Operations on Strings

IC-100

December 12, 2022

Strings

- Last class
  - Strings as a special type of array
    - Char array terminated with ‘\0’
  - String input using scanf, gets and fgets
  - String printing using printf, puts
- This class
  - String specific functions
    - What makes strings superior to char arrays for handling text
  - String operations
    - Using compositions of primitive string function operations

Copying One String to Other

- We **cannot** copy content of one string variable to other using assignment operator

```
char str1[] = "Hello";
char str2[] = str1;
```

WRONG

Array type is not assignable
.

C Pointers needed!

  - This is true for any array variable.
  - Error: Array initializer must be a list or a string.
- We need to do element-wise copying

String Copy

str_copy(char dest[], char src[]);

- Arguments: Two strings: dest and src.
- Copy contents of src into dest.
- We assume that dest is declared with size at least as large as src.
- Note the use of ‘\0’ for loop termination

```
void str_copy(char dest[], char src[]) {
   int i;
   for (i = 0; src[i] != '\0'; i++)
       dest[i] = src[i];
   dest[i] = '\0';
}
```

Comparing Two Strings

- Lexicographical Ordering
  - A string str1 is said to be lexicographically smaller than another string str2 if the first character, where the strings differ, is smaller in str1.
- Examples:
  - "cap" is smaller than "cat".
  - "mat" is smaller than "matter".
- Order of words in a Dictionary or ASCII value.

String Comparison

- We will write a function that compares two strings lexicographically:

  str_compare (char str1[], char str2[])

- Arguments: Two strings str1 and str2
- Return value:
  - 0 if the strings are equal,
  - -1 if str1 is "smaller",
  - 1 if str2 is "smaller".
- Assumption: The strings contain letters of one case (either capital or small).

Code for str_compare

```
int str_compare(char str1[], char str2[]){
   int i=0;
   while (str1[i]==str2[i]){//skip over same chars
      if (str1[i]=='\0')
         break;
      i++;
   }
   if (str1[i] == str2[i])
      return 0;
   else if (str1[i] < str2[i])
      return -1;
   else //str2 < str1
      return 1;
}
```

When can this happen?

At this point, since the first differing characters are such that str1[i] < str2[i], => str1 is smaller

Other String Functions

- Return *length* of a string
- *Concatenates* one string with another
- Search for a *substring* in a given string
- *Reverse* a string
- Find first/last/k-th occurrence of a *character* in a string
- Case sensitive/*insensitive* versions of comparing two strings

string.h

- *Header File with Functions on Strings*
- *strlen(s): returns length of string s (without '\0')*
- *strcpy(d, s): copies s into d*
- *strcat(d, s): appends s at the end of d ('\0' is moved to the end of result)*

string.h

- *strcmp(s1, s2)*: *return an integer less than, equal to, or greater than zero if s1 is found, respectively, to be less than, to match, or be greater than s2.*

- *Example:*

```
char str1[] = "Hello", str2[] = "Helpo";
int i = strcmp(str1,str2);
printf("%d", i);
```

- *Prints the value 'l'-'p' which is -4.*

string.h

- *strncpy(d, s, n)*
- *strncat(d, s, n)*
- *strncmp(d, s, n)*
  - *restrict the function to "n" characters at most (argument n is an integer)*
  - *first two functions-- Truncate the string s to the first "n" characters.*
  - *third function-- Truncate the strings d, s to the first "n" characters.*

```
char str1[] = "Hello", str2[] = "Helpo";
printf("%d",strncmp(str1,str2,3));
```

0

string.h

- *strcasecmp, strncasecmp:*

  case insensitive comparison.

- *Example:*

```
char str1[] = "HELLO", str2[] = "Helpo";
int i = strcmp(str1,str2);
int j = strcasecmp(str1,str2);
printf("%d %d", i, j);
```

```
-32 -4
```

- *strcmp gives -32 because 'E' < 'e' .*
  - *'E'-'e' = -32 .*

string.h

- *Many more utility functions.*

- *strupr(s) : converts lower to upper case.*
- *strlwr(s) : converts upper to lower case.*
- *strstr(S,s) : searches s in S. Returns a pointer to the first occurrence.*

- *All functions depend on '\0' as the end-of-string marker.*

Program Example

- Exercise: Write a program to see whether the phrase “very nice” occurs in [illegible] g i

What is this?

String functions that search within strings return pointers

```
void str_find() {
  char str[1000];
  gets(str);
  char *p = strstr(str, "very nice");
  if(p!=NULL){
      printf("%s\n", p);
      printf("%d\n", p-str+1);
  }else
      printf("Not found");}
```

Could we search for a phrase in the string without using pointers?

Left as (tedious) exercise