C-style string: Pointer-based String Processing
An-Najah university – updated to C++ 2011

Note: C++ string class is preferred for use in new programs

A string is a series of characters treated as a single unit. It may include letters, digits and special characters such as *,&,^, %.

string constants are written in double quotes as: "hello world "

A string is an array of characters ending with the null character ('\0') which marks the end of the string. The string is accessed by a pointer to its first character.

declaration:

A constant string maybe used to initialize an array of characters or a variable of type char *. The following declarations,

char animal[] = "dog";

or

char animal[] = {'d','o','g','\0'};

are valid and creates a 4-elements array of character {'d','o','g','\0'}

The declaration,

char *animalPtr = "dog";

is also valid in C and creates a constant pointer to the first character of "dog", d, also ends with '\0', but this is invalid in c++ (since c++ 2011), but

const char * animalPtr = “dog” ; // this is valid in C++

The compiler determines the size of the array based on the number of characters in the string.

Displaying a string:

The stream injection operator (<<) is overloaded to print the string given the pointer to the first character or the name of the array of characters not the memory address.

```
cout  <<  animal     <<  endl; // print dog
cout  <<  animalPtr  << endl; // prints dog as well
```

Because a string is just an array of characters, we can access a single character using array subscript or pointer offset notation:

```
cout << animal[1]    << endl; // prints 'o'
cout << animalPtr[1] << endl; // prints 'o'
```

Reading a string

A string can be read using the operator (>>). For example, the statements

```
char name[20];
cin >> name;
```

reads characters until a white space or end-of-file character is encountered into the character array of 20. If the user enters "hi there", only "hi" will be read into the name.

The size of the input string should be at most 19 characters to allow appending '\0' character. To ensure that the input string doesn't exceed the size of the array, the setw can be used. (you need to include iomanip)

```
cin  >>  setw(20)  >>  name;
```

will ensue that cin will read at most 19 characters.

If we use a pointer, we must allocate memory before we can read in a string

```
char * name;
cin  >>  name ;  // generates an error
```

but ,

```
char * name = new char[20];
cin  >>  setw(20)  >>  name; // will read at most 19 character
```

To input an entire line of text or a sentence with spaces, we can use the cin member function getline, which takes three arguments.

```
char sentence[100];
cin.getline(sentence, 100, '\n');
```

reads a line of text. The function stops reading characters when the delimiter character '\n' is encountered or 99 character is already reached.

Using const with pointers

The const qualifier tells the compiler that the value of a variable can't be modified after initialization. Typically, they are used in passing parameters to functions.

char ch;

A non-constant pointer to a non-constant data: allow modifying both the data and the pointer
char *p = &ch;
void func(char *p);

A non-constant pointer to a const data: allow modifying the pointer but not the data.
const char *p = &ch;
void func(const char *p);

A constant pointer to non-constant data: allow modifying the data, but not the pointer
char * const p = &ch;
void func(char * const p);

A constant pointer to a constant data: both the pointer and the data can't be modified
const char * const p = & ch;
void func(const char * const p);

Array of pointers (string array):

Arrays can contain pointers which can be used to form an array of pointer-based strings (string array). Each entry in the array of strings is a pointer to the first character of a string.

const char * animals[4] = {"cat", "horse", "dog", "camel"}; // valid when using const pointer

declares an array of 4 elements, where each element is a constant pointer to the first letter of a string.

cout << animal[1] << endl; // prints cat
cout << animals[2] << endl; // prints dog
cout << *(animals + 2) << endl; // prints dog
cout << animals[1][1] << endl; //prints o
animals[1][1] = 'a' ; //error

Pointer-based string manipulation functions

errno_t strcpy_s (char * s1, sizeof(s1), const char * s2);
copies s2 into s1 and returns 0 on success or error code on fail.

errno_t strncpy_s (char * s1, sizeof(s1), const char *s2, size_t n);
copies at most n characters of string s2 into s1 and returns 0 on success or error code on fail

errno_t strcat_s(char *s1, sizeof(s1), const char *s2);
appends s2 to s1 and returns 0 on success or error code on fail

errno_t strncat(char *s1, sizeof(s1), const char* s2, size_t n);
appends at most n characters of s2 to s1

int strcmp (const vhar * s1, const char * s2);
s1 == s2   returns 0
s1 < s2   returns <0
s1 > s2   returns > 0

int strncmp (const vhar * s1, const char * s2);
compares up to n characters

size_t strlen(const char *s)