CIS 129
Advanced Computer Programming

Chapter 5: Arrays and Strings

Mr. Horence Chan

Arrays

- An array is a fixed number of elements of the same type stored sequentially in memory.
- E.g. an integer array holds some number of integers, a character array holds some number of characters.
- The size of the array is referred to as its dimension.
- To declare an array in C++, we write the following:
- `type arrayName[dimension]`

Arrays

- Arrays in C++ are zero-indexed, so the first element has an index of _____
- Example of declare and initialize array:

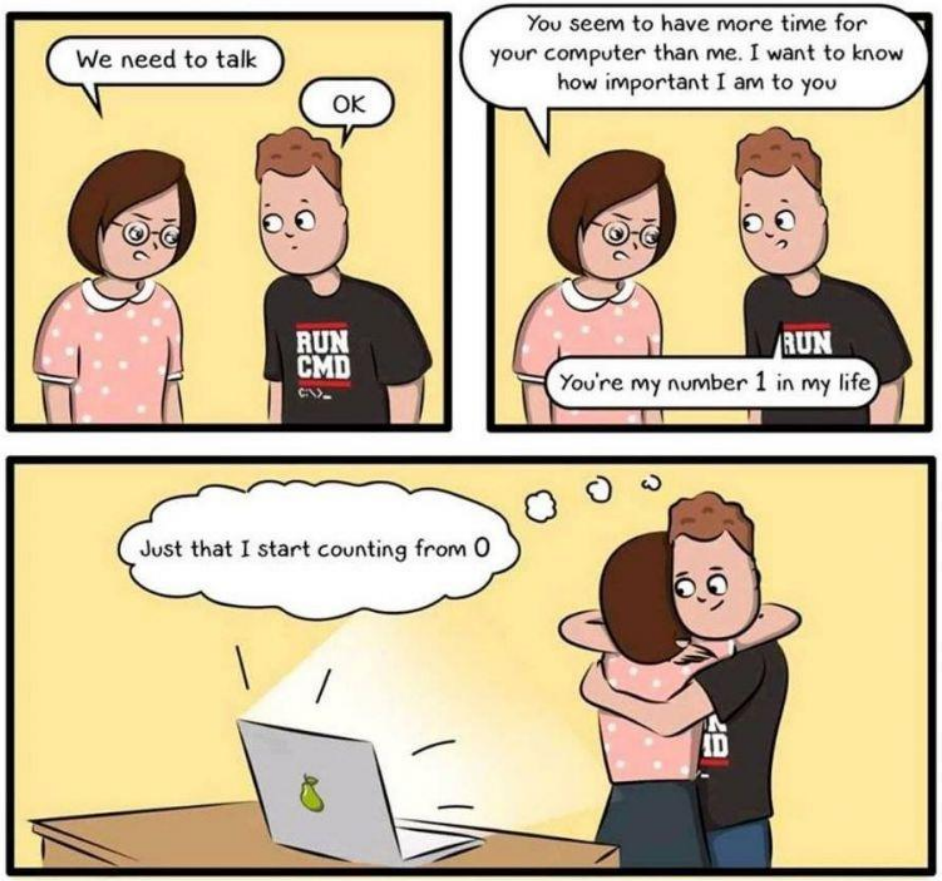
```
int arr[4];
arr[___] = 6;
arr[___] = 0;
arr[___] = 9;
arr[___] = 6;
```

or

```
int arr[4] = { 6, 0, 9, 6 };
```

- We can also leave out the size of the array and let the compiler determine the array's size for us, based on how many elements we give it:

```
int arr[] = { 6, 0, 9, 6, 2, 0, 1, 1 };
```



Arrays

```
#include <iostream>
using namespace std;
int main() {
        int arr[4];
        cout << "Please enter 4 integers: " << endl;
        for(int i = 0; i < 4; i++){
                cin >> arr[i];
        }
        cout << "Values in array are now: ";
        for(int i = 0; i < 4; i++){
                cout << " " << arr[i];
        }
        cout << endl;
        return 0;
}
```

