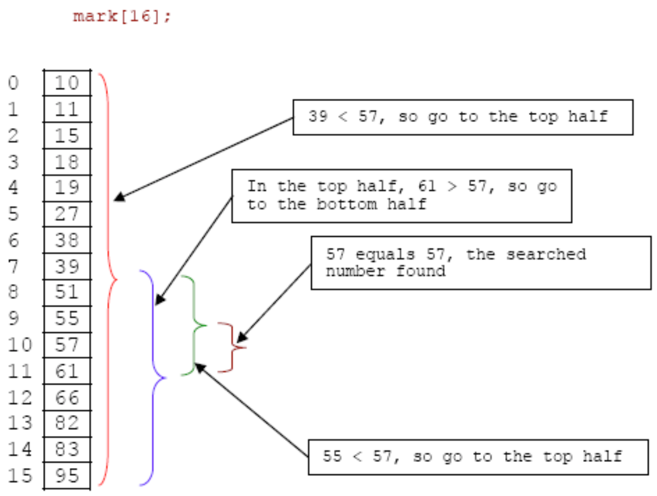
To:
Tenouk

C LAB WORKSHEET 9a_1
C & C++ 1D Array Manipulation Part 4

1. The basic of linear search and binary search.
2. More examples, questions with answers.
3. Tutorial references that should be used together with this worksheet are array part 1 and array part 2.

Binary Search

- Binary searches can be said are more efficient than linear searches for searches involving many elements. The only setback of a binary search is that the elements must be sorted to begin with. However, many other search routines can be implemented efficiently using C++ Standard Template Library (STL).
- In binary search, the first step is to find the midpoint in the array and see the number being searched is less than or greater than the number of the mid-point. Suppose there are 1000 elements in the array. Then see if the number being searched is greater or less than the number at index 500. If the number is greater then search through the top half of the array, otherwise search through the lower half. Then while searching further, continue to apply the same principle of dividing the number of elements to be search by two so the number of searches is cut in half with every search. Hence, it is called a binary search.

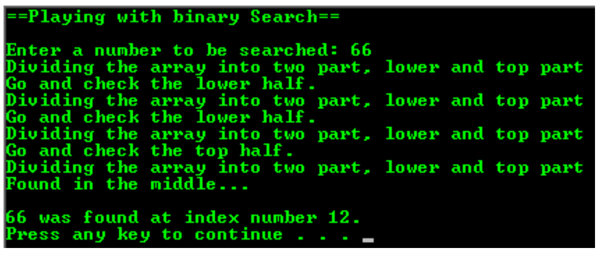

mark[16];
0 10
1 11
2 15
3 18
4 19
5 27
6 38
7 39
8 51
9 55
10 57
11 61
12 66
13 82
14 83
15 95
39 < 57, so go to the top half
In the top half, 61 > 57, so go to the bottom half
57 equals 57, the searched number found
55 < 57, so go to the top half

- In the above Figure, we can see how 57 is searched through the given array. Firstly, the number in the middle of the array, 39 is determined to be less than 57, so the first half/bottom/lower of the array should be ignored. The search is this limited to the range of numbers consisting of the top half of the array. The number in the middle of the top half of the array is 61, which is greater than 57, so the search now is limited to the lower half of that range of the array. This process is continued until the number is found.

```
#include <stdio.h>

void main()
{
    int mark[16] = {10, 11, 15, 18, 19, 27, 38, 39, 51, 55, 57, 61, 66, 82, 83, 95};
    int low = 0,      // lowest index in the range that is being searched
    mid,              // index of the middle in the range
    high = 15,        // index of the highest in the range
    answer = -1,  // this will be the index if lookup is found
    lookup;           // this is what we are searching for in mark[ ]
    // firstly, get the number to be searched.
    printf("==Playing with binary Search==\n");
    printf("\nEnter a number to be searched: ");
    scanf_s("%d", &lookup);
    // start the binary search
    for(; high - low > 1; )
    {
        // find the middle of the range
        printf("Dividing the array into two part, lower and top part\n");
        mid = (high - low) / 2 + low;
        // if the number is at the midpoint, set the answer
        // and break out of loop
        if(mark[mid] == lookup)
        {
            printf("Found in the middle...\n");
            answer = mid;
            break;
        }
        // if the number is not at the midpoint, adjust
        // either high or low
        if(mark[mid] > lookup)
```

A sample input and output.



