**Strings**

1. **C Strings – array of type *char***

A string stored as an array of characters ending with ‘\0’ is called a C string. In C++, a literal string, such as “Hello”, is stored as C string.

*char* s[10];

NOTE that **C-string variable is different from array of characters. A C-string variable must contain the null character '\0' at the end of the C-string value.**

We can initialize the C-string variables. From the above mentioned note:

|  |  |
| --- | --- |
| *char* shortString[] = "abc";  is equivalent to  *char* shortString[4] = "abc"; | *char* shortString[] = "abc"; //places ‘\0’  is different from  *char* shortString[] = {'a', 'b', 'c'}; //3 elements |

If the size is not given in declaration, the C-string variable will be given a size one character longer than the length of the C string (for ‘\0’)

A picture containing text

Description automatically generated

You cannot easily assign the C-string variable, which can only be done in declarations. To assign a value to a C-string variable, you must do something else. The easiest way is to use the predefefined function *strcpy* (will not check the size exceeding) or *strncpy* (with the number of characters).

We can compare strings using *ctrcmp* like C.

🡪 These functions are included in #include <cstring>

**C string arguments and parameters**

C-string variable is an array 🡪 the parameter to a function is simply an array parameter.

**C string I/O**

* C strings can be outputted directly using outStream <<
* C strings can also be inputted directly using inStream >> string\_name. However, **all whitespaces are skipped**.

🡪 We can instead use the member function ***getline*** of the input stream. The function ***getline*** has two arguments. The first argument is a C-string variable to receive the input and the second is the maximum number of array elements in the C-string variable that ***getline*** will be allowed to fill with characters (variable\_name, 5 allows 4 character input, because the function saves 1 space for ‘\0’)

char a[80];

cout << "Enter some input:\n";

cin.getline(a, 80);

cout << a << "END OF OUTPUT\n";

**C-String-to-Number Conversions**

The functions atoi, atol, and atof can be used to convert a C string of digits to the corresponding numeric value. Returns 0 if not correspond #include <cstdlib>

* atoi: converts to integer value of type *int*
* atoll: converts to integer value of type *long*
* atof: converts to value of type *double*

1. **The Standard string class**

To use the string class, we must include

#include <string>

*using namespace* std;

A special note about the string class is that information related to string length is taken care of

🡪 The size of the array automatically changes to allocate memory spaces.

As we noted earlier, quoted strings are really C strings and so they are not literally of type *string*. However, C++ provides automatic type casting of quoted strings to values of type string. So, you can use quoted strings as if they were literal values of type *string*

To declare and initialize a string object, we can do:

Graphical user interface

Description automatically generated with medium confidence

*The reason why we can do string\_name(<C string>) is because there is a second constructor of the string class, which initializes the string object to a value that represents the same string as its C-string argument.*

**I/O with class *string***

* C strings can be outputted directly using outStream <<
* **String inputs with inStream >> still skip whitespaces**. Therefore, we also need to use the *getline* function, but with a different version corresponding to string objects. The first argument has to be the input stream, and the second argument be the string object.

string line;

cout << "Enter a line of input:\n";

getline(cin, line);

cout << line << "END OF OUTPUT\n";

This version of getline is in the library <string>

*istream& getline(istream& ins, string& strVar);*

**\*\*More on getline function**

* We can add another argument to specify when the getline function should stop reading.

getline(cin, line, '?');

*istream& getline(istream& ins, string& strVar, char delimiter);*

* getline actually returns a reference to its first argument, which is the input stream.

Thus, the following will read a line of text into s1 and a string of nonwhitespace characters into s2:

string s1, s2;

getline(cin, s1) >> s2; // The next thing that happen is similar to cin >> s2

*NOTE*: Be careful when mixing the use of >> and getline. If the user input something like this:

42

Hello hitchhiker.

Using cin >> n skips leading whitespace on the input, but leaves the rest of the line, in this case just '\n', for the next input.

A statement like cin >> n; always leaves something on the line for a following getline to read (even if it is just the '\n'). If the next getline sees the '\n' and stops reading, getline read an empty string.

🡪 You can use the .ignore method to solve this problem.

**String Processing with Class string**

* You can access characters in string by indexing, like string arrays. However, there is still no warning if we index outside string range. We can intentionally invoke the error, if any, by using the member function at. It returns the index character if normal, but will detect error if index out of range.

string str("Mary");

cout << str.at(6) << endl; 🡪 Throws runtime error

* With members of class string, you can intuitively use **string comparision operators == >**, **string concatenation +**
* String methods

Text

Description automatically generated

Logo

Description automatically generated

A screenshot of a computer

Description automatically generated with low confidence

**Converting between string Objects and C Strings**

Note that strcpy can only take 2 arguments that are C strings. Therefore, for string objects of class string, we can convert to C Strings using the .c\_str() method

strcpy(aCString, stringVariable.c\_str( ));

*(remember that we can do this, but we cannot assign stringVariable.c\_str() to a C String Variable)*

**Converting Between Strings and Numbers**

* Use stof, stod, stoi, or stol to convert a string to a float, double, int, or long, respectively.
* Use to\_string to convert a numeric type to a string.