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: Create char type variables

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

int main(){
   char ch1 = 'A';
   char ch2 = 70;

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

   return 0;
}
```

Output

```
A
F
```

Example 02: Add a number to char type

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

int main(){
   char ch = 'A';
   int number = ch + 3;

   cout << number;

   return 0;
}
```

Output

```
68
```

1.2 Comparing Characters

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

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

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

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

Example 03:

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

int main(){

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

   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 ch = 'D';

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

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

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

    return 0;
}
```

Output:

```
D is an uppercase letter
```

Practice

Create an array that stores ten alphabets. Convert each alphabet into its next alphabet, and display the array. **Note**: you have to convert `'z'` into `'a'`, and `'Z'` into `'A'`.

1.3 Character Functions

Here are some useful character functions defined in `cctype` standard library:

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**() functions 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
```

Practice

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. `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 Input

To read a string from the user's input, you can use `cin` or `getline()` function.

Example 08: Read a string from the input using `cin`

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

int main(){
    string str;

    cout << "Enter a string: ";
    cin >> str;

    cout << str;

    return 0;
}
```

Output:

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

Example 09: Read a string from the input using `getline()` function

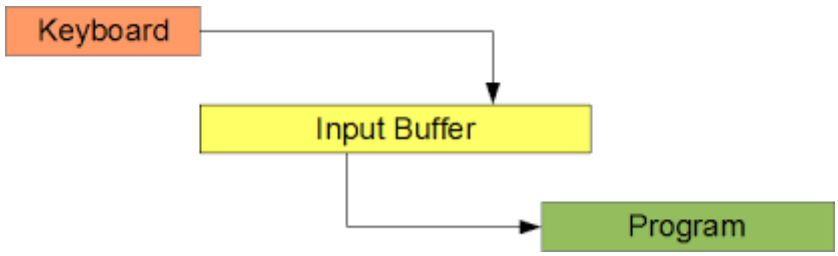
```
#include <iostream>
using namespace std;

int main(){
    string str;

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

    cout << str;

    return 0;
}
```

Output:

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

2.2 Clearing the Buffer

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



On some 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 number;
    string str;

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

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

    cout << number << " and " << str;

    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()` function then reads the leftover newline character from the previous input, making the user not being able to enter a string.

To fix the above issue, we need to clear the buffer before reading the second input. In C++, you can use `fflush(stdin)` or `cin.ignore()` to clear the buffer.

Example 10: Clear the buffer using `fflush(stdin)`

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

int main(){
    int number;
    string str;

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

    fflush(stdin); // clear the buffer

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

    cout << number << " and " << str;

    return 0;
}
```

Output

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

Example 11: Clear the buffer using `cin.ignore()`

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

int main(){
    int number;
    string str;

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

    cin.ignore(); // clear the buffer

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

    cout << number << " and " << str;
    return 0;
}
```

Output

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

When Do We Need to Clear the Buffer?

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

`cin`:

- 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 (space, newline, or tab).
- It leaves a newline character in the buffer after reading the input.

`getline`():

- It consumes the newline character (`'\n'`) left in the buffer from the previous input.
- It reads the entire line of input, including whitespace characters.
- It does NOT leave a newline character in the buffer after reading the input.

Practice

How would you clear the buffer if your program needs to read inputs as below?

```
Reads an integer
Reads a string using cin
Reads a float
Reads an integer
Reads a string using getline() function
Reads a string using getline() function
Reads a char
Reads a double
Reads a char
Reads a string using getline() function
Reads a char
Reads a string using getline() function
Reads a string using cin
Reads a string using getline() function
Reads an integer
```

2.3 Copying Strings

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

Example 12: 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
```

2.4 String Concatenation

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

Example 13: 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
```

2.5 Appending a Character to a String

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

Example 14: Concatenate string with a character.

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

int main(){
    string str = "Hello";
    char ch = 'A';

    str = ch + str + ch;

    cout << str;

    return 0;
}
```

Output

```
AHelloA
```

Note: Appending a character to a string using the + operator in C++ is supported from C++11 onwards.

2.6 Comparing Strings

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

Example 15: Compare two strings.

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

int main(){
    string str1 = "Hello", 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
```

2.7 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 16: 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
```

2.8 Iterating Through a String

You can iterate through a string using loops.

Example 17: 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++ Programming
```

2.9 Converting Between Strings and Numbers

stoi() and **stof**() functions are used to convert strings into numbers.

Example 18: Convert strings into numbers.

```
#include <iostream>
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 are used to convert numerical values into string values.

Example 19: Convert numbers into string values.

```
#include <iostream>
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
```

2.10 Array of Strings

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

Example 20: 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. Remove all consonants from a sentence.
2. Ask the user to enter a word and check if it is a palindrome (if it can be read backwards the same way).
3. 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.
4. Ask the user to enter a sentence, then display the number of consonants and vowels (a, e, i, o and u) in the sentence.
5. (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/