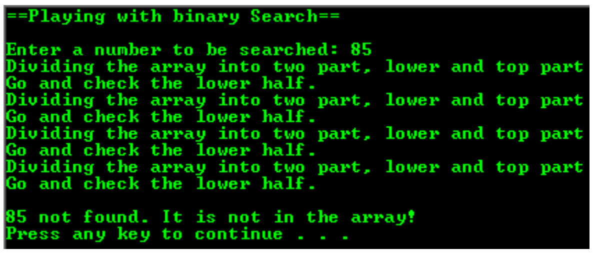
```
==Playing with binary Search==

Enter a number to be searched: 66
Dividing the array into two part, lower and top part
Go and check the lower half.
Dividing the array into two part, lower and top part
Go and check the lower half.
Dividing the array into two part, lower and top part
Go and check the top half.
Dividing the array into two part, lower and top part
Found in the middle...

66 was found at index number 12.
Press any key to continue . . .
```

```
        {
            high = mid;
            printf("Go and check the top half.\n");
        }
        else
        {
            low = mid;
            printf("Go and check the lower half.\n");
        }
    }

    // if the number at high or low, set the answer
    if(answer == -1)
        if(mark[high] == lookup)
        {
            printf("Found in the top half.\n");
            answer = high;
        }
        else
            if(mark[low] == lookup)
            {
                printf("Found in the lower half.\n");
                answer = low;
            }
    // if answer is still unchanged, the number was not found in the array
    if(answer == -1)
        printf("\n%d not found. It is not in the array!\n", lookup);
    else
        printf("\n%d was found at index number %d.\n", lookup, answer);
}
```



```
==Playing with binary Search==

Enter a number to be searched: 85
Dividing the array into two part, lower and top part
Go and check the lower half.
Dividing the array into two part, lower and top part
Go and check the lower half.
Dividing the array into two part, lower and top part
Go and check the lower half.
Dividing the array into two part, lower and top part
Go and check the lower half.

85 not found. It is not in the array!
Press any key to continue . . .
```

- From the previous code, find 3 things of the program that you don't understand. Then discuss with your group members to find the answers. Provide the questions and its answers.

More Practice

1. Answer the following question.

```
int a[5] = {3, 7, 4, 9, 6};
```

Here we have an array with 5 elements. The name of the array is a. It is an array because it is defined with a set of (square) brackets. The 5 inside the brackets indicates that it has 5 elements numbered from 0 to 4. The keyword, int means that each of these 5 indexes holds an integer. a[0] is initialized to 3, a[1] is initialized to 7 and so on.

a. How is a[4] initialized?
b. Is a[5] initialized

a. a[4] initialized to 6.
b. No. a[5] is the sixth element while the array's size just 5.

2. Index/subscript is the term used to refer to a slot number. Answer the following questions based on the previous question.

a. What is the index where 4 is stored in the array?
b. What is the index where 9 is stored?
c. What is the highest index for a[ ]? What is the lowest index for a[ ]?
d. What is the index in the following expression: a[0]? What is the value stored there?

a. The index is 2, a[2] = 4.
b. The index is [3], a[3] = 9.
c. The highest index is 4 and the lowest index is 0.
d. Index is 0 and the value stored is 3.

3. Refer to the following array and answer the questions.

```
int a[5] = {3, 7, 4, 9, 6};
```

a. What is a[1] + a[4]?
b. What is a[0] – a[2]?
c. What is a[0] + 2?
d. What is a[1 + a[0]]?

a. a[1] + a[4] = 7 + 6 = 13.
b. a[0] - a[2] = 3 - 4 = -1.
c. a[0] + 2 = 3 + 2 = 5.
d. a[1 + a[0]] = a[1 + 3] = a[4] = 6.

4. Use the following code and answer the questions.

```
int a[5] = {3, 7, 4, 9, 6};
  for(i = 0; i <= 4; i = i + 1)
        printf("%d\t", a[4 - i]);
```

a. In this code, when i is 0, the value of a[4 – 0] is printed. What is that value?
b. When i is 1, the value of a[4 – 1] is printed. What is that value?
c. When i is 4, what is printed?
d. In conclusion, what is the output of this code?

a. a[4] = 6. 6 was printed.
b. a[3] = 9. 9 was printed.
c. a[4-4] = a[0] = 3. 3 was printed.
d. The output is the array element printed in reverse order that is 6, 9, 4, 7, 3.

