Programming with C I

Fangtian Zhong
CSCI 112

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

2025.04.02

Arrays of Pointers

Consider some examples:

- int data1, data2, *ptr1, *ptr2, *save;

  ➢ data1 = 100; data2 = 200;
  ➢ ptr1 = &data1; ptr2 = &data2;

We could swap the values of the data and store the swapped values in data1 and data2 or we could simply swap the values of the pointers:

  ➢ save =ptr1;
  ➢ ptr1 = ptr2;
  ➢ ptr2 = save;

Arrays of Pointers

- In general, an array of pointers can be used to point to an array of data items.

- The advantage of a array pointer is that the pointers can be reordered in any manner without moving the data items.

- This approach saves a lot of time, with the additional advantage that the data items remain available in the original order.

Arrays of Pointers

- Let us see how we might implement such a scheme.

- **STRPTRS:** Given an array of strings, use pointers to order the strings in sorted form, leaving the array unchanged.

- We will use an array of character pointers to point to the strings declared as follows:

  - **char * flowerptr[MAX];**

Arrays of Pointers

🛡 It is also possible to assign the value of any string pointer to flowerptr[i]; for example, if s is a string, then it is possible to assign the pointer value s to flowerptr[i]:

- **flowerptr[i] = s;**

🛡 In particular, we can read strings into a two dimensional array, flowers[][], and assign each string pointer, flowers[i] to the element of the pointer array, flowersptr[]:

- **for (i = 0; i < MAX; i++)**
- **flowerptr[i] = flowers[i];**

Arrays of Pointers

- The strings can then be accessed either by flowers[i] or by flowerptr[i].

- We can then reorder the pointers in flowerptr[] so that they successively point to the strings in sorted order.

- We can then print the strings in the original order by accessing them through flowers[i] and print the strings in sorted order by accessing them through flowerptr[i].
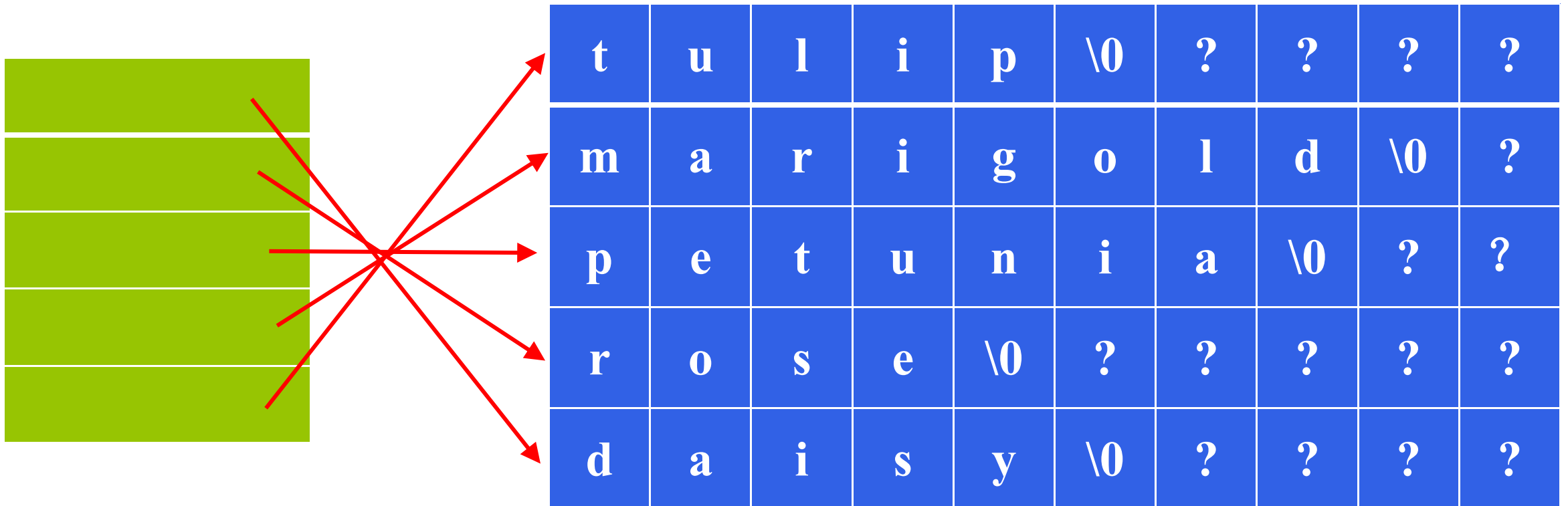
Arrays of Pointers

flowers

t	u	l	i	p	\0	?	?	?	?
m	a	r	i	g	o	l	d	\0	?
p	e	t	u	n	i	a	\0	?	?
r	o	s	e	\0	?	?	?	?	?
d	a	i	s	y	\0	?	?	?	?

