Chapter 8 Characters and Strings

1. Character Data Type and Operations

`char` is used to represent a single character. A character literal is enclosed in single quotation marks `' '`.

1.1 ASCII Code

Most computers use ASCII (American Standard Code for Information Interchange), an 8-bit encoding scheme, for representing all uppercase and lowercase letters, digits, punctuation marks, and control characters. Table 4.4 shows the ASCII code for some commonly used characters. (ASCII Table)

TABLE 4.4 ASCII Code for Commonly Used Characters

Characters	*Code Value in Decimal*
'0' to '9'	48 to 57
'A' to 'Z'	65 to 90
'a' to 'z'	97 to 122

Example 01:

```
#include <iostream>
using namespace std;

int main(){
   char chr1 = 'A';
   char chr2 = 70;

   cout << chr1 << endl;
   cout << chr2 << endl;

   return 0;
}
```

Output

```
A
F
```

Example 02:

```
#include <iostream>
using namespace std;

int main(){
   char chr = 'A';
   int num = chr + 3;

   cout << num;

   return 0;
}
```

Output

```
68
```

1.2 Comparing Characters

Two characters can be compared using the relational operators just like comparing two numbers. This is done by comparing the Unicodes of the two characters. For example,

'a' < 'b' is true because the Unicode for 'a' (97) is less than the Unicode for 'b' (98).

'a' < 'A' is false because the Unicode for 'a' (97) is greater than the Unicode for 'A' (65).

'1' < '8' is true because the Unicode for '1' (49) is less than the Unicode for '8' (56).

Example 03:

```
#include <iostream>
using namespace std;

int main(){

   cout << ('a' < 'b') << endl;
   cout << ('a' < 'A') << endl;
   cout << ('1' < '8') << endl;

   return 0;
}
```

Output

```
1
0
1
```

Often in the program, you need to test whether a character is a number, a letter, an uppercase letter, or a lowercase letter.

Example 04: Check whether a character is an uppercase letter, a lowercase letter, or a digital character.

```
#include <iostream>
using namespace std;

int main(){
    char chr = 'D';

    if (chr >= 'A' && chr <= 'Z')
        cout << chr << " is an uppercase letter";

    else if (chr >= 'a' && chr <= 'z')
        cout << chr << " is a lowercase letter";

    else if (chr >= '0' && chr <= '9')
        cout << chr << " is a numeric character";

    return 0;
}
```

Output:

```
D is an uppercase letter
```

1.3 Character Functions

Here are some useful character functions defined in `cctype` header file you might need:

Function	Description
islower(chr)	returns true if the character is in lowercase. Otherwise, returns false.
isupper(chr)	returns true if the character is in uppercase. Otherwise, returns false.
isdigit(chr)	returns true if the character is a digit (0 – 9). Otherwise, returns false.
isalpha(chr)	returns true if the character is an alphabetic character (a-z, A-Z). Otherwise, returns false.
isalnum(chr)	returns true if the character is a digit (0-9), or an alphabetic character (a-z, A-Z). Otherwise, returns false.
isspace(chr)	returns true if the character is a space. Otherwise, returns false.
tolower(chr)	returns the value of the character in lowercase.
toupper(chr)	returns the value of the character in uppercase.

Example 06: Using Character functions

```
#include<iostream>
#include<cctype>
using namespace std;

int main(){
    char ch = 'A';

    if(islower(ch))
        cout << ch << " is a lowercase character." << endl;

    if(isupper(ch))
        cout << ch << " is an uppercase character." << endl;

    if(isdigit(ch))
        cout << ch << " is a digit." << endl;

    if(isalpha(ch))
        cout << ch << " is an alphabet." << endl;

    if(isalnum(ch))
        cout << ch << " is an alphanumeric character." << endl;

    if(isspace(ch))
        cout << ch << " is a whitespace character.";

    return 0;
}
```

Output

```
A is an uppercase character
A is an alphabet
A is an alphanumeric character
```

Example 07: Using `tolower`() and `toupper`() function to change cases of characters.

```
#include<iostream>
#include<cctype>
using namespace std;

int main(){
    char ch1 = 'H', ch2 = 'a';

    ch1 = tolower(ch1);
    ch2 = toupper(ch2);

    cout << ch1 << endl;
    cout << ch2;

    return 0;
}
```

Output

```
h
A
```

2. `string` class

C++ provides `string` class which is more flexible than C-style strings as it increases and decreases in size automatically.

