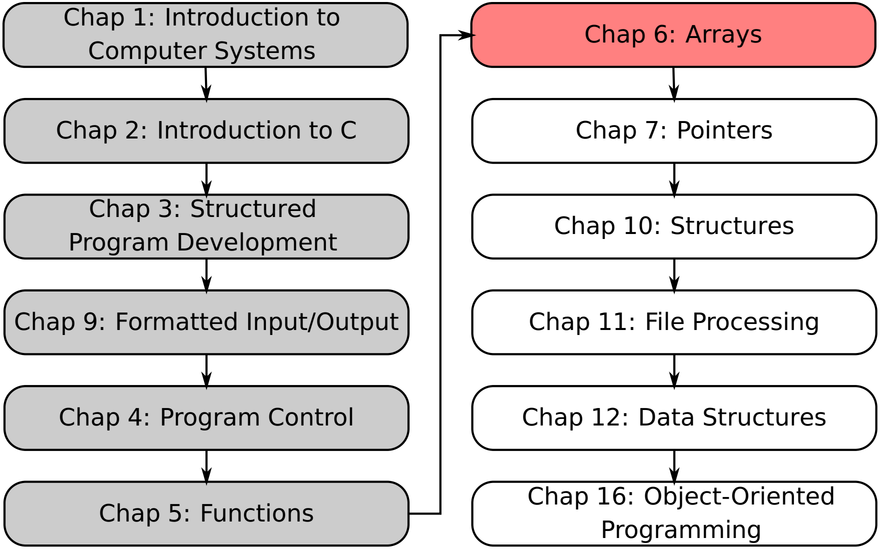
Computer Programming 143 – Lecture 15
Arrays II

Electrical and Electronic Engineering Department
University of Stellenbosch

Prof Johan du Preez
Mr Callen Fisher
Dr Willem Jordaan
Dr Hannes Pretorius
Mr Willem Smit

Copyright & Disclaimer

Module Overview



Chap 1: Introduction to Computer Systems
Chap 2: Introduction to C
Chap 3: Structured Program Development
Chap 9: Formatted Input/Output
Chap 4: Program Control
Chap 5: Functions
Chap 6: Arrays
Chap 7: Pointers
Chap 10: Structures
Chap 11: File Processing
Chap 12: Data Structures
Chap 16: Object-Oriented Programming

Lecture Overview

6.5 Store and Manipulation of Strings I

Character arrays

```
char str1[] = "first";
```

- String **"first"** is really a static array of characters
- Character arrays can be initialized using string literals
  - Null character **'\0'** terminates strings
  - **str1** actually has 6 elements
  - It is equivalent to

```
char str1[] = { 'f', 'i', 'r', 's', 't', '\0'};
```

- Can access individual characters
  - **str1[3]** is character **'s'**
- Array name is address of array, so **&** not needed for **scanf**

  ```
  scanf( "%s", string2 );
  ```

  - Reads characters until whitespace encountered

6.5 Store and Manipulation of Strings II

Problem

- Read a string (array of char) from the keyboard and combine with hard coded string. Also print the read string with spaces inserted between characters.

6.5 Store and Manipulation of Strings III

Pseudocode

```
Declare a character (string1) array of 20 elements
  (used for input, 20 assumed as maximum length)

Initialise a character (string2) array with “string literal”

Prompt the user for a string and read into array string1.
Print string1 and string2

For each character i in string1 up to the ‘\0’ (end of string) character
    print character i of string1 and a space
```

6.5 Store and Manipulation of Strings IV

```c
/* Treating character arrays as strings; Fig. 6.10 in Deitel & Deitel */
#include <stdio.h>

int main( void )
{
   char string1[ 20 ]; // reserves 20 characters
   char string2[] = "string literal"; // reserves 14+1 characters
   int i; // counter

   // read string from user into array string1
   printf( "Enter a string: " );
   scanf( "%s", string1 ); // input ended by whitespace character

   // output strings
   printf( "string1 is: %s\nstring2 is: %s\n",string1, string2 );

   printf(  "string1 with spaces between characters is:\n");
   // output characters until null character is reached
```

