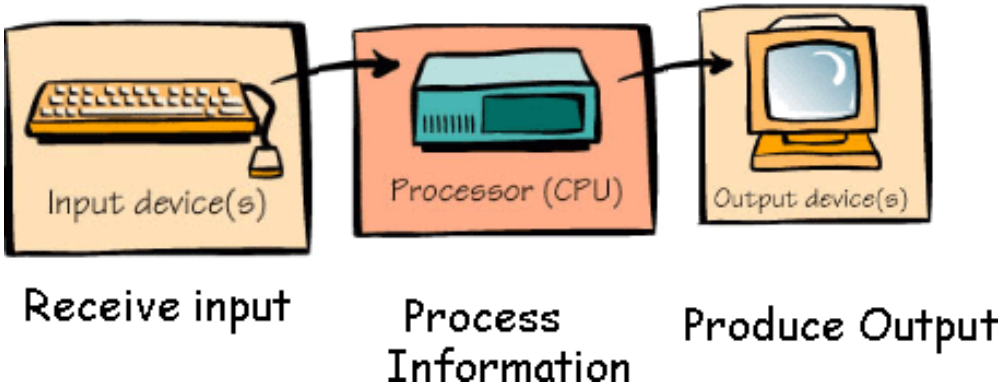
What Computers Do



Week 3 - Lecture 3

Simple Algorithms

Edited by: Dr. Wooi Ping Cheah

Autumn 2022

University of Nottingham
UK | CHINA | MALAYSIA

Overview

- Sorting Algorithms
- Search Algorithms
- Type Casting
- Array of Pointers

Example: Module Mark

- There are 3 students in the year. Each of them has taken 2 modules.
- Read their marks one by one, student by student, then print them out module by module.

```c
#include <stdio.h>

int main(void)
{
    const int NUM_MOD = 2;
    const int NUM_STUDENTS = 3;
    int marks[NUM_MOD][NUM_STUDENTS];

    int s = 0;
    int m = 0;
    for(s = 0; s < NUM_STUDENTS; s++)
    {
        for(m = 0; m < NUM_MOD; m++)
        {
            printf("Student %d, module %d: ", s+1, m+1);

            int mark = 0;
            scanf("%d", &mark);
            marks[m][s] = mark;
        }
    }

    printf("\n\nThe marks entered:\n");
    for(m = 0; m < NUM_MOD; m++)
    {
        for(s = 0; s < NUM_STUDENTS; s++)
        {
            printf("Student %d, module %d: %d\n", s+1, m+1, marks[m][s]);
        }
    }

    return 0;
}
```

Use nested for loops to solve this problem.

marks

m

s	0	1
0	marks[0][0]	marks[0][1]
1	marks[1][0]	marks[1][1]
2	marks[2][0]	marks[2][1]

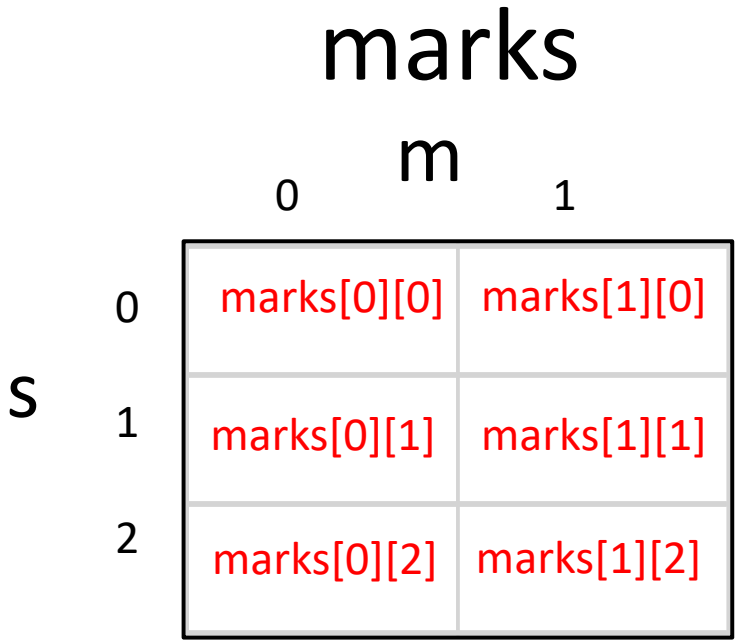
marks[s][m]

marks[0][0]
marks[0][1]
marks[1][0]
marks[1][1]
marks[2][0]
marks[2][1]

3

```
for (s = 0; s < NUM_STUDENTS; s++)
{
        for (m = 0; m < NUM_MOD; m++)
        {
                ..... marks[s][m].....
        }
}
```

2



marks[m][s]

marks[0][0]
marks[0][1]
marks[0][2]
marks[1][0]
marks[1][1]
marks[1][2]

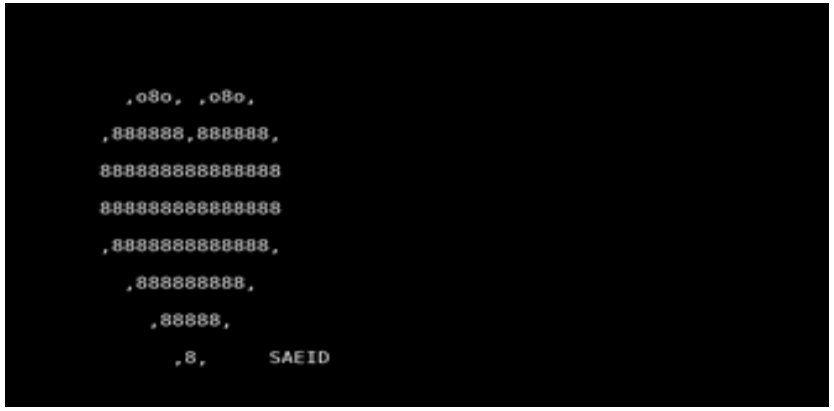
2

```
for (m = 0; m < NUM_MOD; m++)
{
        for (s = 0; s < NUM_STUDENTS; s++)
        {
                ..... marks[m][s].....
        }
}
```

3

Example: 2D shapes

- Create a program that holds a 80 x 25 array of characters to show a shape.
- You can change the values in the array as if you were drawing.
- E.g. http://www.ascii-art.de/ascii/

