Dt: 28/12/2022
Pg.

Module - Character Arrays and 2D Arrays

Why need of Character Arrays?

Because using character variable we can save one character at a time

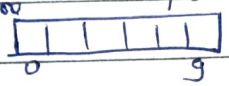
for example - Our Name - 'Ravi'

Strings → 1D Character Arrays terminated by '\0' character.

→ int a[10];

↳ 40 bytes

a = 100



int n;

for(i=0→n){

cin>>arr[i]; }

for(i=0→n) {

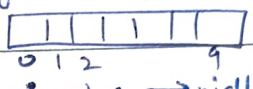
cout<<arr[i]; }

cout<<"\n"; → 100

→ char b[10]; { They behave little differently }

↳ 10 bytes

b = 700



cin>>b; → nidhi

cout<<b; → nidhi

null character

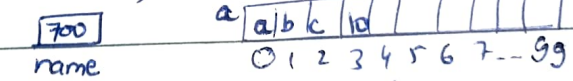
ASCII value = 0

↓

terminator

#

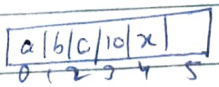
char name[100];



cin>> name; → abc

cout<< name; → abc

```
#include<iostream>
using namespace std;
int main(){
    char name[100];
    cin>>name;        → abc
    cout<<name;       → abc
}
```

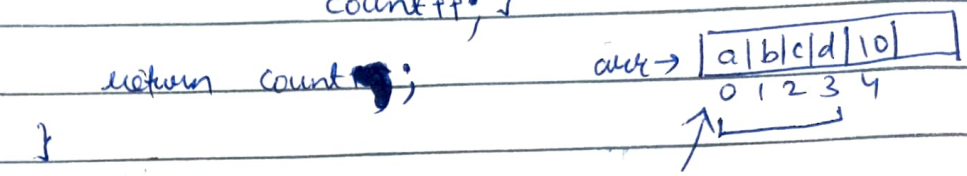


name[4]='x';

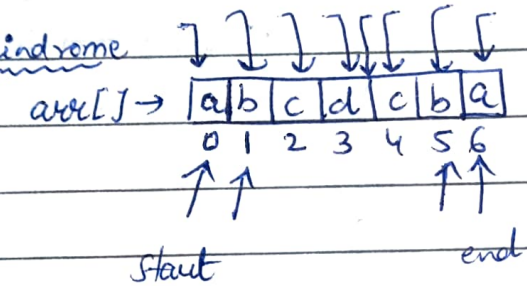
WOW!

Length of an Array

```
int length(char arr[]){ int count=0;
    for(int i=0; arr[i]!='\0'; i++){
        count++; }
    return count;
}
```



Check Palindrome




```
{
int start = 0, end= strlen(input) -1;
  while (start <= end) {
      if ( input[start] != input[end]) return false;
       start++;
       end--;
      }
   return true;
}
```

Replace Character

```
for(int i=0; input[i]!='\0'; i++){
        if (arr[i]==c1) {
            arr[i]=c2;}
        }
```

More About Character Arrays

char input[4]; (n → n+1)

_	_	_	\0
0	1	2	3

cin>>input;
- abcd ✗
- abc ✓
- ab ✓
- a ✓

cin>>input; → abcd → | a | b | c | d | \0 | (memory not mine — it may crash or may not)

cout<<input; → abcd

#

char arr[100];

n	e	l	l	o		w	o	r	l	d	\0

cin>>arr; → hello world → cin breaks on space, tab & enter (stops) taking input

cout<<arr; → hello

Alternative

cin.getline(string-name, len) ← no. of character it can enter

cin.getline(input, 100);
↳ hello world

cin.getline(string-name, len, delimiter) ← default '\n'

cin.getline(input, 100, 'o');
↳ Stops taking on encountering o

char a[10];
cin.getline(a, 10)
↳ 10 → 9
cin.getline(a, 4)
↓
3

hello world

1) input → [ ]
↓
reverse
↓
ip → hello world
op → dlrow olleh

input → | h | e | \0 |

cin.getline(input, len);

char input[10];
↓
10

WOW!