2.1 String Output

Example 11: String output.

```
#include <iostream>
using namespace std;

int main(){
    string name = "C++";

    cout << "Hello " << name << endl;

    return 0;
}
```

Output:

```
Hello C++
```

You cannot use %s and printf() function to print out a string value because it does not support string. It only supports the C-style string. To display strings, you need to use cout. For example:

```
#include <iostream>
#include <cstdio>
using namespace std;

int main(){
    string str = "Hello";

    printf("%s", str);

    return 0;
}
```

Output

```
n■a
```

2.2 String Input

getline() is a function that is used to read a string or a line. It is a part of the <string> header file.

Example 12: String input.

```
#include <iostream>
#include <string>
using namespace std;

int main(){
    string str;

    cout << "Enter a string: ";
    getline(cin, str);

    cout << str <<endl;

    return 0;
}
```

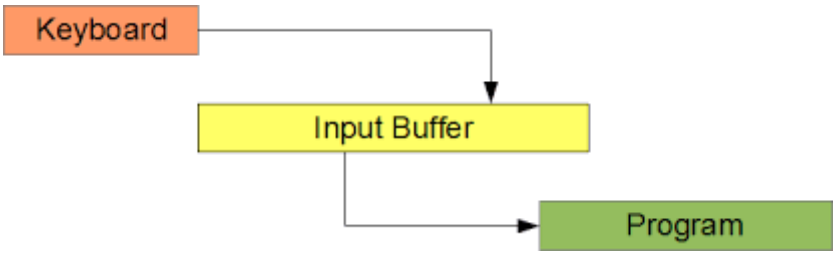
Output:

```
Enter a string: Hello, how are you?
Hello, how are you?
```

Note that the keyboard input is terminated by pressing the return key to create a new-line character ('\n'), but this '\n' will not be stored in a string.

2.3 Clearing the Buffer

Buffer is a temporary storage area. When we input any data to the program via our keyboard, it is not sent to your program directly, instead, it is sent to the operating system's buffer, till the time is allotted to your program.



On various occasions, you may need to clear the buffer of the previous input in order to get the next input, or else the desired input is occupied by a buffer of the previous input.

Let's look at an example below:

```
#include <iostream>
using namespace std;

int main(){
    int num;
    string str;

    cout << "Enter a number: ";
    cin >> num;

    cout << "Enter a string: ";
    getline(cin, str);

    cout << num << " and " << str <<endl;
    return 0;
}
```

Output

```
Enter a number: 10
Enter a string: 10 and
```

In the above example, after the first `cin` reads an input, the newline character (`'\n'`) entered when the user presses "Enter" remains in the buffer. The `getline()` then reads the leftover newline from the previous input as the character, making the user not being able to enter a string.

You generally need to clear the buffer when there are leftover characters (often a newline character `'\n'`) that can interfere with the next input.

To know when clearing the buffer is necessary in C++, you need to know a different between `cin` and `getine()` function.

`cin`:

- Reads a number or a sequence of characters without whitespace characters (space, newline, or tab) from the input.
- It does not consume the newline character (`'\n'`) left in the buffer from the previous input.
- It stops reading when it encounters a whitespace character, so, there will be a newline character left in the buffer.

`getline()`:

- Reads the entire line of input, including whitespace characters, until the newline character is encountered.
- It consumes the newline character left in the buffer from the previous input.
- It reads the entire line, so, there will be no newline character left in the buffer.

In C++, you can use `fflush(stdin)` or `cin.ignore()` to clear the buffer.

Example 13: Clear the buffer using `fflush(stdin)`

```
#include <iostream>
#include <string>

using namespace std;
int main(){
    int num;
    string str;

    cout << "Enter a number: ";
    cin >> num;

    fflush(stdin); // clear the buffer

    cout << "Enter a string: ";
    getline(cin, str);

    cout << num << " and " << str <<endl;
    return 0;
}
```

Output

```
Enter a number: 10
Enter a string: Hello
10 and Hello
```

Example 14: Clear the buffer using `cin.ignore()`

```
#include <iostream>
#include <string>

using namespace std;
int main(){
    int num;
    string str;

    cout << "Enter a number: ";
    cin >> num;

    cin.ignore(); // clear the buffer

    cout << "Enter a string: ";
    getline(cin, str);

    cout << num << " and " << str <<endl;
    return 0;
}
```

Output

```
Enter a number: 10
Enter a string: Hello
10 and Hello
```

2.4 Copying Strings

You can simply copy string objects in C++ using assignment operator (=).

Example 15: Copying strings.

```
#include <iostream>
using namespace std;
int main() {
    string str1, str2 = "C++ programming";

    str1 = str2; // copy str2 to str1
    cout << str1;

    return 0;
}
```

Output

```
C++ programming
```

2.5 String Concatenation

In C++, you can use a + operator to concatenate two strings.

