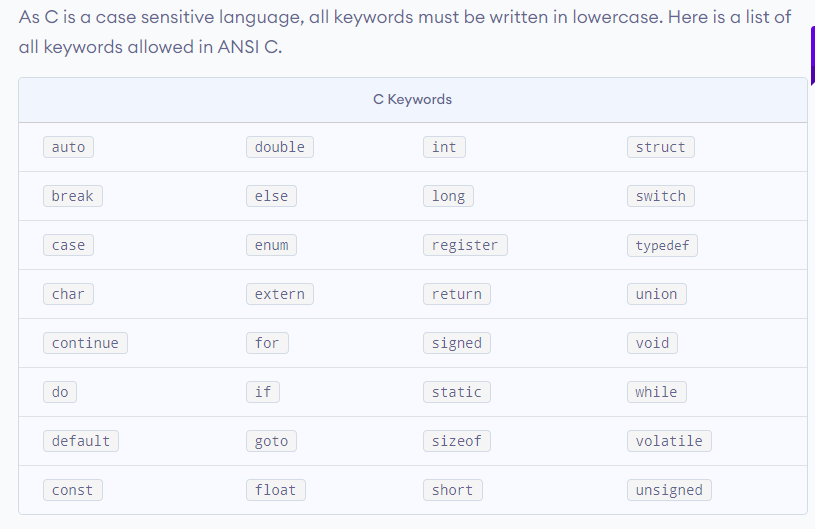
**C Keywords and Identifiers**

## C Keywords

Keywords are predefined, reserved words used in programming that have special meanings to the compiler. Keywords are part of the syntax and they cannot be used as an identifier. For example:

int money;

**Here, int is a keyword that indicates money is a**[**variable**](https://www.programiz.com/c-programming/c-variables-constants)**of type int (integer).**



## C Identifiers

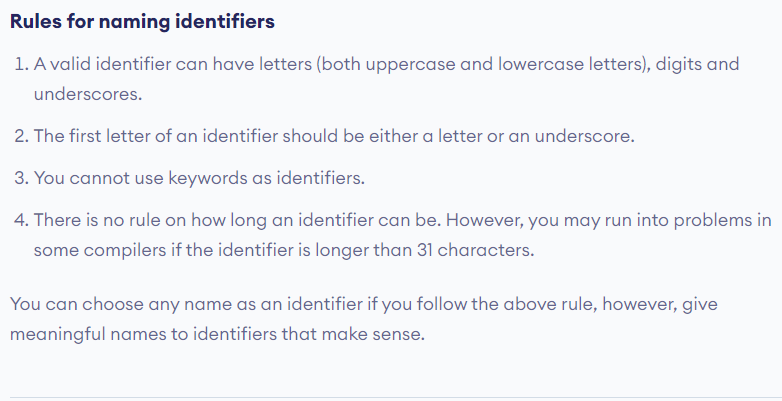
Identifier refers to name given to entities such as variables, functions, structures etc.

Identifiers must be unique. They are created to give a unique name to an entity to identify it during the execution of the program. For example:

int money;

double accountBalance;

Here, money and accountBalance are identifiers.



# C Variables, Constants and Literals

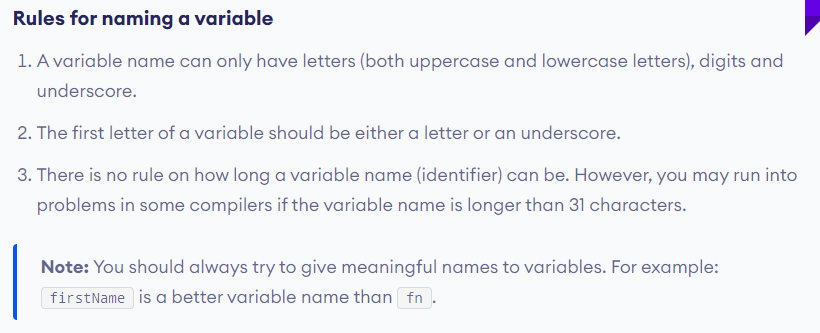
## Variables

In programming, a variable is a container (storage area) to hold data.

