #include <iostream>
3
4  int binarySearch(MyArray const &mya, int valSearchFor)
5  {
6      /* return (any) position if valSearchFor is in sorted
           mya.arr[0..size-1] or -1 if valSearchFor is not
           present */
7      unsigned int lowerBound = 0;
8      unsigned int upperBound = mya.size - 1;
9      while (lowerBound <= upperBound) {
10         unsigned int midpt = (lowerBound + upperBound) / 2;
11         if (mya.arr[midpt] < valSearchFor)
12             lowerBound = midpt + 1;
13         else if (mya.arr[midpt] == valSearchFor)
14             return midpt;
15         else /* mya.arr[midpt] > valSearchFor */
16             upperBound = midpt - 1;
17     }
18     return -1;
19 }
```

Notes

binarySearch()

```
1  #include "binarySearch.h"
2  #include <iostream>
3
4  int binarySearch(MyArray const &mya, int valSearchFor)
5  {
6      /* return (any) position if valSearchFor is in sorted
           mya.arr[0..size-1] or -1 if valSearchFor is not
           present */
7      unsigned int lowerBound = 0;
8      unsigned int upperBound = mya.size - 1;
9      while (lowerBound <= upperBound) {
10         unsigned int midpt = (lowerBound + upperBound) / 2;
11         if (mya.arr[midpt] < valSearchFor)
12             lowerBound = midpt + 1;
13         else if (mya.arr[midpt] == valSearchFor)
14             return midpt;
15         else /* mya.arr[midpt] > valSearchFor */
16             upperBound = midpt - 1;
17     }
18     return -1;
19 }
```

Notes

binarySearch()

```
1  #include "binarySearch.h"
2  #include <iostream>
3
4  int binarySearch(MyArray const &mya, int valSearchFor)
5  {
6      /* return (any) position if valSearchFor is in sorted
           mya.arr[0..size-1] or -1 if valSearchFor is not
           present */
7      unsigned int lowerBound = 0;
8      unsigned int upperBound = mya.size - 1;
9      while (lowerBound <= upperBound) {
10         unsigned int midpt = (lowerBound + upperBound) / 2;
11         if (mya.arr[midpt] < valSearchFor)
12             lowerBound = midpt + 1;
13         else if (mya.arr[midpt] == valSearchFor)
14             return midpt;
15         else /* mya.arr[midpt] > valSearchFor */
16             upperBound = midpt - 1;
17     }
18     return -1;
19 }
```

Notes

binarySearch()

```
1  #include "binarySearch.h"
2  #include <iostream>
3
4  int binarySearch(MyArray const &mya, int valSearchFor)
5  {
6      /* return (any) position if valSearchFor is in sorted
           mya.arr[0..size-1] or -1 if valSearchFor is not
           present */
7      unsigned int lowerBound = 0;
8      unsigned int upperBound = mya.size - 1;
9      while (lowerBound <= upperBound) {
10         unsigned int midpt = (lowerBound + upperBound) / 2;
11         if (mya.arr[midpt] < valSearchFor)
12             lowerBound = midpt + 1;
13         else if (mya.arr[midpt] == valSearchFor)
14             return midpt;
15         else /* mya.arr[midpt] > valSearchFor */
16             upperBound = midpt - 1;
17     }
18     return -1;
19 }
```

Notes

binarySearch()

```
1  #include "binarySearch.h"
2  #include <iostream>
3
4  int binarySearch(MyArray const &mya, int valSearchFor)
5  {
6      /* return (any) position if valSearchFor is in sorted
           mya.arr[0..size-1] or -1 if valSearchFor is not
           present */
7      unsigned int lowerBound = 0;
8      unsigned int upperBound = mya.size - 1;
9      while (lowerBound <= upperBound) {
10         unsigned int midpt = (lowerBound + upperBound) / 2;
11         if (mya.arr[midpt] < valSearchFor)
12             lowerBound = midpt + 1;
13         else if (mya.arr[midpt] == valSearchFor)
14             return midpt;
15         else /* mya.arr[midpt] > valSearchFor */
16             upperBound = midpt - 1;
17     }
18     return -1;
19 }
```

Notes

binarySearch()