Dt
Pg

1) Reverse Character Array

f d c b a

input → | a | b | c | d | f | \0 |
0 1 2 3 4 5 6

output → | a | d | c | b | a | \0 |
0 1 2 3 4 5 6

start end
0 5 (len-1)
↓ ↓
start++ end--

```
void reverseString(char arr[]){
    int i=0, j= length(arr) -1;
    while(i<=j){
        swap(arr[i], arr[j]);
        i++;
        j--;
    }
}
```

Example

d c b a

① input → | a | b | c | d | \0 |
0 1 2 3 4

len = 5
i = 0 2
j = 4 3 2
temp = a

i=0 j=len-1

Trim Spaces

input → | a | b | c | | d | e | f | | g | | h | i | \0 |

Approach → index = 0

```
if(input[i] != ' '){
    input[index] = input[i];
    index++; }
}
input[index] = '\0';
```

Reverse WordWise

input → Welcome to Coding Ninjas

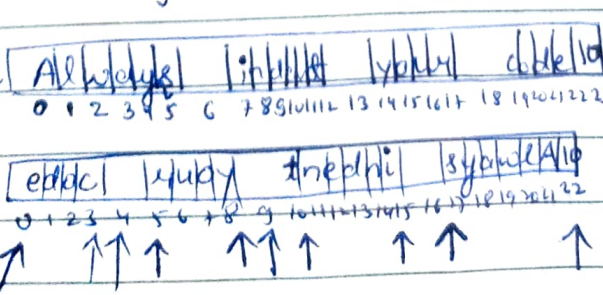
Output → sajniN gnidoC ot emocleW

→ Approach

```
void Reverse (char* arr, int start, int end) {
    while (start <= end) {
        swap (arr[start], arr[end]);
        start++;
        end--; }
    }

void ReverseWordWise (char* arr) { int len = strlen(arr) - 1;
    Reverse (arr, 0, len);
    int i = 0, start = 0, end;
    while (arr[i] != '\0') {
        if (arr[i] == ' ') {
            end = i - 1;
            Reverse (arr, start, end);
            start = i + 1;
            }
        i++;
    }
    Reverse (arr, start, len);
}
```

Example -



Code

WOW!

Inbuilt Functions (Strings)

1) Length of String → strlen(input) #include <cstring>

```
{
int input[100];
cin>>input;
int len=strlen(input);
cout<<len<<endl;
}
```

2) To compare two strings

↓

strcmp (str1, str2);

→ 0 (same) equal
→ non-zero (not same) not equal

	0	1	2	3
input1 →	a	b	c	\0
input2 →	a	c	b	\0
	0	1	2	3

i=0

input1[i] - input2[i]

97-97 = 0

98-97 = 1

97-99 = -2

```
if(strcmp(a1,a2)==0){
    cout<<"Equal"<<endl;}
else{
    cout<<"Not equal";
}
```

3) # To copy string → strcpy(string-name)

destination, source

a → hello

b → hello

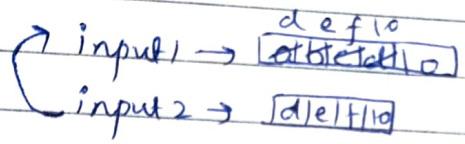
a=abc;
b=def;

strcpy(a,b);
cout<<a; → def

To showcase copying of two strings

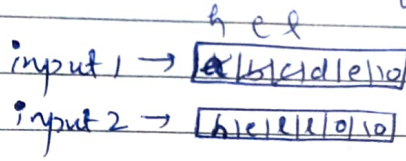
```
char arr1[100] = "abcd";
char arr2[100] = "def";
cout << "Before copying";
cout << "Input1:" << input1 << endl;
cout << "Input2:" << input2 << endl; strcpy(input1, input2);
cout << "After copying:";
cout << "Input1:" << input1 << endl;
cout << "Input2:" << input2 << endl;
```

Example



To copy first n characters of string into another string.