5. To print the array in reverse order for the previous question, how should the for statement be written for this printf()? Think about where i should start and where it should end.

```
for( __________ ; __________ ; ___________ )
printf("%d\t", a[i]);
```

```
for( i=4; i>=0; i=i-1)
printf("%d\t", a[i]);
```

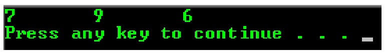
6. How would you write the if statement for the code if all the elements greater than 5 are to be printed in order. That is, first see if 3 is greater than 5. Since it is not, don't print it. Then see if 7 is greater than 5. It is, so print it, etc. The final output then should be 7 9 6.

```
#include <stdio.h>

void main()
{
   int a[5] = {3, 7, 4, 9, 6}, i;
   for(i = 0; i <= 4; i = i + 1)
         if( ________________________ )
               printf("%d\t", a[i]);
}
```

```
#include <stdio.h>

void main()
{
   int a[5] = {3, 7, 4, 9, 6}, i;
   for(i = 0; i <= 4; i = i + 1)
         if( a[i] > 5)
               printf("%d\t", a[i]);
   printf("\n");
}
```



```
7       9       6
Press any key to continue . . .
```

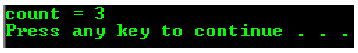
7. Next, instead of printing these numbers, count how many are greater than 5 and print the answer at the end. We will need a variable, call it count, to add this number. Complete the code by adding two more statements, one to calculate count and one to print it. The output should simply be count = 3.

```
#include <stdio.h>

void main()
{
   int count = 0, a[5] = {3, 7, 4, 9, 6}, i;
   for(i = 0; i <= 4; i = i + 1)
         if(a[i] > 5)
               ________________________
         ________________________
}
```

```
#include <stdio.h>

void main()
{
   int count = 0, a[5] = {3, 7, 4, 9, 6}, i;
   for(i = 0; i <= 4; i = i + 1)
         if(a[i] > 5)
              count = count + 1;
         printf("count = %d\n", count);
}
```



```
count = 3
Press any key to continue . . .
```

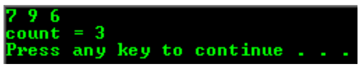
8. Next, don't print only the count, but also print the numbers greater than 5 in the array. You may want to show just the changes in the above solution for this practice.

```
#include <stdio.h>

void main()
{
   int count = 0, a[5] = {3, 7, 4, 9, 6};
   for(i = 0; i <= 4; i = i + 1)
         if(a[i] > 5)
         {
               ________________________
               ________________________
         }
         ______________________
}
```

```
#include <stdio.h>

void main()
{
   int count = 0, a[5] = {3, 7, 4, 9, 6}, i;
   for(i = 0; i <= 4; i = i + 1)
      if(a[i] > 5)
      {
         count = count + 1;
         printf("%d ", a[i]);
      }
   printf("\ncount = %d\n", count);
}
```



```
7 9 6
count = 3
Press any key to continue . . .
```

9. Next, add up only those elements in the array greater than 5 and print their sum. We'll need a variable called sum to accumulate the additions. First 7 will be added to sum. sum will be 7. Then 9 will be added, making sum equal to 16. Finally 6 will be added, making it 22. You'll need one statement to calculate the sum and one to print it.

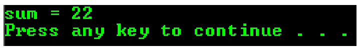
```
#include <stdio.h>

void main()
{
   int count = 0, a[5] = {3, 7, 4, 9, 6};
   for(i = 0; i <= 4; i = i + 1)
         if(a[i] > 5)
               ________________________
         ________________________

}
```

```
#include <stdio.h>

void main()
{
   int count = 0, a[5] = {3, 7, 4, 9, 6}, i, sum=0;
   for(i = 0; i <= 4; i = i + 1)
      if(a[i] > 5)
   sum = sum + a[i];
   printf("sum = %d\n", sum);
}
```



```
sum = 22
Press any key to continue . . .
```

10. Next, print the sum of the numbers greater than 5 and the count of the elements less than or equal to 5. Use two if's. The count should be printed as 2, since 3 and 4 are less than or equal to 5, and the sum should be printed as 22.

