C Programming - Deck 16

Pointers and Strings

Prof. Jyotiprakash Mishra
mail@jyotiprakash.org

Strings and Pointers

- String is an array of characters ending with null terminator
- Character array: `char str[10] = "Hello";`
- String pointer: `char *ptr = "Hello";`
- Array stores string in modifiable memory
- Pointer to string literal points to read-only memory
- String name is a pointer to first character
- Pointer arithmetic works with strings
- Understanding this is crucial for string manipulation

Program 1: String Pointer vs Character Array

```c
#include <stdio.h>
int main() {
  char arr[] = "Hello";
  char *ptr = "World";
  printf("Array: %s\n", arr);
  printf("Pointer: %s\n", ptr);
  arr[0] = 'h';
  printf("Modified array: %s\n", arr);
  printf("arr address: %p\n", (void*)arr);
  printf("ptr address: %p\n", (void*)ptr);
  return 0;
}
```

Output:

```
Array: Hello
Pointer: World
Modified array: hello
arr address: 0x7ffeeb3c4a10
ptr address: 0x10a8e4f28
```

Array is modifiable, pointer to literal is not

Program 2: Traversing String with Pointer

```c
#include <stdio.h>
int main() {
  char str[] = "Hello";
  char *ptr = str;
  printf("Characters:\n");
  while (*ptr != '\0') {
    printf("%c ", *ptr);
    ptr++;
  }
  printf("\n");
  return 0;
}
```

Output:

```
Characters:
H e l l o
```

Moving pointer through string

Program 3: String Length Using Pointer

```c
#include <stdio.h>
int main() {
  char str[] = "Programming";
  char *ptr = str;
  int length = 0;
  while (*ptr != '\0') {
    length++;
    ptr++;
  }
  printf("Length of '%s': %d\n", str, length);
  return 0;
}
```

Output:

```
Length of 'Programming': 11
```

Counting characters until null

Program 4: String Copy Using Pointers

```c
#include <stdio.h>
int main() {
  char src[] = "Hello";
  char dest[20];
  char *ps = src;
  char *pd = dest;
  while (*ps != '\0') {
    *pd = *ps;
    ps++;
    pd++;
  }
  *pd = '\0';
  printf("Source: %s\n", src);
  printf("Destination: %s\n", dest);
  return 0;
}
```

Output:

```
Source: Hello
Destination: Hello
```

Manual string copy with pointers

Program 5: String Concatenation Using Pointers

```c
#include <stdio.h>
int main() {
  char dest[20] = "Hello";
  char src[] = " World";
  char *pd = dest;
  char *ps = src;
  while (*pd != '\0') pd++;
  while (*ps != '\0') {
    *pd = *ps;
    pd++;
    ps++;
  }
  *pd = '\0';
  printf("Result: %s\n", dest);
  return 0;
}
```

Output:

```
Result: Hello World
```

Appending one string to another

Program 6: String Comparison Using Pointers

```c
#include <stdio.h>
int main() {
  char str1[] = "Hello";
  char str2[] = "Hello";
  char *p1 = str1;
  char *p2 = str2;
  while (*p1 != '\0' && *p2 != '\0') {
    if (*p1 != *p2) break;
    p1++;
    p2++;
  }
  if (*p1 == *p2) {
    printf("Strings are equal\n");
  } else {
    printf("Strings are not equal\n");
  }
  return 0;
}
```

Output:

```
Strings are equal
```

Character-by-character comparison

Program 7: Reverse String Using Pointers

```c
#include <stdio.h>
int main() {
  char str[] = "Hello";
  char *left = str;
  char *right = str;
  char temp;
  while (*right != '\0') right++;
  right--;
  while (left < right) {
    temp = *left;
    *left = *right;
    *right = temp;
    left++;
    right--;
  }
  printf("Reversed: %s\n", str);
  return 0;
}
```

