Characters and Strings

Fundamentals of Strings and Characters

- Characters
  - Building blocks of programs
    - Every program is a sequence of meaningfully grouped characters
  - Character constant
    - An `int` value represented as a character in single quotes
    - `'z'` represents the integer value of `z`
- Strings
  - Series of characters treated as a single unit
    - Can include letters, digits and special characters (`*`, `/`, `$`)
  - String literal (string constant) - written in double quotes
    - `"Hello"`
  - Strings are arrays of characters
    - String a pointer to first character
    - Value of string is the address of first character

Fundamentals of Strings and Characters

- String declarations
  - Declare as a character array or a variable of type `char *`

```
char color[] = "blue";
char *colorPtr = "blue";
```

  - Remember that strings represented as character arrays end with `'\0'`
    - `color` has `5` elements
- Inputting strings
  - Use `scanf`

```
scanf("%s", word);
```

    - Copies input into `word[]`
    - Do not need `&` (because a string is a pointer)
  - Remember to leave room in the array for `'\0'`

Standard Input/Output Library Functions

- Functions in `<stdio.h>`
  - Used to manipulate character and string data

Function prototype	Function description
`int getchar( void );`	Inputs the next character from the standard input and returns it as an integer.
`char *gets( char *s );`	Inputs characters from the standard input into the array `s` until a newline or end-of-file character is encountered. A terminating null character is appended to the array.
`int putchar( int c );`	Prints the character stored in `c`.
`int puts( const char *s );`	Prints the string `s` followed by a newline character.

String Manipulation Functions of the String Handling Library

- String handling library has functions to
  - Manipulate string data
  - Search strings
  - Tokenize strings
  - Determine string length

Function prototype	Function description
`char *strcpy( char *s1, const char *s2 )`	Copies string **s2** into array **s1**. The value of **s1** is returned.
`char *strncpy( char *s1, const char *s2, size_t n )`	Copies at most **n** characters of string **s2** into array **s1**. The value of **s1** is returned.
`char *strcat( char *s1, const char *s2 )`	Appends string **s2** to array **s1**. The first character of **s2** overwrites the terminating null character of **s1**. The value of **s1** is returned.
`char *strncat( char *s1, const char *s2, size_t n )`	Appends at most **n** characters of string **s2** to array **s1**. The first character of **s2** overwrites the terminating null character of **s1**. The value of **s1** is returned.

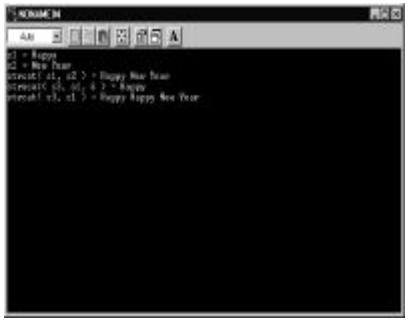
Example 7A

```
/* 7 A
 Using strcat and strncat */
#include<stdio.h>
#include <string.h>
#include<conio.h>

int main()
{
  char s1[ 20 ] = "Happy ";
  char s2[] = "New Year ";
  char s3[ 40 ] = "";

  printf( "s1 = %s\ns2 = %s\n", s1, s2 );
  printf( "strcat( s1, s2 ) = %s\n", strcat( s1, s2 ) );
  printf( "strncat( s3, s1, 6 ) = %s\n", strncat( s3, s1, 6 ) );
  printf( "strcat( s3, s1 ) = %s\n", strcat( s3, s1 ) );
  getch ();
  return 0;
}
```


Output Result



Comparison Functions of the String Handling Library

- Comparing strings
  - Computer compares numeric ASCII codes of characters in string
  - Appendix D has a list of character codes
- `int strcmp( const char *s1, const char *s2 );`
  - Compares string **s1** to **s2**
  - Returns a negative number if **s1 < s2**, zero if **s1 == s2** or a positive number if **s1 > s2**
- `int strncmp( const char *s1, const char *s2, size_t n );`
  - Compares up to **n** characters of string **s1** to **s2**
  - Returns values as above

Some Other Functions of the String Handling Library

Function prototype	Function description
`int strlen (char *s)`	Returns the length of the string s
`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 );`	Determin es 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.
`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"—logical pieces such as words in a line of text—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.

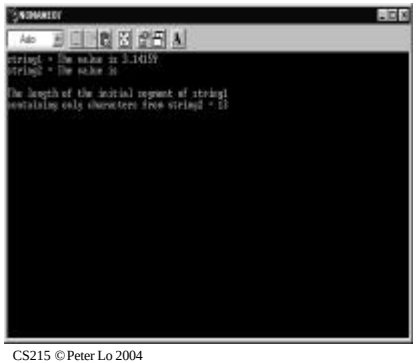
Example 7B

```
/* 7B
Using strspn */
#include <stdio.h>
#include <string.h>
#include<conio.h>

int main()
{
  const char *string1 = "The value is 3.14159";
  const char *string2 = "The value is ";

  printf( "%s%s\n%s%s\n\n%s\n%s%u\n",
          "string1 = ", string1, "string2 = ", string2,
          "The length of the initial segment of string1",
          "containing only characters from string2 = ",
          strspn( string1, string2 ) );
  getch ();
  return 0;
}
```


Output Result

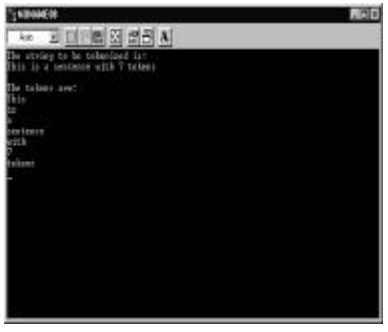


Example 7C

```
/* 7C
Using strtok */
#include <stdio.h>
#include <string.h>
#include <conio.h>
int main()
{
  char string[] = "This is a sentence with 7 tokens";
  char *tokenPtr;
  printf( "%s\n%s\n\n%s\n",
          "The string to be tokenized is:", string,
          "The tokens are:" );

  tokenPtr = strtok( string, " " );
  while ( tokenPtr != NULL ) {
    printf( "%s\n", tokenPtr );
    tokenPtr = strtok( NULL, " " );
  }
  getch();
  return 0;
}
```

Output Result



Implementing String using Function strlen

```
/* 7D
Determine the length of a string */
#include <stdio.h>
#include <conio.h>
int STRLEN  (char str[]);
char s[]="Hello";
main()
{
  clrscr();
  printf ("The length of %s is %d",s,STRLEN(s));
  getch();
  return 0;
}

int STRLEN  (char s[])
{  int count = 0;
    while (s[count] != \0)
      count++;
   return count;
}
```


Output Result