Example 16: String Concatenation.

```
#include <iostream>
using namespace std;
int main(){
    string str1 = "Hello", str2 = "World";
    string str3 = str1 + " " + str2;

    cout << str3;

    return 0;
}
```

Output

```
Hello World
```

2.6 Appending a Character to a String

You can append a character to the beginning or the end of the string using + operator.

Example 17: Concatenate string with a character.

```
#include <iostream>
using namespace std;
int main(){
    string str = "Hello";
    char chr = 'A';

    str = chr + str + chr;

    cout << str;

    return 0;
}
```

Output

```
HelloA
```

Note: Appending a character to a string using the + operator in C++ is supported from C++11 onwards. In C++11, the + operator was overloaded to support concatenation of `std::string` with char.

2.7 Comparing Strings

In C++, you can use comparison operators (==, !=, >, <, >=, <=) to compare two strings.

Example 18: Compare two strings.

```
#include <iostream>
using namespace std;

int main(){
    string str1 = "Hello";
    string str2 = "Hi";

    if(str1 == str2)
        cout << "str1 is equal to str2";

    else if (str1 > str2)
        cout << "str1 is greater than str2";

    else
        cout << "str2 is greater than str1";

    return 0;
}
```

Output

```
str2 is greater than str1
```

2.8 String Length

We can get the length of a string in C++ by using string methods: `size()` and `length()`. The `size()` and `length()` are just synonyms and they both do the same thing.

Example 19: Get the length of a string.

```
#include <iostream>
using namespace std;

int main(){
    string str = "Learn C++";

    cout << "The length of our string is " << str.size() << endl;
    cout << "The length of our string is " << str.length();

    return 0;
}
```

Output

```
The length of our string is 9
The length of our string is 9
```

2.9 Iterating Through a String

You can iterate through a string using loops.

Example 20: Iterate through a string

```
#include <iostream>
using namespace std;

int main(){
    string str = "C++ Programming";

    for(int i = 0; i < str.length(); i++){
        cout << str[i] << " ";
    }

    return 0;
}
```

Output

```
C ++ P r o g r a m m i n g
```

2.10 Converting Between Strings and Numbers

stoi() and **stof**() functions defined in standard library <string> are used to convert strings into numbers.

Example 21: Convert strings into numbers.

```
#include <iostream>
#include <string>
using namespace std;

int main(){
    string str1 = "10";
    string str2 = "2.5";

    int number1 = stoi(str1);
    float number2 = stof(str2);

    cout << number1 + number2;

    return 0;
}
```

Output

```
12.5
```

`to_string()` function defined in standard library `string` are used to convert numerical values into string values.

Example 22: Convert numbers into string values.

```
#include <iostream>
#include <string>
using namespace std;
int main(){
    int number1 = 10;
    float number2 = 2.5;

    string str1 = to_string(number1);
    string str2 = to_string(number2);

    cout << str1 + " and " + str2;

    return 0;
}
```

Output

```
10 and 2.500000
```

2.11 Array of Strings

You can create an array of strings using `string` class.

Example 23: Using array of strings.

```
#include <iostream>
using namespace std;
int main(){
    string color[] = {"Blue", "Red", "Orange", "Yellow" };

    // Display the array elements
    for (int i = 0; i < 4; i++)
        cout << color[i] << "\t";
}
```

Output

```
Blue    Red    Orange    Yellow
```

Exercises

Write the following programs:

1. Check whether a character input by a user is an alphabet or not. If it is an alphabet, check if it is a vowel or a consonant.
2. Find common elements between two arrays of strings.
3. Convert each alphabet of a sentence into its next alphabet and print the sentence.
4. Remove all consonants from a sentence.
5. Ask the user to enter a word and check if it is a palindrome (if it can be read backwards the same way).
6. Ask the user to enter a sentence, then capitalizes the first character of each word in the sentence. Note that the other letters beside the first letter of each word must be in the lowercase.
7. Ask the user to enter a sentence, then display the number of consonants and vowels (a, e, i, o and u) in the sentence.
8. (Check Valid Password) Some websites impose certain rules for passwords. Suppose the rules are:
   - A password must have exactly 8 characters.
   - A password must consist of only digits and letters.
   - A password must always start with a digit.
   - A password must contain at least one uppercase letter.

   Write a program that asks the user to enter a password and displays if the password entered is valid or invalid. If it is invalid, let the user know why.

Reference

[1] Y. Daniel Liang. 'Introduction to Programming with C++', 3e – 2014

[2] Dey, P., & Ghosh, M. 'Computer Fundamentals and Programming in C', 2e – 2013

[3] https://cplusplus.com/doc/tutorial/

[4] https://www.programiz.com/cpp-programming

[5] https://www.studytonight.com/cpp/