Advanced Programming
(BETC 1353)

Topic 4 : Characters and Strings

ROSZIANA BINTI HASHIM
rosziana@utem.edu.my

Learning Outcome

At the end of this session, you should be able:

- To explain the strings and characters fundamental.
- To use the character handling functions
- To perform the string processing using
- To convert string format using general utilities library functions

Strings & Characters Fundamentals

- Characters
  - The program block is built from a sequence of meaningfully grouped characters
  - Characters in single quotes is represented as **`int`** value known as characters constant

    `example : 'm'` represents the int value of **`m`**

- Strings
  - Described as sequences of characters treated as a single unit
    - May consists of any characters listed in ASCII table.
    - String is a pointer pointed to the first (most-left) character.
  - Strings literal must be written using double quotes " "
    - **`Ex: "My name is Lola."`**

Type of Strings

1. C-Strings (Originated from C language)
2. String Object (New version of string in C++, from class **string**)

- Understanding each type of string is essential because each string type has different handling

Fundamentals of Strings and Characters

- C-string :

1. Sequence or array of characters
2. Stored in the memory adjacently
3. Ended by **NULL** '\0' character

- A C-string can be initialized as follow:

```
char cStr[] = "Hi Man!";
```

It is saved in the memory as:

H	i		M	a	n	!\0

Character Handling Library

- Contains useful functions to
  - test the character
  - manipulate the character

- Possible argument receive by each function
  - int
  - EOF

- All character handling functions are available in
  - **`ctype.h`** (C program)
  - **`cctype`** (C++ program)

Functions in cctype Library

Function	Description
`tolower (z)`	Converting lowercase z to uppercase z, only if z is a lowercase letter. Else, z remains unchanged.
`toupper (z)`	Converting uppercase z to lowercase z, only if z is a uppercase letter. Else, z remains unchanged.
`isspace (z)`	Checking whether z is a whitespace, newline, carriage return, form feed, horizontal or vertical tab, or not.
`iscntrl (z)`	Checking if z is a control character.
`ispunct (z)`	Checking if **z** is a printing character excluding alphabets, digits and whitespace.
`isupper (z)`	Checking if **z** is an uppercase alphabet.
`islower (z)`	Checking if **z** is a lowercase alphabet.
`isdigit (z)`	Checking if **z** is a digit.
`isalpha (z)`	Checking if **z** is an alphabet (lowercase or uppercase).
`isalnum (z)`	Checking if **z** is a digit or an alphabet.
`isxdigit (z)`	Checking if **z** is a digit in hexadecimal.
`isprint (z)`	Checking if **z** is a printable character, including the whitespace.
`isgraph (z)`	Checking if **z** is a printable character, excluding the whitespace.

Functions for String Conversion

- Conversion functions shall be available in the library of General utilities.
  - **`<stdlib.h>`** for C program
  - **`<cstdlib >`** for C++ program
- Useful in converting strings containing digits, to integer or floating-point values.

Prototype	Description
`atof ( myStr )`	Converting string **`myStr`** to **`double`**.
`atoi ( myStr )`	Converting string **`myStr`** to **`int`**.
`atol ( myStr )`	Converting string **`myStr`** to long **`int`**.

atoi, atol, atof

Function	Description	Example
atoi	Convert alphanumeric to int `int atoi (char *num_Str)` Return 0 if not a number	`int v1;` `numb = atoi("187");`
atol	Convert alphanumeric to long `long atol(char *num_Str)` Return 0 if not a number	`long v2;` `v2 = atol("1234567");`
atof	Convert alphanumeric to a double-precision floating point number `double atof(char *num_Str)` Returns 0.0 if not a digit	`double v3;` `v3 = atof("6.3791641");`

`atoi`, `atol`, `atof`

- Results are undefined if converted C- string is contained others than digits and may give return result as below ;
  - Return conversion up to first non-digit
  - Return 0

String Manipulation Functions (String Handling Library)

- Table below shows the string manipulation functions of the string handling library

Function prototype	Function description
`char *strcpy( char *str1, const char *str2 )`	String **`str2 is copied`** into array **`str1`**. Return value **`str1`**
`char *strncpy( char *str1, const char *str2,size_t n )`	**`n`** characters of string **`str2 are copied`** into array **`str1`**. Return value **`str1`**.
`char *strcat( char *str1, const char *str2 )`	String **`str2`** is appended to array **`str1`**. The terminating null character of **`str1`** is overwrite by the first character of **`str2.`** Return value **`str1`**
`char *strncat( char *str1, const char *str2, size_t n )`	**`n`** characters of string **`s2`** are appended to array **`s1`**. . The terminating null character of **`str1`** is overwrite by the first character of **`str2.`** Return value **`str1`**