6.5 Store and Manipulation of Strings V

```
   for ( i = 0; string1[ i ] != '\0'; i++ ) {
     printf( "%c ", string1[ i ] );
   } // end for

   printf( "\n" );
   return 0; // indicates successful program termination
} // end main
```

Output

```
Enter a string: Hello there
string1 is: Hello
string2 is: string literal
string1 with spaces between characters is:
H e l l o
```

6.7 Passing Arrays to Functions I

Passing arrays

- To pass an array argument to a function, specify the name of the array without any brackets

```
        int myArray[ 24 ];
⋮
        myFunction( myArray, 24 );
```

- Array arguments passed call-by-reference
- Name of array is address of first element
- Function “knows” *where* the array is stored, but not array size
- Therefore, we usually pass the array size as a separate argument

Any changes modifies the data at original memory locations

because arrays are passed by reference

6.7 Passing Arrays to Functions II

Function prototype

- Prototype for a function that takes an array as argument:

```
void myFunction( int b[], int arraySize );
```

  - Specifies that the first argument of function **myFunction** is an array of integers
- To prevent the function from modifying the array, use the keyword **const**:

```
void myFunction( const int b[], int arraySize );
```

  - See program listing in Fig. 6.14 in Deitel & Deitel

6.7 Passing Arrays to Functions III

Passing array elements

- Passed by call-by-value
- Pass subscripted name (i.e., **`myArray[ 3 ]`**) to function
- Value of the element is copied into the parameter of the function
- Original element in array is unaffected by function

Refer to Fig. 6.13 in Deitel & Deitel for example of passing arrays and array elements to functions

6.10 Searching Arrays I

Introduction

- When working with large amounts of data
- Search to see if it can match one of the array values to a **key value**
- Two searching techniques
  - Linear search
  - Binary search

6.10 Searching Arrays II

Linear search (Search array for **key value**)

- Compare each element of array with key value
- Useful for small and unsorted arrays
- Assumes unique key values, e.g. student numbers
- Array does not have to be sorted

6.10 Searching Arrays III

```
int linearSearch( const int array[], int key, int size )
{
    int n = 0; // counter
    int keyLocation = -1;  // store location of key

    // loop through array
  do{
        if (array[ n ] == key ) {
            keyLocation = n; // stores location of key
        } // end if
        n++;
    } while((keyLocation == -1) && (n < size));  //end do...while

    return keyLocation;
} // end function linearSearch
```

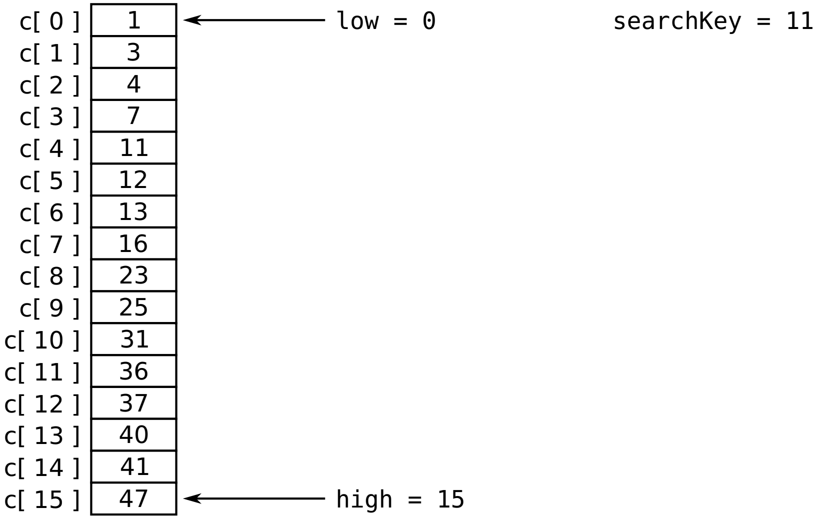
See Fig. 6.18 in Deitel & Deitel for an example of linear search

