Intro. Computing with the C Programming Language

Strings

Shin Hong

16 October 2023

String Variables

- a string is stored as an array of characters terminated by null (‘\0’, or NULL).
  - there is no separate type of string in C

- Initializing character arrays with string

```
char first[] = “Hello“ ;

char first[6] = {‘H’, ‘e’, ‘l’, ‘l’, ‘o’, ‘\0’} ;

printf(“%s\n”, first) ;
printf(“%c %c\n”, first[0], first[1]) ;
```

String Operations

- `string.h` provides a set of functions for manipulating strings
  - https://www.gnu.org/software/libc/manual/html_node/String-and-Array-Utilities.html

- `strlen()` returns the length of a string
  - the length of a string is the number of characters before NULL
  - it is not same as the length of the container array

- Example
  - locate.c

More String Operations

- determine whether two strings contain the same text
- determine which of two strings precedes in lexicographical order

Pointer

- Every variable is located at a specific memory address (or memory location)
- In C, we can directly access a memory address as a number
- A pointer variable is a variable with a special type to hold a memory address

```
int number = 5 ;
int * i_p ;
i_p = &number ;     // address-of
int m = *number;    // content-of
```

String Concatenation

- Put the content of a string to the tail of another string

```
char fruit[20] = "banana";
char bakedGood[] = " nut bread";
strncat(fruit, bakedGood, 10);
printf ("%s\n", fruit);
```

Assigning New Values to String

- It is not possible to assign a string constant directly to a string variable
  - e.g., `fruit = "orange";`
    ```
    /* Wrong: Cannot assign directly! */
    ```
- `strncpy()` is to copy the content of a string to another string
  - e.g
    ```
    strncpy (greeting, "Hello, world!", 14);
    strncpy (greeting, "Hello, world!", 5);
       /*only Hello is copied*/
    greeting[5] = ’\0’;
    ```

Character Classification

- C provide a library of functions that determines the character classification
  - e.g.,
    ```
    char letter = ’a’;
    if (isalpha(letter))  {
       printf("The character %c is a letter.",
         letter);
    }
    ```

- other functions
  - `int isalnum(int c);`
  - `int isalpha(int c);`
  - `int iscntrl(int c);`
  - `int isdigit(int c);`
  - `int isgraph(int c);`
  - `int islower(int c);`
  - `int isprint(int c);`
  - `int ispunct(int c);`
  - `int isspace(int c);`
  - `int isupper(int c);`
  - `int isxdigit(int c);`

Getting User Input (1/2)

- scanf() returns the number of items that have been successfully read

```
int main (void) {
   int success, x;
   /* prompt the user for input */
   printf ("Enter an integer: \n");
   /* get input */
   success = scanf("%i", &x);
   /* check and see if the input statement succeeded */
   if (success == 1)
   {
      /* print the value we got from the user */
      printf ("Your input: %i\n", x);
      return EXIT_SUCCESS;
   }
   printf("That was not an integer.\n");
   return EXIT_FAILURE ;
}
```

Get User Inputs (2/2)

- getchar() reads one character at a time
  - e.g.,

```
char ch; /* helper variable stores discarded chars*/
while (success != 1) {
   printf("That isn’t a number. Please try again:\n");

   ch = getchar() ;
   while (ch != ’\n’ && ch != EOF) {
      ch = getchar() ;
   }
   success = scanf("%i", &x);
```