FEU INSTITUTE OF TECHNOLOGY
COLLEGE OF ENGINEERING • COLLEGE OF COMPUTER STUDIES

String Manipulation

CSPROG2

Computer Programming 2 for CS

C-STRING

A string stored as an array of characters terminated with '\0'

An Array type for Strings

greet[6]="Hello" (where Hello is a literal string)

greet[0] = 'H'

greet[1] = 'e'

greet[2] = 'l'

greet[3] = 'l'

greet[4] = 'o'

greet[5] = '\0' – end marker, null character (sentinel value)

C-STRING

C-String variable is just an array of characters.

The length of the longest string that the array can hold is one less than the size of the array.

char shortString[]= “abc” is equivalent to char shortString[4] = “abc” (where shortString[3]=’\0’)

C-STRING

Syntax

char Array_Name[Maximum_C-String_Size + 1];

char shortString[] = “abc” is not equal to char shortString[] = {‘a’,’b’,’c’}

Note: When manipulating the indexed variable you should be very careful not to replace the null character ‘\0’ with some other value.

C-STRING

illegal

```
char aString[10];
aString = "Hello";          //You cannot do it
   anywhere else in the program
```

legal

```
strcpy(aString,"Hello");
```

C-STRING

illegal

```
char aString[10] = "Hello";
if(aString == "Hello")
   cout << "Hello";
```

legal

```
char aString[10] = "Hello";
if(strcmp(aString,"Hello")==0)
   cout << "Hello";
```

C-STRING

illegal

```
char aString[10] = "Hello";
if(aString == "Hello")
   cout << "Hello";
```

legal

```
char aString[10] = "Hello";
if(strcmp(aString,"Hello")==0)
   cout << "Hello";
```

TABLE: SOME PREDEFINED C-STRING FUNCTIONS IN <cstring>

FUNCTION	DESCRIPTION
strcpy(Target_String_Var,Src_String)	**Copies the C-string value Src_String into the C-string variable Target_String_Var**
strncpy(Target_String_Var,Src_String,limit)	The same as the two argument strcpy except that at most Limit characters are copied
strcat(Target_String_Var,Src_String)	**Concatenates the C-String value Src_String onto the end of C-string in the C-string variable Target_String_Var**
strncat(Target_String_Var,Src_String, Limit)	The same as the two argument strcat except that at most Limit characters are appended.
strlen(Src_String)	**Returns an integer equal to the length of Src_String. (The null character, '\0', is not counted in the length.**
strcmp(String_1, String_2)	**Returns 0 if String_1 and String_2 are the same. Returns a value < 0 if String_1 is less than String_2. Returns a value > 0 if String_1 is greater than String_2 (that is, returns a nonzero value if String_1 and String_2 are different). The order is lexicographic.**
strncmp(String_1, String_2, Limit)	The same as the two-argument strcmp except that at most Limit characters are compared.

TABLE: SOME PREDEFINED C-STRING FUNCTIONS IN <cstring>

```
#include <iostream>
#include<cstring>
using namespace std;
int main()
{   char x[]="good morning ";
    char y[25],  z[15];
    cout<<"The string character in array x is " << x<<endl;
    cout<<"The string character in array y is " << strcpy(y,x)<<endl;
    strncpy(z,y,4);
    z[5]='\0';
    cout<<"The string character in array z is " << z<<endl;
    system("pause>0");
    return 0;                    }
```

FUNCTION	DESCRIPTION
strcpy(Target_String_Var,Src_String)	**Copies the C-string value Src_String into the C-string variable Target_String_Var**
strncpy(Target_String_Var,Src_String,limit)	The same as the two argument strcpy except that at most Limit characters are copied

TABLE: SOME PREDEFINED C-STRING FUNCTIONS IN <cstring>

Source Code:

```
#include<iostream>
using namespace std;
int main()
{
    char str1[20] = "Happy";
    char str2[20];
    //char str3[
    cout << strlen(str1) << endl;
    strcpy(str2," Day Morning");
    strcat(str1,str2);
    cout << strlen(str1) << endl;
    cout << strcmp(str1,"Happy day Morning");
    cout << endl;
    if(!strcmp(str1,"Happy Day Morning"))
        cout << str1;
    system("pause>0");
    return 0;
}
```