- This program shows how the user can input some values to an array
- Also print the elements of an array
- Note that when accessing an array the index given must be a _____ from 0 to n-1, where n is the dimension of the array.
- Other example of accessing elements in an array:
  - `arr[5];`
  - `arr[i];`
  - `arr[i+3];`

Arrays

```
#include <iostream>
using namespace std;
int sum(const int array[], const int length);
int main() {
        int arr[] = {1, 2, 3, 4, 5, 6, 7};
        cout << "Sum: " << sum(arr, 7) << endl;
        return 0;
}
int sum(const int array[], const int length) {
        long sum = 0;
        for(int i = 0; i < length; sum += array[i++]);
        return sum;
}
```

- This program calculate the sum of an array, where output is `Sum: 28`
- Note that `sum += array[i++]` has the same meaning of _____
- `i++` means _____

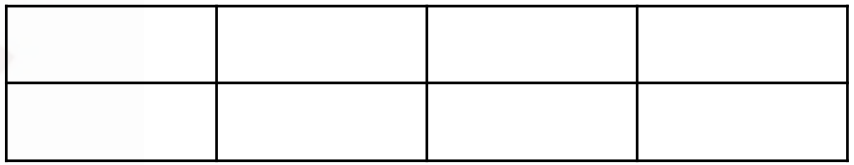
- The for loop in the function simplify some steps when calculating the sum, here is a longer expression:

```
for(int i = 0; i < length; i++){
        sum += array[i];
}
```

Multidimensional Arrays

- Two-dimensional array:
- `type arrayName[dimension1][dimension2];`
- Example of declare and initialize two-dimensional array:

```
int twoDimArray[2][4];
twoDimArray[0][0] = 6;
twoDimArray[0][1] = 0;
twoDimArray[0][2] = 9;
twoDimArray[0][3] = 6;
twoDimArray[1][0] = 2;
twoDimArray[1][1] = 0;
twoDimArray[1][2] = 1;
twoDimArray[1][3] = 1;
```



Multidimensional Arrays

- The array can also be initialized at declaration in the following ways:
  - `int twoDimArray[2][4] = { 6, 0, 9, 6, 2, 0, 1, 1 };` or
  - `int twoDimArray[2][4] = _____;`

- Print out the above the array using this loop:

```
for(int i = 0; i < 2; i++){
    for(int j = 0; j < 4; j++){
            cout << twoDimArray[i][j];
    }
cout << endl;
}
```

- The output is an array with two rows and four columns:

6 0 9 6
2 0 1 1

Multidimensional Arrays

- Note that _____ must *always* be provided when initializing multidimensional arrays (the first dimension is optional)
  - e.g. `int aFunction(int arr[][4]) { … }`

- Declare multidimensional arrays help programmers to identify the elements in the program
- As all of the elements in the array are sequential in computer memory.
- For example, computer memory consider `int arr[2][4];` and `int arr[8]` are the _____

Character Arrays

```
#include <iostream>
using namespace std;
int main() {

char itsfine[] = { 'G', 'w',
'a', 'e', 'n', ' ', 'C', 'h',
'a', ' ', 'N', 'a', '!', '\0' };

cout << itsfine << endl;
      return 0;
}
```

- Output: _____
- \0: _____, this character is used to indicate the end of the string.
- Character arrays can also be initialized using string literals. In this case, no null character is needed, as the compiler will automatically insert one:
  - `char itsfine[] = "Gwaen Cha Na!";`

Character Arrays

- When manipulating character arrays, the following functions may need to include in a program through the use of the `#include` directive:
  - cctype: character handling
  - cstdio: input/output operations
  - cstdlib: general utilities
  - cstring: string manipulation

```
#include <iostream>
#include <cctype>
using namespace std;
int main (){
    char messyString[] = "t6H0I9s6.iS.999a9.STRING";
    char current= messyString[0];
    for (int i = 0; current != '\0'; i++){
        current = messyString[i];
        if (isalpha (current)){
            if(isupper (current)){
                current = tolower (current);
            }
        cout << current;
        }
        else if (ispunct (current)){
            cout << ' ';
        }
    }
    cout << endl;
    return 0;}
```