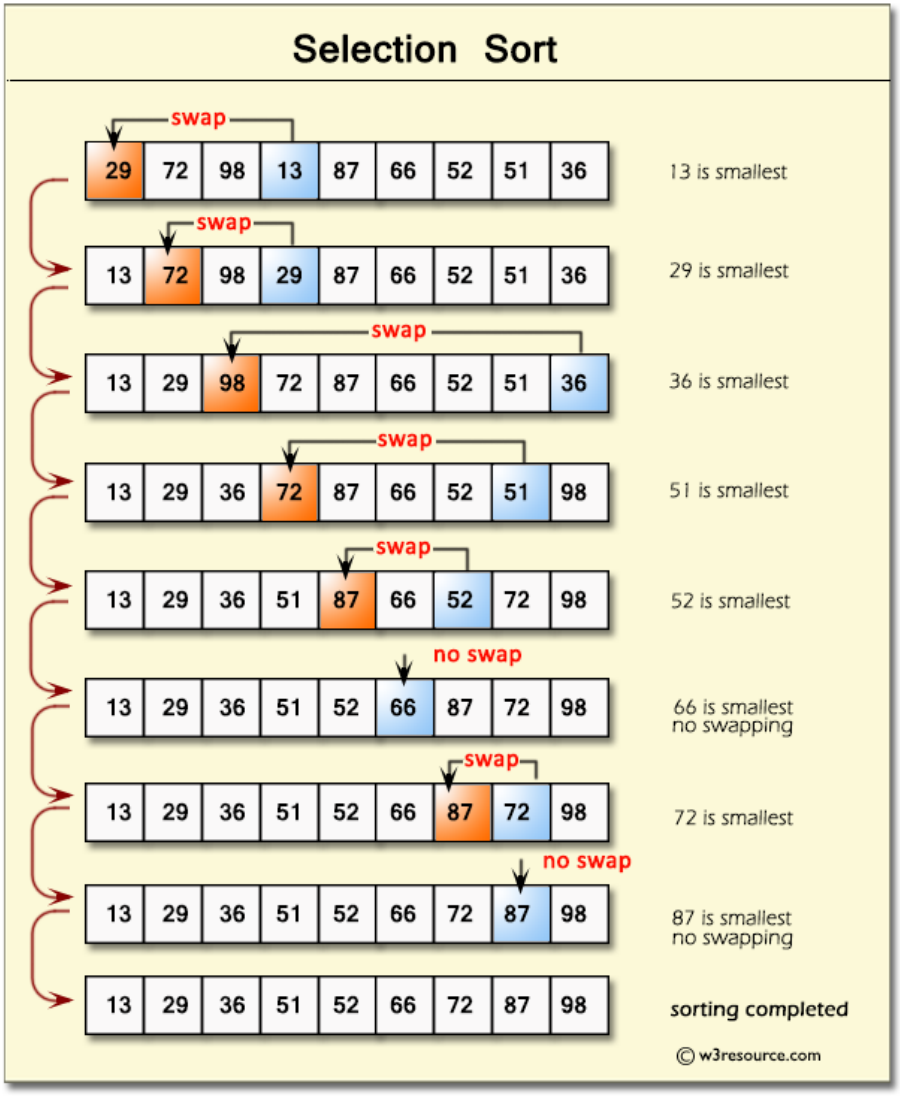


Overview

- **Sorting Algorithms**
- Searching Algorithms
- Type Casting
- Array of Pointers

Selection Sort

- Repeatedly **find the minimum element** and swap with the element at the index where it is supposed to be (from the beginning of the array).



https://www.w3resource.com/php-exercises/searching-and-sorting-algorithm/searching-and-sorting-algorithm-exercise-4.php

Selection Sort (2)

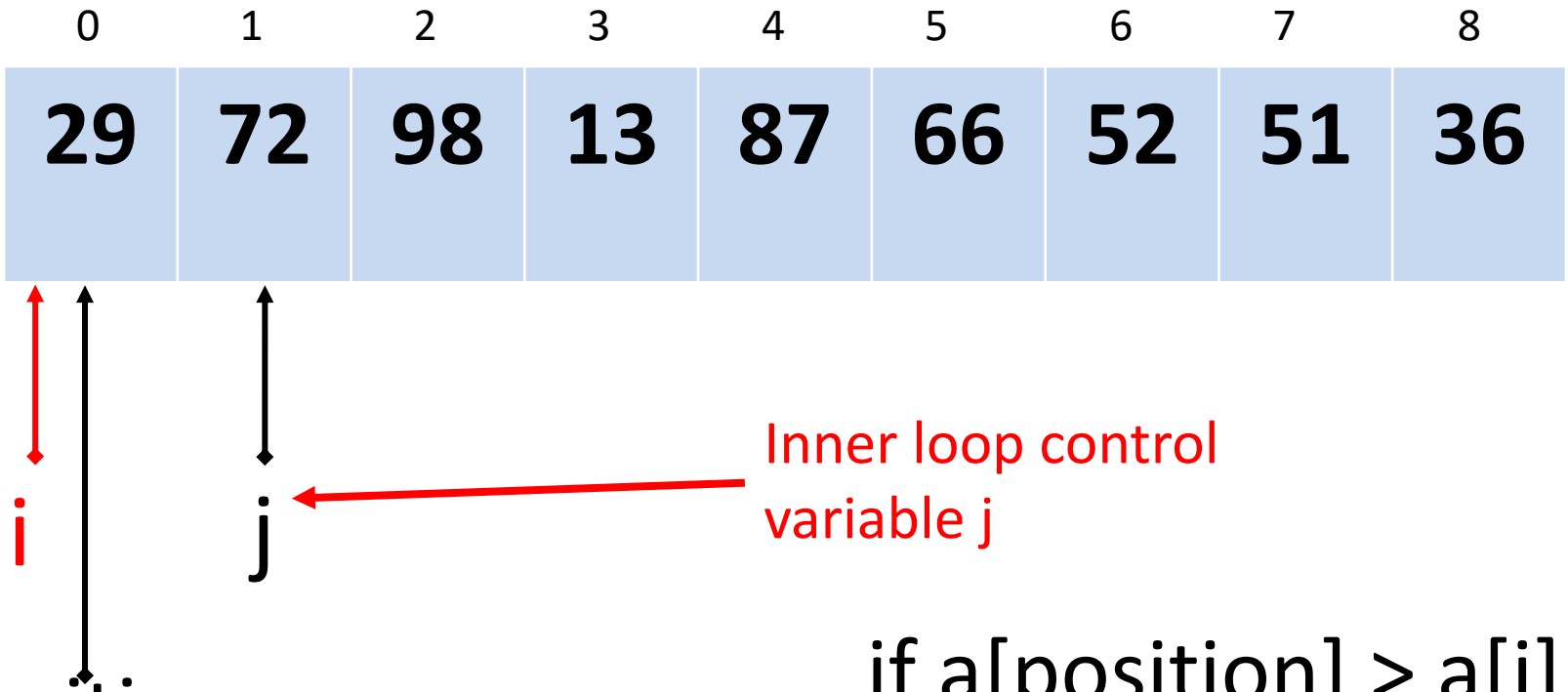
```
for(i = 0; i < n - 1; i++)
{
        position=i;
        for(j = i + 1; j < n; j++)
        {
                if(a[position] > a[j])
                        position=j;
        }
        if(position != i)
                {
                        temp=a[i];
                        a[i]=a[position];
                        a[position]=temp;
                }
}

printf("Sorted Array/n");
for(i = 0; i < n; i++)
        printf("%dn", a[i]);
return 0;
}
```