↳ It does not append \0 by itself.



strncpy (dest_string, source_string, n);
(input1, input2, 3)

input1 → abcd

all prefixes →
- a
- ab
- abc
- abcd

a → 0, 0
ab → 0, 1
abc → 0, 2
abcd → 0, 3

```
void AllPrefixes(char input[]){
    input[i] = '\0';
for(int i=0; i< len; i++){
    for(int j=0; j<=i; j++){
        cout << input[j];
    }
cout << endl;
}
```

WOW!

Dt
Pg

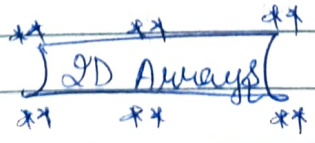
Print All Substrings

input → a b c \0
0 1 2 3

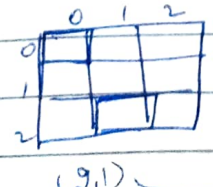
a → 0,0
ab → 0,1
abc → 0,2
b → 1,1
bc → 1,2
c → 2,2

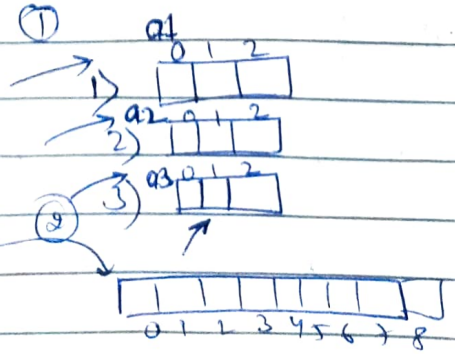
```
for(int i=0; input[i]!='\0';i++){ → start
    for(int j=i; input[j]!='\0'; j++){ → end
        for(int k=i; k<=j; k++){ → print
            cout<< input[k];
        }
    }
cout<<endl;
}
```

I { a, ab, abc
II { b, bc
III { c



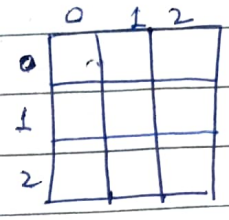
Tic Tac Toe



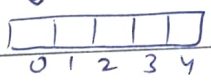


Dt
Pg

2D Arrays



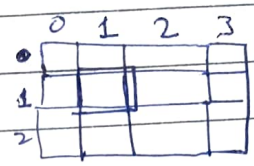
int arr[5];



arr[1] = 10;

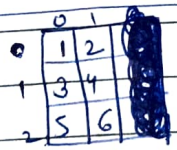
int arr[3][3]; Rows=3 columns=3

Rows columns

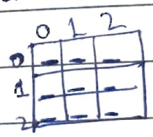


arr[1][1] = 10;

```
int arr[100];
int n;
cin>>n;
for(0 → n-1){
    cin>>arr[i]; }
for(0 → n-1){
    cout << arr[i];
```

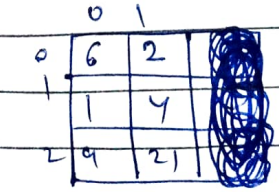



```
int b[100][100];
rows → m = 3
columns → n = 2
for(0 → 2){
    for(0 → 1){
        cin>>arr[i][j]; }}
```



arr[0][0]
arr[0][1]
arr[0][2]
arr[1][0]
arr[1][1]
arr[1][2]
arr[2][0]
arr[2][1]
arr[2][2]

```
for(0 → 1){
    for(0 → 2){
        cout << arr[i][j]; }}
```



6 1 9
2 4 21

```
for(0 → 2){
    for(0 → 1){
        cout << arr[i][j];
    } cout << endl;
}
```

WOW!

Dt
Pg

Column Wise Sum

	0	1
0	4	2
1	1	2
2	3	4
3	5	6
4	7	8

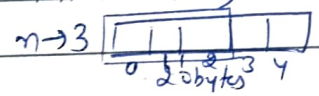
	0	1	2	3	4	
0	4	2	1	2	3	= 12
1	4	5	6	7	8	

Column → 0 → 20
→ 1 → 22

How are 2D Arrays Stored?