OUTPUT:

```
5
17
-1
Happy Day Morning
```

TABLE: SOME PREDEFINED C-STRING FUNCTIONS IN <cstring>

FUNCTION	DESCRIPTION
strcmp(String_1, String_2)	Returns 0 if String_1 and String_2 are the same. Returns a value < 0 if String_1 is less than String_2. Returns a value > 0 if String_1 is greater than String_2 (that is, returns a nonzero value if String_1 and String_2 are different). The order is lexicographic.

```
char x[]="good morning ";
char y[25]="Good Morning";
char z[25]="good afternoon";

cout<<x <<" compare "<< y <<"\t"<<strcmp(x,y) <<endl;
cout<<x <<" compare "<< "good morning" <<"\t"<<strcmp(x,"good morning ") <<endl;
cout<<z <<" compare "<< x <<"\t"<<strcmp(z,x) <<endl;
```

OUTPUT:

```
good morning  compare Good Morning      1
good morning  compare good morning      0
good afternoon compare good morning     -1
```

getline

The member function getline can be used to read a line of input and place the string of characters on that line into a C-string variable.

Source Code:

syntax:

cin.getline(String_Var, Max_Characters + 1);

```
#include<iostream>
using namespace std;
int main()
{
    char str[80];
    cout << "Enter a phrase: ";
    cin >> str;
    cout << "The phrase: " << str;
    system("pause>0");
    return 0;
}
```

OUTPUT:

```
Enter a phrase: the quick brown fox
The phrase: the
```

getline

The member function getline can be used to read a line of input and place the string of characters on that line into a C-string variable.

syntax:

cin.getline(String_Var, Max_Characters + 1);

Source Code:

```
#include<iostream>
using namespace std;
int main()
{
    char str[80];
    cout << "Enter a phrase: ";
    cin.getline(str,80);
    cout << "The phrase: " << str;
    system("pause>0");
    return 0;
}
```

OUTPUT:

```
Enter a phrase: the quick brown fox
The phrase: the quick brown fox
```

CHARACTER MANIPULATION TOOLS

get function allows your program to read in one character of input and store it in a variable of type char

SOURCE CODE:

```
char a,b,c;
cout<<"Enter any 3 characters ";
cin.get(a);
cin.get(b);
cin.get(c);
cout<<c<<b<<a;
```

OUTPUT:

```
Enter any 3 characters advanced programming
```

CHARACTER MANIPULATION TOOLS

SOURCE CODE:

```
#include<iostream>
using namespace std;
int main()
{
    cout << "Enter a line of input and I will echo it:\n";
    char symbol;
    do
    {
        cin.get(symbol);
        cout << symbol;
    }while(symbol != '\n');
    system("pause > 0");
    return 0;
}
```

CHARACTER MANIPULATION TOOLS

put This function member is analogous to the member function get except that it is used for output rather than input. The function put allows your program to output one character

SOURCE CODE:

```
#include<iostream>
using namespace std;
int main()
{
    cout.put('A');
    cout.put('p');
    cout.put('p');
    cout.put('l');
    cout.put('e');
    system("pause > 0");
    return 0;
}
```

OUTPUT:

```
Apple
```

CHARACTER-MANIPULATING FUNCTIONS

Table: Some functions in <cctype>

FUNCTION	S
toupper(Char_Exp)	Returns the uppercase version of Char_Exp (as value of type int).
tolower(Char_Exp)	Returns the lowercase version of Char_Exp (as value of type int).
isupper(Char_Exp)	Returns true provided Char_Exp is an uppercase letter; otherwise, **returns false**.
islower(Char_Exp)	Returns true provided Char_Exp is an lowercase letter; otherwise, **returns false**.
isalpha(Char_Exp)	Returns true provided Char_Exp is a letter of the alphabet; otherwise **returns false**.
isdigit(Char_Exp)	Returns true provided Char_Exp is one of the digits '0' through '9'; otherwise, **returns false**.
isalnum(Char_Exp)	Returns true provided Char_Exp is either a letter or a digit; otherwise, **returns false**.
isspace(Char_Exp)	Returns true provided Char_Exp is a whitespace character, such as the blank or newline character, otherwise, **returns false**.
ispunct(Char_Exp)	Returns true provided Char_Exp is a printing character other than whitespace, a digit, or a letter; otherwise **returns false**.
isprint(Char_Exp)	Returns true provided Char_Exp is a printing characters includes blank space; otherwise **returns false**.
isgraph(Char_Exp)	Returns true provided Char_Exp is a printing characters; otherwise **returns false**.
isctrl(Char_Exp)	Returns true provided Char_Exp is a control character; otherwise, **returns false**.