Outer Loop

Inner Loop

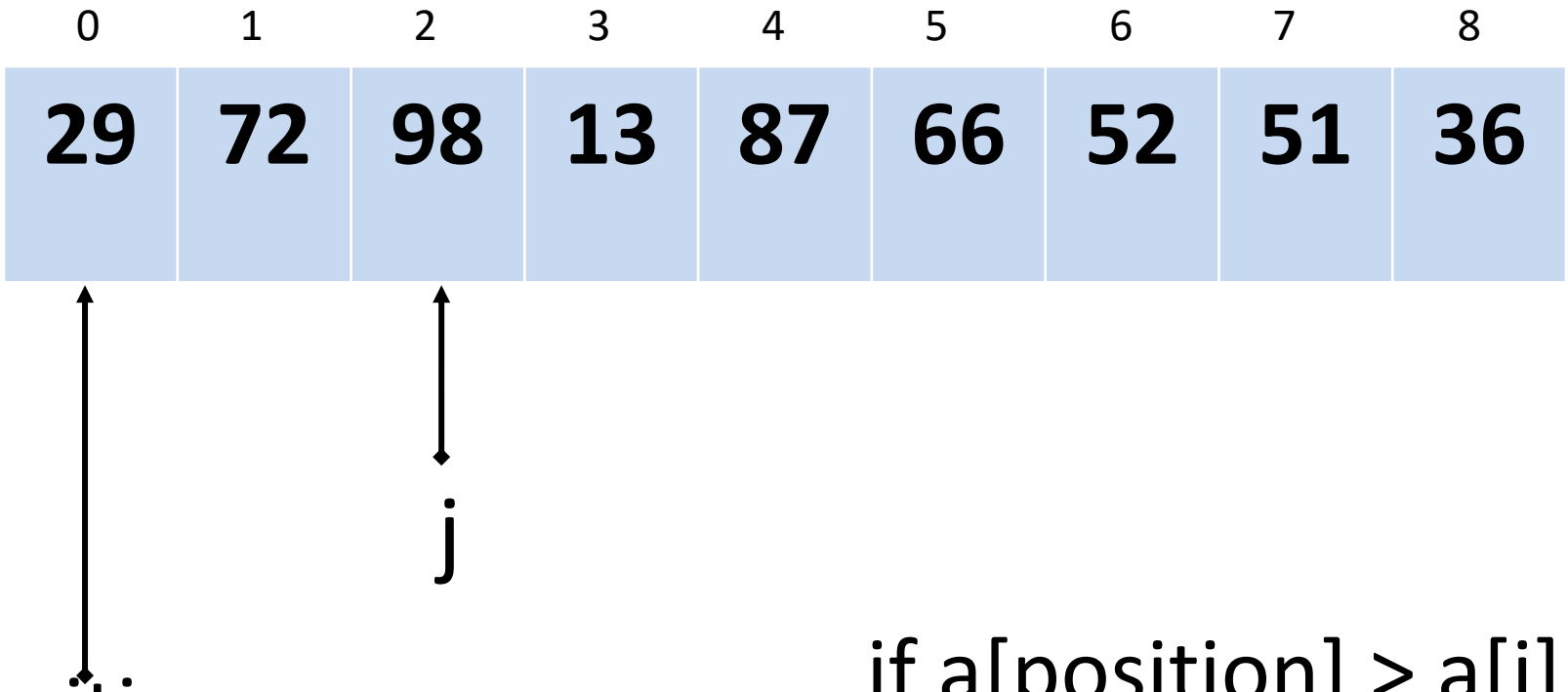
swap



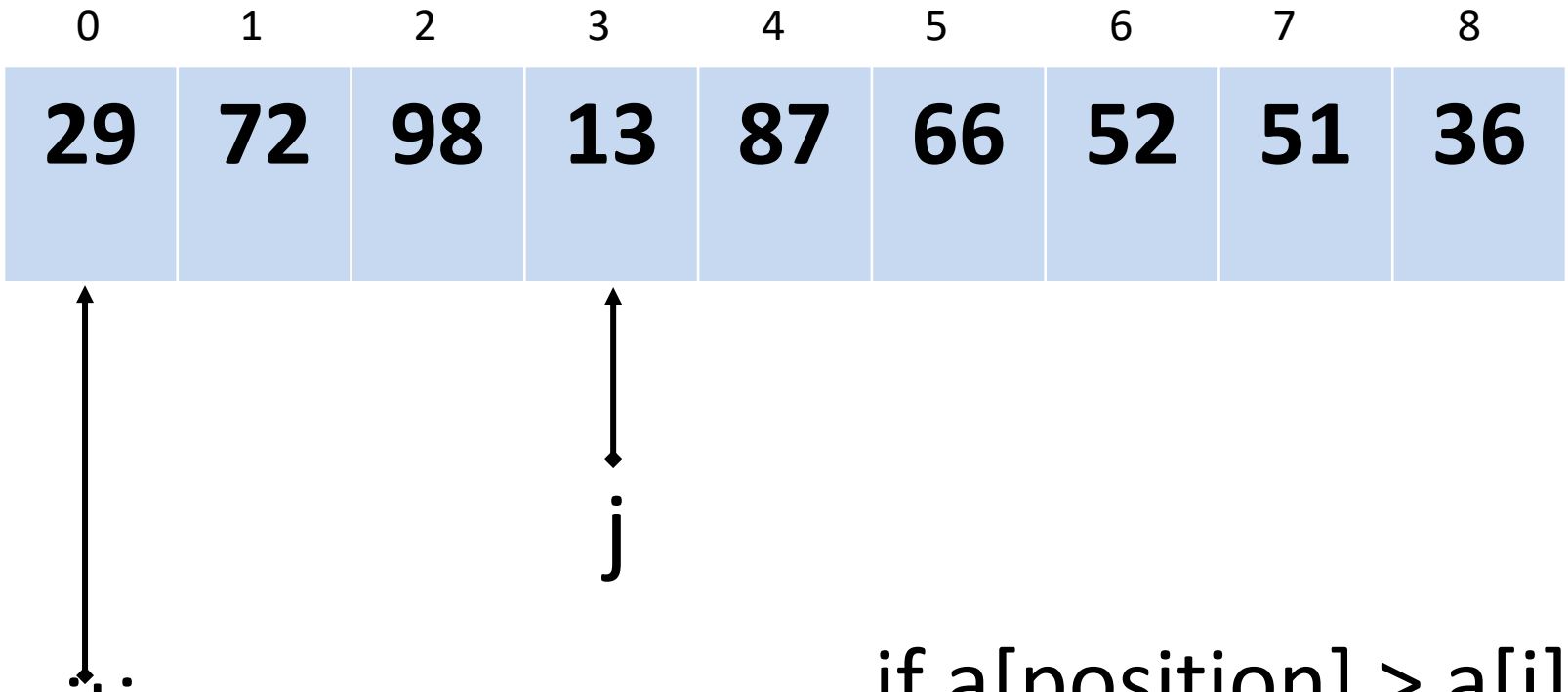

```
if a[position] > a[j];
position = j;
```