- Functions from `cctype` library:
  - `isalpha`: Check if character is _____
  - `isupper`: Check if character is _____ letter
  - `ispunct`: Check if character is a _____ character
  - `tolower`: Convert _____ letter to _____

```
#include <iostream>
#include <cctype>
using namespace std;
int main (){
    char messyString[] = "t6H0I9s6.iS.999a9.STRING";
    char current= messyString[0];
    for (int i = 0; current != '\0'; i++){
        current = messyString[i];
        if (isalpha (current)){
            if(isupper (current)){
                current = tolower (current);
            }
        cout << current;
        }
        else if (ispunct (current)){
            cout << ' ';
        }
    }
    cout << endl;
    return 0;}
```

- On each iteration of the for loop:
  - If the current character is _____ and _____, it is converted to _____ and then displayed.
  - If it is already lowercase it is simply displayed.
  - If the character is a _____ mark, a _____ is displayed.
- All other characters are ignored.
- Output: _____

Character Arrays

```
#include <iostream>
#include <cstring>
using namespace std;
int main() {
        char fragment1[] = "I'm a s";
        char fragment2[] = "tring!";
        char fragment3[20];
        char finalString[20] = "";
        strcpy(fragment3, fragment1);
        strcat(finalString, fragment3);
        strcat(finalString, fragment2);
        cout << finalString;
        return 0;
}
```

- Functions from `cstring` library:
  - `Strcpy(a, b):` _____ character array from b to a
  - `Strcat(a, b):` _____ character arrays, and store to a
- Output: _____

Reference of Array

- As parameters to functions, arrays are passed by _____ only.
- Because as parameters, arrays are passed by _____ only, when declaring an array as a formal parameter, you do not use the symbol _____ after the data type.
- Example:
  - `void getName(string& Names[]);` ✘
  - `void getName(string Names[]);` ✓

get()

```
#include <iostream>
#include <fstream>
using namespace std;
int main(){
   ifstream infile;
   char ch;
   infile.open("textfile.txt");

   _______________
   while (infile){
         cout << ch << " ";

         _______________
   }
   infile.close();
   return 0;
}
```

- get(): receive input from a file _____ by _____
- Input file:
- Konnichiwa
- Output (in console):

More on Conversation

```
#include <iostream>
#include <fstream>
using namespace std;
int main(){
cout << static_cast<int>('A') << endl;
cout << static_cast<int>('B') << endl;
cout << static_cast<char>(67) << endl;
cout << static_cast<char>(68) << endl;
return 0;
}
```

Output:

- Conversation between character and integer follows _____

Decimal	Hex	Char	Decimal	Hex	Char
64	40	@	96	60	`
65	41	A	97	61	a
66	42	B	98	62	b
67	43	C	99	63	c
68	44	D	100	64	d
69	45	E	101	65	e
70	46	F	102	66	f
71	47	G	103	67	g
72	48	H	104	68	h
73	49	I	105	69	i
74	4A	J	106	6A	j
75	4B	K	107	6B	k
76	4C	L	108	6C	l
77	4D	M	109	6D	m
78	4E	N	110	6E	n
79	4F	O	111	6F	o
80	50	P	112	70	p
81	51	Q	113	71	q
82	52	R	114	72	r
83	53	S	115	73	s
84	54	T	116	74	t
85	55	U	117	75	u
86	56	V	118	76	v
87	57	W	119	77	w
88	58	X	120	78	x
89	59	Y	121	79	y
90	5A	Z	122	7A	z

