→ Character arrays / Strings

→ multi-dimensional array (2D-array)

Pointer type-casting: same as normal type-casting

difference is, pointers type are represented with additional *,

- (void*) also exists

} uses

References / Reference variable / func. with ref. variables

Returning pointers from function =

Array and its relation with pointers.

Pointer Arithmetic.

Bubble Sort, Selection sort using functions ?? HW

Q: Why we need to pass the size of an array to a func. ??

→ When we pass array to a function, it is passed as a pointer to (arr, arr[0]).

⇒ it has no information about the no. of elements.

Post-increment & Pre-increment

a++ → a+=1

++a → a+=1;

a should increase by one

difference ??

when to increase

Operator Precedence

Associativity !!

```
a = 5
x = a++;
```

⇓

```
x = a;
a += 1;
```

first use the value, then inc.

```
a = 5
x = ++a;
```

⇓

```
a += 1;
x = a
```

first inc., then use the value.

Same for a-- and --a;

17 a++; ⇒ a += 1;

++a;

Square Root

1 → 1 110 → 10

2 → 1

3 → 1

4 → 2

5 → 2

6 → 2

Nearest power of 2.

1, 2, 4, 8, 16, 32, 64, 128, 256

N ← nearest

abs difference minimum

N ≥ 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, ...

Square root → 1, 2, 3, 4, 5, 6, ...

Sq. root	N
1	< 4 2^2
2	< 9 3^2
3	< 16 4^2
4	< 25 5^2

→ iterate from $i=1$ onwards, (no. of steps unknown) (14)

and simply check whether the number is less than $(i+1)^2$

$i=1$ 2^2 ×

$i=2$ 3^2 ×

$i=3$ 4^2 ✓ ans

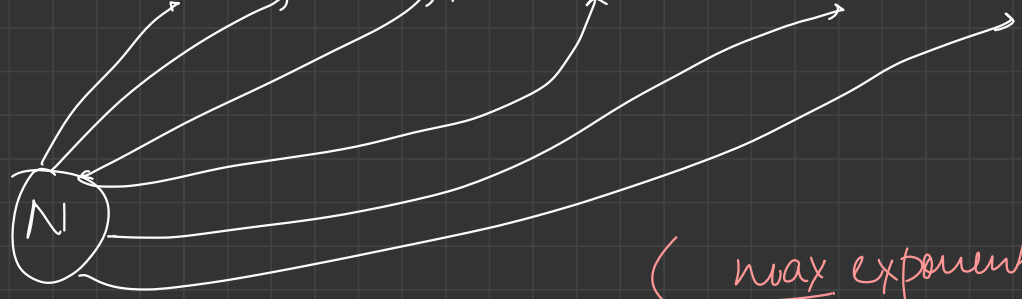
if yes then → i is the output

else, i++;

loop

while

Print 1, 2, 4, 8, 16, 32, 64, 128, 256, 512



N can be 2^x ←

or can be b/w 2^{x-1} — 2^x

{ max exponent of (2), 2^x which is less / equal, to N }

2^{x-1} —(×2)→ 2^x

N → nearest (2^{x-1}, 2^x)

Character Arrays

Strings

sequence of chars / symbols

an array of chars

```
char a[40];
```

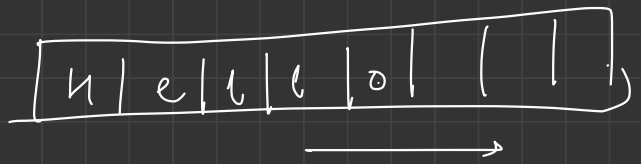
a[0] = "H"

a[1] = "e"

a[2] = "l"

a[3] = "l"

a[4] = "o";



Sequence of chars

words

sentence

Can I use character array??

Strings : are stored / behind the scenes are char arrays.

:- we have derived data type known as strings.

⇓

a sequence of chars

↓

behaves like char[]

- also has some more features

to represent the char[] in a more subtle way, we made a string data-type

In C/C++, every string is simulated (behind-the-scenes) using char arrays.

char [] ⟺ strings

They are almost same thing.

Q. Can we deduce, why we can store a string ("hi") in a char []. ??

→ Convention: Strings always end with NULL character (ascii → 0) ('\0')

null character ('\0') marks the end of a string.

String: sequence of chars

- implemented using char[]
- always ends with '\0'

Observe: char array behaves as a string during input/output.

char array ⟺ string : cout: treats a char array as a strings not as an array.

Overloaded <<

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

address

char str[5] = {"H", 'e', 'l', 'l', 'o'};

char *

cout

Characters / some rubbish

not the address

str ⇔ (char *)

cout

not address

not address

cout << str;

not as an array

output

Hello

cout treated this char[] as a string and prints it collectively.

→ cout treats, char address (char[], char*) differently

→ if we give it any other address (int, bool, double, ...) it just prints the address.

→ It will print each character byte-by-byte, until it finds a '\0'. (Starting from the address of array)

char → 1 byte

&str → 100

cout << str;

cout << *(100);
*(101);
*(102);
*(103);
⋮
until the char is null.

cout << str;

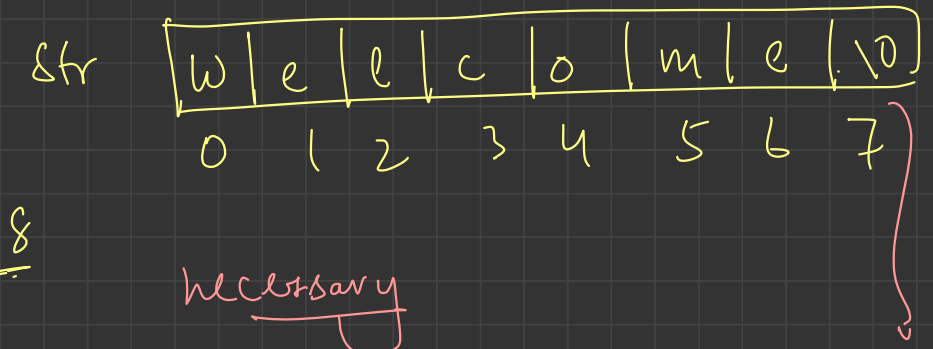
*str;
*(str+1);
*(str+2);
⋮
until '\0'.

→ it doesn't care about the size of array / space-allocated.

char[] are treated as strings, and strings end with '\0'.

Cout prints char[] until a null char, which is to be found at the end of string.

char str[ ] = "Welcome";



size = 8

necessary

cout << str; ⇒ Welcome

character arrays behave as a string (seq. of chars ends with '\0')

Input to a char array

Q. Check palindrome using function.

Q. Read N, and then N strings, & print the largest

Q. Write a function to rotate a string by k.

Shift right by one place

input: Hello ⇒ oHell : output

k = 3