6.10 Searching Arrays IV

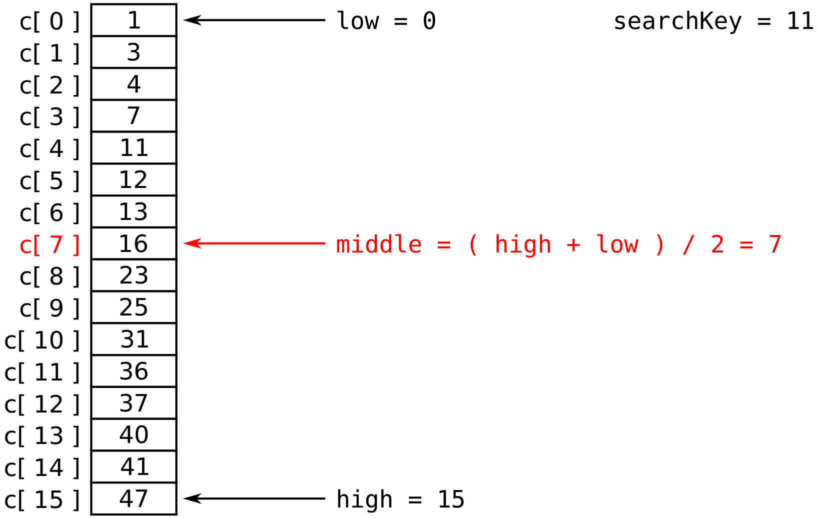
Binary search

- Only for arrays sorted by key
- Compares **`middle`** element with **`key`**
  - If equal, match found
  - If **`key < middle`**, looks further in lower half of array
  - If **`key > middle`**, looks further in upper half of array
  - Repeat
- Very fast; at most n steps, where $2^n >$ number of elements
  - A 30 element array takes at most 5 steps
  - $2^4 < 30 < 2^5$ so at most 5 steps
- More efficient than linear search