Outer loop control variable i

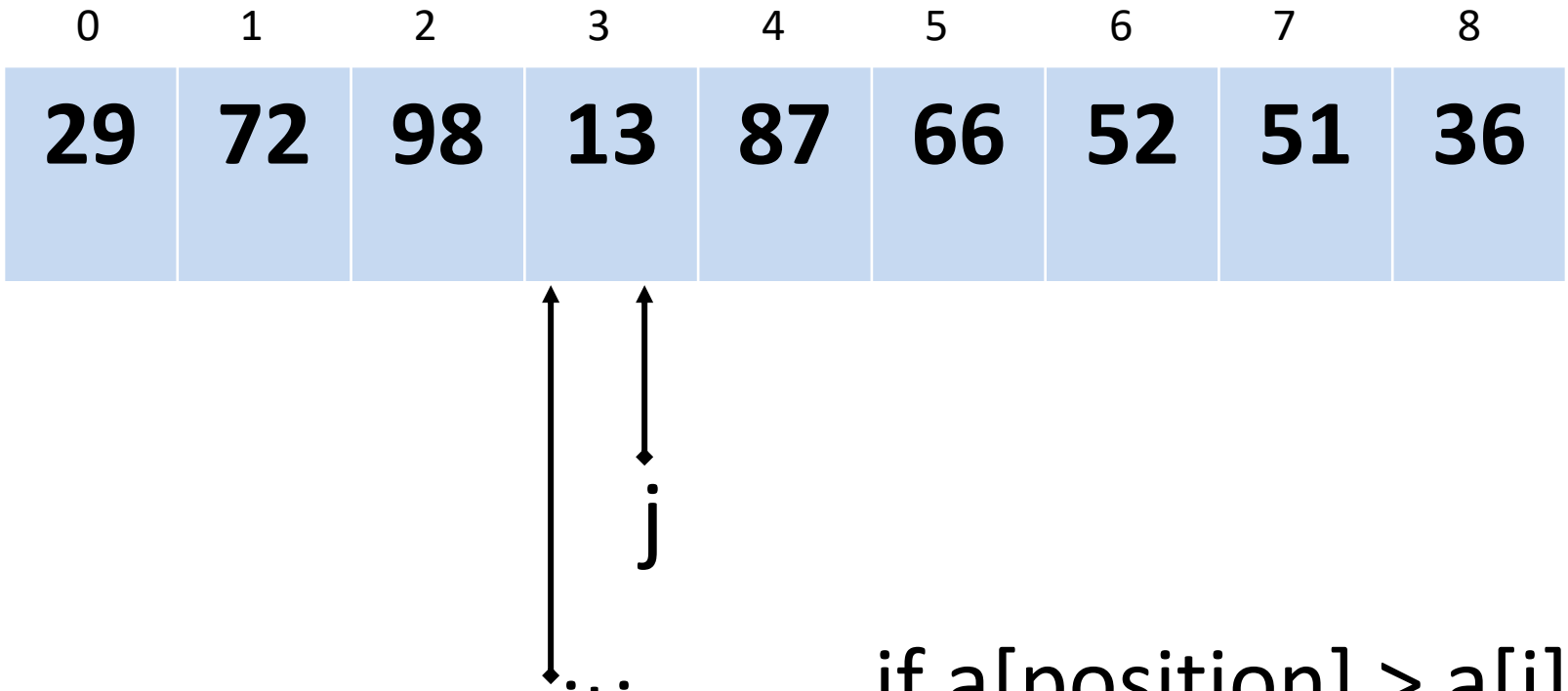
```
i = 0;
position = i;
```



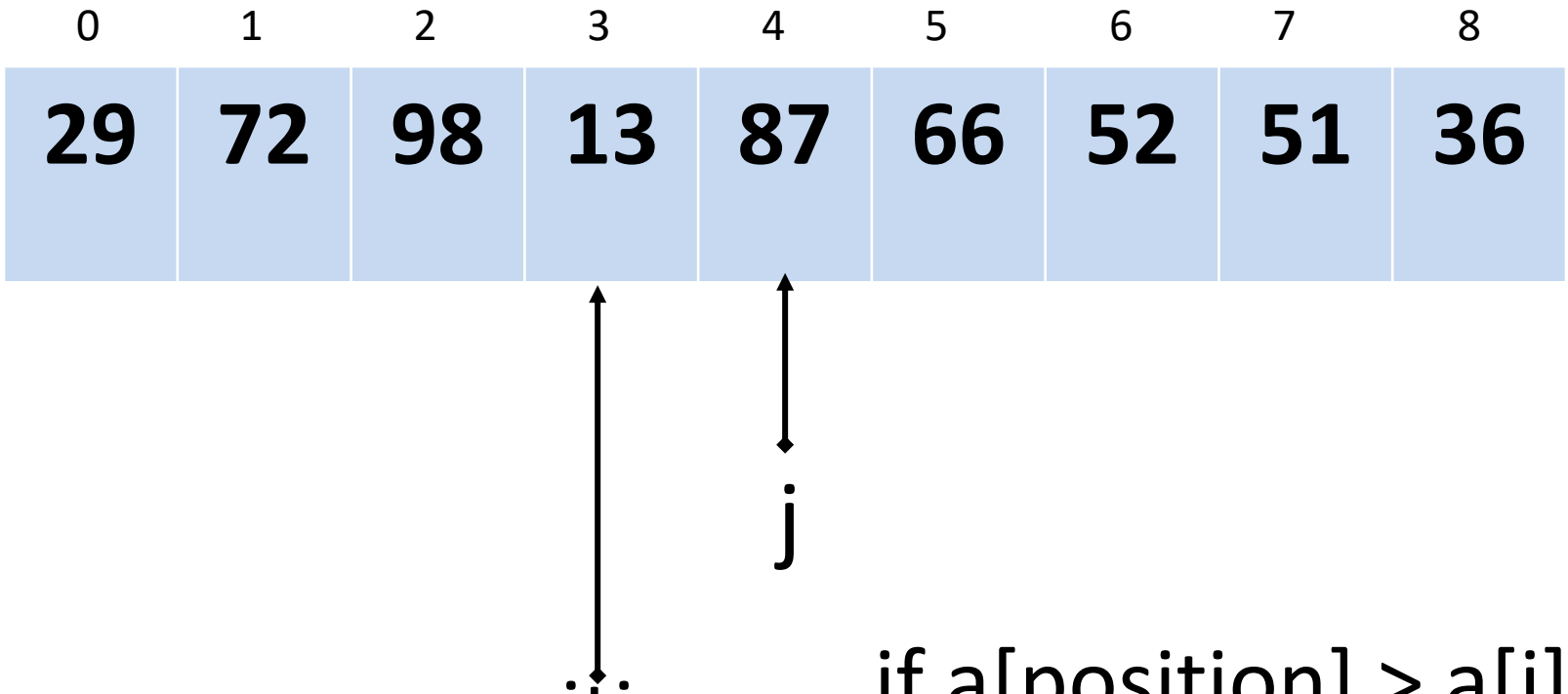
if a[position] > a[j];
position = j;



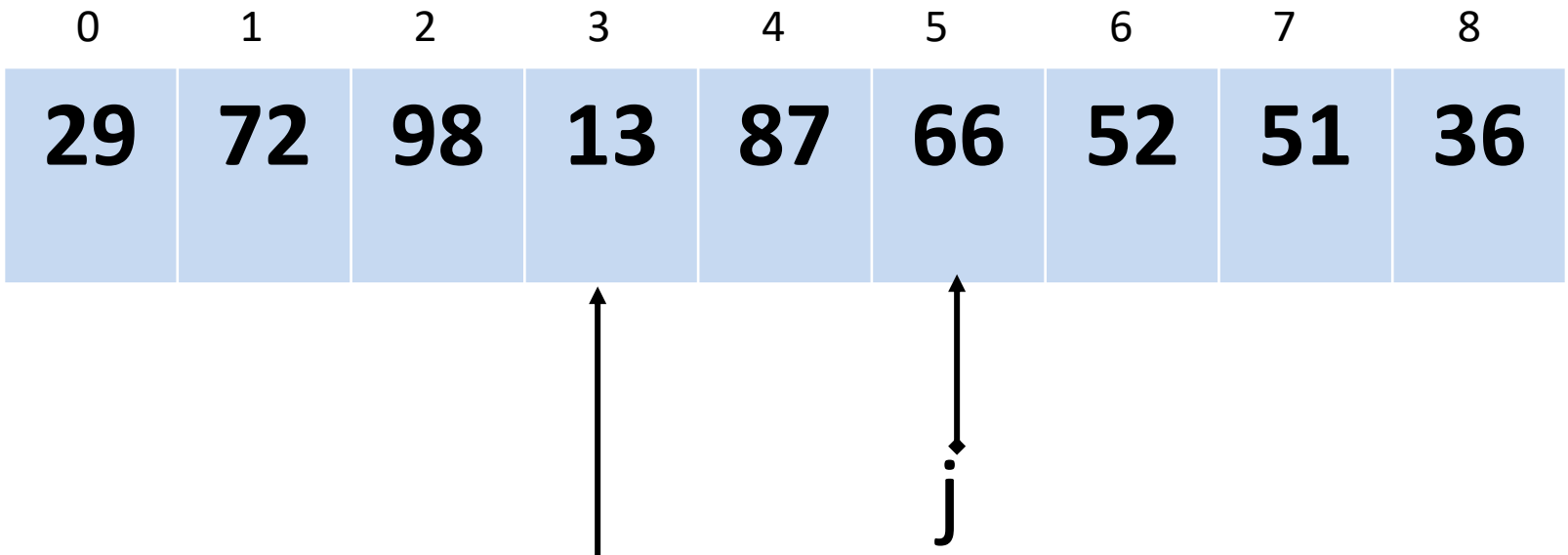
if a[position] > a[j];
position = j;