```
1  #include "binarySearch.h"
2  #include <iostream>
3
4  int binarySearch(MyArray const &mya, int valSearchFor)
5  {
6      /* return (any) position if valSearchFor is in sorted
           mya.arr[0..size-1] or -1 if valSearchFor is not
           present */
7      unsigned int lowerBound = 0;
8      unsigned int upperBound = mya.size - 1;
9      while (lowerBound <= upperBound) {
10         unsigned int midpt = (lowerBound + upperBound) / 2;
11         if (mya.arr[midpt] < valSearchFor)
12             lowerBound = midpt + 1;
13         else if (mya.arr[midpt] == valSearchFor)
14             return midpt;
15         else /* mya.arr[midpt] > valSearchFor */
16             upperBound = midpt - 1;
17     }
18     return -1;
19 }
```

Notes

binarySearch()

```
1  #include "binarySearch.h"
2  #include <iostream>
3
4  int binarySearch(MyArray const &mya, int valSearchFor)
5  {
6      /* return (any) position if valSearchFor is in sorted
           mya.arr[0..size-1] or -1 if valSearchFor is not
           present */
7      unsigned int lowerBound = 0;
8      unsigned int upperBound = mya.size - 1;
9      while (lowerBound <= upperBound) {
10         unsigned int midpt = (lowerBound + upperBound) / 2;
11         if (mya.arr[midpt] < valSearchFor)
12             lowerBound = midpt + 1;
13         else if (mya.arr[midpt] == valSearchFor)
14             return midpt;
15         else /* mya.arr[midpt] > valSearchFor */
16             upperBound = midpt - 1;
17     }
18     return -1;
19 }
```

Notes

binarySearch()

```
1  #include "binarySearch.h"
2  #include <iostream>
3
4  int binarySearch(MyArray const &mya, int valSearchFor)
5  {
6      /* return (any) position if valSearchFor is in sorted
           mya.arr[0..size-1] or -1 if valSearchFor is not
           present */
7      unsigned int lowerBound = 0;
8      unsigned int upperBound = mya.size - 1;
9      while (lowerBound <= upperBound) {
10         unsigned int midpt = (lowerBound + upperBound) / 2;
11         if (mya.arr[midpt] < valSearchFor)
12             lowerBound = midpt + 1;
13         else if (mya.arr[midpt] == valSearchFor)
14             return midpt;
15         else /* mya.arr[midpt] > valSearchFor */
16             upperBound = midpt - 1;
17     }
18     return -1;
19 }
```

Notes

binarySearch()

```
1  #include "binarySearch.h"
2  #include <iostream>
3
4  int binarySearch(MyArray const &mya, int valSearchFor)
5  {
6      /* return (any) position if valSearchFor is in sorted
           mya.arr[0..size-1] or -1 if valSearchFor is not
           present */
7      unsigned int lowerBound = 0;
8      unsigned int upperBound = mya.size - 1;
9      while (lowerBound <= upperBound) {
10         unsigned int midpt = (lowerBound + upperBound) / 2;
11         if (mya.arr[midpt] < valSearchFor)
12             lowerBound = midpt + 1;
13         else if (mya.arr[midpt] == valSearchFor)
14             return midpt;
15         else /* mya.arr[midpt] > valSearchFor */
16             upperBound = midpt - 1;
17     }
18     return -1;
19 }
```

Notes

binarySearch()

```
1  #include "binarySearch.h"
2  #include <iostream>
3
4  int binarySearch(MyArray const &mya, int valSearchFor)
5  {
6      /* return (any) position if valSearchFor is in sorted
           mya.arr[0..size-1] or -1 if valSearchFor is not
           present */
7      unsigned int lowerBound = 0;
8      unsigned int upperBound = mya.size - 1;
9      while (lowerBound <= upperBound) {
10         unsigned int midpt = (lowerBound + upperBound) / 2;
11         if (mya.arr[midpt] < valSearchFor)
12             lowerBound = midpt + 1;
13         else if (mya.arr[midpt] == valSearchFor)
14             return midpt;
15         else /* mya.arr[midpt] > valSearchFor */
16             upperBound = midpt - 1;
17     }
18     return -1;
19 }
```

Notes

binarySearch()

```
1  #include "binarySearch.h"
2  #include <iostream>
3
4  int binarySearch(MyArray const &mya, int valSearchFor)
5  {
6      /* return (any) position if valSearchFor is in sorted
           mya.arr[0..size-1] or -1 if valSearchFor is not
           present */
7      unsigned int lowerBound = 0;
8      unsigned int upperBound = mya.size - 1;
9      while (lowerBound <= upperBound) {
10         unsigned int midpt = (lowerBound + upperBound) / 2;
11         if (mya.arr[midpt] < valSearchFor)
12             lowerBound = midpt + 1;
13         else if (mya.arr[midpt] == valSearchFor)
14             return midpt;
15         else /* mya.arr[midpt] > valSearchFor */
16             upperBound = midpt - 1;
17     }
18     return -1;
19 }
```

Notes

binarySearch()