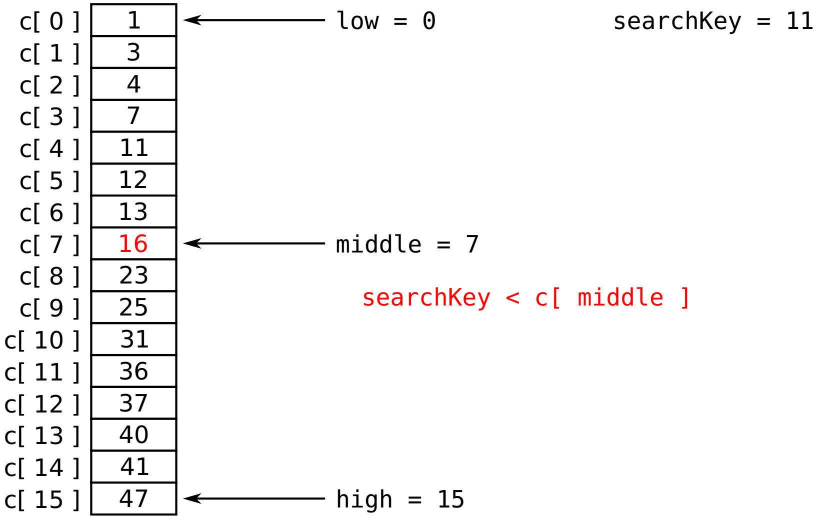
6.10 Binary Search



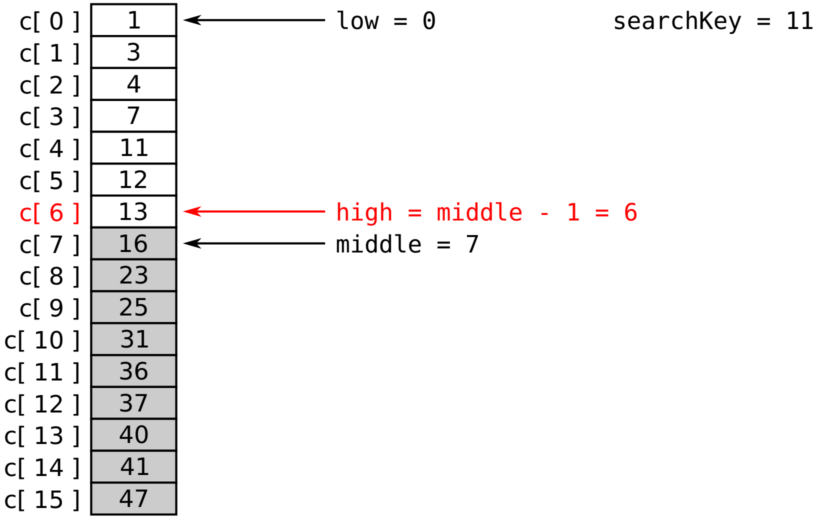
6.10 Binary Search



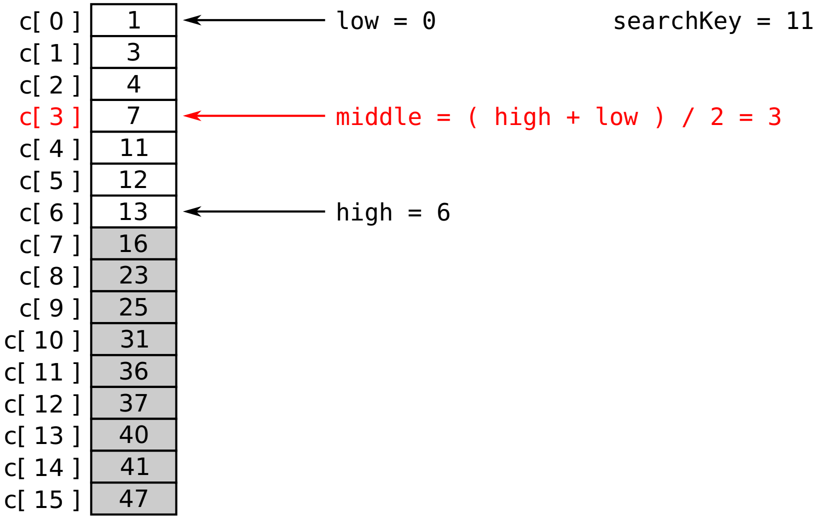
6.10 Binary Search



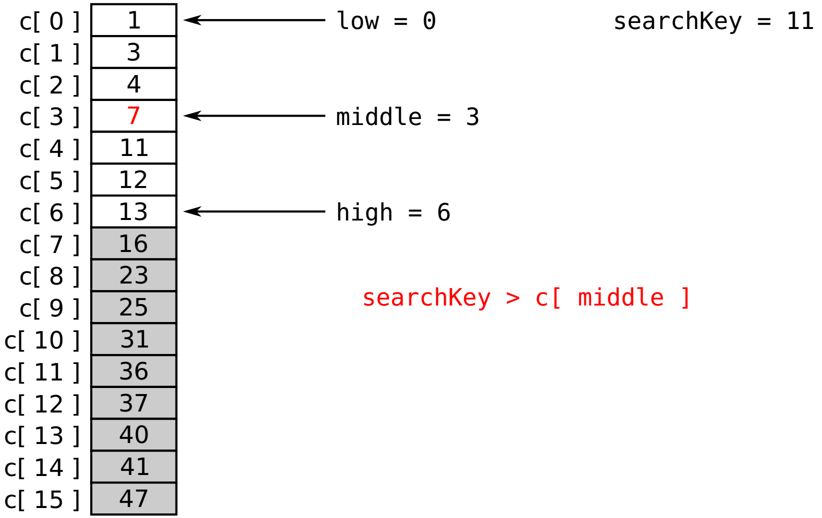
6.10 Binary Search



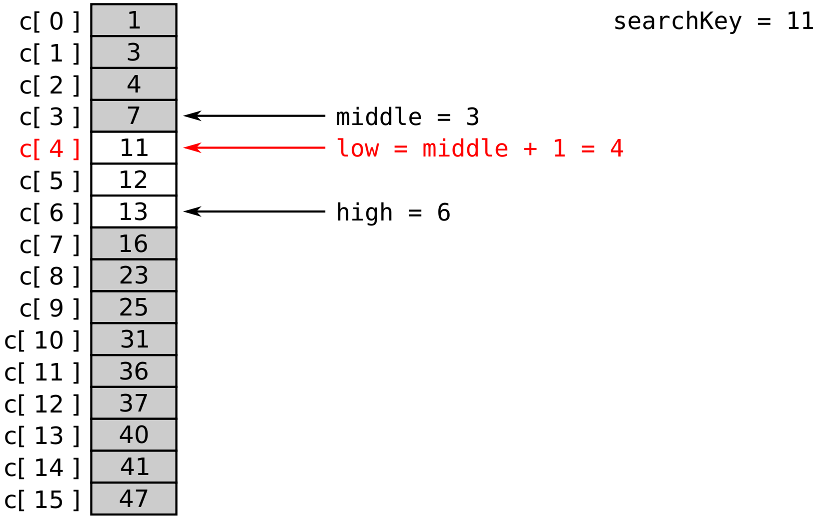
6.10 Binary Search



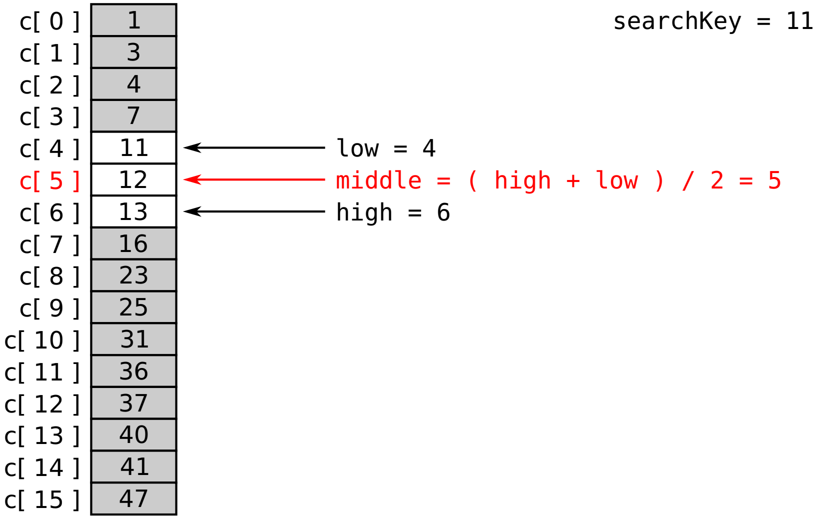
6.10 Binary Search



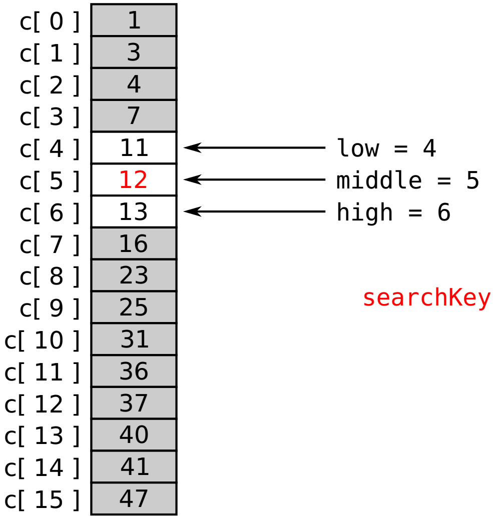
6.10 Binary Search



6.10 Binary Search



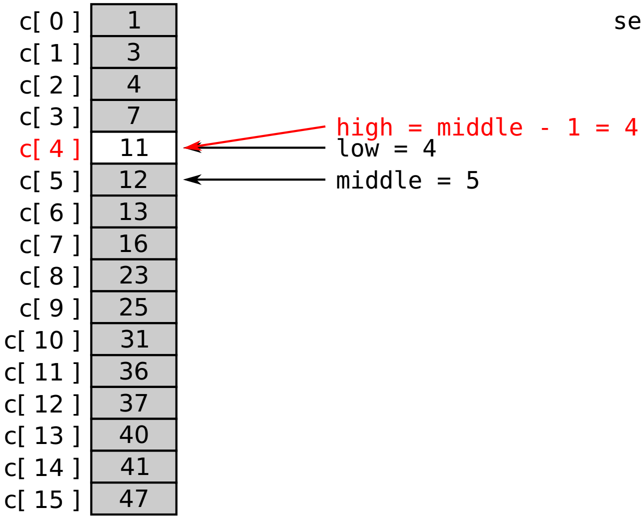
6.10 Binary Search



`searchKey = 11`

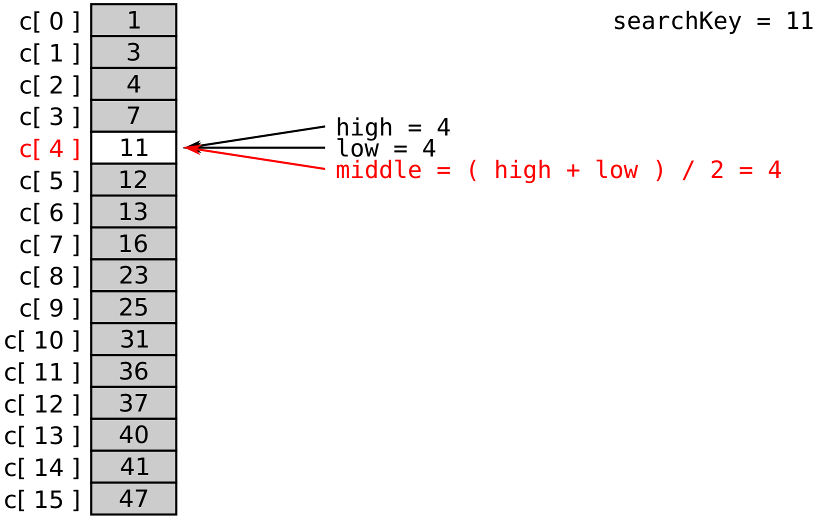
