Character arrays and 2D arrays

Strings

Till now, we have seen how to work with single characters but not with a sequence of characters. For storing a character sequence, we use strings. Strings are the form of 1D character arrays which are terminated by null character. Null character is a special character with a symbol '\0'.

Declaration syntax:

```
string  variableName;
```

string is the datatype in C++ to store continuous characters that terminates on encountering a space or a newline.

For example:

Suppose the input is:

```
Hello! How are you?
```

If you try to take input using string datatype over the given example, then it will only store **Hello!** in the form of an array.

To store the full sentence, we would need a total of four-string variables (though there are other methods to handle this (getline), we will soon read about them.)

Note: To use string datatype, don't forget to include the header file:

```
#include <cstring>
```

Character arrays

Declaration syntax:

```
char Name_of_array[Size_of_array];
```

All the ways to use them are the same as that of the integer array, just one difference is there. In case of an integer array, suppose we want to take 5 elements as input, an array of size 5 will be good for this.

In case of character arrays, if the length of the input is 5 then we would have to create an array of size at least 6, meaning **character arrays require one extra space in the memory from the given size.** This is because the character arrays store a NULL character at the last of the given input size as the mark of termination in the memory.

Now talking about the memory consumption, as each character requires 4 bytes in memory, the character array will require 4 multiplied by the character array size.

Also to take the character array as the input, you don't need to necessarily run the loop for each element. You can directly do the same as follows:

```
cin >> Name_of_array;
```

If we take above the example and run this statement over that and let the name of the array is arr and the size of array is at least 7, then the memory representation is as follows:

0	1	2	3	4	5	6												
'H'	**'e'**	**'l'**	**'l'**	**'o'**	**'!'**	**'\0'**												

Here, also the cin statement terminates taking input if it encounters any of the following:

- Space
- Tab
- \n

At the last you can notice some special character is placed, that is the NULL character about which we were talking earlier.

In the same way, you can directly use a cout statement to print the character array instead of running the loop.

Till now we haven't solved the "string with spaces" problem. Let's check that also...

Getline function:

In C++, we have another function named as **cin.getline()** which takes 3 arguments:

- Name of the string or character array name
- Length of the string
- Delimiter (optional argument)

Syntax:

```
cin.getline(string_name, length_of_string, delimiter);
```

Note: This function breaks at new line. This also initializes the last position to NULL.

Delimiter denotes the ending of each string. Generally, it is '\n' by default.

Let's look at some examples:

Suppose the input we want to take is: **Hello how are you?**

If the input commands are:

- **cin.getline(input, 100);**: this will input all the characters of input
- **cin.getline(input, 3);**: this will input only the first 3 characters of input and discard the rest instead of showing error.

In-built functions for character array:

- **strlen(string_name)**: To calculate the length of the string
- **strcmp(string1, string2)**: Comparison of two strings returns an integer value. If it returns 0 means equal strings. If it returns a positive value, it means that string2 is greater than string1 and if it returns a negative value, it means that string1 is greater than string2.

- **strcpy(destination_string, source_string) :** It copies the source_string to destination_string.
- **strncpy(destination_string, source_string, number_of_characters_to_copy):** it copies only a specified number of characters from source_string to destination_string.

2D arrays

Combination of many 1D arrays forms a 2D array.

Syntax to declare a 2D array:

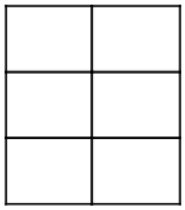
```
datatype array_name[rows][columns];
```

where rows imply the number of rows needed for the array and column implies the number of columns needed.

For example: If we want to declare an integer array of size 2x3, then:

```
int arr[2][3];
```

It looks like:



In this array you can store the values as required. Suppose in the above array you want to store 3 at every index, you can do so using the following code:

```
for (int i = 0; i < 2; i++)
{
        for (int j = 0; j < 3; j++)
        {
                arr[i][j] = 3;
        }
}
```

where, arr[i][j] invokes the element of the ith row and jth column.

How are 2D arrays stored in the memory?

They are actually stored as a 1D array of size (number_of_rows * number_of_columns).

Like if you have an array of size 5 x 5, so in the memory, a 1D array is created of size 25 and if you want to get the value of the element (2,1) in this array, it will invoke (2*5 + 1 = 11)th position in the array. We don't need to take care of this calculation, these are done by the compiler itself.

If we want to pass this array to a function, we can simply pass the name of the array as we did in the case of 1D arrays. But in the function definition, we can leave the first dimension empty, though the second dimension always needs to be specified.

Syntax for function call:

```
int fun (int arr[ ][number_of_columns], int n, int m) {
.................
}
```

Let's move to some questions now:

Practice problems:

Arrays and strings:

https://www.hackerearth.com/challenges/competitive/code-monk-array-strings/problems/

2D arrays:

https://www.hackerrank.com/challenges/2d-array/problem
https://www.techgig.com/practice/data-structure/two-dimensional-arrays

Rest there are many questions available for practice on codezen too. You can try them also...