

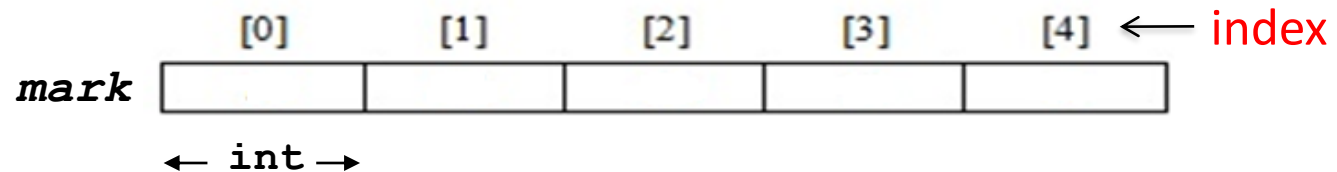
Chapter 1 - Arrays & Strings

- Learning outcomes:
 - *Describe the idea of an array*
 - *Apply declaration & initialization of arrays*
 - *Perform accessing elements of an array*
 - *Discuss & create multi-dimensional arrays*
 - *Describe C++ string representation*
 - *Apply declaration & initialization of string*
 - *Practice reading strings from the keyboard*
 - *Evaluate C++ standard library for string*

Introduction

- **Array**

- a **group of elements** of the **same type** placed in **contiguous memory** locations,
 - under **one common** name or (same name).
- e.g., array ***mark*** to contain 5 integer values:



- First: **index=0**, Last: **index=N-1**, where N is the array size.
- an array **must be declared** before it is used.

Declaring Array

- Declaring array variable,
 - type of array (e.g, int, char, float...etc.)
 - variable name (identifier)
 - number of elements (size)
- Syntax: `type arrayName[arraySize];`
 - e.g., `int mark[5];` *// declare mark, an array of 5 integers*
- Declaring multiple arrays of same type,
 - e.g., `int a[100], b[27], c[10];` *// use comma separated list*

Cont'd...

- Regular arrays are blocks of non-dynamic memory:
 - static entity (or have the same size throughout the program)
 - its size must be determined when declared (before execution)
- Regular arrays (“arrays of known constant size”):
 - size can be specified by a constant integer variable
 - e.g., `const int size=20; //size cannot be changed`
`int x[size];`
 - constant variables must be initialized when declared

Initializing Arrays

- Specify **each element values** when array declared,
 - e.g., `int mark [5]={10, 25, 30, 14, 50};` *// initialize mark*

	[0]	[1]	[2]	[3]	[4]
<i>mark</i>	10	25	30	14	50

- When **not enough initializers**, sets **all rightmost** elements
 - e.g., `int n[5]= {0};` *// sets every element to 0*
- When **array size is omitted**, array **size = number of initializers**
 - `int n[]={1, 2, 3, 4, 5};` *// n has an array size of 5*

Accessing Array Elements

- To refer to array element,
 - specify array name and index
- **Syntax:** `arrayName[index]`
- Array `c` with N-element: `c[0]`, `c[1]`... `c[n-1]`
 - first element: `index=0`; the Nth element: `index=N-1`
- E.g., array `int mark[5]`,
 - `mark[0]; mark[1]; ...mark[4];`

Cont'd...

- Trying to **access mark[5]**, you may encounter
 - “**undefined behavior**” (program can crash, freeze, random output ...etc.)
- Like other variables, **array elements** can be:
 - **assigned** or **printout** values
 - e.g., `mark[1]=75; cout<<mark[1];`
 - or perform **operations** (inside the subscript)
 - e.g., `int a=1; mark[a]=75;`

`int b; b=mark[a+2];` *// same as `b=mark[3]`*

`mark [mark[a]]= mark[2] + 5;`

Cont'd...