`searchKey < c[ middle ]`

6.10 Binary Search

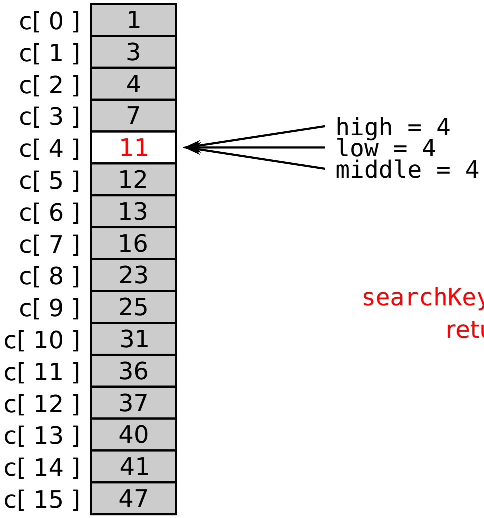


searchKey = 11

6.10 Binary Search



6.10 Binary Search



```
searchKey = 11
```

```
searchKey = c[ middle ]
```
return middle

```c
/* function to perform binary search of an array */
int binarySearch( const int b[], int searchKey, int low, int high )
{
   int middle;
   int keyLocation = -1;

   while ((keyLocation == -1) && (low <= high)) {

      middle = ( low + high ) / 2;  /* get middle element*/

      if ( searchKey == b[ middle ] ) {
            keyLocation = middle;
      } else if ( searchKey < b[ middle ] ) {
            high = middle - 1;  /* search low end of array */
      } else {
            low = middle + 1;   /* search high end of array */
      }
   }
   return keyLocation;                  /* searchKey not found */
}
```

See Fig. 6.19 in Deitel & Deitel for an example of binary search

Perspective

Today

Arrays II

- Character arrays
- Passing arrays to functions
- Searching arrays

Next lecture

Arrays III

- Sorting arrays

Homework

1. Study Sections 6.5, 6.7 and 6.10 in Deitel & Deitel
2. Do Self Review Exercise 6.2(e)
3. Do Exercises 6.6(a)-(f), 6.33