To indicate the storage area, each variable should be given a unique name ([identifier](https://www.programiz.com/c-programming/c-keywords-identifier)). Variable names are just the symbolic representation of a memory location. For example:

int playerScore = 95;

Here, playerScore is a variable of int type. Here, the variable is assigned an integer value 95.



C is a strongly typed language. This means that the variable type cannot be changed once it is declared. For example:

int number = 5; // integer variable

number = 5.5; // error

double number; // error

Here, the type of number variable is int. You cannot assign a floating-point (decimal) value 5.5 to this variable. Also, you cannot redefine the data type of the variable to double. By the way, to store the decimal values in C, you need to declare its type to either double or float.

## Literals

Literals are data used for representing fixed values. They can be used directly in the code. For example: 1, 2.5, 'c' etc.

Here, 1, 2.5 and 'c' are literals. Why? You cannot assign different values to these terms.

### 1. Integers

An integer is a numeric literal(associated with numbers) without any fractional or exponential part. There are three types of integer literals in C programming:

* decimal (base 10)
* octal (base 8)
* hexadecimal (base 16)
* For example:
* Decimal: 0, -9, 22 etc
* Octal: 021, 077, 033 etc
* Hexadecimal: 0x7f, 0x2a, 0x521 etc
* In C programming, octal starts with a 0, and hexadecimal starts with a 0x.

### 2. Floating-point Literals

A floating-point literal is a numeric literal that has either a fractional form or an exponent form. For example:

-2.0

0.0000234

-0.22E-5

**Note:** E-5 = 10-5

### 3. Characters

A character literal is created by enclosing a single character inside single quotation marks. For example: 'a', 'm', 'F', '2', '}' etc.

### 4. Escape Sequences

Sometimes, it is necessary to use characters that cannot be typed or has special meaning in C programming. For example: newline(enter), tab, question mark etc.

In order to use these characters, escape sequences are used.

| Escape Sequences | |
| --- | --- |
| Escape Sequences | Character |
| \b | Backspace |
| \f | Form feed |
| \n | Newline |
| \r | Return |
| \t | Horizontal tab |
| \v | Vertical tab |
| \\ | Backslash |
| \' | Single quotation mark |
| \" | Double quotation mark |
| \? | Question mark |
| \0 | Null character |

### 5. String Literals

A string literal is a sequence of characters enclosed in double-quote marks. For example:

"good" //string constant

"" //null string constant

" " //string constant of six white space

"x" //string constant having a single character.

"Earth is round\n" //prints string with a newline

## Constants

If you want to define a variable whose value cannot be changed, you can use the const keyword. This will create a constant. For example,

const double PI = 3.14;

Notice, we have added keyword const.

Here, PI is a symbolic constant; its value cannot be changed.

const double PI = 3.14;

PI = 2.9; //Error

# C Data Types

## Basic types

Here's a table containing commonly used types in C programming for quick access.

| Type | Size (bytes) | Format Specifier |
| --- | --- | --- |
| int | at least 2, usually 4 | %d, %i |
| char | 1 | %c |
| float | 4 | %f |
| double | 8 | %lf |
| short int | 2 usually | %hd |
| unsigned int | at least 2, usually 4 | %u |
| long int | at least 4, usually 8 | %ld, %li |
| long long int | at least 8 | %lld, %lli |
| unsigned long int | at least 4 | %lu |
| unsigned long long int | at least 8 | %llu |
| signed char | 1 | %c |
| unsigned char | 1 | %c |
| long double | at least 10, usually 12 or 16 | %Lf |

### void

void is an incomplete type. It means "nothing" or "no type". You can think of void as **absent**.

For example, if a function is not returning anything, its return type should be void.

Note that, you cannot create variables of void type.

### short and long

If you need to use a large number, you can use a type specifier long. Here's how:

long a;

long long b;

long double c;

Here variables a and b can store integer values. And, c can store a floating-point number.

If you are sure, only a small integer ([−32,767, +32,767] range) will be used, you can use short.

short d;

You can always check the size of a variable using the sizeof() operator.