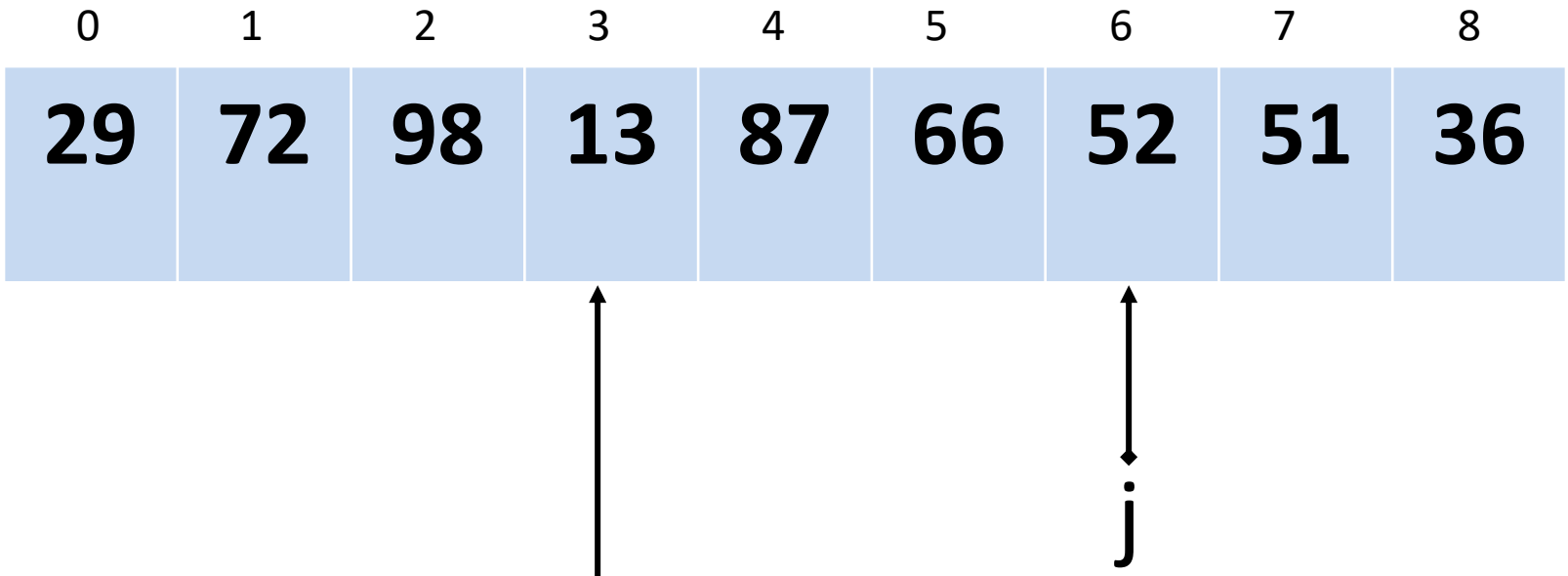
if a[position] > a[j];
position = j;



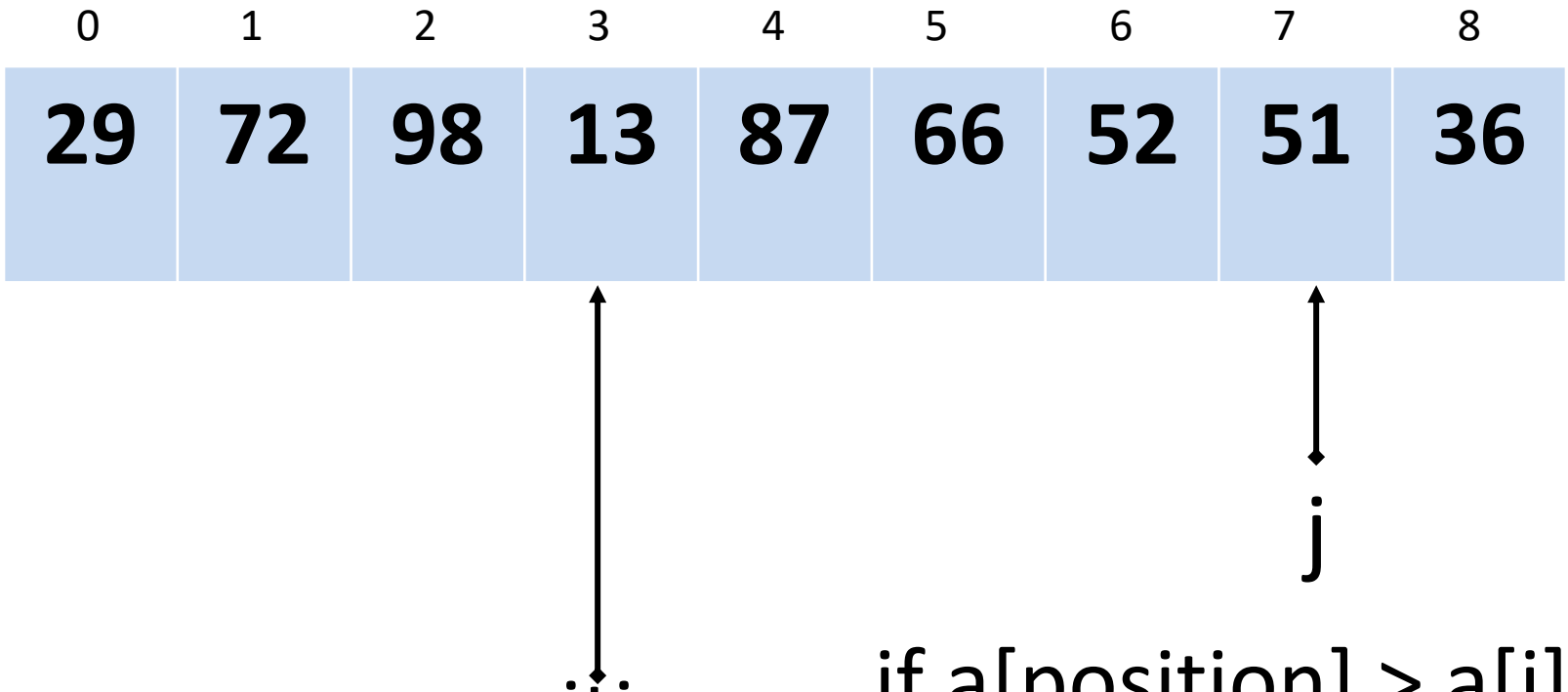
if a[position] > a[j];
position = j;

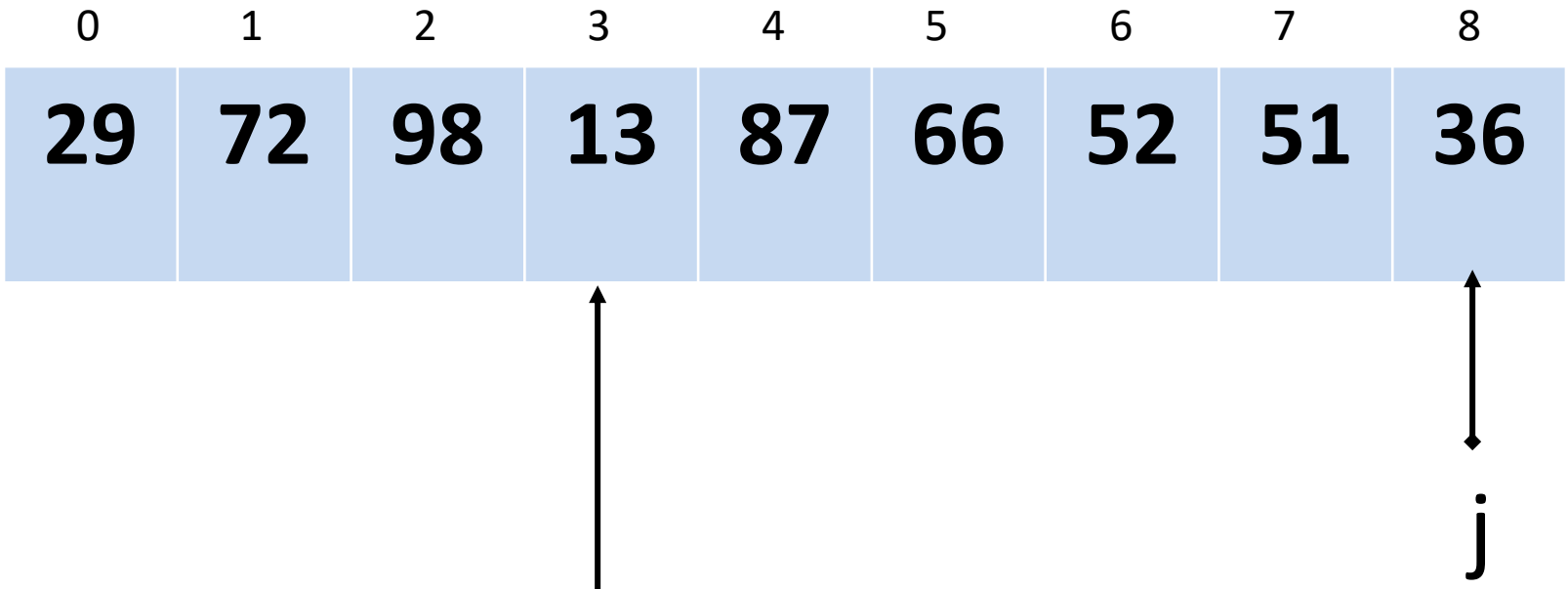


if a[position] > a[j];
position = j;