- To **read or write** to/from an array,
 - often **use `for` loop** control structure
- E.g., for **reading values** from array `mark[5]`

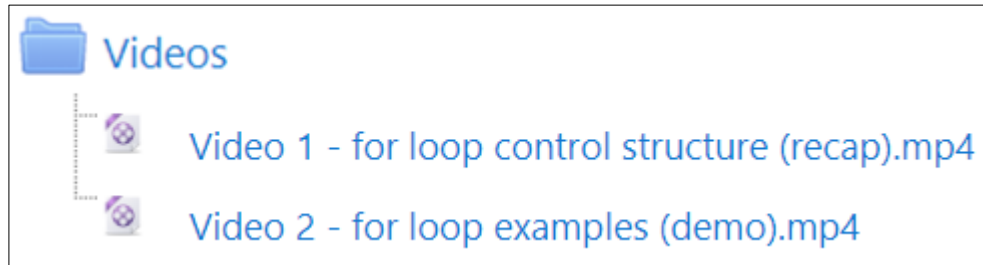
```
for(int i=0; i<5; i++)  
    cout<<mark[i] ; // to write cin>>mark[i] ;
```

- Often **format I/O stream**,
 - using stream **manipulators** in `<iomanip>`
 - e.g., **`setw()`** *// for setting field widths*
- Note: **No Array-to-Array Assignments, No Bounds Checking**

Activity 0 - Watch videos!

Instruction:

- Please **login to your Moodle¹ course page** to complete this activity.



¹ <https://elearning.amu.edu.et/>

Example 1

```
1 //Example 1 - Declaring and initializing an integer array
2 #include <iostream>
3 #include <iomanip> //to use setw()
4
5 using namespace std;
6
7 int main() {
8
9     // use initializer list to initialize integer array n
10    int n[10]={ 32, 27, 64, 18, 95, 14, 90, 70, 60, 37};
11
12    // set field 13 positions wide
13    cout<<"Element"<<setw(13)<<"Value"<<endl;
14
15    // output contents of array n in tabular format
16    // set field 13 positions wide
17    for(int i=0; i<10; i++)
18        cout<<setw(7)<<i<<setw(13)<<n[i]<<endl;
19
20    return 0;
21 }
```

Example 2

```
1 // Example 2 - Accept 5 integers from a user, store them in an integer array,  
2 // and display them back  
3 #include <iostream>  
4  
5 using namespace std;  
6  
7 int main() {  
8  
9     int mark [5]; // declaring integer array  
10  
11     cout<<"Enter 5 integers into mark[5]:"<<endl;  
12  
13     for(int i=0; i<5; i++) // storing values into array elements  
14         cin>>mark[i];  
15  
16     cout<<"\nmark[5] contains: ";  
17  
18     for(int j=0; j<5; j++) // displaying values from array elements  
19         cout<<mark[j]<<" ";  
20  
21     cout<<endl;  
22  
23     return 0;  
24 }
```

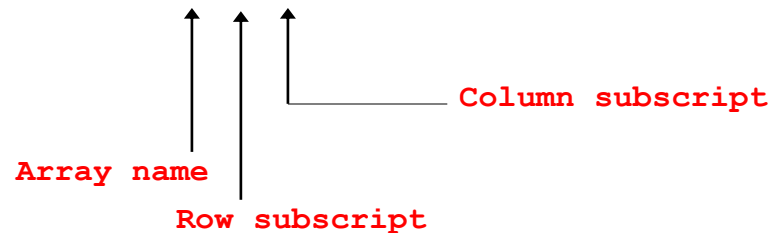
Multi-dimensional Arrays

- One dimensional (1D) arrays
 - e.g., `int x[10], char y[5]...etc.`
- Multidimensional arrays (“arrays of arrays”)
 - e.g., `int a[3][4], ... a[i][j]...[n]`
- Most common: two dimensional (2D) arrays
 - e.g., `a[i][j]` *// have two subscripts*
 - can be considered as a **table**: rows & columns
 - **i** specify **row**, **j** specify **column**

Cont'd...