#include <stdio.h>

int main() {

short a;

long b;

long long c;

long double d;

printf("size of short = %d bytes\n", sizeof(a));

printf("size of long = %d bytes\n", sizeof(b));

printf("size of long long = %d bytes\n", sizeof(c));

printf("size of long double= %d bytes\n", sizeof(d));

return 0;

}

### signed and unsigned

In C, signed and unsigned are type modifiers. You can alter the data storage of a data type by using them. For example,

unsigned int x;

int y;

Here, the variable x can hold only zero and positive values because we have used the unsigned modifier.

Considering the size of int is 4 bytes, variable y can hold values from -231 to 231-1, whereas variable x can hold values from 0 to 232-1.

Other data types defined in C programming are:

* bool Type
* Enumerated type
* Complex types

## Derived Data Types

Data types that are derived from fundamental data types are derived types. For example: arrays, pointers, function types, structures, etc.

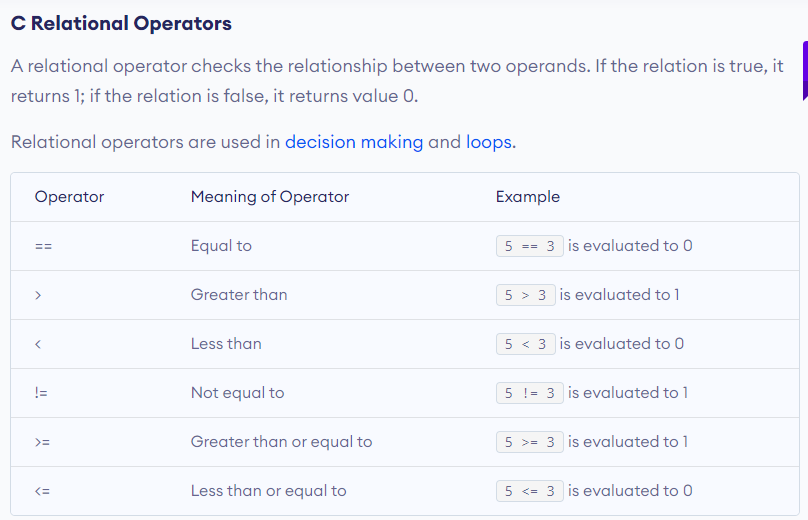
# C Programming Operators

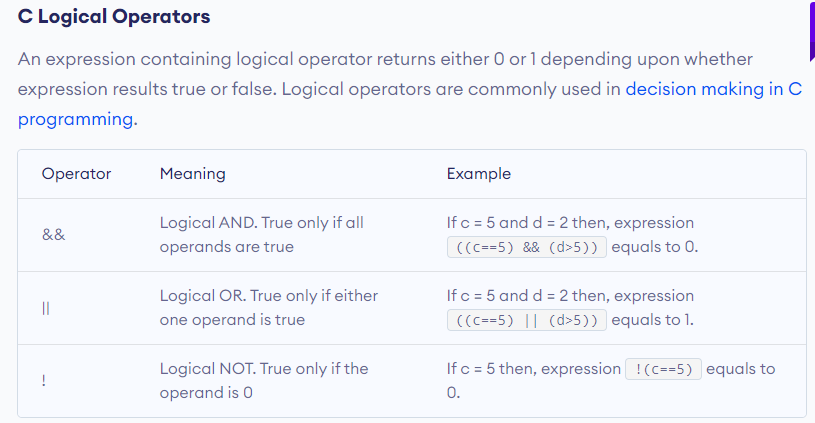
# 

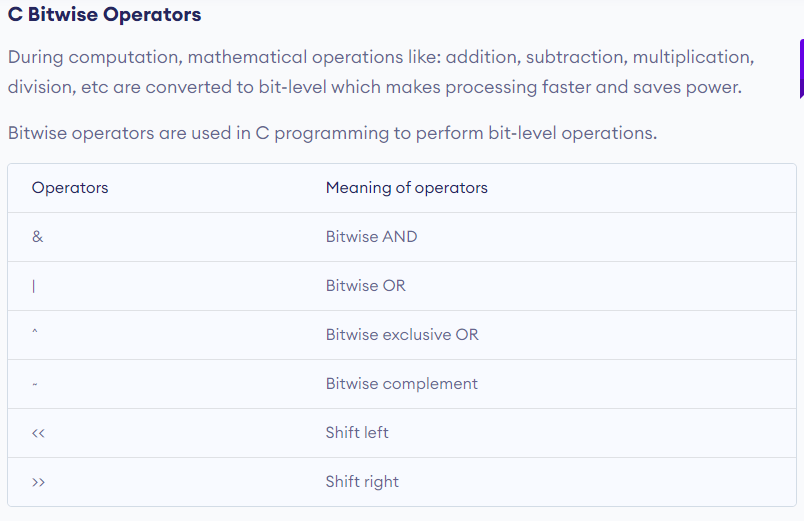
## C Increment and Decrement Operators

C programming has two operators increment ++ and decrement -- to change the value of an operand (constant or variable) by 1.