CHARACTER-MANIPULATING FUNCTIONS

```
#include<iostream>
#include<cctype>
using namespace std;
int main()
{
    char phrase[]="12. The Quick \n Brown Fox #";
    char ltr;
    int n(0);
    cout << "Original Phrase" << endl;
    cout << phrase;
    cout << "\n\ntoupper" << endl;
    for(int i=0;i<strlen(phrase);i++)   {
        ltr = toupper(phrase[i]);
        cout << ltr;
    }
```

```
Original Phrase
12. The Quick
 Brown Fox #
```

toupper(Char_Exp)
Returns the uppercase version of Char_Exp (as value of type int).

```
toupper
12. THE QUICK
 BROWN FOX #
```

CHARACTER-MANIPULATING FUNCTIONS

```
#include<iostream>
#include<cctype>
using namespace std;
int main()
{
    char phrase[]="12. The Quick \n Brown Fox #";
    char ltr;
    int n(0);
   cout << "\n\ntolower" << endl;
   for(int i=0;i<strlen(phrase);i++)   {
       ltr = tolower(phrase[i]);
       cout << ltr;
   }
```

tolower(Char_Exp)

Returns the lowercase version of Char_Exp (as value of type int).

```
tolower
12. the quick
 brown fox #
```

CHARACTER-MANIPULATING FUNCTIONS

```
#include<iostream>
#include<cctype>
using namespace std;
int main()
{
    char phrase[]="12. The Quick \n Brown Fox #";
    char ltr;
    int n(0);
     cout << "\n\nisupper" << endl;
     for(int i=0;i<strlen(phrase);i++)   {
         if(isupper(phrase[i]))
             cout << phrase[i];
     }
```

isupper(Char_Exp)
Returns true provided Char_Exp is an uppercase letter; otherwise, **returns false**.

```
isupper
TQBF
```

CHARACTER-MANIPULATING FUNCTIONS

```
#include<iostream>
#include<cctype>
using namespace std;
int main()
{
    char phrase[]="12. The Quick \n Brown Fox #";
    char ltr;
    int n(0);
    cout << "\n\nislower" << endl;
    for(int i=0;i<strlen(phrase);i++)   {
        if(islower(phrase[i]))
            cout << phrase[i];
    }
```

islower(Char_Exp)
Returns true provided Char_Exp is an lowercase letter; otherwise, **returns false**.

```
islower
heuickrownox
```

CHARACTER-MANIPULATING FUNCTIONS

```
#include<iostream>
#include<cctype>
using namespace std;
int main()
{
    char phrase[]="12. The Quick \n Brown Fox #";
    char ltr;
    int n(0);
    cout << "\n\nisalpha" << endl;
    for(int i=0;i<strlen(phrase);i++)   {
        if(isalpha(phrase[i]))
            cout << phrase[i];
    }
```

isalpha(Char_Exp)
Returns true provided Char_Exp is a letter of the alphabet; otherwise **returns false**.

```
isalpha
TheQuickBrownFox
```

CHARACTER-MANIPULATING FUNCTIONS

```
#include<iostream>
#include<cctype>
using namespace std;
int main()
{
    char phrase[]="12. The Quick \n Brown Fox #";
    char ltr;
    int n(0);
    cout << "\n\nisdigit" << endl;
    for(int i=0;i<strlen(phrase);i++)   {
        if(isdigit(phrase[i]))
            cout << phrase[i];
    }
```

isdigit(Char_Exp)
Returns true provided Char_Exp is one of the digits '0' through '9'; otherwise, **returns false**.

```
isdigit
12
```