- To declare
 - e.g., `int x[3][4];`

	Column 0	Column 1	Column 2	Column 3
Row 0	<code>a[0][0]</code>	<code>a[0][1]</code>	<code>a[0][2]</code>	<code>a[0][3]</code>
Row 1	<code>a[1][0]</code>	<code>a[1][1]</code>	<code>a[1][2]</code>	<code>a[1][3]</code>
Row 2	<code>a[2][0]</code>	<code>a[2][1]</code>	<code>a[2][2]</code>	<code>a[2][3]</code>



- referred as a "3 by 4" integer array
- the size of array `x` will be $3 \times 4 = 12$ elements

Cont'd...

- To **initialize**
 - initializers are **grouped by row** in braces

- e.g., `int a[2][2]={{1,2},{3,4}};`
Row 0 Row 1

a

1	2
3	4

- `int b[][2]={{1},{3,4}};`

b

1	0
3	4

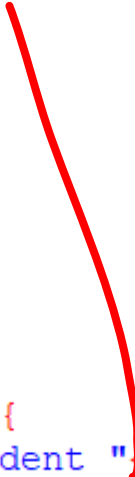
- **Referenced** like array variable
 - `cout<<b[0][1];` *// the default is 0; outputs 0*
 - `cout<<b[][1];` *// error message*

Example 3

```
1 // initializing multidimensional array
2 #include <iostream>
3
4 using namespace std;
5
6 int main() {
7
8     int a[2][3]={{1, 2, 3}, {4, 5, 6}};
9
10    cout<<"Values in array by row are:\n"<<endl;
11
12    for(int i=0; i< 2; i++){ // output row values
13        for(int j=0; j<3; j++) // output column values
14            cout<<a[i][j]<<' ';
15
16        cout<<endl;
17    }
18
19    return 0;
20 }
```

Example 4

```
1 // A program to record marks of 3 students for 4 tests
2 #include <iostream>
3 #include <iomanip>
4
5 using namespace std;
6
7 int main(){
8
9     const int no_st=3;
10    const int no_ts=4;
11
12    float grade[no_st][no_ts];
13
14    int i, j;
15
16    for(i=0; i<no_st; i++){
17        for(j=0; j<no_ts; j++){
18            cout<<"\nEnter student "<<(i+1)<<"\tmark "<<(j+1)<<": ";
19            cin>>grade[i][j];
20        }
21    }
```




Cont'd...

```
23     cout<<"\n\n";
24
25     cout<<"           Mark    1           2           3           4 \n";
26
27     for(i=0; i<no_st; i++){
28         cout<<"\nStudent " <<(i+1);
29
30         for(j=0; j<no_ts; j++){
31             cout<<setiosflags(ios::fixed)
32                 <<setiosflags(ios::showpoint)
33                 <<setprecision(1)
34                 <<setw(9)
35                 <<grade[i][j];
36         }
37     }
38     cout<<endl;
39
40     return 0;
41 }
```

Activity 1 – Live poll!

Instruction:

- Please **login to your Moodle¹ course page** to complete this activity.

 Activity 1

This activity is a **live poll**.

It is your **in-class exercise**, and has **no mark**.

You are also **required** to complete this activity, **to obtain full grades** in **marked activities**.


Please **click-once the Number** (1, 2, 3, 4, or 5) that **best answers** the question.

¹ <https://elearning.amu.edu.et/>

Discussion forum 1 – Hands-on exercise!

Instruction:

- Please **login to your Moodle¹ course page** to complete this activity.

 Discussion forum 1

Post a piece of **C++ code** that you wrote.

Respond to one **other student's code**.

Carefully examine the code, and **give your feedback** briefly.

Your feedback can be about the **design**, **functionality**, or **organization** of the code.

You can also **ask for an explanation** if you have any doubts.

¹ <https://elearning.amu.edu.et/>

Assignment 1 – Individual assignment!

Instruction:

- Please [login to your Moodle¹ course page](#) to complete this activity.

Assignment 1

This is your **individual assignment**, and it has **15 marks**.

Use CodeBlocks IDE to write the **complete C++ program**.

You should **submit your C++ program file** via Moodle's **CodeGrade tool**.

- Your C++ program **file should be named** as "matrix_addition.cpp".
- You have **3 chances** to make **trial submissions**, before your final submission.