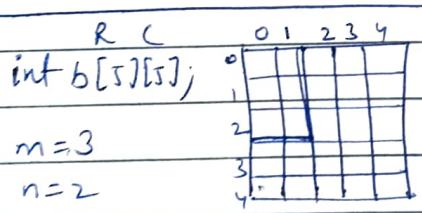
int arr[5];

n → 3



int b[5][5];

m = 3
n = 2



arr[i][j] → (Cxi+j

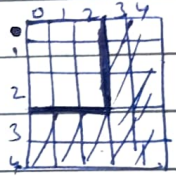
```
void printArray (int b[][100], int m, int n){
    for(int i=0; i<m; i++){
        for(int j=0; j<n; j++){
            cout<<b[i][j]<<" ";
        }
        cout<<endl;
    }
}
```

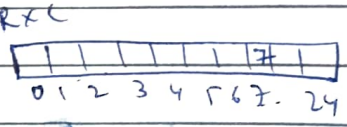
m = 3
n = 2

arr →

arr[0][0] → (Cxi+j) → 100

int b[5][5]; m = 3 n = 2





a[2][1] = 7

(2,1)

(Column x 2 + 1)

(Cxi+j)

WOW!

```
int arr[] = {1, 2, 3};
int b[][2] = { {1,2}, {3,4}, {5,6} };
```

```
int arr[10] = {1, 2, 3, 4, 5};
```

arr → | 1 | 2 | 3 | 4 | 5 | 0 | 0 | 0 | 0 | 0 |
0 1 2 3 4 5 6 7 8 9

```
int arr[2][5] = { {1, 2}, {3, 4} };
```

arr →	0	1	2	3	4
0	1	2	0	0	0
1	3	4	0	0	0

Largest Row or Largest Column

	0	1	2
0	2	1	3
1	6	4	5
2	9	8	7

maxSum = ~~0~~ ~~2~~ ~~3~~ ~~5~~ ~~6~~ ~~15~~ 24

~~2 3 15~~ 24

RowSum = ~~0~~

RowIndex = ~~0~~ ~~1~~ 2

ColSum = ~~0~~

ColIndex = 0

Wave Print

arr →	0	1	2	3
0	6	23	27	31
1	1	21	26	30
2	2	20	25	32
3	3	16	29	33

[6 | 1 | 2 | 3 | 16 | 20 | 21 | 23]

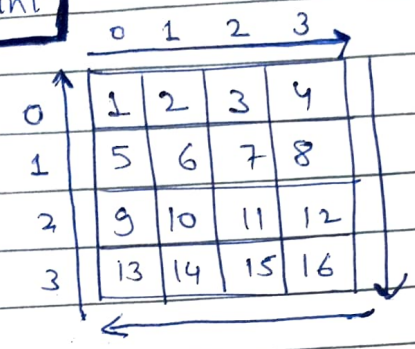
⇒ [25 | 26 | 28 | 29 | 31 | 30 | 32 | 33]

```
for(int j=0; j<cols; j++){
    if(j%2==0){
        for(int i=0; i<rows; i++){
            cout<<arr[i][j]<<" ";
        }
    }
    else{
        for(int j=m-1; j>=0; j--){
            cout<<arr[i][j]<<" ";
        }
    }
}
```

WOW!

Dt
Pg

Spiral Print



i (current row)	j (current column)	Total Rows	Total Columns
~~0~~ 1	0 - 3	4	4
1 - 3	~~3~~		
~~2~~	2 - 0		

Problem Code

```
void SpiralPrint(int arr[][cols], int rows, int cols){
    int i;
    int currRow=0, currCol=0;
    while(currRow<rows && currCol<cols){
        for(i=currCol; i<cols; i++){
            cout<<arr[currRow][i]<<" ";}
        currRow++;
        for(i=currRow; i<rows, i++){
            cout<<arr[i][cols-1]<<" ";}
        cols--;
        if(currRow<rows){
            for(i=cols-1; i>=currCol; i--){
                cout<<arr[rows-1][i]<<" ";
            }
            rows--;
        }
        if(currCol<cols){
            for(i=rows-1; i>=currRow; i--){
                cout<<arr[i][currCol];
            }
            currCol++;
        }
```