Letter count

```
int main(){
    ifstream infile;
    char ch;
    int letterCount[4];
    int index, tmp;
    infile.open("textfile.txt");

    ________________
    while (infile){
          index = static_cast<int>(ch)

                                  ____________________________
          letterCount[index]++;

          ________________          }
    infile.close();
...}
```

- Given an input file, we would like to count the occurrence of each letter
- Input file:

abcddaadbd

- Minus each character by the ASCII value of _____
- Therefore, the index value is within the length of an array (_____)
- _____ the occurrence of each character by one

Letter count

```
int main(){
...
   for (index = 0; index < 4; index++){

         tmp = index _____________________________

         cout << _____________________________
              << " count = "
              << letterCount[index]<< endl;
   }
return 0;
}
```

- Input file:

abcddaadbd

Output:

____ count = 3

____ count = 2

____ count = 1

____ count = 4

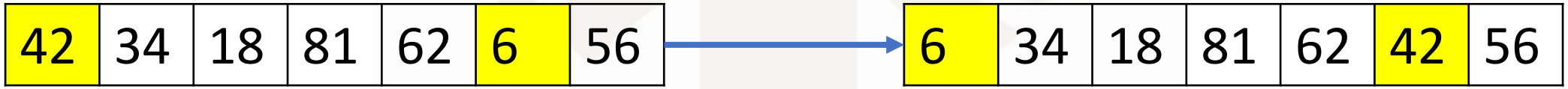
- Find the integer value of a character by adding ASCII value of _____
- Convert the integer to a _____
- _____ of each character

Selection Sort

- For example there is a list of unsorted numbers, and we would like to arrange in ascending order:

42	34	18	81	62	6	56

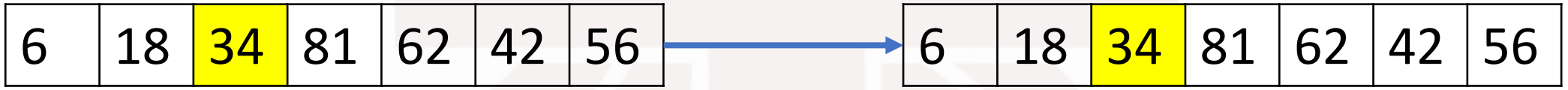
1. Find the _____ number, then swap with the _____ number



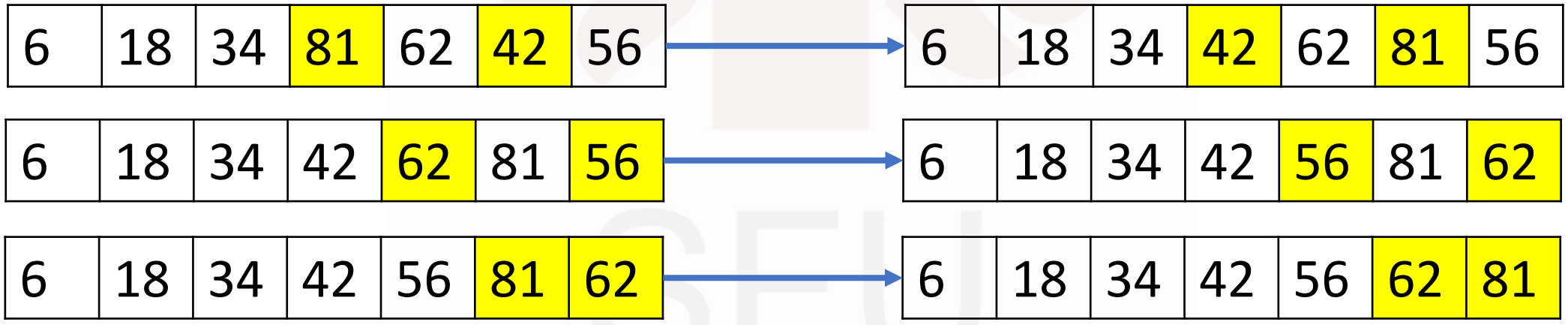
2. Find the _____ number in the remaining list, then swap with the _____ number

