Programming with C I

Fangtian Zhong
CSCI 112

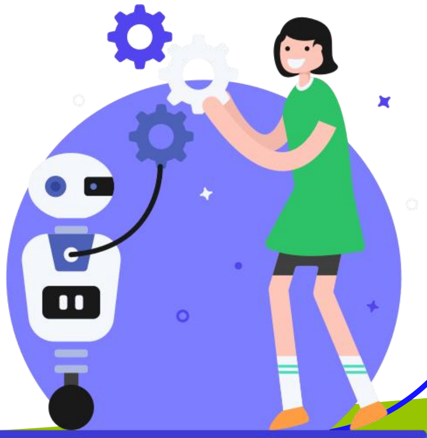
Gianforte School of Computing
Norm Asbjornson College of Engineering
E-mail: fangtian.zhong@montana.edu

2024.02.23

Linear Search Example Using While

```
// Example: Search array using while
int scores[MAX_SCORES];
int scoresCount, scoreNdx, targetScore;

// Assume array has been loaded,
// count = scoreCount, and search value = targetScore
scoreNdx = 0;
while (scoreNdx<scoreCount && scores[scoreNdx]!=targetScore)
     scoreNdx++;
if (ScoreNdx>=scoreCount) {
     // Whatever you want to do if not found
}
else {
     // Whatever you want yo do if found
}
```



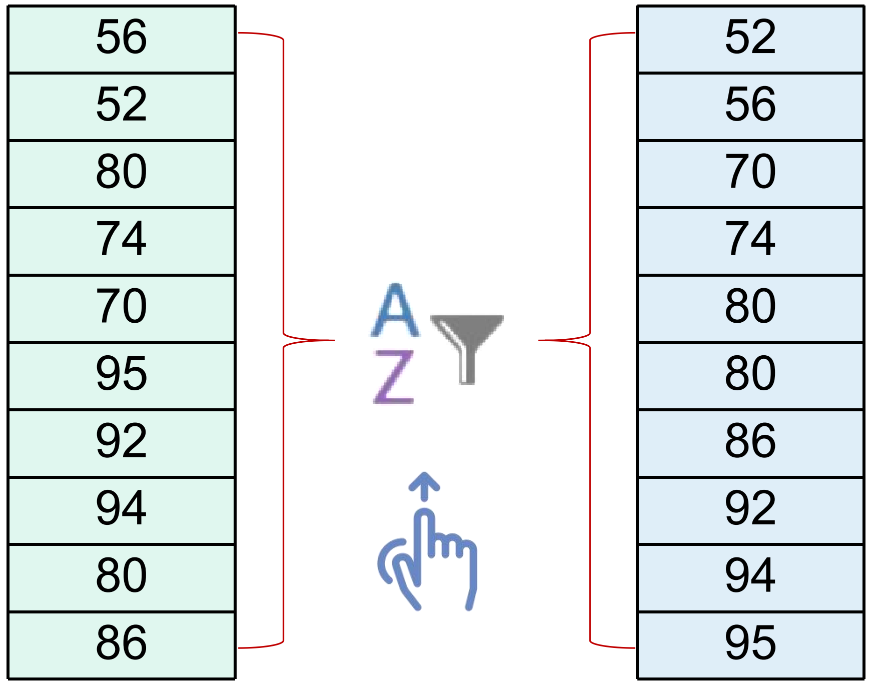
Linear Search Example Using For

```
// Example: Search array using for
int scores[MAX_SCORES];
int scoresCount, scoreNdx, targetScore;

// Assume array has been loaded,
// count = scoreCount, and search value = targetScore
for (scoreNdx=0;
            scoreNdx<scoreCount && scores[scoreNdx]!=targetScore;
            scoreNdx++)  /* null */;
   // Note: Above for statement has empty basic block by design
if (scoreNdx>=scoreCount) {
   // Whatever you want to do if not found
}
else {
   // Whatever you want to do if found
}
```

Sorting

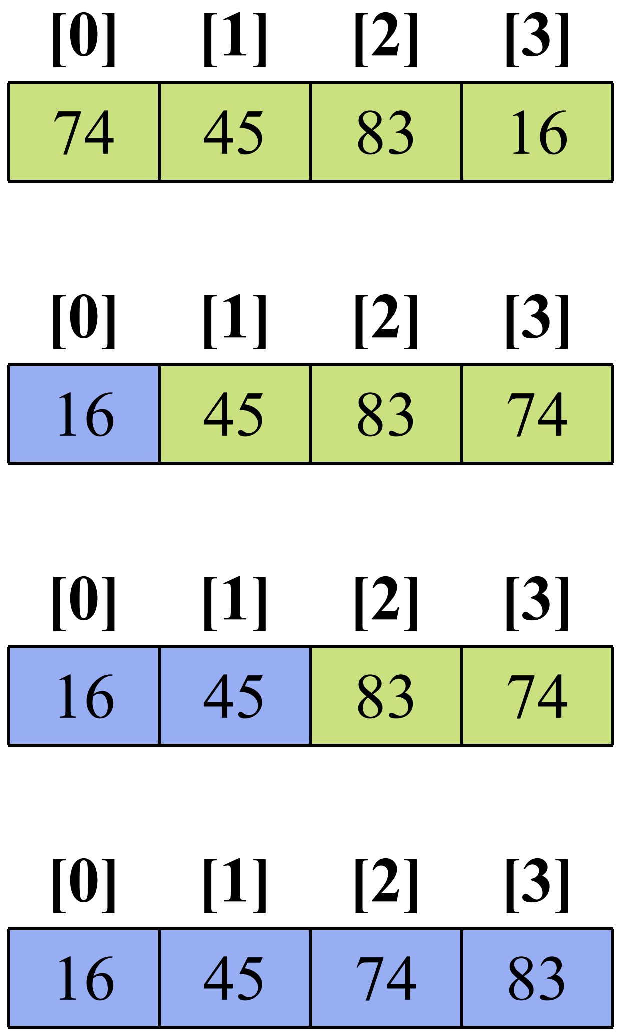
- Place array into some order
  - Ascending or descending
- Many types
  - Simple: Selection
  - More intelligent: Bubble, selection, insertion, shell, comb, merge, heap, quick, counting, bucket, radix, distribution, timsort, gnome, cocktail, library, cycle, binary tree, bogo, pigeonhole, spread, bead, pancake, …