You **receive a notification** when your **grade and feedback** are **ready**.

For support, directly **ask your questions** on the source code, which appears **as inline comments** for **course Tutor**.

Please look at the **assignment resources attached** for detailed information.

¹ <https://elearning.amu.edu.et/>

C++ Strings

- Recap: character constant, placed in **single quotes** (' ')
 - e.g., `char x = 'z' ;` *// 'z' is the integer value of z, 122 in ASCII*
 - letters (a-z), digits (0-9), symbols (*), non-printable (enter or '\n')
- String
 - a **series of characters**, treated as **single unit**
 - e.g., name of a person
- String constant, placed in **double quotes** (" ")
 - e.g., "abebe"

Cont'd...

- String (c-style)
 - “...can be stored in an **array of characters**” (Bjarne S.)
 - i.e., **NO** need for a **‘string’** keyword
- To **declare**
 - e.g., `char name[6];` *// no declare array of 6 characters*
- To **initialize**
 - e.g., `char name[6] = "abebe";` *// initialize a string name*
- Others: **character pointer** or C++ **‘string’ keyword** (next chapters)


String Initialization

- Initialization
 - e.g., `char color[5]="blue";`
 - create `char` array `color` with 5 elements
 - add `NULL ('\0 '` character at the end (`color[4]`)
- Alternatively,
 - e.g., `char color[]={ 'b', 'l', 'u', 'e', '\0' };`
- String
 - an array of characters, `NULL ('\0 '` terminated.

Accessing Strings

- Individual element,
 - e.g., `color[0]='c' ; // assign c to color[0]`
`cout<<color[2] ; // output u`
- Entire string
 - e.g., `cout<<color ; // output clue`
 - print the characters until **NULL** is found
 - **caution**: do **not work** for other array types (e.g., int)

Cont'd...

- Input from keyboard,
 - e.g., `char str[5];` *// declare char array str*
`cin>>str;` *// accept 5 characters*
 - `cin>>` stops at the first **whitespace** (or blank space)
 - add **NULL** (`'\0'`) character implicitly
 - Input string exceed array size,
 - modern C++: `cin>>` accept string above 5 char
 - i.e., by getting **some memory overwritten**
 - solution: `cin>>setw(5)>>str;` *// setw() discard extra chars*
- 

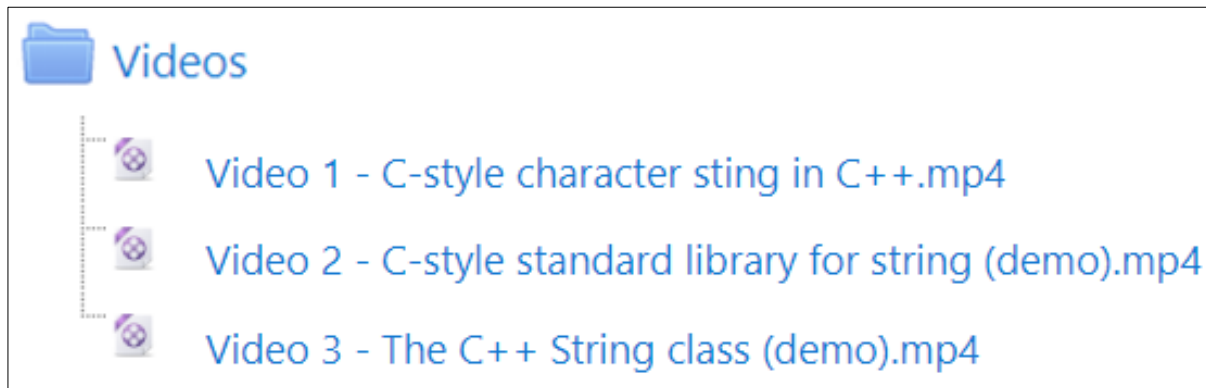
Cont'd...