### Comma Operator

Comma operators are used to link related expressions together. For example:

int a, c = 5, d;

**C Pointers**

Pointers (pointer variables) are special variables that are used to store addresses rather than values.

### Pointer Syntax

Here is how we can declare pointers.

int\* p;

Here, we have declared a pointer p of int type.

You can also declare pointers in these ways.

int \*p1;

int \* p2;

Let's take another example of declaring pointers.

int\* p1, p2;

Here, we have declared a pointer p1 and a normal variable p2.

### Assigning addresses to Pointers

Let's take an example.

int\* pc, c;

c = 5;

pc = &c;

Here, 5 is assigned to the c variable. And, the address of c is assigned to the pc pointer.

### Get Value of Thing Pointed by Pointers

To get the value of the thing pointed by the pointers, we use the \* operator. For example:

int\* pc, c;

c = 5;

pc = &c;

printf("%d", \*pc); // Output: 5

Here, the address of c is assigned to the pc pointer. To get the value stored in that address, we used \*pc.

**Note:**In the above example, pc is a pointer, not \*pc. You cannot and should not do something like \*pc = &c;

By the way, \* is called the dereference operator (when working with pointers). It operates on a pointer and gives the value stored in that pointer.

### Changing Value Pointed by Pointers

Let's take an example.

int\* pc, c;

c = 5;

pc = &c;

c = 1;

printf("%d", c); // Output: 1

printf("%d", \*pc); // Ouptut: 1

We have assigned the address of c to the pc pointer.

Then, we changed the value of c to 1. Since pc and the address of c is the same, \*pc gives us 1.

Let's take another example.

int\* pc, c;

c = 5;

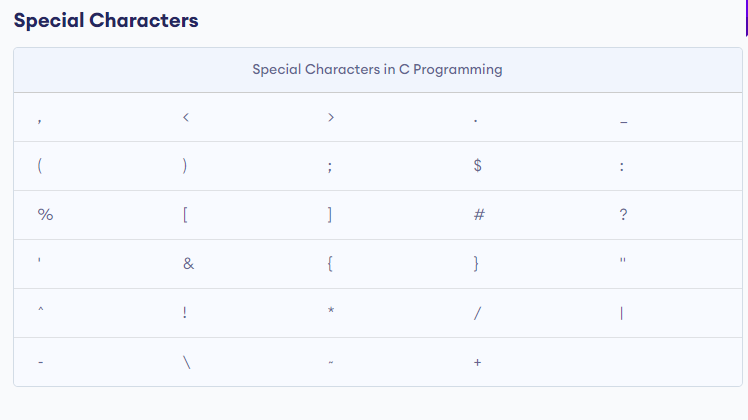
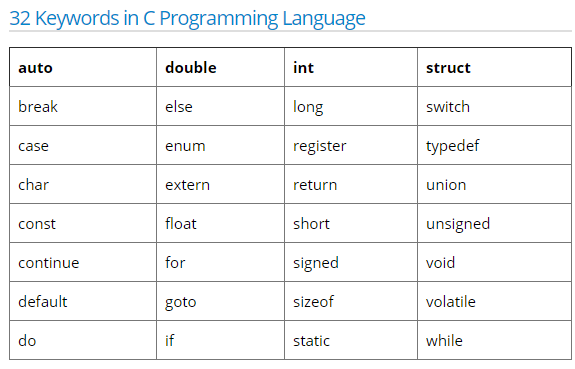
pc = &c;

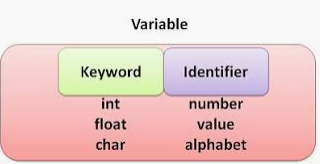
\*pc = 1;