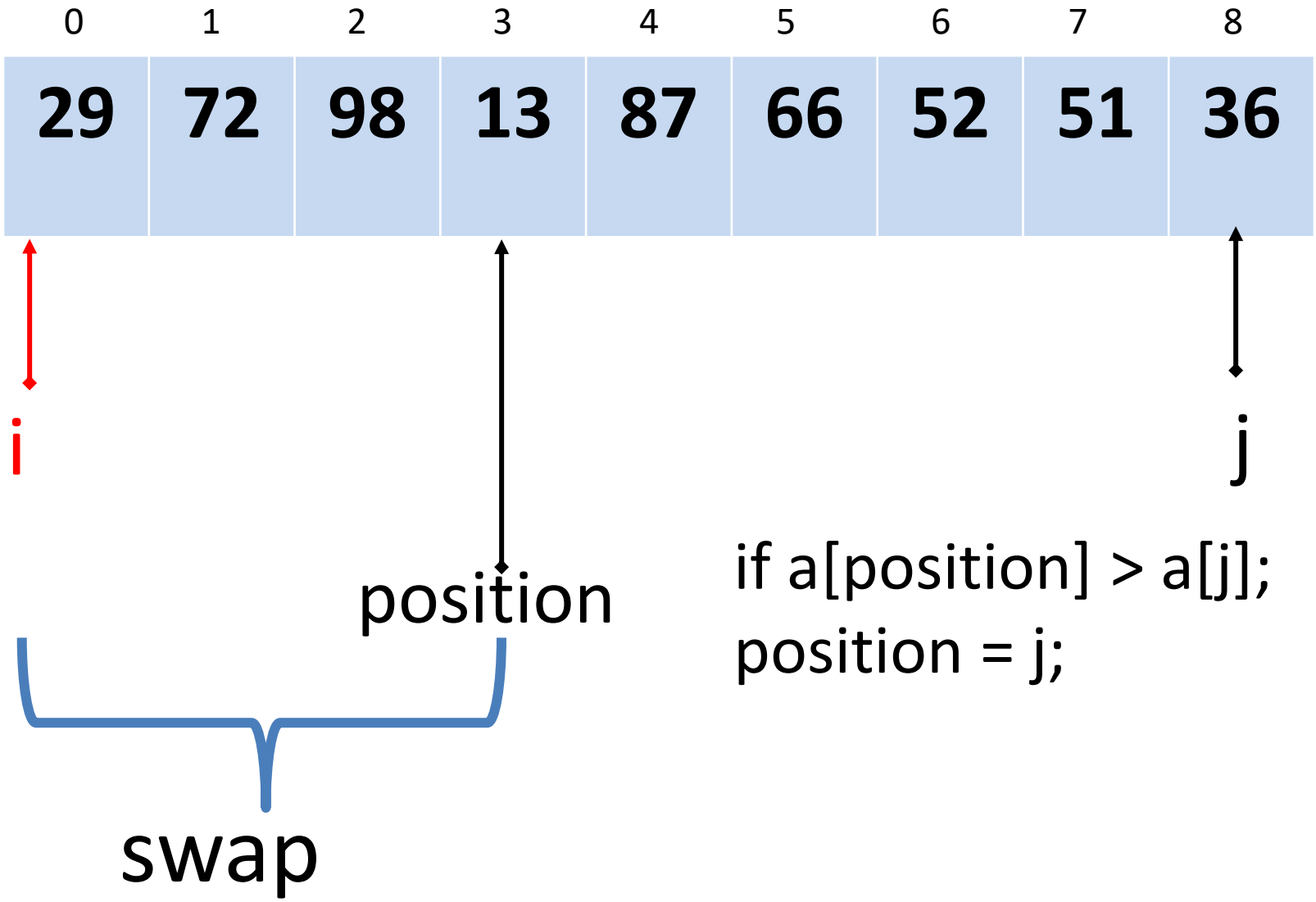


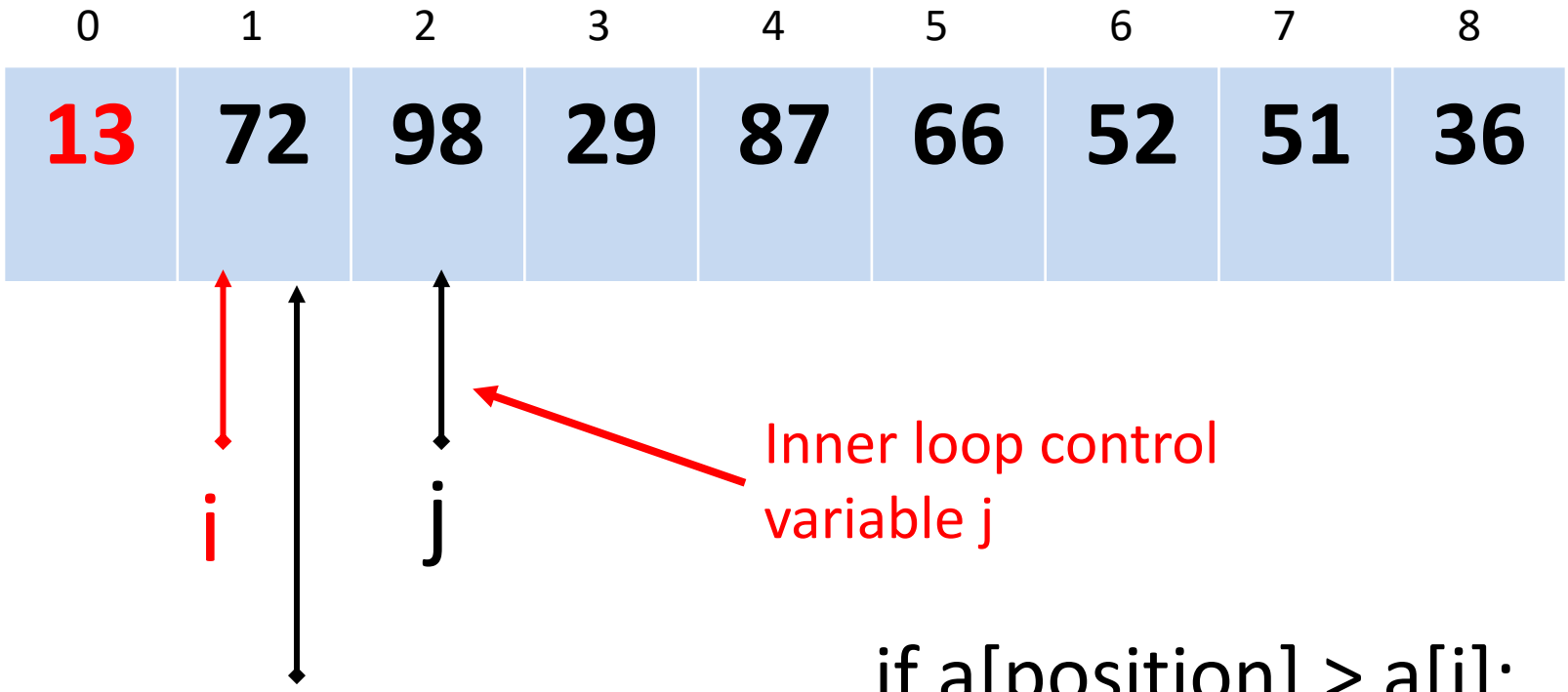
if a[position] > a[j];
position = j;





if a[position] > a[j];
position = j;






```
if a[position] > a[j];
position = j;
```