```
1  #include "binarySearch.h"
2  #include <iostream>
3
4  int binarySearch(MyArray const &mya, int valSearchFor)
5  {
6      /* return (any) position if valSearchFor is in sorted
           mya.arr[0..size-1] or -1 if valSearchFor is not
           present */
7      unsigned int lowerBound = 0;
8      unsigned int upperBound = mya.size - 1;
9      while (lowerBound <= upperBound) {
10         unsigned int midpt = (lowerBound + upperBound) / 2;
11         if (mya.arr[midpt] < valSearchFor)
12             lowerBound = midpt + 1;
13         else if (mya.arr[midpt] == valSearchFor)
14             return midpt;
15         else /* mya.arr[midpt] > valSearchFor */
16             upperBound = midpt - 1;
17     }
18     return -1;
19 }
```

Notes

binarySearch()

```
1  #include "binarySearch.h"
2  #include <iostream>
3
4  int binarySearch(MyArray const &mya, int valSearchFor)
5  {
6      /* return (any) position if valSearchFor is in sorted
           mya.arr[0..size-1] or -1 if valSearchFor is not
           present */
7      unsigned int lowerBound = 0;
8      unsigned int upperBound = mya.size - 1;
9      while (lowerBound <= upperBound) {
10         unsigned int midpt = (lowerBound + upperBound) / 2;
11         if (mya.arr[midpt] < valSearchFor)
12             lowerBound = midpt + 1;
13         else if (mya.arr[midpt] == valSearchFor)
14             return midpt;
15         else /* mya.arr[midpt] > valSearchFor */
16             upperBound = midpt - 1;
17     }
18     return -1;
19 }
```

Notes

binarySearch()

```
1  #include "binarySearch.h"
2  #include <iostream>
3
4  int binarySearch(MyArray const &mya, int valSearchFor)
5  {
6      /* return (any) position if valSearchFor is in sorted
           mya.arr[0..size-1] or -1 if valSearchFor is not
           present */
7      unsigned int lowerBound = 0;
8      unsigned int upperBound = mya.size - 1;
9      while (lowerBound <= upperBound) {
10         unsigned int midpt = (lowerBound + upperBound) / 2;
11         if (mya.arr[midpt] < valSearchFor)
12             lowerBound = midpt + 1;
13         else if (mya.arr[midpt] == valSearchFor)
14             return midpt;
15         else /* mya.arr[midpt] > valSearchFor */
16             upperBound = midpt - 1;
17     }
18     return -1;
19 }
```

Notes

binarySearch()

```
1  #include "binarySearch.h"
2  #include <iostream>
3
4  int binarySearch(MyArray const &mya, int valSearchFor)
5  {
6      /* return (any) position if valSearchFor is in sorted
           mya.arr[0..size-1] or -1 if valSearchFor is not
           present */
7      unsigned int lowerBound = 0;
8      unsigned int upperBound = mya.size - 1;
9      while (lowerBound <= upperBound) {
10         unsigned int midpt = (lowerBound + upperBound) / 2;
11         if (mya.arr[midpt] < valSearchFor)
12             lowerBound = midpt + 1;
13         else if (mya.arr[midpt] == valSearchFor)
14             return midpt;
15         else /* mya.arr[midpt] > valSearchFor */
16             upperBound = midpt - 1;
17     }
18     return -1;
19 }
```

Notes

binarySearch()

```
1  #include "binarySearch.h"
2  #include <iostream>
3
4  int binarySearch(MyArray const &mya, int valSearchFor)
5  {
6      /* return (any) position if valSearchFor is in sorted
           mya.arr[0..size-1] or -1 if valSearchFor is not
           present */
7      unsigned int lowerBound = 0;
8      unsigned int upperBound = mya.size - 1;
9      while (lowerBound <= upperBound) {
10         unsigned int midpt = (lowerBound + upperBound) / 2;
11         if (mya.arr[midpt] < valSearchFor)
12             lowerBound = midpt + 1;
13         else if (mya.arr[midpt] == valSearchFor)
14             return midpt;
15         else /* mya.arr[midpt] > valSearchFor */
16             upperBound = midpt - 1;
17     }
18     return -1;
19 }
```

Notes

binarySearch()