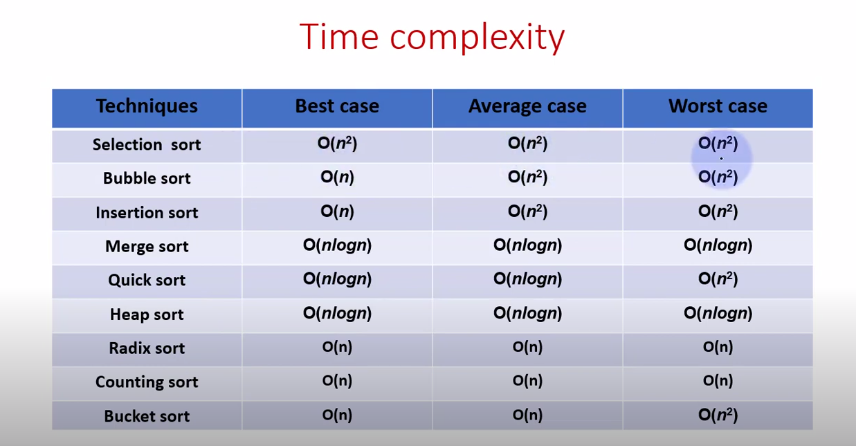
printf("%d", \*pc); // Ouptut: 1

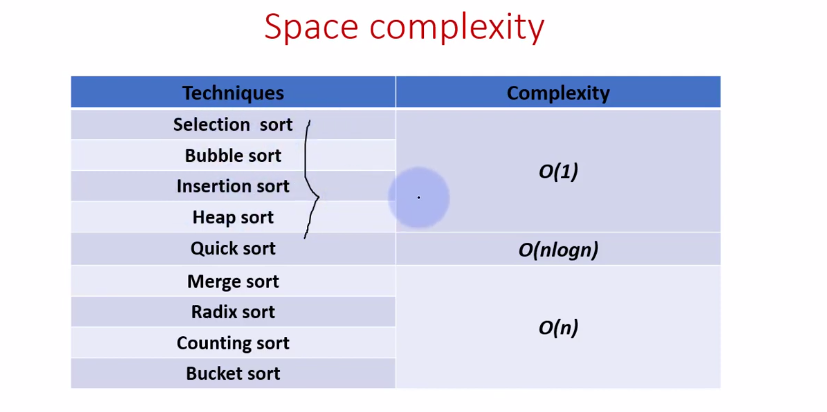
printf("%d", c); // Output: 1

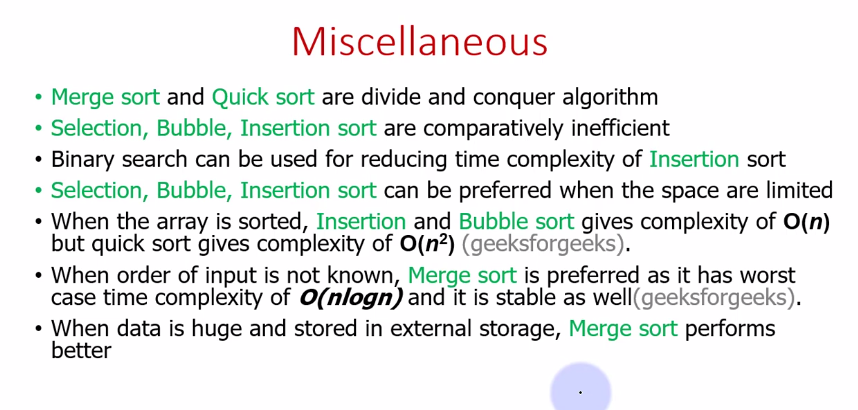
We have assigned the address of c to the pc pointer.