Selection Sort

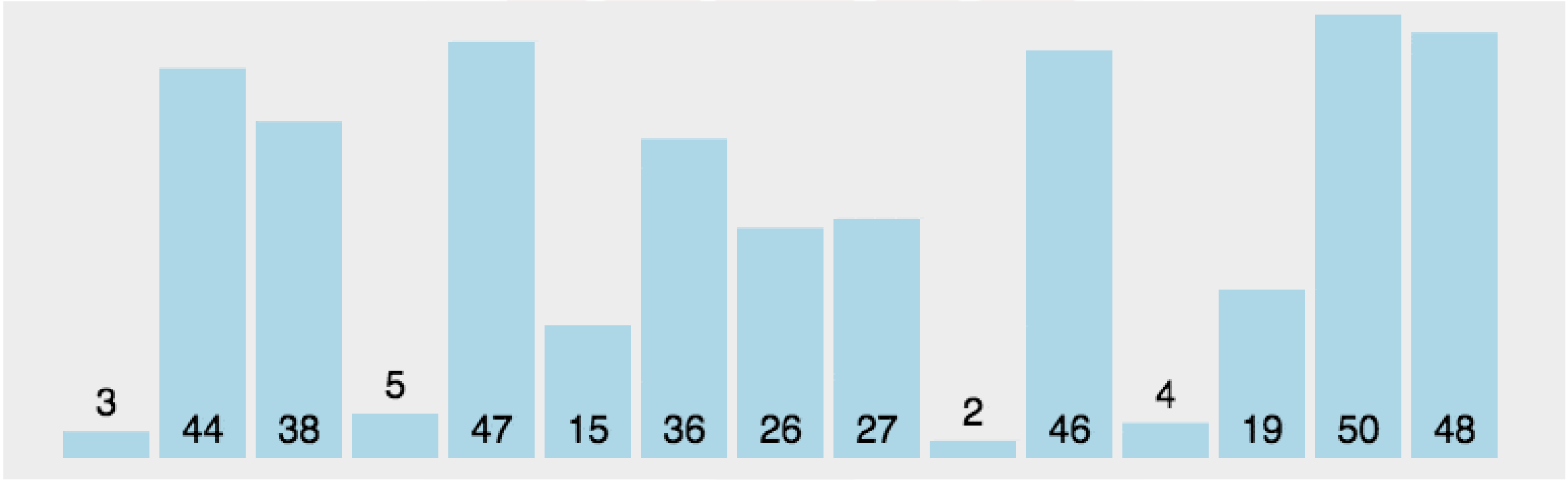
3. Find the _____ number in the remaining list, then swap with the _____ number



4. Continue the sorting process:



Selection Sort



https://www.runoob.com/w3cnote/selection-sort.html

Selection Sort

```
void selectionsort(int num[], int rows){
int i, j, min, temp;
for (i = 0; i < rows - 1; i++){
min = i;
        for (j = i + 1; j < rows; j++){
                if (_____________________){
                min = j;
                }
        }
        temp = _______________;
        ________________________
        _______________= temp;
}
}
```

- This function sort the numbers using selection sort
- This for loop find the _____________ number
- _____________ the smallest number to the front of the array

Selection Sort

```
#include <iostream>
#include <string>
using namespace std;
void selectionsort(int num [], int rows);
int main(){
int arr[7]={42,34,18,81,62,6,56};
int arr_size=7;
selectionsort(arr, arr_size);
      for(int i = 0; i < arr_size; i++){
            cout <<  " " << arr[i] ;
      }
return 0;
}
```

Output:

_____

Selection Sort

```
void selectionsort(string str[], int rows){
int i, j, min;
string temp;
for (i = 0; i < rows - 1; i++){
min = i;
        for (j = i + 1; j < rows; j++){
                if (____________________){
                min = j;
                }
        }
        temp = _______________;
        ________________________
        _______________ = temp;
}
}
```