Comparison Functions : String Handling Library

- Comparing strings

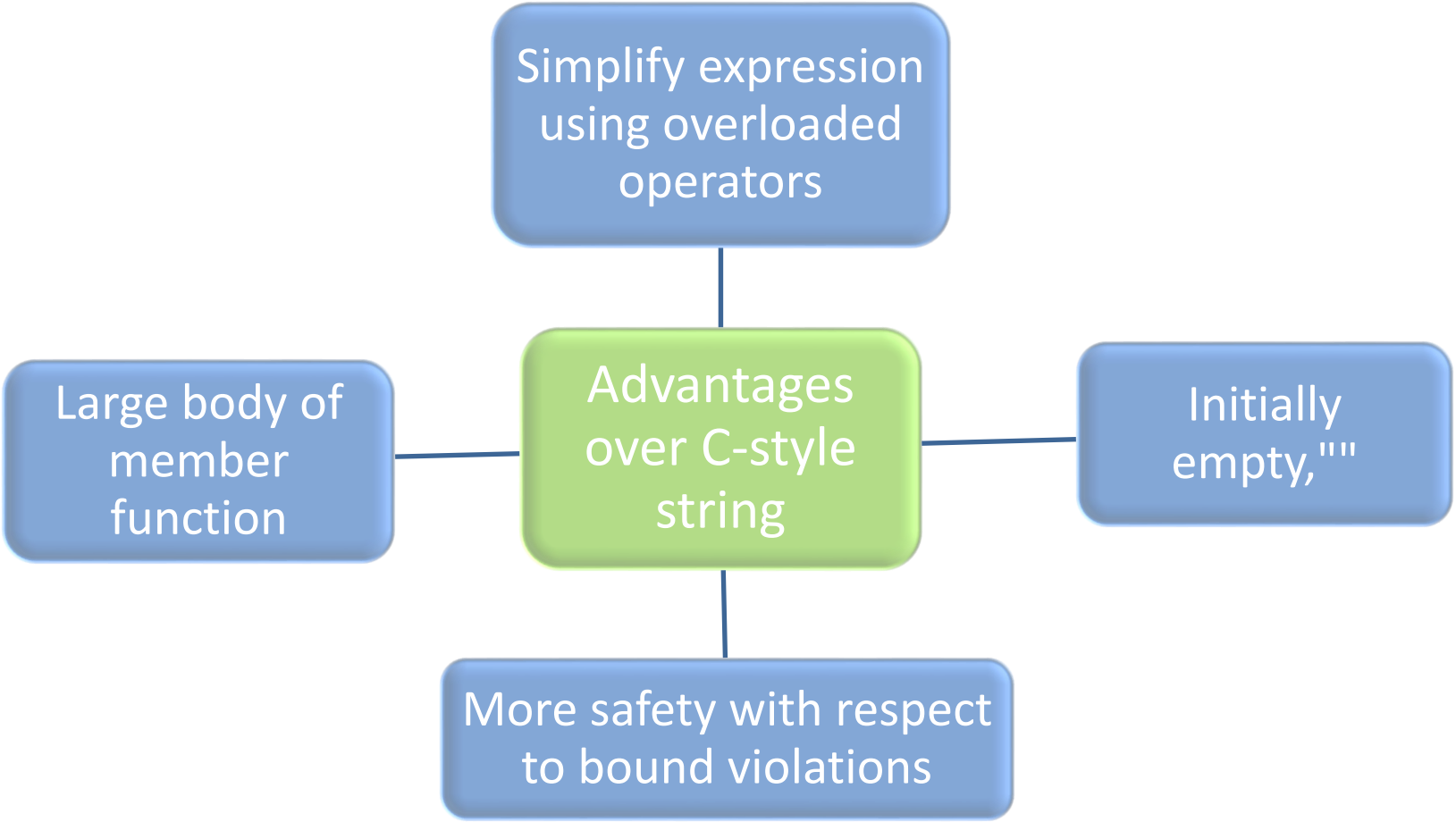
  Numeric ASCII codes' characters contain in string is compared by computer

