

Weeks 1–6

- Programming: from Turtle to C; data types and operators.
- Input and output; if-else, switch, and loops.
- Numerical methods implemented in C.
- Single-dimensional arrays.

Where we are: Week 7

Weeks 1–6

- Programming: from Turtle to C; data types and operators.
- Input and output; if-else, switch, and loops.
- Numerical methods implemented in C.
- Single-dimensional arrays.

Week 7

Strings and multi-dimensional arrays in C.

Character arrays

A character array can hold a string terminated by '`\0`'.

```
char name[20] = "AVANI";  
/* Equivalent explicit initialization: */  
char other[20] = {'A', 'V', 'A', 'N', 'I', '\0'};
```

Character arrays

A character array can hold a string terminated by '`\0`'.

```
char name[20] = "AVANI";  
/* Equivalent explicit initialization: */  
char other[20] = {'A', 'V', 'A', 'N', 'I', '\0'};
```

Or read one word into a character array:

```
char word[20];  
scanf("%19s", word);
```

Character arrays

A character array can hold a string terminated by `'\0'`.

```
char name[20] = "AVANI";  
/* Equivalent explicit initialization: */  
char other[20] = {'A', 'V', 'A', 'N', 'I', '\0'};
```

Or read one word into a character array:

```
char word[20];  
scanf("%19s", word);  
  
char name[20];  
name = "AVANI";           /* Incorrect: cannot assign to an array. */
```

The earlier “null char” placeholder is written as `'\0'`. The input width leaves room for the terminator.

What is the output of this program?

```
#include<stdio.h>
int main(void) {
    char name[20] = "AVANI";
    int i;

    for (i=10; i<20; i++) {
        name[i] = 'X';
    }

    printf("name = %s\n", name);

    for (i=0; i<20; i++) {
        printf("%c %d\n", name[i], name[i]);
    }
}
```

Source: `printascii.c`. Only the signature of `main` was made explicit; the array and loops are unchanged.

Character arrays and standard library support

- Character arrays or strings occur very often.
- C provides a standard library [string.h](#)
- exposes several useful functions:
 - `strlen`
 - `strcmp`
 - `strcpy`
 - `strstr`
- But we can create libraries.

Example 1: Finding the length of a given string

Task: Read a word and find its length.

Pseudo-code:

- Start at index 0.
- Advance until the null character is reached.
- Output the index of that null character.

Example 1: Finding the length of a given string

Task: Read a word and find its length.

Pseudo-code:

- Start at index 0.
- Advance until the null character is reached.
- Output the index of that null character.

The terminating semicolon is intentional: the for-loop has an empty body.

Program segment:

```
char s[1000];  
int i;  
scanf("%999s", s);  
  
for(i=0 ; s[i] != '\0'; ++i);  
  
printf("Length : %d", i);
```

Source: `strlen.c`. Changed the counter from `char` to `int`, and bounded `scanf` to 999 characters for the 1000-byte array.

Example 2: Compare two strings

Task: Check whether two strings have the same contents.

```
if (s1 == s2)
```

This does not compare their contents.

A first approach:

Compare lengths first, then compare corresponding characters.

Can we combine the steps?

Scan both strings together. Stop on a mismatch or a terminator.

Example 2: Compare two strings

Task: Check whether two strings have the same contents.

```
if (s1 == s2)
```

This does not compare their contents.

A first approach:

Compare lengths first, then compare corresponding characters.

Can we combine the steps?

Scan both strings together. Stop on a mismatch or a terminator.

Program segment:

```
// strings are in arrays a and b
int i = 0;
while (a[i] == b[i]) {
    if ((a[i] == '\0') || (b[i] == '\0'))
        break;
    i++;
}
if (a[i] == '\0' && b[i] == '\0')
    printf("SAME");
else
    printf("NOT SAME");
```

Source: string-compare.c. Added its missing closing parenthesis in the if-condition; comparison logic is unchanged.

Example 3 : Palindromes

A string is a palindrome iff $\text{string} == \text{reverse}(\text{string})$

Example 3 : Palindromes

A string is a palindrome iff $\text{string} == \text{reverse}(\text{string})$

- malayalam
- neveroddoreven
- dontnod

Example 3 : Palindromes

A string is a palindrome iff `string == reverse(string)`

- malayalam
- neveroddoreven
- dontnod

Write a program to determine if the given string is a palindrome.

The code compares characters exactly; for phrase examples, remove spaces first.

Example 3: Palindromes

Task: Given a string, check if it is a palindrome.

Pseudo-code:

- Start at the first and last characters.
- Compare the characters, then move inward.
- If a pair differs, declare NOT PALINDROME.
- If all pairs match, declare PALINDROME.

Does this program implement that last step correctly?

From `palindrome-run.c`:

```
int l = 0;
char str[] = "abcba";
int h = strlen(str) - 1;
while (h > l) {
    if (str[l++] != str[h--]) {
        printf("NOT PALINDROME");
        break;
    }
}
if (h == l)
    printf("PALINDROME");
```

Main-body excerpt; unused declaration omitted.