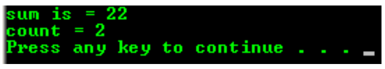
```
#include <stdio.h>

void main()
{
  int a[5] = {3, 7, 4, 9, 6};
  for(i = 0; i <= 4; i = i + 1)
  {
      if( ___________________________ )
          _____________________________
          if( _______________________ )
              _________________________
  }
  printf(" ______________________________ ");
  printf(" ______________________________ ");
}
```

You may use the if-else statement to replace the two if's.

```
#include <stdio.h>

void main()
{
   int a[5] = {3, 7, 4, 9, 6}, i, sum=0, count=0;
   for(i = 0; i <= 4; i = i + 1)
   {
      if( a[i] > 5 )
        sum = sum + a[i];
      if( a[i] <= 5 )
        count = count + 1;
   }
   printf("sum is = %d\n", sum);
   printf("count = %d\n", count);
}
```



```
sum is = 22
count = 2
Press any key to continue . . . _
```

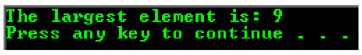
11. Starting with the following lines, complete the code for finding and printing the largest number in the array.

```
#include <stdio.h>

void main()
{
  int largest = a[0], a[5] = {3, 7, 4, 9, 6};
  for(i = 1; i <= 4; i = i + 1)
      ...
      ...
      ...
}
```

```
#include <stdio.h>

void main()
{
   int a[5] = {3, 7, 4, 9, 6}, i, largest = a[0];
   for(i = 1; i <= 4; i = i + 1)
      if(a[i] > largest)
   largest = a[i];
   printf("The largest element is: %d\n", largest);
}
```

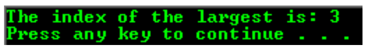


```
The largest element is: 9
Press any key to continue . . .
```

12. Next, print only the index of the largest number in the array. Keeping track of the largest number is not sufficient. We have to keep track of the index. Once we know the index, we know the number in that index/slot.

```
#include <stdio.h>

void main()
{
   int a[5] = {3, 7, 4, 9, 6}, i, largest = a[0], item_idx=0;
   for(i = 1; i <= 4; i = i + 1)
      if(a[i] > largest)
     {
        largest = a[i];
        item_idx = i;
      }
     printf("The index of the largest is: %d\n", item_idx);
}
```



```
The index of the largest is: 3
Press any key to continue . . .
```

13. Next, let us change our problem type. Using the given data items as input, read in values into the two arrays, a[ ] and b[ ]. Complete the code and show the contents of these arrays. The data items are 3, 8, 2, 9, 1, 5, 7, 6, 0, 4. What happens when i is 0?

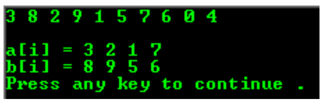
```
#include <stdio.h>

void main()
{
      ...
      ...
      for(i = 0; i <= 3; i = i + 1)
          scanf_s("%d %d", &a[i], &b[i]);
      ...
      ...
      ...
}
```

Hint: The contents of a[ ] starting at index 0 are "3 2 1 7". The contents of b [ ] starting at index 0 are "8 9 5 6".

```
#include <stdio.h>

void main()
{
   int i, a[5], b[5];
   for(i = 0; i <= 3; i = i + 1)
   {
      scanf_s("%d %d", &a[i], &b[i], sizeof(int), sizeof(int));
    }
   printf("\na[i] = ");
   for(i = 0; i <= 3; i = i + 1)
      printf("%d ", a[i]);
   printf("\nb[i] = ");
   for(i = 0; i <= 3; i = i + 1)
      printf("%d ", b[i]);
   printf("\n");
}
```



```
3 8 2 9 1 5 7 6 0 4

a[i] = 3 2 1 7
b[i] = 8 9 5 6
Press any key to continue .
```

When i is 0, a[0] = 3 and b[0] = 8.

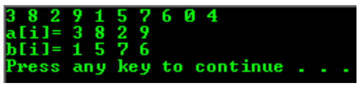
14. Using the following input data: 3, 8, 2, 9, 1, 5, 7, 6, 0, 4. Complete the code and show the contents of the two arrays after executing the following code.

```
#include <stdio.h>

void main()
{
        ...
        ...
        for(i = 0; i <= 3; i = i + 1)
              scanf_s("%d", &a[i]);
        for(i = 0; i <= 3; i = i + 1)
              scanf_s("%d", &b[i]);
        ...
        ...
        ...
}
```

```
#include <stdio.h>

void main()
{
    int i, a[5], b[5];
    for(i = 0; i <= 3; i = i + 1)
        scanf_s("%d", &a[i]);
    for(i = 0; i <= 3; i = i + 1)
        scanf_s("%d", &b[i]);
    printf("a[i]= ");
    for(i = 0; i <= 3; i = i + 1)
        printf("%d ", a[i]);
    printf("\nb[i]= ");
    for(i = 0; i <= 3; i = i + 1)
        printf("%d ", b[i]);
    printf("\n");
}
```



```
3 8 2 9 1 5 7 6 0 4
a[i]= 3 8 2 9
b[i]= 1 5 7 6
Press any key to continue . . .
```

15. Complete the following code and show the contents of the arrays after executing the following code. Data inputs are 3, 8, 2, 9, 1, 5, 7, 6, 0, 4. Notice the k > 1? condition.