CHARACTER-MANIPULATING FUNCTIONS

```
#include<iostream>
#include<cctype>
using namespace std;
int main()
{
    char phrase[]="12. The Quick \n Brown Fox #";
    char ltr;
    int n(0);
    cout << "\n\nisalnum" << endl;
    for(int i=0;i<strlen(phrase);i++)   {
        if(isalnum(phrase[i]))
            cout << phrase[i];
    }
```

isalnum(Char_Exp)
Returns true provided Char_Exp is either a letter or a digit; otherwise, **returns false**.

```
isalnum
12TheQuickBrownFox
```

CHARACTER-MANIPULATING FUNCTIONS

```
#include<iostream>
#include<cctype>
using namespace std;
int main()
{
    char phrase[]="12. The Quick \n Brown Fox #";
    char ltr;
    int n(0);
    cout << "\n\nisspace" << endl;
    for(int i=0;i<strlen(phrase);i++)   {
        if(isspace(phrase[i]))
            n++;
    }
    cout << n;
```

isspace(Char_Exp)
Returns true provided Char_Exp is a whitespace character, such as the blank or newline character, otherwise, **returns false**.

```
isspace
7
```

CHARACTER-MANIPULATING FUNCTIONS

```
#include<iostream>
#include<cctype>
using namespace std;
int main()
{
    char phrase[]="12. The Quick \n Brown Fox #";
    char ltr;
    int n(0);
    cout << "\n\nisprint" << endl;
    for(int i=0;i<strlen(phrase);i++)   {
        if(isprint(phrase[i]))
            cout << phrase[i];
    }
```

isprint(Char_Exp)
Returns true provided Char_Exp is a printing characters includes blank space; otherwise **returns false**.

```
isprint
12. The Quick  Brown Fox #
```

CHARACTER-MANIPULATING FUNCTIONS

```
#include<iostream>
#include<cctype>
using namespace std;
int main()
{
    char phrase[]="12. The Quick \n Brown Fox #";
    char ltr;
    int n(0);
    cout << "\n\nisgraph" << endl;
    for(int i=0;i<strlen(phrase);i++)   {
        if(isgraph(phrase[i]))
            cout << phrase[i];
    }
```

isgraph(Char_Exp)
Returns true provided Char_Exp is a printing characters; otherwise **returns false**.

```
isgraph
12.TheQuickBrownFox#
```

CHARACTER-MANIPULATING FUNCTIONS

```
#include<iostream>
#include<cctype>
using namespace std;
int main()
{
    char phrase[]="12. The Quick \n Brown Fox #";
    char ltr;
    int n(0);
    cout << "\n\nispunct" << endl;
    for(int i=0;i<strlen(phrase);i++)   {
        if(ispunct(phrase[i]))
            cout << phrase[i];
    }
```

ispunct(Char_Exp)
Returns true provided Char_Exp is a printing character other than whitespace, a digit, or a letter; otherwise **returns false**.

```
ispunct
.#
```

CHARACTER-MANIPULATING FUNCTIONS

Table: Member Functions of the Standard Class String

Example	Remarks
Constructor	
string str;	Default constructor; creates empty string object str.
string str("string");	Creates string object with data "string".
string str(aString);	Creates a string object str that is a copy of aString. aString is an object of the class string.
Element Access	
str[i]	Returns read/write reference to character in str at index i.
str.at(i)	Returns read/write reference to character in str at index i.
str.substr(position, length)	Returns the substring of the calling object starting at position and having length characters.

CHARACTER-MANIPULATING FUNCTIONS

Table: Member Functions of the Standard Class String

Example	Remarks
Assignment/Modifiers	
str1 = str2	Allocates space and initializes it to Str2's data, releases memory allocated for str1, and set str1's size to that of str2.
str1 += str2	Character data of str2 is concatenated to the end of str1; the size is set appropriately.
str.empty( )	Returns true if str is an empty string; returns false otherwise.
str1 + str2	Returns a string that has str2's data concatenated to the end of str1's data. The size is set appropriately.
str.insert(pos, str2)	Insert str2 into str beginning at position pos.
str.remove(pos, length)	Removes substring of size length, starting at position pos.