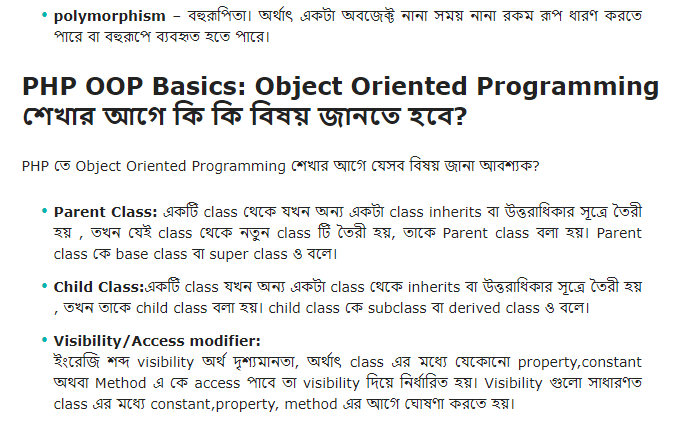
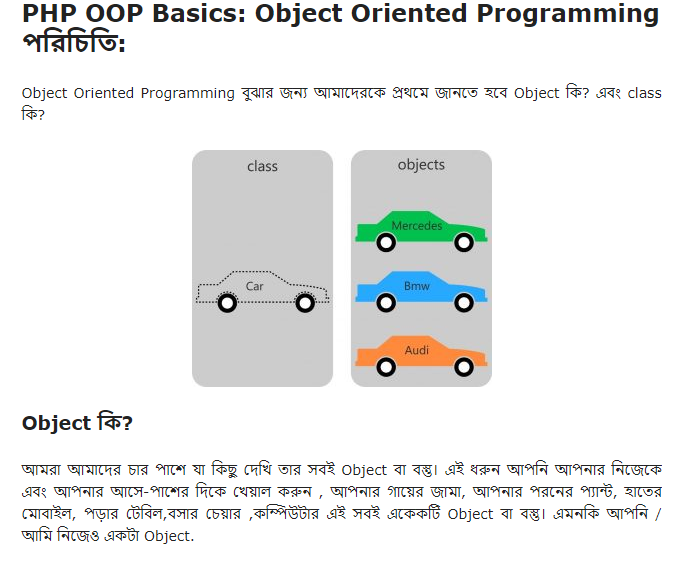
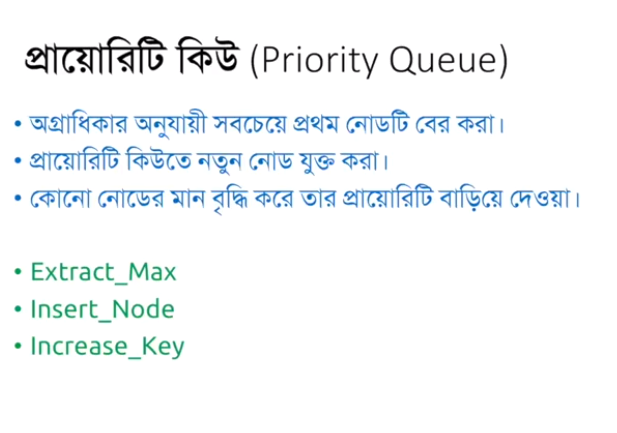
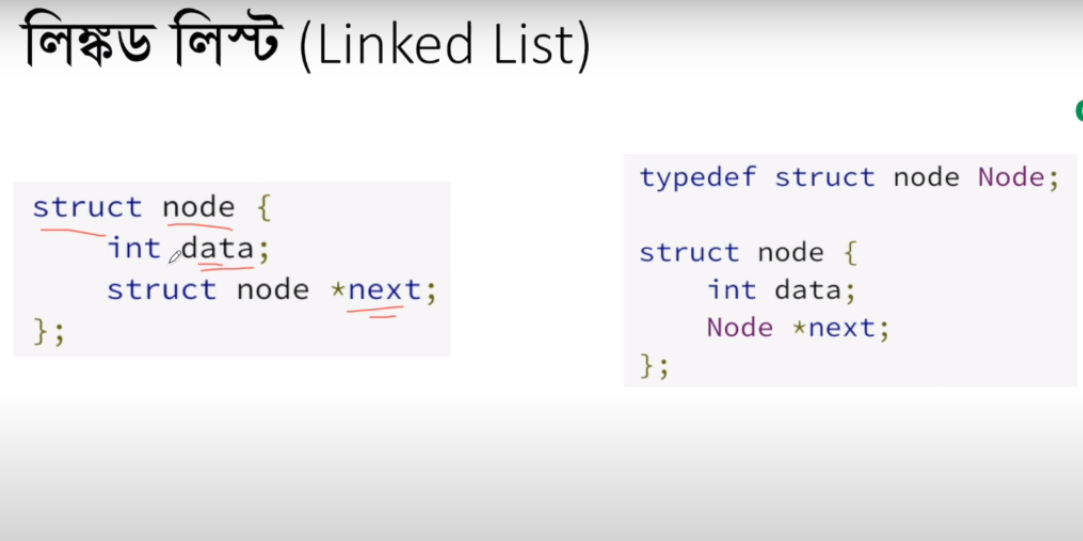