- Input string with **blank space**,
 - use **`cin.get(array, size/cin.getline(array, size)`**
- **`cin.getline()`** *// used later in file I/O operation (CH-6)*
 - e.g., **`cin.getline(array, size, delimiter);`**
 - **Stops** either **delimiter** character is entered
 - or **size-1** character limit is reached
- E.g., **to input a poem**
 - **`char poem[80];`**
`cin.getline(poem, 80, '*');`

Activity 0 - Watch videos!

Instruction:

- Please [login to your Moodle¹ course page](#) to complete this activity.



¹ <https://elearning.amu.edu.et/>

Example 1

```
1 // Treating character array as string
2 #include <iostream>
3
4 using namespace std;
5
6 int main(){
7
8     char str1[]="String Literal"; // 14 characters
9     char str2[20];
10
11     // output entire string
12     cout<<"String 1: "<<str1<<"\n\n";
13
14     // output individual characters
15     for(int i=0; str1[i] !='\0'; i++)
16         cout<<str1[i]<<' ';
17
18     // read string from user into array str2
19     cout<<"\n\nEnter the string \"Hello There\": ";
20     cin>>str2; // space terminates input
21
22     cout<<"\nString 2: "<<str2<<endl;
23
24     return 0;
25 }
```

Activity 2 – Live poll!

Instruction:

- Please [login to your Moodle¹ course page](#) to complete this activity.



Activity 2

This activity is a **live poll**.

It is your **in-class exercise**, and has **no mark**.

You are also **required** to complete this activity, **to obtain full grades** in **marked activities**.

Please **click-once the Option** (A, B, C, or D) that **best answers** the question.

¹ <https://elearning.amu.edu.et/>

Standard Library Functions for String

- Modern C++ also support C standard library functions,
 - defined in `<cstring>`, for string manipulation
- Most commons, e.g.,
 - `strcpy()` // *copy* string
 - `strcat()` // *concatenate* strings
 - `strcmp()` // *compare* strings
 - `Strtok()` // *tokenize* string
 - `strlen()` // *get string length*

Cont'd...

<code>char *strcpy(char *s1, const char *s2);</code>	Copies the string s2 into the character array s1 . The value of s1 is returned.
<code>char *strncpy(char *s1, const char *s2, size_t n);</code>	Copies at most n characters of the string s2 into the character array s1 . The value of s1 is returned.
<code>char *strcat(char *s1, const char *s2);</code>	Appends the string s2 to the string s1 . The first character of s2 overwrites the terminating null character of s1 . The value of s1 is returned.
<code>char *strncat(char *s1, const char *s2, size_t n);</code>	Appends at most n characters of string s2 to string s1 . The first character of s2 overwrites the terminating null character of s1 . The value of s1 is returned.
<code>int strcmp(const char *s1, const char *s2);</code>	Compares the string s1 with the string s2 . The function returns a value of zero, less than zero or greater than zero if s1 is equal to, less than or greater than s2 , respectively.

Cont'd...

<pre>int strncmp(const char *s1, const char *s2, size_t n);</pre>	<p>Compares up to n characters of the string s1 with the string s2. The function returns zero, less than zero or greater than zero if s1 is equal to, less than or greater than s2, respectively.</p>
<pre>char *strtok(char *s1, const char *s2);</pre>	<p>A sequence of calls to strtok breaks string s1 into “tokens”—logical pieces such as words in a line of text—delimited 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.</p>
<pre>size_t strlen(const char *s);</pre>	<p>Determines the length of string s. The number of characters preceding the terminating null character is returned.</p>

String Copy

- `char *strcpy(char *s1, const char *s2)`
 - copy string **s2** to **s1**, plus the **NULL** character
 - **s1** must be large enough to hold **s2**, plus the **NULL** character
 - return the value of **s1**
- `char *strncpy(char *s1, const char *s2, size_t n)`
 - copy the first **n** number of characters of **s2** to **s1**
 - do not copy **NULL** character necessarily
 - append **NULL** character at the end of **s1**
 - return the value of **s1**

Example 2

```
1 // Example 2 - Using strcpy() and strncpy()
2 #include <iostream>
3 #include <cstring> // to use strcpy() & strncpy()
4
5 using namespace std;
6
7 int main() {
8
9     char x[]="Happy Birth Day to You";
10    char y[25], z[15];
11
12    // copy content of x into y
13    strcpy(y, x);
14
15    cout<<"The string in array x is: "<<x<<endl;
16
17    cout<<"\nThe string in array y is: "<<y<<endl;
18
19    // copy the first 14 characters of x into z
20    strncpy(z, x, 14); // does not copy null character
21
22    z[14]='\0'; // append '\0' at the end of z
23
24    cout<<"\nThe string in array z is: "<<z<<endl;
25
26    return 0;
27 }
```

String Concatenation

- `char *strcat(char *s1, const char *s2)`
 - append the copy of **s2** to **s1**, plus the **NULL** character
 - the first character of **s2** replaces the **NULL** character of **s1**
 - **s1** must be large enough to hold **s2**, plus the **NULL** character
 - return the value of **s1**
- `char *strncat(char *s1, const char *s2, size_t n)`
 - append the first **n** number of characters of **s2** to **s1**
 - plus append **NULL** character at the end **s1**
 - return the value of **s1**

Example 3

```
1 // Using strcat() and strncat()
2 #include <iostream>
3 #include <cstring> // to use strcat() & strncat()
4
5 using namespace std;
6
7 int main() {
8
9     char s1 [20]="Happy ";
10    char s2 []= "New Year ";
11    char s3 [40]=" ";
12
13    cout<<"s1 = "<<s1<<"\ns2 = "<<s2;
14
15    // concatenate s2 to s1
16    strcat(s1, s2);
17
18    cout<<"\n\nAfter strcat(s1, s2):\ns1 = "<<s1<<"\ns2 = "<<s2;
19
20    // concatenate the first 6 characters of s1 into s3
21    strncat(s3, s1, 6); // places '\0' after last character
22
23    cout<<"\n\nAfter strncat (s3, s1, 6):\ns1 = "<<s1<<"\ns3 = "<<s3;
24
25    return 0;
26 }
```

String Comparison

- Characters are represented as numeric codes
 - strings are compared using numeric codes of characters
- Character codes (character sets)
 - ASCII (American Standard Code for Information Interchange)¹
 - e.g., ASCII code for 'z' is 122, and 90 for 'Z' (in decimal)
 - EBCDIC (Binary Coded Decimal Interchange Code)²
 - Both represent a character by 8-bits (i.e., 1 byte of memory)

¹ <https://theasciicode.com.ar/>

² <https://en.wikipedia.org/wiki/EBCDIC>

Cont'd...

- `int strcmp(const char *s1, const char *s2)`
 - compare string **s1** against **s2**, character by character
 - performs a binary comparison of the characters
 - return:
 - zero (0), if **s1** = **s2**
 - negative value (-ve), if **s1** < **s2**
 - positive value (+ve), if **s1** > **s2**
- `int strncmp(const char *s1, const char *s2, size_t n)`
 - compare up to **n** number of characters of **s1** against **s2**

Example 4

```
1 // Example 4 - Using strcmp and strncmp
2 #include <iostream>
3 #include <cstring> // to use strcmp() & strncmp()
4 #include <iomanip> // to use setw()
5
6 using namespace std;
7
8 int main() {
9
10     char s1[]="Happy New Year";
11     char s2[]="Happy New Year";
12     char s3[]="Happy Holidays";
13
14     cout<<"s1 = "<<s1<<endl;
15     cout<<"s2 = "<<s2<<endl;
16     cout<<"s3 = "<<s3<<endl;
17
18     cout<<"\nstrcmp(s1, s2) = "<<setw(2)<<strcmp(s1, s2)<<endl;
19     cout<<"strcmp(s1, s3) = "<<setw(2)<<strcmp(s1, s3)<<endl;
20     cout<<"strcmp(s3, s1) = "<<setw(2)<<strcmp(s3, s1)<<endl;
21     cout<<"\nstrncmp(s1, s3, 6) = "<<setw(2)<<strncmp(s1, s3, 6)<<endl;
22     cout<<"strncmp(s1, s3, 7) = "<<setw(2)<<strncmp(s1, s3, 7)<<endl;
23     cout<<"strncmp(s3, s1, 7) = "<<setw(2)<<strncmp(s3, s1, 7)<<endl;
24
25     return 0;
26 }
```