```
1  #include "binarySearch.h"
2  #include <iostream>
3
4  int binarySearch(MyArray const &mya, int valSearchFor)
5  {
6      /* return (any) position if valSearchFor is in sorted
           mya.arr[0..size-1] or -1 if valSearchFor is not
           present */
7      unsigned int lowerBound = 0;
8      unsigned int upperBound = mya.size - 1;
9      while (lowerBound <= upperBound) {
10         unsigned int midpt = (lowerBound + upperBound) / 2;
11         if (mya.arr[midpt] < valSearchFor)
12             lowerBound = midpt + 1;
13         else if (mya.arr[midpt] == valSearchFor)
14             return midpt;
15         else /* mya.arr[midpt] > valSearchFor */
16             upperBound = midpt - 1;
17     }
18     return -1;
19 }
```

Notes

binarySearch()

```
1  #include "binarySearch.h"
2  #include <iostream>
3
4  int binarySearch(MyArray const &mya, int valSearchFor)
5  {
6      /* return (any) position if valSearchFor is in sorted
           mya.arr[0..size-1] or -1 if valSearchFor is not
           present */
7      unsigned int lowerBound = 0;
8      unsigned int upperBound = mya.size - 1;
9      while (lowerBound <= upperBound) {
10         unsigned int midpt = (lowerBound + upperBound) / 2;
11         if (mya.arr[midpt] < valSearchFor)
12             lowerBound = midpt + 1;
13         else if (mya.arr[midpt] == valSearchFor)
14             return midpt;
15         else /* mya.arr[midpt] > valSearchFor */
16             upperBound = midpt - 1;
17     }
18     return -1;
19 }
```

Notes

`binarySearch()`

```
1  #include "binarySearch.h"
2  #include <iostream>
3
4  int binarySearch(MyArray const &mya, int valSearchFor)
5  {
6      /* return (any) position if valSearchFor is in sorted
           mya.arr[0..size-1] or -1 if valSearchFor is not
           present */
7      unsigned int lowerBound = 0;
8      unsigned int upperBound = mya.size - 1;
9      while (lowerBound <= upperBound) {
10         unsigned int midpt = (lowerBound + upperBound) / 2;
11         if (mya.arr[midpt] < valSearchFor)
12             lowerBound = midpt + 1;
13         else if (mya.arr[midpt] == valSearchFor)
14             return midpt;
15         else /* mya.arr[midpt] > valSearchFor */
16             upperBound = midpt - 1;
17     }
18     return -1;
19 }
```

Notes

Returned to `main`

```
1  #include <iostream>
2  #include "MyArray.h"
3  #include "linearSearch.h"
4  #include "binarySearch.h"
5
6  using namespace std;
7
8  int main()
9  {
10     MyArray ma1;
11     ma1.arr = new int[7];
12     ma1.capacity = 7;
13     ma1.size = 7;
14     for (unsigned int i = 0; i < ma1.size; ++i)
15         ma1.arr[i] = i * 10;
16
17     unsigned int idxofvalue = linearSearch(ma1, 20);
18
19     unsigned int idxofvalue2 = binarySearch(ma1, 20);
20
21     delete [] ma1.arr;
22
23     return 0;
24 }
```

Notes

Returned to `main`

```
1  #include <iostream>
2  #include "MyArray.h"
3  #include "linearSearch.h"
4  #include "binarySearch.h"
5
6  using namespace std;
7
8  int main()
9  {
10     MyArray ma1;
11     ma1.arr = new int[7];
12     ma1.capacity = 7;
13     ma1.size = 7;
14     for (unsigned int i = 0; i < ma1.size; ++i)
15         ma1.arr[i] = i * 10;
16
17     unsigned int idxofvalue = linearSearch(ma1, 20);
18
19     unsigned int idxofvalue2 = binarySearch(ma1, 20);
20
21     delete [] ma1.arr;
22
23     return 0;
24 }
```

Notes

Returned to `main`

```
1  #include <iostream>
2  #include "MyArray.h"
3  #include "linearSearch.h"
4  #include "binarySearch.h"
5
6  using namespace std;
7
8  int main()
9  {
10     MyArray ma1;
11     ma1.arr = new int[7];
12     ma1.capacity = 7;
13     ma1.size = 7;
14     for (unsigned int i = 0; i < ma1.size; ++i)
15         ma1.arr[i] = i * 10;
16
17     unsigned int idxofvalue = linearSearch(ma1, 20);
18
19     unsigned int idxofvalue2 = binarySearch(ma1, 20);
20
21     delete [] ma1.arr;
22
23     return 0;
24 }
```

Notes

Notes

Notes