Outer loop control variable i

```
i = 1;
position = i;
```

Bubble Sort

Bubble sort or the sinking sort: the smaller values gradually "bubble" their way upward to the top of the array like air bubbles rising in water, while the larger values sink to the bottom of the array.

https://www.coderarticles.com/bubble-sort-algorithm/

Bubble Sort (2)

```
395     int temp = 0;
396     int j = 0;
397     for(i = 0; i < count; i++)
398     {
399         for(j = 0; j < (count - 1); j++)
400         {
401             if(myInt[j] > myInt[(j + 1)])
402             {
403                 temp = myInt[j];
404                 myInt[j] = myInt[(j + 1)];
405                 myInt[(j + 1)] = temp;
406             }
407         }
408     }
```

swap

Please refer to the following set of PPT Slides for an animation of Bubble Sort:

https://www.cc.gatech.edu/~bleahy/cs1311/cs1311lecture16wdl.ppt

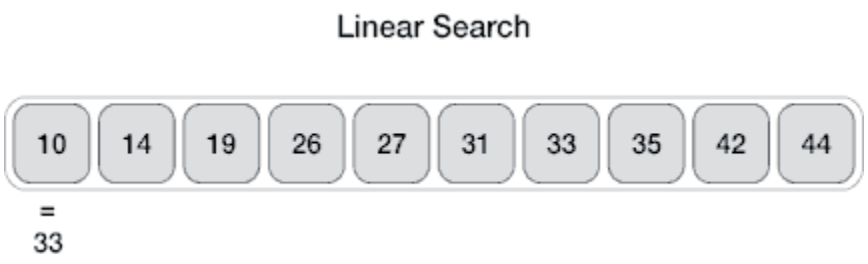
Slides 24-96

Overview

- Sorting Algorithms
- **Searching Algorithms**
- Type Casting
- Array of Pointers

Linear Search

- A sequential search, every item is checked until the end of the data collection or the item is found.




```
450     int index = -1;
451     for(i = 0; i < count; i++)
452     {
453         if(hunt == myInt[i])
454         {
455             index = i;
456             break;
457         }
458     }
```

Source: https://www.tutorialspoint.com/data_structures_algorithms/linear_search_algorithm.htm

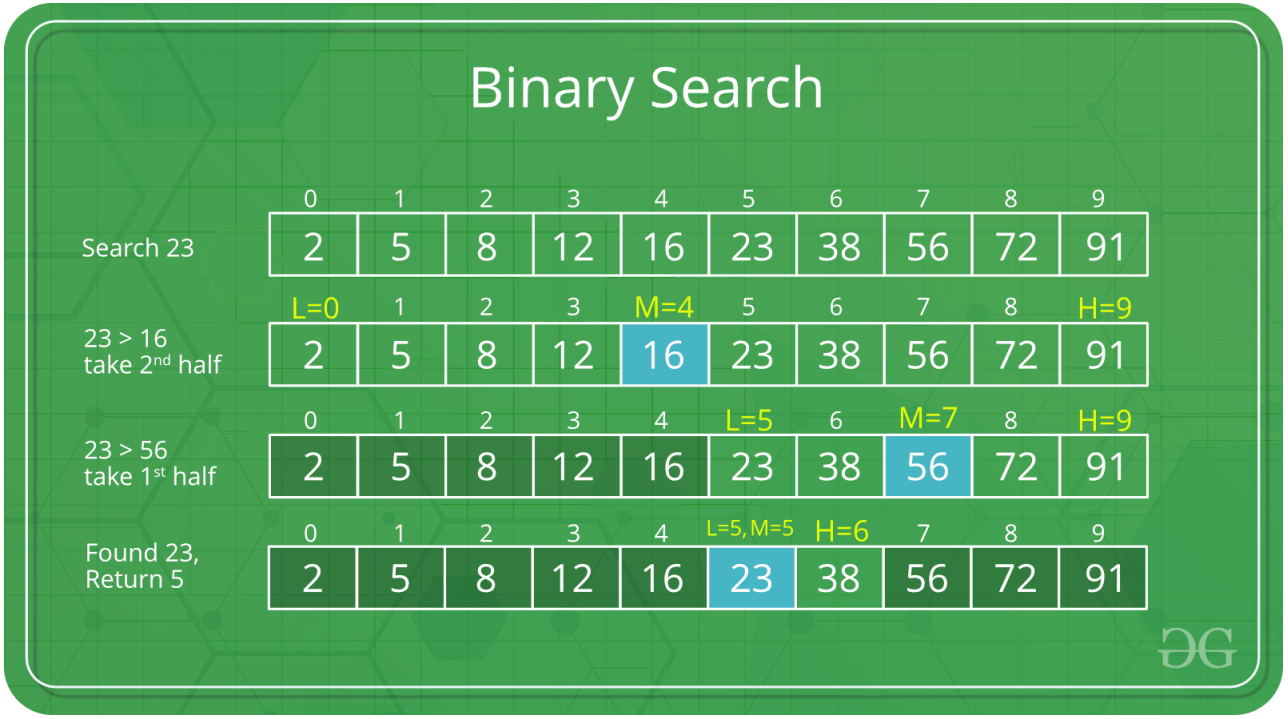
Binary Search

- The linear searching method works well for *small* or *unsorted* arrays.
- However, for large arrays linear searching is *inefficient*.
- If the array is sorted, the high-speed binary search technique can be used.
- The binary search algorithm eliminates from consideration *one-half* of the elements in a sorted array after each comparison.

Binary Search (2)

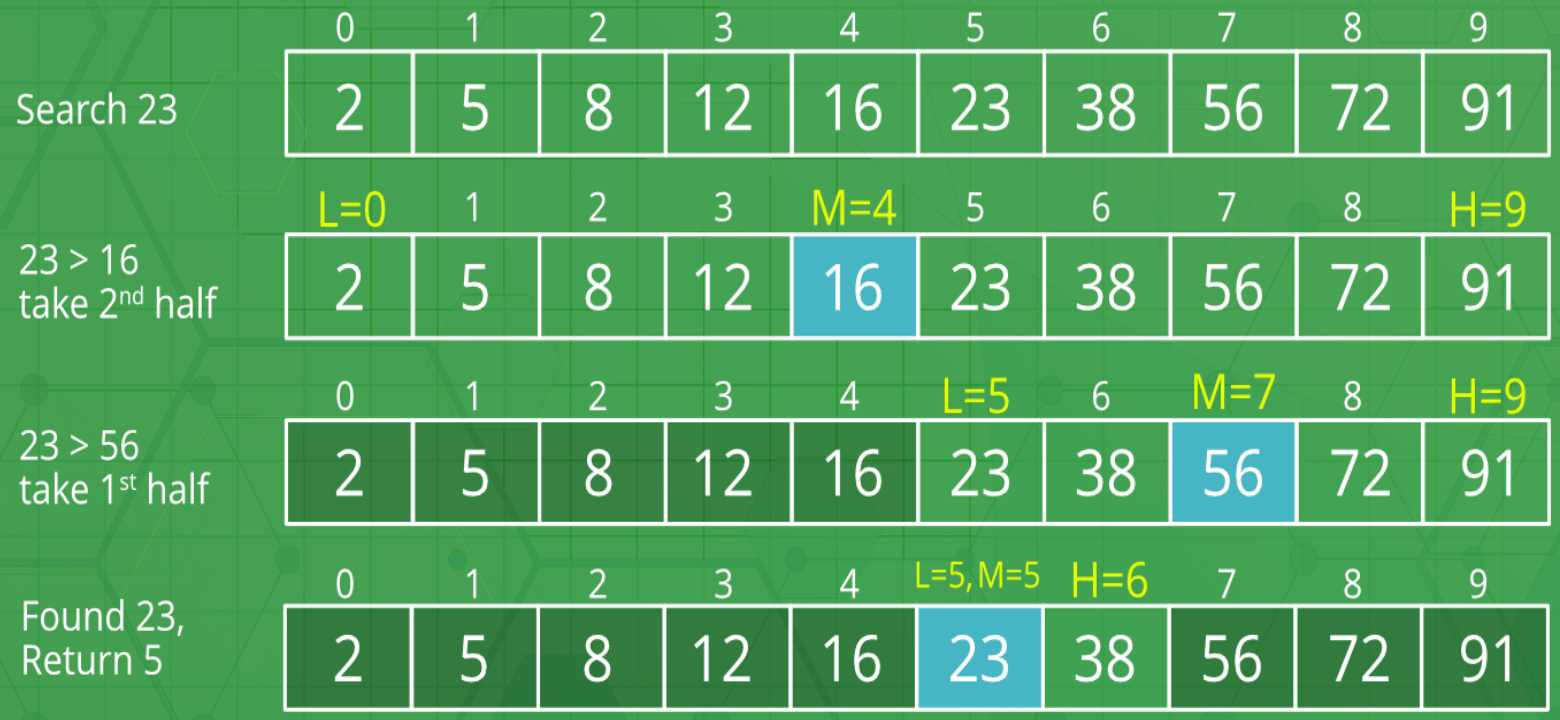
- An array of 1,048,576 (2^{20}) elements takes a maximum of only 20 comparisons to find the search key.
- This is a tremendous increase in performance over the linear search that required comparing the search key to an average of half of the array elements.
- Binary search, focuses on Divide and Conquer, requires sorted array.