String Tokenizing

- Tokenizing
 - break a **string into tokens**, separated by **delimiting characters**
 - **tokens** usually are **logical units**, e.g., **words** separated by “ ”
 - **"This is my string"** = has **4 word-tokens**
- `char *strtok(char *s1, const char *s2)`
 - requires **multiple function calls**:
 - **1st call**, `strtok(s1, delimiter)`
 - replace the **first delimiter** character of string **s1** by **NULL**
 - **subsequent** calls, `strtok(NULL, delimiter)`
 - continue **tokenizing** the rest of string **s1**

Example 5

```
1 // Example 5 - Using strtok()
2 #include <iostream>
3 #include <cstring> // to use strtok()
4 #include <iomanip> // to use setw()
5
6 using namespace std;
7
8 int main() {
9
10     char sentence[]="This is a sentence with 7 tokens";
11     char *tokenPtr; // character pointer as string variable
12
13     cout<<"The string to be tokenized: \""<<sentence<<"\"<<endl;
14
15     cout<<"\nThe delimiting character: \" \"<<endl;
16
17     cout<<"\nThe tokens are: "<<endl;
18
19     // begin tokenization of sentence
20     tokenPtr=strtok(sentence, " ");
21
22     // continue tokenizing sentence until tokenPtr becomes NULL
23     while(tokenPtr != NULL) {
24
25         cout<<setw(3)<<"- "<<tokenPtr<<endl;
26
27         tokenPtr=strtok(NULL, " "); // get the next token
28     }
29
30     cout<<"\nAfter strtok(), sentence = "<<sentence<<endl;
31
32     return 0;
33 }
```

String Length

- `size_t strlen(const char *s)`
 - return the number of characters in a string (i.e., an integer value)
 - length not include the **NULL** character
- Examples:
 - `char name[]="abebe";`
 - `name` has a length of 5 (chars)
 - `char str[100]="test string";`
 - `str` has a length of 11 (chars) *// do not confuse with size of str*


Example 6

```
1 // Example 6 - Using strlen()
2 #include <iostream>
3 #include <cstring> // to use strlen()
4
5 using namespace std;
6
7 int main() {
8
9     char string1[]="abcdefghijklmnopgrstuvwxyz";
10    char string2[]="four";
11
12    cout<<"The length of \""<<string1<<"\": "<<strlen(string1)<<endl;
13    cout <<"\nThe length of \""<<string2<<"\": "<<strlen(string2)<<endl;
14
15    return 0;
16 }
```

Discussion forum 2 – Homework!

Instruction:

- Please [login to your Moodle¹ course page](#) to complete this activity.

 Discussion forum 2

This is your **homework**, and it has **10 marks**.

Use CodeBlocks IDE to write the complete C++ program.

Post the complete C++ code of your solution that **best satisfies** the question.

You **receive a notification** when your **grade and feedback** are available.

Refer to the **grading rubric** for the **evaluation guidelines** used for this homework.

Please **ask the course Tutor** if you need **any support**.

1 <https://elearning.amu.edu.et/>

Quiz 1 – Self-evaluation!

Instruction:

- Please **login to your Moodle¹ course page** to complete this activity.



Quiz 1

This is your **quiz** exercise, and it has **no mark**.

The **aim** is to help you **self-check**, that you **have attained** the **learning outcomes** of Chapter 1.

Be sure you **have studied** the **materials**, and **completed activities** of **Chapter 1**.

Quiz 1 has **10 questions**, and you have **10 minutes** to complete.

- You have **only one chance** to attempt.
- So **do not open** until you are ready.

It's also **open-notes** and book **exams**.

- **Do not** use your **CodeBlocks IDE**, and test code snippets.
- You **must complete** the quiz **yourself**.

Good luck!!

¹ <https://elearning.amu.edu.et/>