Dt
Pg

Check Permutation

str1 → | a | b | c | d | e | \0 |
0 1 2 3 4 5

str2 → | b | a | e | d | c | \0 |
0 1 2 3 4 5

freq[256] = | | | | | | | |
0 1 2 3 ... 255

```
bool CheckPermutation(string str1, string str2){
    int freq[256]={0};
    for(int i=0; str1[i]!='\0'; i++){
        int index = str1[i];
        freq[index]++; }
    for(int i=0; i<256; i++){
        if(freq[i]!=0) return false; }
    return true;
}
```

```
for(int i=0; str2[i]!='\0'; i++){
    int index=str2[i];
    freq[index]--; }
```

Remove Consecutive Duplicates

str1 → | x | x | y | y | z | x | x |
0 1 2 3 4 5 6

index=1; lastChar = x

str1 → | x | y | z | x | \0 |
0 1 2 3 4

Approach

arr[] → | a | x | x | b | c | c | d | d | \0 |
0 1 2 3 4 5 6 7 8

→ | a | x | b | c | d | \0 | | | |
0 1 2 3 4 5 6 7 8

index ~~1~~ 2

nextIndex ~~1~~ ~~2~~ ~~3~~

lastChar ~~a~~ ~~x~~ b c d \0

Example → arr[] = | a | a | a | b | b | c | d | d | \0 |
0 1 2 3 4 5 6 7 8

WOW!

Dt
Pg

Remove Characters

```
void RemoveCharacter(char input[], char c1){ int j=0;
    for(int i=0; input[i]!='\0'; i++){
        if(input[i]!=c1){
            input[j++]=input[i];
        }
    }
    input[j]='\0';
}
```

Highest Occuring Character

freq → | | | | | | | | (256 characters)
0 1 2 3 4 ... 255

```
max=INT_MIN; {
for(0 → len){
    freq[str[i]]++; }
for(0 → len){
    if(max < freq[str[i]]){
        max = freq[str[i]];
        result = str[i]; }
}
return result;
}
```

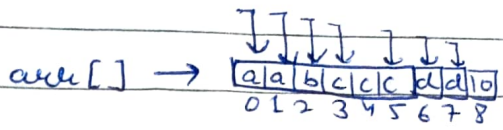
Compress the String

char → | a | a | b | b | b | b | b | d | d | e | e | a |
0 1 2 3 4 5 6 7 8 9 10 11

Approach

Dt
Pg

Approach

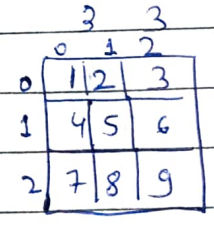


string ans;
int count = 1;

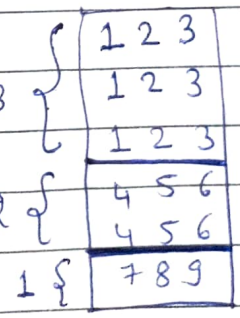
- a2b1c3d2

```
string getCompressed(string input){
    string ans;
    int len = strlen(input); int count = 1;
    for(int i = 0; i < len; i++){
        char arr = input[i];
        if(arr != input[i+1]){
            if(count == 1) ans += x;
            else ans += x + count + '0';
        } count = 1;
    else {
        count++;
    }
}
return ans;
}
```

Print 2D Array



⇒



WOW!

Minimum length word

t	h	i	s		i	s		t	e	s	t		i	s		t	e	s	t		a		w	o	r	d
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19							

```
int startIndex = 0; min = INT_MAX; int i = ; int count = 0;
for(i=0; input[i] != '\0'; i++){
        if(input[i] == ' '){
                if(count < min){
                        min = count;
                        startIndex = i - min; }
                count = 0; }

        }
        else {
                count++; }
        if(count < min){
                min = count;
                startIndex = i - min; }
        int k = 0;
        while(k < min){
                output[k++] = input[startIndex++]; }
```

Print leaders in an Array

arr[ ] = { 3, 12, 34, 2, 0, -1 }

```
for(0 → len-1){
        bool isLeader = 1;
        for(i+1 → len){
                if(arr[j] > arr[i]) isLeader = 0; break; }
        if(isLeader) cout << arr[i] << " "; }
cout << arr[len-1]; }
```