Binary Search



Source: https://www.geeksforgeeks.org/binary-search/

Binary Search



Source: https://www.geeksforgeeks.org/binary-search/

Please refer to the following set of PPT Slides for an animation of Binary Search:

https://www.cc.gatech.edu/~bleahy/cs1311/cs1311lecture15wdl.ppt

Slides 130-136

Binary Search: code sample

```
#include <stdio.h>
int main()
{
  int c, first, last, middle, n, search,
array[100];

  printf("Enter number of elements\n");
  scanf("%d", &n);
  printf("Enter %d integers\n", n);

  for (c = 0; c < n; c++)
    scanf("%d", &array[c]);

  printf("Enter value to find\n");
  scanf("%d", &search);

  first = 0;
  last = n - 1;
  middle = (first+last)/2;

  while (first <= last) {
    if (array[middle] < search)
      first = middle + 1;
    else if (array[middle] == search) {
      printf("%d found at location %d.\n", search,
middle+1);
      break;
    }
    else
      last = middle - 1;

    middle = (first + last)/2;
  }
  if (first > last)
    printf("Not found! %d isn't present in the list.\n",
search);

  return 0;
}
```

```
#include <stdio.h>
int main()
{
  int c, first, last, middle, n, search,  array[100];

  printf("Enter number of elements\n");
  scanf("%d", &n);
  printf("Enter %d integers\n", n);

  for (c = 0; c < n; c++)
    scanf("%d", &array[c]);

  printf("Enter value to find\n");
  scanf("%d", &search);

  first = 0;
  last = n - 1;
  middle = (first+last)/2;
```

```
  while (first <= last) {
      if (array[middle] < search)
          first = middle + 1;
      else if (array[middle] == search) {
          printf("%d found at location %d.\n", search, middle+1);
          break;
      }
      else
          last = middle - 1;
      middle = (first + last)/2;
  }

  if (first > last)
      printf("Not found! %d isn't present in the list.\n", search);

  return 0;
}
```

Binary Search: new solution, what do you think?

```
#include <stdio.h>
int binarySearch(int [], int, int, int);

int main()
{
 int c, first, last, n, search, array[100], index;

 printf("Enter number of elements\n");
 scanf("%d", &n);

 printf("Enter %d integers\n", n);

 for (c = 0; c < n; c++)
  scanf("%d", &array[c]);

 printf("Enter value to find\n");
 scanf("%d", &search);

 first = 0;
 last = n - 1;

 index = binarySearch(array, first, last, search);

 if (index == -1)
   printf("Not found! %d isn't present in the list.\n", search);
 else
   printf("%d is present at location %d.\n", search, index +
1);

 return 0;
}

int binarySearch(int a[], int s, int e, int f) {
 int m;

 if (s > e) // Not found
   return -1;

 m = (s + e)/2;

 if (a[m] == f)  // element found
   return m;
 else if (f > a[m])
   return binarySearch(a, m+1, e, f);
 else
   return binarySearch(a, s, m-1, f);
}
```

Overview

- Sorting Algorithms
- Searching Algorithms
- **Type Casting**
- Array of Pointers

Type Casting

- Convert the type of an expression to another type

- (data_type) expression

- float a, b, c = 2.34;
  b = (int)(c+4.6)

b is 6

Explicit Type Conversions

- Implicit type conversion

```
float f;
int i;
i = f;
```

- Explicit type conversion

```
float f;
int i;
i = (int) f;
```

Same result, but for explicit type conversion the reader knows for sure that it was intentional!!

Initialisation

Example: variable initialized where declared

```
int max = 0;
/* use of max is within a page of where it is declared */
for (i=0; i<n; i++)
    if (vec[i] > max)
        max = vec[i];
```

Example: variable initialized where used

Use an assignment statement just before the for loop:

```
int max;
...
/* several pages between declaration and use */
...
max = 0;
for (i=0 ; i<n ; i++)
    if (vec[i] > max)
        max = vec[i];
```

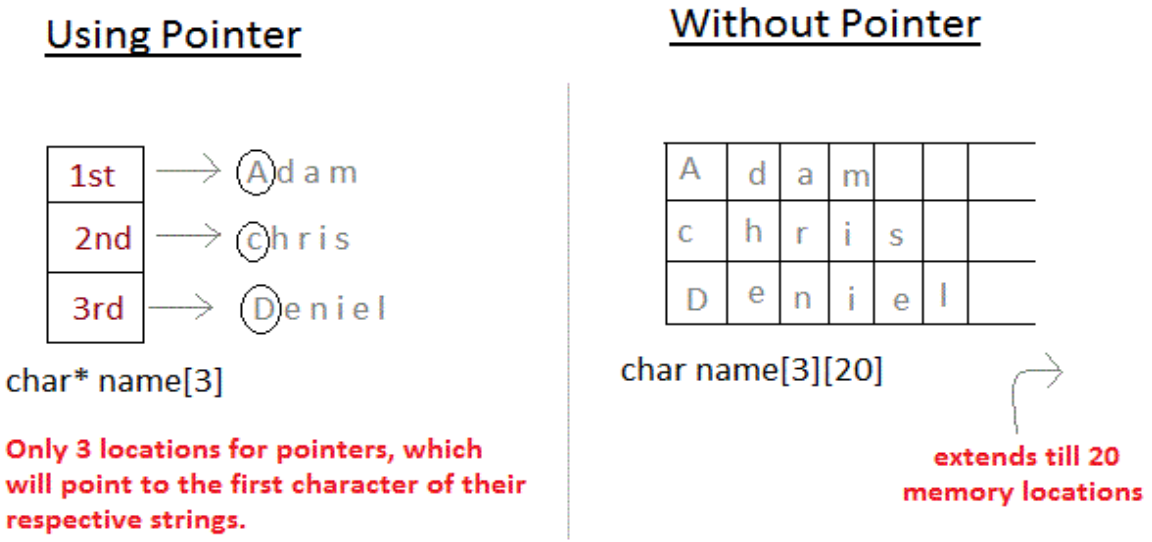
Overview

- Sorting Algorithms
- Search Algorithms
- Type Casting
- **Array of Pointers**

2D Array and Array of Pointers

- Array of pointers can be helpful in handling character array with varying length.

```
char *name[3] = {
    "Adam",
    "chris",
    "Deniel"
};
//Now lets see same array without using pointer
char name[3][20] = {
    "Adam",
    "chris",
    "Deniel"
};
```



Source: https://www.studytonight.com/c/pointers-with-array.php

Summary

- Sorting Algorithms
- Searching Algorithms
- Type Casting
- Array of Pointers