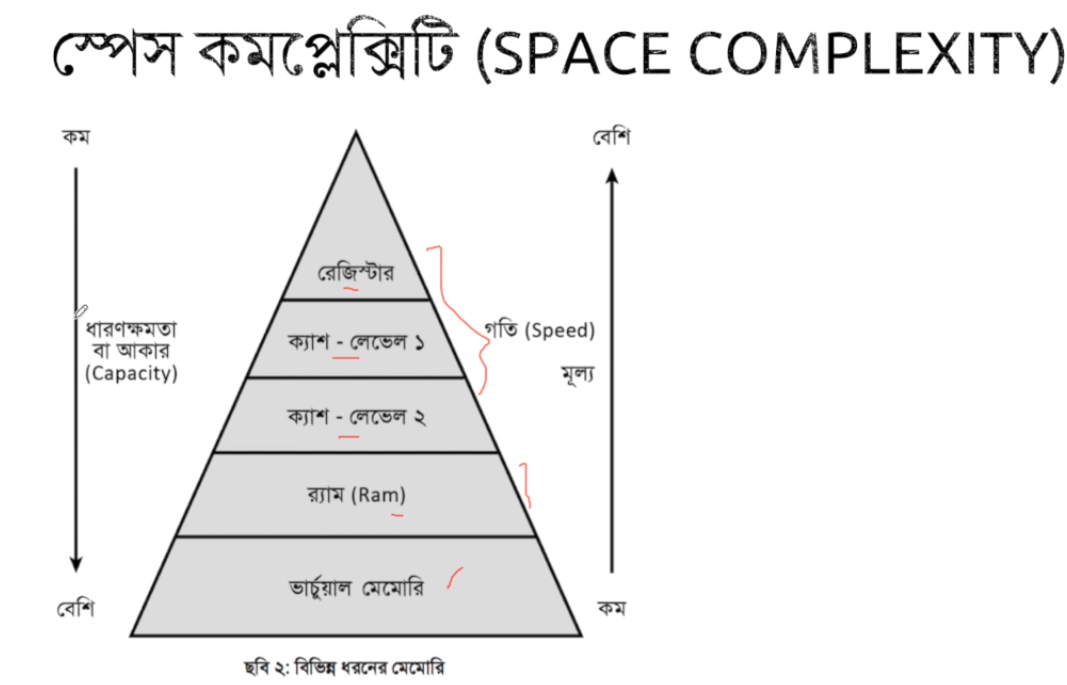
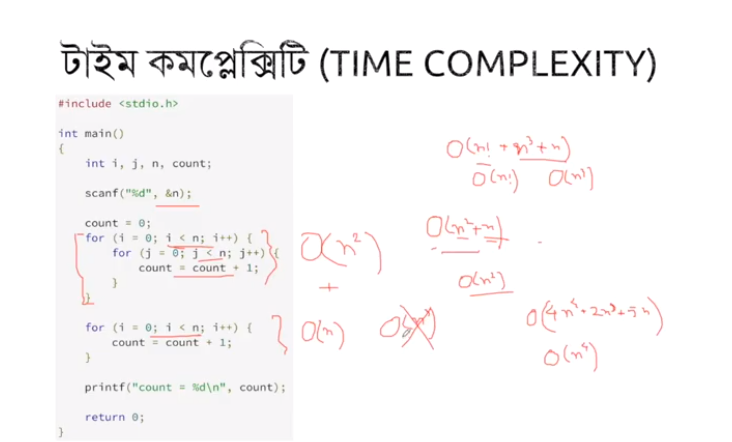












#### PHP তে Visibility তিন ধরনের:

