

# Char Arrays

Arrays of characters is known as Character Arrays.

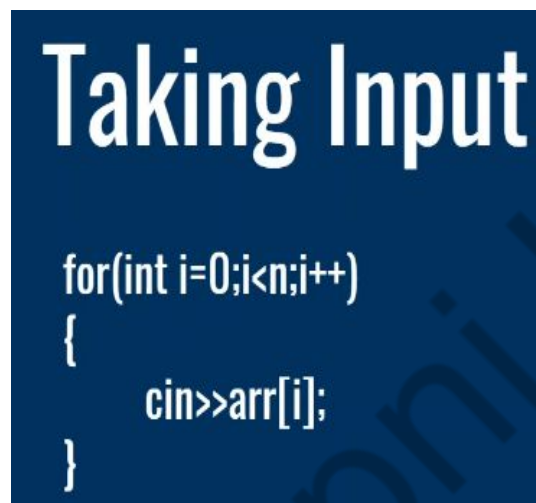
## Declaration

To declare a character array of n size, we do

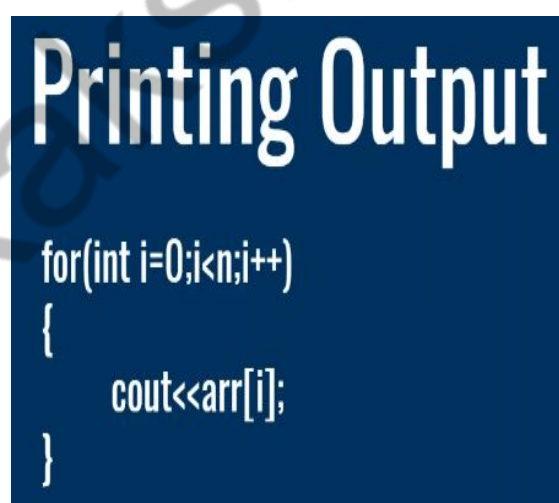
```
char arr[n+1];
```

Note: We declare an array of n+1 as 0 to n-1 indices store the actual string and nth character stores '\0' (null character).

## Taking input and Printing output

A dark blue rectangular box with white text. The title "Taking Input" is at the top in a large, bold font. Below it is a C++ code snippet for taking input from a character array.

```
for(int i=0;i<n;i++)  
{  
    cin>>arr[i];  
}
```

A dark blue rectangular box with white text. The title "Printing Output" is at the top in a large, bold font. Below it is a C++ code snippet for printing the contents of a character array.

```
for(int i=0;i<n;i++)  
{  
    cout<<arr[i];  
}
```

We can also directly take input if there are no spaces between the characters in the word

```
cin >> arr;
```

In the similar way, we can output the character array

```
cout << arr;
```

## Important Questions

### 1. Check if a given character array is a palindrome or not.

Palindrome: Given a string  $s$ , on reversing the string we get the same string we call that string is a palindrome.

For example:



Algorithm:

- 1) Let the length of the character array be  $n$ .
- 2) Keep a boolean variable `ans` to store the result and initialize it with `true`.
- 2) Iterate over the string and check if  $i^{\text{th}}$  character is equal to  $(n-i-1)^{\text{th}}$ , there can be 2 cases
  - a) If equal, then do nothing
  - b) If unequal, then put `ans = false`
- 3) When the loop ends, if `ans` is `true`, then the string is palindrome else it is not a palindrome.

Code:

The screenshot shows a C++ IDE with a file named `temp.cpp` open. The code implements a function to check if a string is a palindrome by comparing characters from both ends. The `input.txt` file contains the string "racecar", and the `output.txt` file shows the result: "racecar is a pallindrome".

```
#include<bits/stdc++.h>
using namespace std;

int main()
{
    #ifndef ONLINE_JUDGE
    freopen("input.txt", "r", stdin);
    freopen("output.txt", "w", stdout);
    #endif

    int n;
    cin >> n;

    char arr[n];
    cin >> arr;

    bool check = 1;
    for(int i=0; i<n; i++)
    {
        if(arr[i] != arr[n-1-i])
            check=0;
    }

    if(check)
        cout << arr << " is a pallindrome" << endl;
    else
        cout << arr << " is not a pallindrome" << endl;
}
```

input.txt

```
1 7
2 racecar
```

output.txt

```
1 racecar is a pallindrome
2
```

## 2. Largest word in a sentence

To input a complete sentence, we use the `getline()` function.

`cin.getline(arr, n)`

where `arr` is the character array and `n` is the total length of sentence.

### Approach

1. Iterate over the sentence and keep variables `currLen` and `maxLen` which store the current length of the present word being iterated and the overall maximum length word's length.
2. Whenever we encounter a space during iteration, we will maximize our `maxLen` variable.

$maxLen = \max(maxLen, currLen)$

### Code

```
void solve()
{
    char arr[100] = "apple";
    int n;
    cin >> n;
    cin.ignore();
    cin.getline(arr, n);
    cin.ignore();

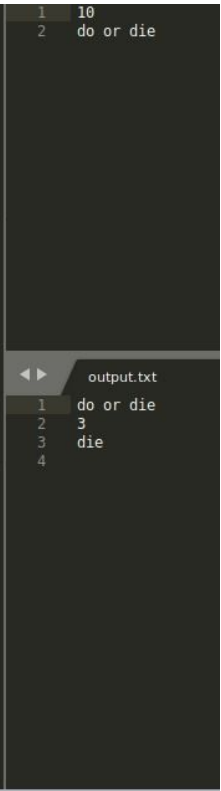
    int i=0;
    int currLen = 0;
    int maxLen = 0;
    int st = 0, maxSt = 0;
    while(i<n)
    {
        if(arr[i] == ' ' || arr[i] == '\0')
        {
            if(currLen > maxLen)
            {
                maxSt = st;
                maxLen = currLen;
            }

            st = i+1;
            currLen = 0;
        }
        else
            currLen++;

        if(arr[i] == '\0')
            break;
        i++;
    }

    cout << arr << endl;
    cout << maxLen << endl;

    for(int i=0; i<maxLen; i++)
        cout << arr[maxSt+i];
    cout << endl;
}
```



The screenshot shows a C++ IDE with a code editor on the left and a console/output window on the right. The code defines a `solve()` function that reads a sentence into a character array `arr` of size 100. It then iterates through the array to find the longest word. The input sentence is "do or die". The output shows the original sentence "do or die", the length of the longest word (3), and the longest word itself ("die").