**From One Translation Unit to Another**   
  
Workshop 1

In this workshop, you implement aspects of linkage, storage duration, namespaces, guards, and operating system interfaces.

**LEARNING OUTCOME**

Upon successful completion of this workshop, you will have demonstrated the ability

* to link to a variable in another translation unit
* to declare a local variable that lasts the lifetime of the program
* to guard a class definition from repetition
* to define a class within its own namespace
* to pass arguments to program from the command line

**SPECIFICATIONS**

This workshop consists of three modules:

* **w1**
* **process**
* **CString**

**CString Module**

Write the header and implementation files for a class named **CString**.  Wrap your header file in a conditional macro that guards against repeated use.  Embed your class definition and its implementation in a namespace named **w1**.  Include in your class definition:

* a constant definition of the number of characters to be stored by an object of your class
* a constructor that receives the address of a C-style null-terminated string and stores the first MAX characters of the string.  Check for receipt of the null address.  Store an empty string in that case.
* a member query named **display()**that receives a reference to an **ostream** object and displays the string as stored in your **CString** object
* a helper non-friend operator that inserts the stored string into the left **ostream** operand.  This operator prefaces the string with the number of the insertion and increment that number

Do not use the **string** class of the standard library in this workshop.  Use the **cstring** functions.

Include in your implementation file:

* a definition of a global variable initialized to the number of characters stored by an object of your class.

**process Module**

Write the header and implementation files for a function named **process**.  Your function receives the address of a C-style null-terminated string and uses an object of your **CString** class to store a possibly truncated version of the string.  Leave your prototype and function definition in the global namespace.  In your definition:

* construct a **CString** object from the string received
* insert the **CString** object into the **std::cout** object and terminate with a newline

**main Module**

Write the implementation file for a **main()** function that processes command-line input.  The command-line input includes a user-defined number of strings.  Your function echoes the command-line input and uses your **process**module to process each string separately.  Before processing any string, your function outputs the maximum number of characters stored in the processing as shown below.  If the user does not enter any arguments, your function displays an error message and returns an error code of 1 to the operating system.  Your function returns an error code of 0 on successful processing.

**Sample Output**

The output from your program looks like:

|  |
| --- |
| **Command Line : w1 oop345 btp305**  **Maximum number of characters stored : 3**  **0: oop**  **1: btp**  **Command Line : w1**  **Insufficient number of arguments (min 1)** |

**Test Runs**

Run your program at the command line on the following two platforms

* Visual Studio cl compiler
* GNU - g++ compiler

**SUBMISSION**

**Typescript**

On matrix, create a typescript of your complete solution using the following commands:

**+ At the prompt, type: script w1.txt**

**+ At the prompt, type: whoami**

**+ At the prompt, type: cat w1\_cpp.cpp**

**+ At the prompt, type: g++ -o w1 w1.cpp CString.cpp process.cpp**

**+ At the prompt, type: w1 oop345 btp305**

**+ At the prompt, type: w1**

**+ At the prompt type: exit**

These commands will produce a file named **w1.txt**.

Download your typescript file to your local computer.

**Moodle**

* Login to [OOP244](https://scs.senecac.on.ca/~oop244/index.html)
* Select OOP345 if necessary
* Select W1 under Workshops
* Upload your typescript file to Moodle
* Press "Edit"
* Summarize to your instructor the concepts that you have learned in doing this particular workshop.  Add any other comments you wish to make.
* Press "Save Changes"
* When ready to submit, press "Send for Marking"

**MySeneca**

* Login to [IPC144](https://scs.senecac.on.ca/~ipc144/index.html)
* Select OOP345 if necessary
* Select Assignments or Workshops
* Select W1
* Press "Browse My Computer" to upload your typescript
* Press "Edit"
* Summarize to your instructor the concepts that you have learned in doing this particular workshop.  Add any other comments you wish to make in the comment box provided.
* Press "Submit" IMPORTANT: If you "Save As Draft" your instructor does not receive your submission unitl you press "Submit"