Decimal	Hex	Char	Decimal	Hex	Char
64	40	@	96	60	`
65	41	A	97	61	a
66	42	B	98	62	b
67	43	C	99	63	c
68	44	D	100	64	d
69	45	E	101	65	e
70	46	F	102	66	f
71	47	G	103	67	g
72	48	H	104	68	h
73	49	I	105	69	i
74	4A	J	106	6A	j
75	4B	K	107	6B	k
76	4C	L	108	6C	l
77	4D	M	109	6D	m
78	4E	N	110	6E	n
79	4F	O	111	6F	o
80	50	P	112	70	p
81	51	Q	113	71	q
82	52	R	114	72	r
83	53	S	115	73	s
84	54	T	116	74	t
85	55	U	117	75	u
86	56	V	118	76	v
87	57	W	119	77	w
88	58	X	120	78	x
89	59	Y	121	79	y
90	5A	Z	122	7A	z

- If we change some "`int`" to "`string`", the function can sort the words in _____ order (_____ order)

Selection Sort

```
#include <iostream>
#include <string>
using namespace std;
void selectionsort(string numbers[], int rows);
int main(){
string name[7]={"Peter", "Tony", "Mary", "Amy", "John", "Alex", "June"};
int arr_size=7;
selectionsort(name, arr_size);
	for(int i = 0; i < arr_size; i++){
		cout << name[i] << endl;
	}
return 0;
}
```

Output:

Linear Search

- Given an array, we would like to find the index of a certain value

```
arr[7]={6,18,34,42,56,62,81}
```

- For example, we would like to find the index of "56"
  - Compare the ____________ and compare it to the target (56).
  - If the item is at the same, we will return the position of the _________________. Otherwise, we will move to the _________________.

value	6	18	34	42	56	62	81
index	0	1	2	3	4	5	6

value = 56 ?

value	6	18	34	42	56	62	81
index	0	1	2	3	4	5	6

value = 56 ?

value	6	18	34	42	56	62	81
index	0	1	2	3	4	5	6

value = 56 ?

value	6	18	34	42	56	62	81
index	0	1	2	3	4	5	6

value = 56 ?

value	6	18	34	42	56	62	81
index	0	1	2	3	4	5	6

value = 56 ?

Linear Search

```
int linearSearch(int sortnum[], int noOfRows, int target) {
    bool found = false;
    for(int i = 0; i< noOfRows; i++){
        if (sortnum[i] == target) {
            found = true;
            return i;
        }
    }
    return -1;
}
```

- If the array size is larger, it may take a lot of time to find the target

when you implement a linear search into your database instead of a binary search

Binary Search

- Given a sorted array, we would like to find the index of a certain value

value	6	18	34	42	56	62	81
index	0	1	2	3	4	5	6

- For example, we would like to find the index of “56”