Selection Sort

- **for each value of fill from 0 to n-2**
  - Find index_of_min, the index of the smallest element in the unsorted subarray list[fill] through list[n-1]
  - if fill is not the position of the smallest element (index_of_min)
    - Exchange the smallest element with the one at position fill.

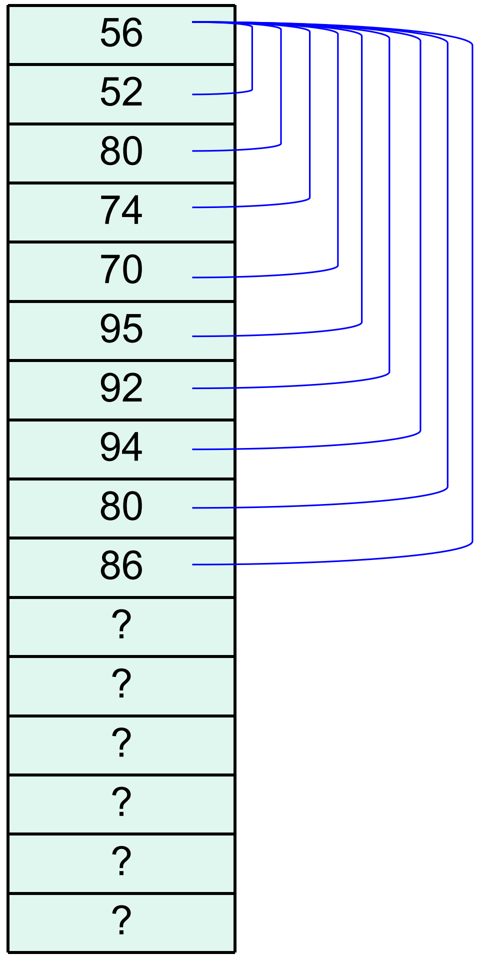
Figure Trace of Selection Sort



- fill is 0. Find smallest element in subarray list[1] through list[3] and swap it with list[0].
- fill is 1. Find the smallest element in subarray list[1] through list[3] - no exchange needed.
- fill is 2. Find the smallest elment in subarray list[2] through list[3] and swap it with list [2].

Brute Force Sort

- Compare element to all elements below and then move to next element, swap when appropriate

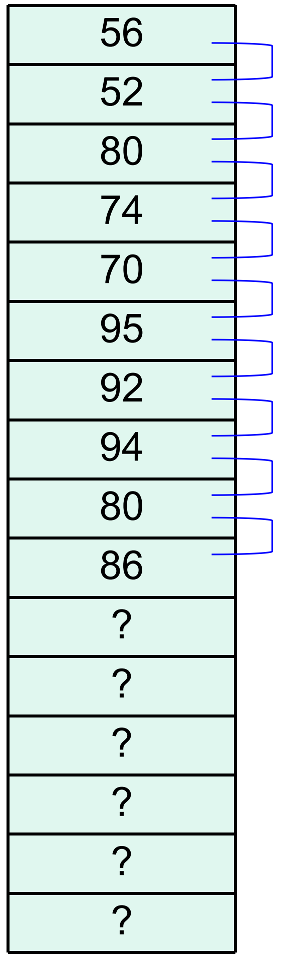


```
void sort_values(int values[ ], int count) {
        // Sort values in ascending order
        // using selection sort
        int sub1, sub2, temp;

        for (sub1=0; sub1<count-1; sub1++)
                for (sub2=sub1+1; sub2<count; sub2++)
                        if (values[sub1]>values[sub2]) {
                                temp = values[sub1];  //swap
                                values[sub1] = values[sub2];
                                values[sub2] = temp;
                        }
}
```

Bubble/Sinking Sort

- Compare adjacent elements, swap when appropriate
- Stop if no swaps on a pass



```
void sort_values(int values[ ], int count) {
        // Sort values in ascending order
        // using selection sort
        int sub1, sub2, temp, sorted = 0;

        for (sub1=0; !sorted && sub1<count-1; sub1++) {
                sorted = 1;        // Assume sorted on each pass
                for (sub2=count-2; sub2>=sub1; sub2--)
                        if (values[sub2]>values[sub2+1]) {
                                temp = values[sub2];  //swap
                                values[sub2] = values[sub2+1];
                                values[sub2+1] = temp;
                                sorted = 0;          // Assume unsorted after swap
                        }
        }
}
```

MONTANA
STATE UNIVERSITY

THE END

Fangtian Zhong
CSCI 112

Gianforte School of Computing
Norm Asbjornson College of Engineering
E-mail: fangtian.zhong@montana.edu

2024.02.23