Output:

```
Reversed: olleH
```

Two-pointer approach for reversal

Program 8: Count Vowels Using Pointer

```c
#include <stdio.h>
int main() {
  char str[] = "Programming";
  char *ptr = str;
  int count = 0;
  while (*ptr != '\0') {
    char ch = *ptr;
    if (ch=='a'||ch=='e'||ch=='i'||
        ch=='o'||ch=='u'||ch=='A'||
        ch=='E'||ch=='I'||ch=='O'||
        ch=='U') {
      count++;
    }
    ptr++;
  }
  printf("Vowels in '%s': %d\n", str, count);
  return 0;
}
```

Output:

```
Vowels in 'Programming': 3
```

Checking each character for vowel

Program 9: Convert to Uppercase Using Pointer

```c
#include <stdio.h>
int main() {
  char str[] = "hello world";
  char *ptr = str;
  printf("Before: %s\n", str);
  while (*ptr != '\0') {
    if (*ptr >= 'a' && *ptr <= 'z') {
      *ptr = *ptr - 32;
    }
    ptr++;
  }
  printf("After: %s\n", str);
  return 0;
}
```

Output:

```
Before: hello world
After: HELLO WORLD
```

Converting lowercase to uppercase

Array of String Pointers

- Array where each element points to a string
- Syntax: `char *arr[5];`
- Each pointer can point to different string literals
- Useful for managing multiple strings
- Common for command-line arguments: `char *argv[]`
- Strings can have different lengths
- Memory efficient for read-only strings
- Example: `char *days[] = {"Mon", "Tue", "Wed"};`

Program 10: Array of String Pointers

```c
#include <stdio.h>
int main() {
  char *fruits[5] = {
    "Apple",
    "Banana",
    "Cherry",
    "Date",
    "Elderberry"
  };
  int i;
  printf("Fruits:\n");
  for (i = 0; i < 5; i++) {
    printf("%d. %s\n", i+1, fruits[i]);
  }
  return 0;
}
```

Output:

```
Fruits:
1. Apple
2. Banana
3. Cherry
4. Date
5. Elderberry
```

Each pointer points to a string literal

Program 11: Printing Strings with Pointers

```c
#include <stdio.h>
int main() {
  char *days[] = {
    "Monday",
    "Tuesday",
    "Wednesday",
    "Thursday",
    "Friday"
  };
  char **ptr = days;
  int i;
  for (i = 0; i < 5; i++) {
    printf("%s\n", *(ptr + i));
  }
  return 0;
}
```

Output:

```
Monday
Tuesday
Wednesday
Thursday
Friday
```

Pointer to pointer for string array

Program 12: Find Longest String

```c
#include <stdio.h>
int main() {
  char *words[] = {"Hi", "Hello", "Hey", "Greetings"};
  int i, len, maxLen = 0;
  int maxIndex = 0;
  char *ptr;
  for (i = 0; i < 4; i++) {
    len = 0;
    ptr = words[i];
    while (*ptr != '\0') {
      len++;
      ptr++;
    }
    if (len > maxLen) {
      maxLen = len;
      maxIndex = i;
    }
  }
  printf("Longest: %s (%d)\n",
    words[maxIndex], maxLen);
  return 0;
}
```

Output:

```
Longest: Greetings (9)
```

Finding longest string in array

Program 13: Search String in Array

```c
#include <stdio.h>
int main() {
  char *colors[] = {"Red", "Green", "Blue", "Yellow"};
  char *search = "Blue";
  int found = 0;
  int i;
  char *p1, *p2;
  for (i = 0; i < 4; i++) {
    p1 = colors[i];
    p2 = search;
    while (*p1 && *p2 && *p1 == *p2) {
      p1++;
      p2++;
    }
    if (*p1 == *p2) {
      printf("Found '%s' at index %d\n", search, i);
      found = 1;
      break;
    }
  }
  if (!found) printf("Not found\n");
  return 0;
}
```

Output:

```
Found 'Blue' at index 2
```

Searching for specific string

Program 14: Count Total Characters in String Array

```c
#include <stdio.h>
int main() {
  char *words[] = {"C", "is", "powerful"};
  int total = 0;
  int i;
  char *ptr;
  for (i = 0; i < 3; i++) {
    ptr = words[i];
    while (*ptr != '\0') {
      total++;
      ptr++;
    }
  }
  printf("Total characters: %d\n", total);
  return 0;
}
```