1. Compare “56” with the _____ value (“42”)
2. Since 56 > 42, so the index value should be in the range of index _____

Binary Search

3. Compare “56” with the _____ number of the remaining values (“62”)

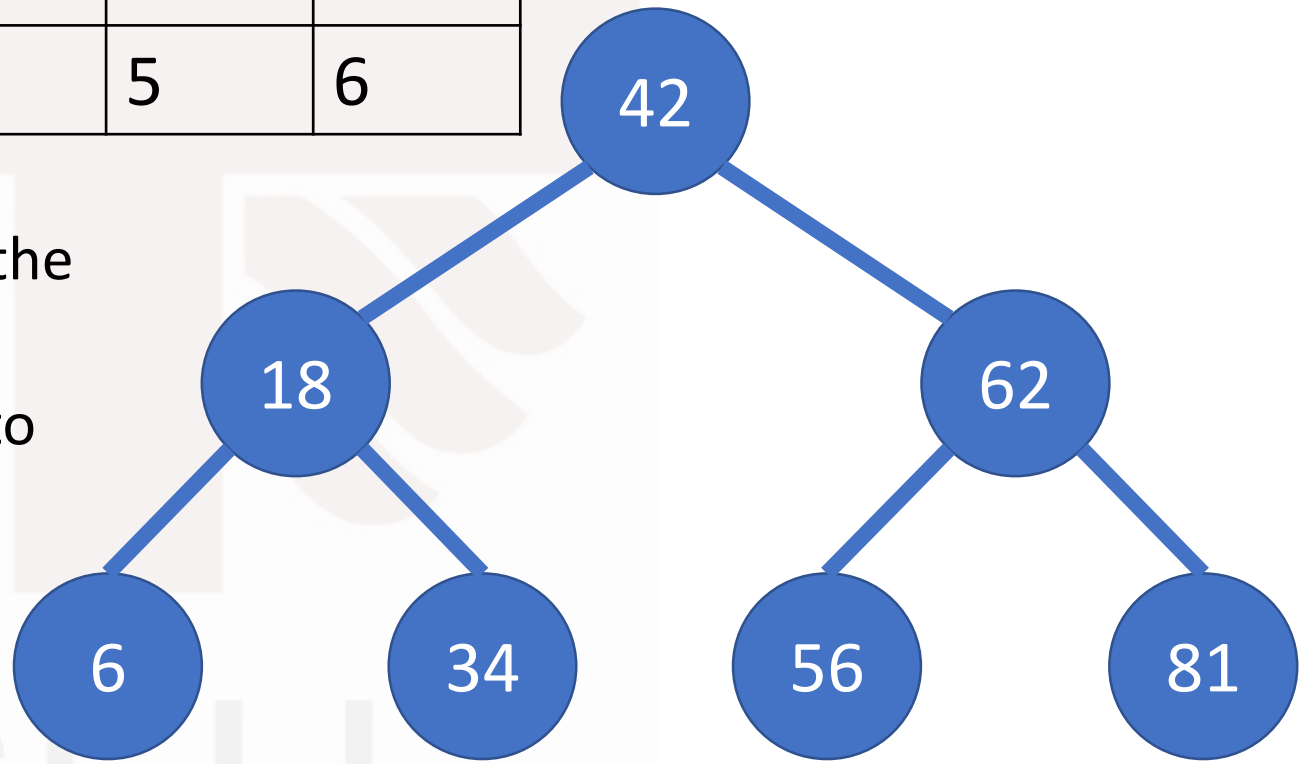
value	~~6~~	~~18~~	~~34~~	~~42~~	56	62	81
index	~~0~~	~~1~~	~~2~~	~~3~~	4	5	6

