Outline

- String Basics
- String Library Functions
- String Comparison
- Array of Pointers
- String-to-Number and Number-to-String Conversion
- Case Study
- Common Programming Errors
String Basics

- String: array of characters with the null character (\0) to mark the end.
- The data type char alone is not very useful, but string is.

Example:
- Word Processing
- Internet Search
- Spell Check...
Declaring and Initializing String

- Use `#define`

```c
#define ERROR_MESSAGE "Error has occurred!"
```

- Declaring as an array of type `char`:
  ```c
  char string_var[30];
  ```

- Initialization:
  ```c
  char str[20] = "Initial value";
  ```
Array of Strings

- An array of strings a two-dimensional array of characters in which each row is a string.

- Example:
  ```
  #define NUM_PEOPLE 30
  #define NAME_LEN 25
  ..
  char names[NUM_PEOPLE][NAME_LEN]
  ```

- Initialization:
  ```
  char month[12][4]=
  ```
Input/Output

- printf:
  ```c
  printf("Topic: %s\n",string_var);
  ```

- Right vs left-adjusted (use -)
  ```c
  printf("Topic: %-20s\n",string_var);
  ```

- scanf: no need to have & since array is always passed by reference.
#include <stdio.h>
#define STRING_LEN 10

int main(void)
{
    char dept[STRING_LEN];
    int course_num;
    char days[STRING_LEN];
    int time;

    printf("Enter department code, course number, days and ");
    printf("time like this:\n> COSC 2060 MWF 1410\n> ");
    scanf("%s%d%s%d", dept, &course_num, days, &time);
    printf("%s %d meets %s at %d\n", dept, course_num, days, time);
    return (0);
}
String Library Functions

- Cannot use assignment in the following manner:
  ```c
  char one_str[20];
  one_str = "Test string";
  ```
- Same for <, >, or ==, ...
- `<string.h>`
String Library Functions (1)

- `strcpy (char *dest, const char *source)`
  ```c
  strcpy(s1, “hello”);
  ```

- `strncpy(char *dest, const char *source, size_t n)`
  where `size_t` is an unsigned integer.
  ```c
  strncpy(s1, “Inevitable”, 5);
  ```

  Note: \0 not added

- `strcat(char *dest, const char *source)`
  ```c
  strcat(s1, “and more”);
  ```

- `strncat(char *dest, const char *source, size_t n)`
  ```c
  strncat(s1, “and more”, 10);
  ```
String Library Functions (2)

- `strcmp(const char *s1, const char *s2)`
  
  ```c
  if (strcmp(name1, name2) == 0) ... 
  > 0 if s2 precedes s1, <0 if s1 precedes s2
  ```

- `strncmp(const char *s1, const char *s2, size_t n)`
  
  ```c
  if (strncmp(s1, s2, 10) == 0) ... 
  ```

- `strlen(const char *s)`
  
  ```c
  strlen("What") returns 4 
  ```
Can use `strncpy` to get substrings.

But you’ll need to take care of the termination character `\0`.

Example:
```c
strncpy(result, s1, 9);
result[9] = '\0';
```

To extract a middle substring:
```c
strncpy(result, &s1[5], 2);
result[2] = '\0';
```
Concatenation

- strcat, strncat
- Example:
  ```c
  #define STRSIZ 15
  ```
- This is OK: `strcat(f1,last);`
- This is not, why? `strcat(f2,last);`
- Important: It’s up to the programmer to check whether there is enough space to store the whole string, and make sure that the string ends with ‘\0’.
Distinction between char and strings

- ‘Q’ --> character
- “Q” --> string
- Remember the termination character!
- Use gets to scan a complete line of data.
String Comparison

- `strcmp`, `strncmp`
- But it this OK? (assuming `str1` and `str2` are strings)
  \[ str1 < str2 \]
- This answer is: YES

But, it compares the memory location, not the string content !!!
Array of Pointers

- Array of strings is equivalent to array of pointers to char.
- So the following declaration makes sense: `char *alpha[5];`
- See page 452 of text for an example of using array of pointers to show two orderings of one list.
Character Operations

- Character input:
  - `scanf("%c", &ch);`
  - `ch = getchar();`

- Character output:
  - `putchar('a');`

- Character classification and conversion:
  `<ctype.h>`
  - `isalpha, isdigit, islower, tolower, toupper...`
String <--> Number

- **Number to string conversion:** `sprintf`
  ```c
  sprintf(s, "%d/%d/%d", mon, day, year);
  ```

- **String to number conversion:**
  ```c
  sscanf(" 85 96.2 hello", "%d%lf%s", &num, &val, word);
  ```
Case Study

- Text Editor to perform editing on a line of text. Should be able to:
  - locate a specified target substring
  - delete a substring
  - insert a substring at a specified location
- Source string is less than 80 characters in length.
Your Homework

- Read Chapter 9 of textbook
- Case Study: text editor (page 468-476 of text)
- Due on 12/8/2000. Please send the source code to your TA via e-mail.