Output:

```
Total characters: 12
```

Summing lengths of all strings

Program 15: Pointer Arithmetic with Strings

```c
#include <stdio.h>
int main() {
  char str[] = "Hello World";
  char *ptr = str;
  printf("Full string: %s\n", ptr);
  printf("From index 6: %s\n", ptr + 6);
  printf("Character at 0: %c\n", *ptr);
  printf("Character at 4: %c\n", *(ptr + 4));
  printf("Third char: %c\n", ptr[2]);
  return 0;
}
```

Output:

```
Full string: Hello World
From index 6: World
Character at 0: H
Character at 4: o
Third char: l
```

Using pointer arithmetic on strings

Program 16: Remove Spaces Using Pointer

```c
#include <stdio.h>
int main() {
  char str[] = "H e l l o";
  char *pr = str;
  char *pw = str;
  while (*pr != '\0') {
    if (*pr != ' ') {
      *pw = *pr;
      pw++;
    }
    pr++;
  }
  *pw = '\0';
  printf("Result: %s\n", str);
  return 0;
}
```

Output:

```
Result: Hello
```

Two pointers: read and write

Program 17: Check Palindrome Using Pointers

```c
#include <stdio.h>
int main() {
  char str[] = "madam";
  char *left = str;
  char *right = str;
  int isPal = 1;
  while (*right != '\0') right++;
  right--;
  while (left < right) {
    if (*left != *right) {
      isPal = 0;
      break;
    }
    left++;
    right--;
  }
  if (isPal) printf("'%s' is palindrome\n", str);
  else printf("'%s' is not palindrome\n", str);
  return 0;
}
```

Output:

```
'madam' is palindrome
```

Checking from both ends

Program 18: Count Words Using Pointer

```c
#include <stdio.h>
int main() {
  char str[] = "Hello World from C";
  char *ptr = str;
  int words = 0;
  int inWord = 0;
  while (*ptr != '\0') {
    if (*ptr == ' ') {
      inWord = 0;
    } else if (inWord == 0) {
      inWord = 1;
      words++;
    }
    ptr++;
  }
  printf("Word count: %d\n", words);
  return 0;
}
```

Output:

```
Word count: 4
```

Counting words separated by spaces

Program 19: Find Character in String

```c
#include <stdio.h>
int main() {
  char str[] = "Programming";
  char ch = 'g';
  char *ptr = str;
  int pos = -1;
  int i = 0;
  while (*ptr != '\0') {
    if (*ptr == ch) {
      pos = i;
      break;
    }
    ptr++;
    i++;
  }
  if (pos != -1) {
    printf("'%c' found at position %d\n", ch, pos);
  } else {
    printf("'%c' not found\n", ch);
  }
  return 0;
}
```

Output:

```
'g' found at position 3
```

Finding first occurrence of character

Program 20: Replace Character Using Pointer

```c
#include <stdio.h>
int main() {
  char str[] = "Hello World";
  char *ptr = str;
  char oldCh = 'o';
  char newCh = '*';
  int count = 0;
  printf("Before: %s\n", str);
  while (*ptr != '\0') {
    if (*ptr == oldCh) {
      *ptr = newCh;
      count++;
    }
    ptr++;
  }
  printf("After: %s\n", str);
  printf("Replacements: %d\n", count);
  return 0;
}
```

Output:

```
Before: Hello World
After: Hell* W*rld
Replacements: 2
```

Replacing all occurrences

Key Takeaways

- String is a character array with null terminator
- String literal creates pointer to read-only memory
- Character array is modifiable, string literal is not
- Pointer arithmetic works naturally with strings
- Array of string pointers: `char *arr[]`
- Pointers simplify string manipulation operations
- Two-pointer technique useful for many operations
- Always ensure null terminator when modifying strings
- Understanding pointers with strings is essential for C