4. Since 56 < 62, the index of “56” should be _____ than 5
5. Therefore, the index of “56” is 4.

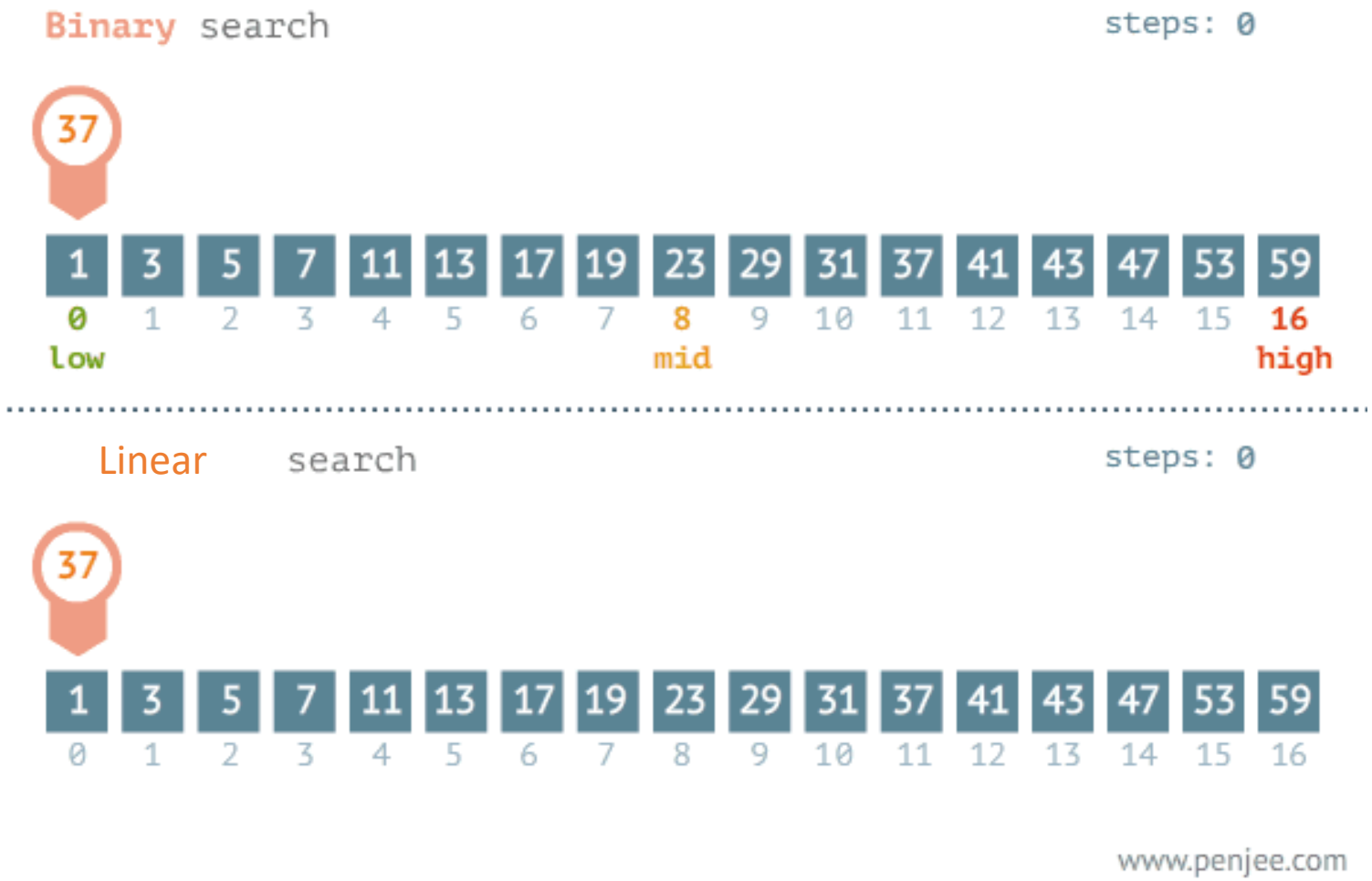
Binary Search

value	6	18	34	42	56	62	81
index	0	1	2	3	4	5	6

- In binary search, we can represent the array using binary tree
- How many comparisons is needed to find “56” ?



Binary Search vs Linear Search



https://devopedia.org/binary-search

Binary Search

```
int binSearch(int sortnum[], int noOfRows, int target){
int first, last, mid;
bool found;
first = 0;
last = noOfRows - 1;
found = false;
while (!found && first <= last){
        mid = (first + last) / 2;
        if (sortnum[mid] == target){
                found = true;}
        else if (sortnum[mid] > target){
                last = mid - 1;}
        else{
                first = mid + 1;}
    }
```

- This function find the index of a number using binary search
- With some modification, we can also use binary search to find the index of a string array

```
if (found){
        return mid;}
    else{
        return -1;}
}
```

Binary Search

```
#include <iostream>
#include <string>
using namespace std;
int binSearch(int sortnum[], int noOfRows, int target);
int main(){
int idx;
int arr[7]={6,18,34,42,56,62,81};
int arr_size=7;
idx = binSearch(arr, arr_size,56);
            cout << "Index of 56: " << idx << endl;
return 0;
}
```

Output:

Index of 56: 4