CHARACTER-MANIPULATING FUNCTIONS

Table: Member Functions of the Standard Class String

Example	Remarks
Comparisons	
str1 == str2 str1 != str2	Compare for equality or inequality; returns a Boolean value.
str1 < str2str1 > str2 str1 <= str2 str1 >= str2	Four comparisons. All are lexicographical comparisons
str.find(str1)	Returns index of the first occurrence of str1 in str.
str.find(str1, pos)	Returns index of the first occurrence of string str1 in str; the search starts at position pos.
str.find_first_of(str1, pos)	Returns the index of the first instance in str of any character in str1, starting the search at position pos.
str.find_first_not_of(str1,pos)	Returns the index of the first instance in str of any character not in str1, starting search at position pos.

SOURCE CODE – String Class

```
#include<iostream>
#include<string>
using namespace std;
int main()
{
    string str1("the quick ");
    string str2("brown fox...");
    string str3(str2);
    cout << str3 <<endl;
    cout << str1 << endl;
    cout << str1.at(5) << endl;
    cout << str1.substr(4,5) << endl;
    str3 = str1;
    cout << str3 <<endl;
    str3 += str2;
    cout << str3 <<endl;
    cout << str3.empty() << endl;
    cout << str1 + str2 << endl;
    str3 = str1;
    cout << str3.insert(4,str2) << endl;
    str3 = str1 + str2.substr(0,9) + " jumps over the lazy dog";
    cout << str3.find("the") << endl;
    cout << str3 << endl;
    cout << str3.find_first_of("the",4) << endl;
    cout << str3.find_first_not_of(str1,0);
   system("pause>0");
   return 0;
}
```

SOURCE CODE

```
#include<iostream>
#include<string>
using namespace std;
int main()
{
    string str1("the quick ");
    string str2("brown fox...");
    string str3(str2);
    cout << str3 <<endl;
    cout << str1 << endl;
    cout << str1.at(5) << endl;
    cout << str1.substr(4,5) << endl;
    str3 = str1;
    cout << str3 <<endl;
    str3 += str2;
    cout << str3 <<endl;
    cout << str3.empty() << endl;
    cout << str1 + str2 << endl;
    str3 = str1;
    cout << str3.insert(4,str2) << endl;
    str3 = str1 + str2.substr(0,9) + " jumps over the lazy dog";
    cout << str3.find("the") << endl;
    cout << str3 << endl;
    cout << str3.find_first_of("the",4) << endl;
    cout << str3.find_first_not_of(str1,0);
  system("pause>0");
  return 0;
}
```

```
brown fox...
the quick
u
quick
the quick
the quick brown fox...
0
the quick brown fox...
the brown fox...quick
0
the quick brown fox jumps over the lazy dog
28
10
```

SOURCE CODE

```
#include <iostream>
#include <string>
using namespace std;
int main()
{
    string str1("The quick brown fox jumps over the");
    string str2(" lazy dog near the bank of the river ");
    string str3;
    cout << str1.substr(10,15);
    str3 = str1;
    str3 += str2;

    if(str3.empty())
        cout << "str3 is empty";
    else
        cout << endl << str3;
    str1.insert(9,str2);
    cout << endl << str1;
    cout << endl << str1.find(str2);

    cout << endl << str3.find("the",20);

    system("pause>0");
    return 0;
}
```

SOURCE CODE

```
#include <iostream>
#include <string>
using namespace std;
int main()
{
    string str1("The quick brown fox jumps over the");
    string str2(" lazy dog near the bank of the river ");
    string str3;
    cout << str1.substr(10,15);
    str3 = str1;
    str3 += str2;

    if(str3.empty())
        cout << "str3 is empty";
    else
        cout << endl << str3;
    str1.insert(9,str2);
    cout << endl << str1;
    cout << endl << str1.find(str2);

    cout << endl << str3.find("the",20);

    system("pause>0");
    return 0;
}
```

```
brown fox jumps
The quick brown fox jumps over the lazy dog near the bank of the river
The quick lazy dog near the bank of the river  brown fox jumps over the
9
31
```