Prototyped	Description
int strcmp(const char *str1, const char *str2);	Compares string str1 to str2 If str1 < str2 ; return negative (-) number If str1 == str2; return 0 If str1 > str2 ; return positive (+) number
int strncmp(const char *str1, const char *str2, size_t n);	Compares up to n characters of string str1 to str2 If str1 < str2 ; return negative (-) number If str1 == str2; return 0 If str1 > str2 ; return positive (+) number

Other Functions : String Handling Library

Prototyped	Description
`char *strerror( int error_num );`	Creates error message (system-dependent message) based on error_num Return a pointer to the string
size_t strlen(const char *str);	Return the number of characters in strings **str** (before terminating character **'\0'**)

`string` Class (C++)



Must include the <**string**> header file

`string` Class (C++)

```
// getline with strings
#include <iostream>
#include <string>
using namespace std;

int main ()
{
  string str_MyName;
  cout << "What is your name?: ";
  getline (cin, str_MyName);
  cout << "Thank you, " << str_MyName << endl;
}
```

`string` Class (C++)

OPERATOR	DESCRIPTION
>>	reads whitespace-delimited strings into string object
<<	outputs string object to a stream
=	assigns the right string to left string object
+=	appends the right string to end of contents of left string

`string` Class (C++)

OPERATOR	DESCRIPTION
`+`	appends two strings
`[]`	references character using array notation – but NO BOUNDS CHECKING!
`>, >=, <, <=, ==, !=`	relational operators. Return **`true`** or **`false`**

C++ `string` Class

```
string word_1, phrases; // initially ""
string word_2 = "shirt";
cin >> word_1; // user enters "your"
               // word1 has "your"
phrases = word_1 + word_2; // phrase has
                           // "your shirt"
phrases += " in a closet";
for (int i = 0; i < phrases.length(); i++)
   cout << phrases[i];  // displays
                        // "your shirt in a
  closet"
```

string Member Functions

Categories	String member
C-strings conversion	data, c_str
Modification	append, clear, replace, assign, erase, insert, swap, copy
Space management	resize, empty, capacity, length, size
Substrings	substr,find
Comparison	compare

C-strings Conversion

- Two functions below convert **string** object to the equivalent C-string
  - `data()`
  - `c_str()`
- Useful when using a function which expect C-string as a parameter

```
char greet[20] = "May you have a";
string str("nice day");
strcat(greet, str.data());
```

`string` objects Modification

- `str.append(string m)`

  concatenates `m` to end of `str`

```
str += m;  // same as...
```

- A C-string can be passed in place of a string object

```
string str("you look ");
str.append("nice today");
```

- `append` is an overloaded function for more flexibility

Modification of `string` objects

- `str.insert(int p, string k)`

  inserts string `k` after the `p`th character in string `str`
- `insert` is an overloaded function, for more flexibility
  - A C-string can be passed in place of a **string** object

```
string myStr = "Hello ";
myStr.insert(6, "Everyone!");
  // myStr is now "Hello Everyone"
```

Self-Review Questions

1) Write a C++ program which:

- create a string s1 and initialized it to : **MK0373abc32#**
- using the string s1, calculate and display the following:

Number of digits:
Number of lowercase letters:
Number of uppercase letters:
Number of others:

Self-Review Questions

Answers:

```
char s1[] = "MK0373abc32#";
int dig = 0;
int low = 0;
int upp = 0;
int oth = 0;

for (int i = 0; i < strlen(s1); i++){
    if (isdigit(s1[i])){
        dig++;
    } else if (islower(s1[i])){
        low++;
    } else if (isupper(s1[i])){
        upp++;
    } else {
        oth++;
    }
}
```

```
cout << "Number of digits: " << dig;
cout << "Number of lowercase letters: " << low;
cout << "Number of uppercase letters: " << upp;
cout << "Number of others: " << oth;
```

Self-Review Questions

2) Determine the result of the following statements:

1. strcmp (“Fratelia”, “fratelia”);
2. strcmp (“Programing”, “Programming”);
3. strncmp (“Programing”, “Programming”, 7);

Self-Review Questions

Answers:

a) **negative** as first string is smaller than second string

b) **negative** as first string is smaller than second string

c) **zero** as the first 7 characters of both strings are the same

Self-Review Questions

3) Determine the final value of string1 and string2, after the following statements:

```
strcpy (string1, "My name is Mike.");
strcpy (string2, " I am a great football ");
strcpy (string3, " coach.");

strcat (string2, string3);
strcat (string1, string2);
```

Self-Review Questions

Answers:

string1 = "My name is Mike. I am a great football player.”

string2 = “I am a great football player.”