Strings

A character array is a string (and vice versa).

A C string is an array of characters that ends with the ASCII 0 null character (`'\0'`). The '\0' character is called the **null terminator**. It denotes the end of a character string.

The null terminator must be taken into account when declaring character arrays. You need to set aside an extra character (array element) for it.

The standard input object, cin, is a member of the class istream. One of the member functions of this class is getline (name, size, delimiter). The default delimiter is '\n' .

Example:

```
//getstring.cpp

#include <iostream>
int main (){
   const int MAX = 80;
   char name [MAX];

   //testing getline function
   cout << "Enter your full name:  ";
   cin.getline (name, MAX);
   cout << "Hello " << name << ", how are you?\n";

   //testing get function
   cout << "What is your favorite flavor of ice cream?  ";
   cin.get (name, MAX);
   cout << "One order of " << name << " coming up!\n";
   cout << "\n\nPress any key to close console window."; char c; cin >>
c;
   return 0;
}
```

Output:

```
Enter your full name:  President Barak Obama
Hello President Barak Obama, how are you?
What is your favorite flavor of ice cream?  low fat vanilla
One order of low fat vanilla coming up!


Press any key to close console window.
```

Many of the functions useful for use with C strings are declared in the string.h (or cstring.h) header file. For example, the function **strlen(str)** returns the number of characters in the string. This may be less than the size of the string. Its return type is unsigned integer.

Assignment:
We cannot use the = operator on arrays. To assign the value of one string to another, we use the **strcopy(str1, str2)** function. However, when a character array is declared in a string class, we can assign one object to another object using =.

[operator overload]

Concatenation:

strcat(str1, str2) concatenates the two strings and stores the result in str1. str1 must have enough room for the added characters. If we use the string class we could write
str3 = str1 + str2;

String Comparison:

When strings are declared as arrays of characters, we can't use relational operators on them. Instead we use the function **strcmp (str1, str2)** which compares the two strings and returns an integer.

It returns a negative number if str1 is less than str2.
It returns a zero if str1 is equal to str2.
It returns a positive number if str1 is more than str2.

Example

```
//comparestrings.cpp
//modified from SAMS ex 7.5, p. 229

#include <iostream>
#include <cstring>
#include <fstream>

const unsigned STR_SIZE = 40;
const unsigned ARRAY_SIZE = 11;
ofstream cprn ("f:\\cplus\\reports\\comparestrings.txt");

int main (){
   char strArray [STR_SIZE][ARRAY_SIZE] = {"California", "Virginia",
      "Alaska", "New York", "Michigan", "Nevada", "Ohio", "Florida",
      "Washington", "Oregon", "Arizona"};
   cprn << "Unordered array of strings:  " << endl;
   for (int i=0; i<ARRAY_SIZE; i++) cprn << strArray[i] << endl;
```

```
   cprn << endl;
            //Bubble sort
   for (int i=0; i<ARRAY_SIZE; ++i)
      for (int j=0; j<i; ++j)
         if (strcmp(strArray[i], strArray[j]) < 0) {
            char temp[STR_SIZE];
            strcpy(temp, strArray[i]);
            strcpy(strArray[i], strArray[j]);
            strcpy(strArray[j], temp);
         }
   cprn << "Sorted array of strings:  " << endl;
   for (int i=0; i<ARRAY_SIZE; i++) cprn << strArray[i] << endl;
   return 0;
}
```

Output:

```
Unordered array of strings:
California
Virginia
Alaska
New York
Michigan
Nevada
Ohio
Florida
Washington
Oregon
Arizona

Sorted array of strings:
Alaska
Arizona
California
Florida
Michigan
Nevada
New York
Ohio
Oregon
Virginia
Washington
```

Functions in the string class library are more natural in this regard.

Text Replacement:

Example

```
//textreplacement.cpp
//modified from SAMS ex 7.4 p. 225

#include <iostream>
#include <cstring>

const unsigned MAX1 = 40;
const unsigned MAX2 = 80;

int main(){
   char smallStr[MAX1+1];
   char bigStr[MAX2+1];
   char findChar, replChar;

   cout << "Enter first string:" << endl;
   cin.getline(bigStr, MAX2);
   cout << "Enter second string:" << endl;
   cin.getline(smallStr, MAX1);

   cout << "String 1 has " << strlen(bigStr) << " characters" << endl;
   cout << "String 2 has " << strlen(smallStr) << " characters"
      << endl;

   //concatenate the strings
   strcat(bigStr, smallStr);
   cout << "Concatenated string is:" <<endl << bigStr << endl;
   cout << "New string has " << strlen(bigStr) << " characters"
      << endl;

   //get the search and replacement charac ters
   cout << "Enter search character: ";
   cin >> findChar;
   cout << "Enter replacement character: ";
   cin >> replChar;

   //replace characters in string bigStr
   for(int i=0; i< strlen(bigStr); i++)
      if (bigStr[i] == findChar) bigStr[i] = replChar;

   //display the updated string bigStr
   cout << "New string is:" << endl << bigStr;
   cout << "\n\nPress any key to close console window.";
   char c; cin >> c;
   return 0;
}
```

Output:

```
Enter first string:
The rain in Spain stays
Enter second string:
 mainly in the plain.
String 1 has 23 characters
String 2 has 21 characters
Concatenated string is:
The rain in Spain stays mainly in the plain.
New string has 44 characters
Enter search character: a
Enter replacement character: A
New string is:
The rAin in SpAin stAys mAinly in the plAin.

Press any key to close console window.
```

strlwr(str) converts str to all lower case characters.
strupr(str) converts str to all upper case characters.
strrev(str) converts str to its reverse.

Example:

```
//textprocessg.cpp
//modified from SAMS ex 7.6 p. 233

#include <iostream>
#include <cstring>

const unsigned SSIZE = 40;

int main(){
   char str1[SSIZE+1];
   char str2[SSIZE+1];
   bool isLowerCase, isUpperCase, isSymmetrical;

   cout << "Enter a string: ";
   cin.getline(str1, SSIZE);
   cout << "Input: " << str1 << endl;

   strcpy (str2, str1);
   strlwr(str2);
   isLowerCase = (strcmp(str1, str2) == 0) ? true : false;
   cout << "Lowercase: " << str2 << endl;

   strupr(str2);
   isUpperCase = (strcmp(str1, str2) == 0) ? true : false;
   cout << "Uppercase: " << str2 << endl;

   strcpy(str2, str1);
   strrev(str2);
   isSymmetrical = (strcmp(str1, str2) == 0) ? true : false;
   cout << "Reversed: " << str2 << endl;

   if (isLowerCase)
      cout << "Your input has no uppercase letters." << endl;
   if (isUpperCase)
      cout << "Your input has no uppercase letters." << endl;
   if (isSymmetrical)
      cout << "Your input has symmetrical characters." << endl;

   cout << "\n\n\nPress any key to close console window.";
   char c; cin >> c;
   return 0;
}
```

```
Enter a string: level
Input: level
Lowercase: level
Uppercase: LEVEL
Reversed: level
Your input has no uppercase letters.
Your input has symmetrical characters.
```

Finding a Substring in a String:

A substring is also called a ***token***.

strstr (str, substr) returns the index of the first character of the substr if it is contained within str. If not found, the function returns a 0.

Use strrev(str) and strrev(substr) before strstr(str, substr) to find the last occurrence of the substr in str.

Using the string Class:

Lafore pp. 278-281