Characters and Strings (Cont.)

Character Handling Library `stdio.h`

Prototypes and Function Descriptions
`int getchar(void);`
Inputs the next character from the std input and returns it as an integer.
`char *fgets(char *s, int n, FILE *stream);`
Inputs characters from the stream into array s until a newline or end-of-file character is encountered. If a newline character is encountered it is included in the strings and stored in s. \0 is appended at the end of the array.
`int putchar(int c);`
Prints character stored in c and returns it as an integer.

Prototypes and Function Descriptions
`int putschar(const char *s);`
Prints the string s followed by a newline character. Returns a nonzero if successful, or EOF if an error occurs.
`int sprintf(char *s, char char *format, ...);`
Equivalent to printf, except the output is stored in the array s instead of printed on the screen. Returns the number of characters written to s, or EOF if an error occurs.
`int sscanf(char *s, char char *format, ...);`
Equivalent to scanf, except the input is read from the array s rather than from the keyboard. Returns the number of items successfully read by the function, or EOF if an error occurs.

```c
#include <stdio.h>
#define SIZE 80
int main() {
  int x1, x2;
  double y1, y2;
  puts("Enter an integer and a double:");
  scanf("%d%lf", &x1, &y1);
  char s1[SIZE];
  sprintf(s1, "Integer:%6d\nDouble:%7.2f", x1, y1);
  printf("String: %s\nInteger: %d\nDouble: %f\n",
  s1, x1, y1);

  char s2[] = "31324 76.375";
  sscanf(s2, "%d%lf", &x2, &y2);
  printf("String: %s\nInteger: %d\nDouble: %f",
  s2, x2, y2);

  return 0;
}
```

Character Handling Library `string.h`

Prototype and Function Description
`char *strcpy(char *s1, const char *s2);`
Copy string s2 into array s1 and returns s1.
`char *strncpy(char *s1, const char *s2, size_t n);`
Copy n characters of string s2 into array s1 and returns s1.
`char *strcat(char *s1, const char *s2);`
Appends string s2 to array s1. First character of s2 overwrites \0 of s1, returns s1.
`char *strncat(char *s1, const char *s2, size_t n);`
Appends n characters of string s2 to array s1. First character of s2 overwrites \0 of s1. The value of s1 is returned.

`strcpy` and `strncpy`

- Function `strncpy` is equivalent to `strcpy`, except that `strncpy` specifies the number of characters to be copied.
- First argument array must be large enough to store the second string and its terminating '\0'.
- Function `strncpy` does not necessarily copy the terminating '\0' of its second argument.
- Function `strncpy` causes a missing terminating '\0' for first argument when n is < or = length of second argument
- A terminating '\0' must be appended to array after the call to `strncpy` in the program does not.

```c
#include <stdio.h>
#include <string.h>
#define SIZE1 25
#define SIZE2 15
int main() {
  char a[] = "Happy Birthday to You";
  char b[SIZE1];
  char c[SIZE2];

  strcpy(b, a);
  strncpy(c, a, SIZE2-1);
  c[SIZE2 -1] = '\0';

  return 0;
}
```

`strcat` and `strncat`

- Function `strncat` is equivalent to `strcat`, except that `strncat` specifies the number of characters to be appended.
- First argument array must be large enough to store the second string and its terminating '\0'.
- First character of the second argument replaces the terminating '\0' of the first argument.
- A terminating '\0' is appended to the result.

```
char s1[20] = "Happy";
char s2[] = " New Year";
char s3[40] = "";

strcat(s1, s2);
strncat(s3, s1, 6);
```

Prototype and Function Description
`int strcmp(const char *s1, const char *s2);`
Compares the string s1 with the string s2. Returns 0 if s1 is equal, < 0 if s1 is less than or >0 if s2 is greater than.
`int strncmp(const char *s1, const char *s2, size_t n);`
Compares up to the n characters of the string s1 with the string s2. Returns 0 if s1 is equal, < 0 if s1 is less than or >0 if s2 is greater than.

`strcmp` and `strncmp`

- Compares first and second string character by character.
- Alphabetical order plays into comparisons
- All characters are represented as numeric codes in character sets such as ASCII and Unicode

```
const char *s1 = "Happy New Year";
const char *s2 = "Happy New Year";
const char *s3 = "Happy Holidays";

strcmp(s1, s2);
strcmp(s1, s3);
strcmp(s3, s1);

strncmp(s1, s3, 6);
strncmp(s1, s3, 7);
strncmp(s3, s1, 7);
```

Prototypes and Function Descriptions
`char *strchr(const char *s, int c);`
Locates the first occurrence of character c in string s. If c is found, a pointer to c in s is returned. Otherwise, a NULL pointer is returned.
`size_t strcspn(const char *s1, const char *s2);`
Determines and returns the length of the initial segment of string s1 consisting of characters not contained in string s2.
`size_t strspn(const char *s1, const char *s2);`
Determines and returns the length of the initial segment of string s1 consisting only of characters contained in string s2.
`char *strpbrk(const char *s1, const char *s2);`
Locates the first occurrence in string s1 of any character in string s2. If a character from string s2 is found, a pointer to the character in string s1 is returned. Otherwise, a NULL pointer is returned.

Prototypes and Function Descriptions
`char *strrchr(const char *s, int c);`
Locates the last occurrence of c in string s. If c is found, a pointer to c in string s is returned. Otherwise, a NULL pointer is returned.
`char *strstr(const char *s1, const char *s2);`
Locates the first occurrence in string s1 of string s2. If the string is found, a pointer to the string in s1 is returned. Otherwise, a NULL pointer is returned.
`char *strtok(char *s1, const char *s2);`
A sequence of calls to strtok breaks string s1 into tokens separated by characters contained in string s2. The first call contains s1 as the first argument, and subsequent calls to continue tokenizing the same string contain NULL as the first argument. A pointer to the current token is returned by each call. If there are no more tokens when the function is called, NULL is returned.

```
const char *string1 = "This is a test apple";
const char *string2 = "The value is 3.14159";
const char *string3 = "1234567890";
const char *string4 = "aehi lsTuv";
const char *string5 = "beware";
const char *string6 = "456";
char letter1 = 'a';
char letter2 = 'z';

if (strchr(string1,letter1)) { printf("Found\n"); }
else { printf("Not Found\n"); }

if (strchr(string1,letter2)) { printf("Found\n"); }
else { printf("Not Found\n"); }

printf("Length (#) of string1 having none of strings3
characters = %u\n", strcspn(string2, string3));

printf("Length (#) of string2 containing only strings4
characters = %u\n", strspn(string2, string4));
```

```
const char *string1 = "This is a test apple";
const char *string2 = "The value is 3.14159";
const char *string3 = "1234567890";
const char *string4 = "aehi lsTuv";
const char *string5 = "beware";
const char *string6 = "456";
char letter1 = 'a';
char letter2 = 'z';

printf("First appearing character of string5
in string1 = %c\n", strpbrk(string1, string5));

printf("Remainder of string1 starting with the
last occurence of character is \"%s\"\n",
strrchr(string1, letter1));

printf("Remainder of string3 with the first occurence
of string6 \"%s\"\n", strstr(string3, string6));
```

Questions?