alphap

original

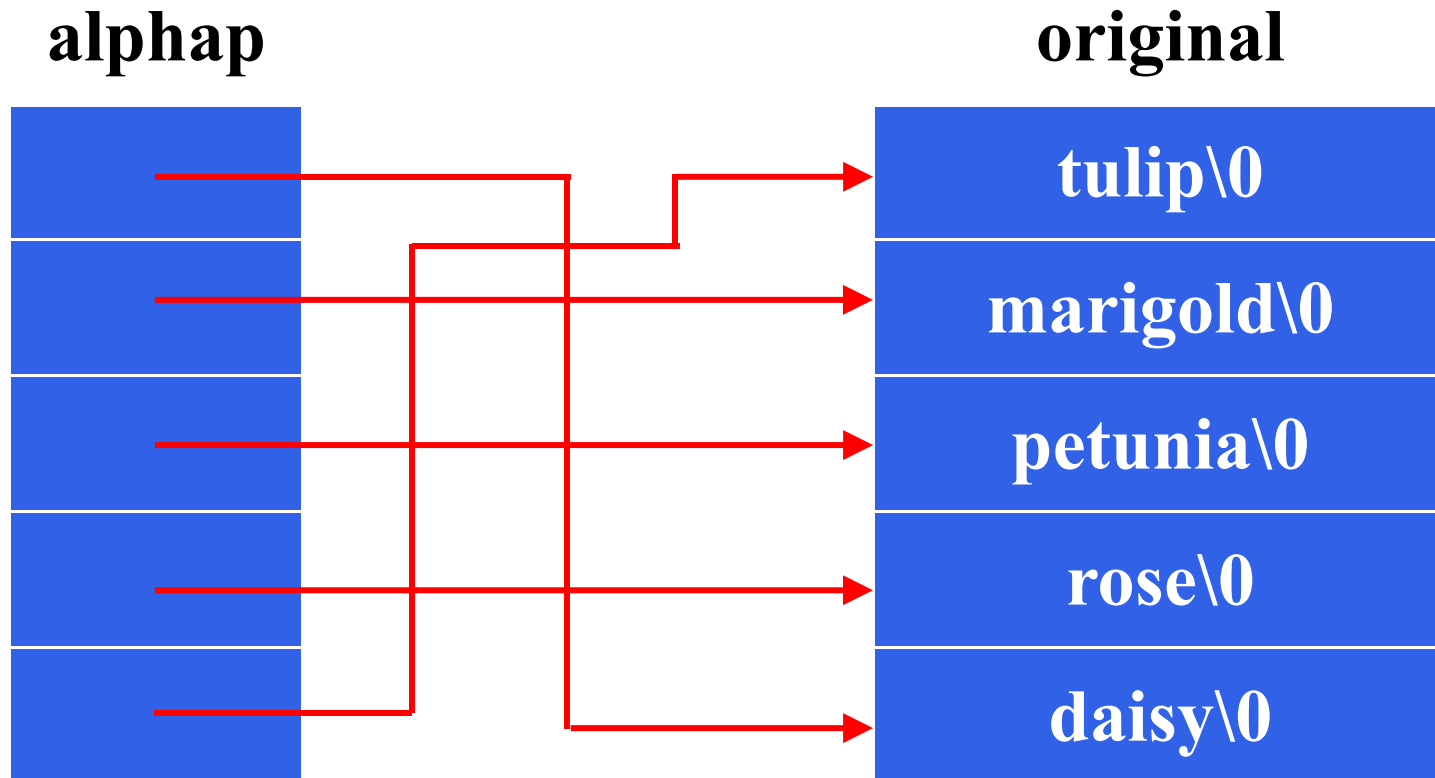| |
|---|
| tulip\0 |
| marigold\0 |
| petunia\0 |
| rose\0 |
| daisy\0 |

Driver for Sorting Pointer Array Program

```c
#include <stdio.h>
#include <string.h>

#define NUM_FLOWERS 5
#define MAX_LEN 20

void selectionSort(char *flowerptr[], int n) {
    int i, j;
    char *temp;

    for (i = 0; i < n - 1; i++) {
        int minIndex = i;
        for (j = i + 1; j < n; j++) {
            if (strcmp(flowerptr[j], flowerptr[minIndex]) < 0) {
                minIndex = j;
            }
        }
        if (minIndex != i) {
            // Swap pointers in flowerptr array
            temp = flowerptr[minIndex];
            flowerptr[minIndex] = flowerptr[i];
            flowerptr[i] = temp;
        }
    }
}
```

```
int main() {
    // Two-dimensional array of flower names
    char flowers[NUM_FLOWERS][MAX_LEN] = {"tulip", "marigold", "petunia", "rose", "daisy"};

    // Array of pointers to strings
    char *flowerptr[NUM_FLOWERS];

    // Assign each string pointer to the corresponding element of the pointer array
    for (int i = 0; i < NUM_FLOWERS; i++) {
        flowerptr[i] = flowers[i];
    }

    // Print original order
    printf("Original order:\n");
    for (int i = 0; i < NUM_FLOWERS; i++) {
        printf("%s ", flowers[i]);
    }
    printf("\n");

    // Sort the flower pointers
    selectionSort(flowerptr, NUM_FLOWERS);

    // Print sorted order
    printf("\nSorted order:\n");
    for (int i = 0; i < NUM_FLOWERS; i++) {
        printf("%s ", flowerptr[i]);
    }
    printf("\n");

    return 0;
}
```
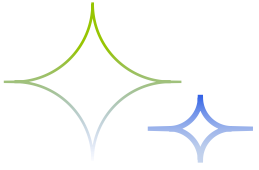
THE END

Fangtian Zhong
CSCI 112

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

2025.04.02