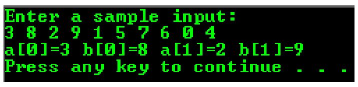
```
#include <stdio.h>

void main()
{
        ...
        scanf_s("%d", &k);
        for(i = 0; k > 1; i = i + 1)
        {
              a[i] = k;
              scanf_s("%d", &b[i]);
              scanf_s("%d", &k);
        }
        ...
        ...
        ...
}
```

```
#include <stdio.h>

void main()
{
    int i, a[5], b[5], k;

    printf("Enter a sample input:\n");
    scanf_s("%d", &k, sizeof(int));
    for(i = 0; k > 1; i = i + 1)
    {
        a[i] = k;
        scanf_s("%d", &b[i], sizeof(int));
        scanf_s("%d", &k, sizeof(int));
        printf("a[%d]=%d ", i, a[i]);
        printf("b[%d]=%d ", i, b[i]);
    }
    printf("\n");
}
```

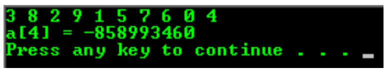


```
Enter a sample input:
3 8 2 9 1 5 7 6 0 4
a[0]=3 b[0]=8 a[1]=2 b[1]=9
Press any key to continue . . .
```

16. Complete the following code and show the contents of only array a[4] after executing the following code. Data items are 3, 8, 2, 9, 1, 5, 7, 6, 0, 4. Notice that the index is not i.

```
#include <stdio.h>

void main()
{
        ...
        scanf_s("%d", &k);
        for(i = 0; i <= 2; i = i + 1)
        {
              scanf_s("%d", &a[k]);
              scanf_s("%d", &k);
        }
        ...
        ...
        ...
}
```



```
3 8 2 9 1 5 7 6 0 4
a[4] = -858993460
Press any key to continue . . . _
```

The content of a[4] is rubbish.

18. Write a loop involving only the b team. This loop will determine the score of the fastest runner (which is 4.7). Then write a second loop that will count the number of players from the a team that this player could have beaten. The player with the 4.7 could have beaten the players of the a that received a score of 7.2, 4.9 and 8.1. Complete the following two for loops and the whole code.

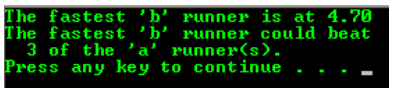
```
#include <stdio.h>

void main()
{
   ...
   ...
   float fastest = b[0];
   // loop to find the fastest in team b
   for(i = 1; i <= 4; i = i + 1)
        _________________________
          _________________________

   // loop to count the slower ones in a
   for(i = 0; i <= 4; i = i + 1)
        _________________________
        _________________________
   printf("The fastest b runner could beat %d of the a runner\n", count);
}
```

```
#include <stdio.h>

void main()
{
    // the f & F modifiers used to indicate that
    // the type is float, by default it is treated
    // as double for floating point. without the
    // modifier compiler will generate warnings...
    float a[5] = {7.2F, 3.4F, 4.9F, 2.2F, 8.1F},
    b[5]= {5.6f, 12.8f, 4.7f, 6.2f, 7.3f},
    fastest = b[0];
    int i, count = 0;
    // loop to find the fastest in team b
    for(i = 1; i <= 4; i = i + 1)
    {
        if(b[i] < fastest)
            fastest = b[i];
    }
    printf("The fastest 'b' runner is at %.2f\n", fastest);
    // loop to count the slower ones in a
    for(i = 0; i <= 4; i = i + 1)
    {
        if(a[i] > 4.7)
          count = count +1;
    }
    printf("The fastest 'b' runner could beat\n %d of the 'a' runner(s).\n", count);
}
```



```
The fastest 'b' runner is at 4.70
The fastest 'b' runner could beat
  3 of the 'a' runner(s).
Press any key to continue . . .
```

-------------------------End of 1D Array--------------------