1)what is variable
In C programming, a variable definition is the process of declaring a variable and, crucially, allocating memory for it. It's the point where the compiler reserves space in memory for the variable and, optionally, assigns it an initial value. While you can declare a variable multiple times, it can only be defined once within a given scope (file, function, or block of code). [1, 2]
Example:

```
#include <stdio.h>

int main() {
    // Variable definition and initialization
    int age = 30;

    // Variable definition without immediate initialization
    float price;
    price = 99.99; // Assignment of a value later

    printf("Age: %d\n", age);
    printf("Price: %.2f\n", price);

    return 0;
}
```

In this example:

- int age = 30; is a variable definition. It declares an integer variable named age, allocates memory for it, and initializes its value to 30.
- float price; is also a variable definition. It declares a floating-point variable named price and allocates memory. At this point, price might hold a garbage value.
- price = 99.99; is an assignment operation, giving price a specific value after its definition.

2) what is different data types in c programme
C programming language categorizes data types into several groups to handle different kinds of data. These categories include:
1. Basic (or Primitive) Data Types: These are the fundamental building blocks.

- int: Used to store whole numbers (integers). Can be modified with short, long, signed, and unsigned to control range and memory usage.
    - Example: int age = 30;

- char: Used to store single characters. Can also be signed or unsigned.
    - Example: char initial = 'J';

- float: Used to store single-precision floating-point numbers (numbers with decimal points).
  - Example: float price = 19.99f;

- double: Used to store double-precision floating-point numbers, offering higher precision and range than float.
  - Example: double pi = 3.1415926535;

- _Bool (or bool in C99 and later with &lt;stdbool.h&gt;): Used to store Boolean values (true or false).
  - Example: bool is_active = true;

2. Derived Data Types: These are built upon the basic data types.

- Arrays: Collections of elements of the same data type.
  - Example: int numbers[5] = {1, 2, 3, 4, 5};

- Pointers: Variables that store memory addresses of other variables.
  - Example: int *ptr;

- Functions: Blocks of code designed to perform a specific task. While not a data type in the traditional sense, function types define the return type and parameters.

3. User-Defined Data Types: These allow programmers to create custom data types.

- struct (Structures): Allow grouping of different data types under a single name.
  - Example:

```
struct Person {
    char name[50];
    int age;
};
```

- union (Unions): Similar to structures, but all members share the same memory location.
  - Example:

```
union Data {
    int i;
    float f;
};
```

- enum (Enumerations): Assign symbolic names to integer constants, improving readability.
  - Example: enum Day {MONDAY, TUESDAY, WEDNESDAY};

4. void Data Type:

- void: Indicates the absence of a value. It is commonly used as a function's return type when it doesn't return any value, or as a pointer type that can point to any data type (a generic pointer).
    - Example: void print_message(); or void *generic_ptr;

3) what is format specfiers

format specifier is a placeholder in a string used by functions like printf() and scanf() to specify the data type and format of data being read or displayed. These specifiers, which start with a percent sign (%), tell the compiler how to interpret and format variables, such as integers (%d), floating-point numbers (%f), and strings (%s).

Key functions and examples

printf() (Output): Used to print formatted data to the console.

printf("%d", my_integer); prints the value of the variable my_integer as a decimal integer.

printf("Hello, %s!", my_name); prints a string.

scanf() (Input): Used to read formatted data from the user.

scanf("%d", &my_integer); reads a decimal integer from the input and stores it in my_integer.

Common format specifiers

Specifier	Data Type
%d	Signed decimal integer
%i	Signed decimal integer (similar to %d)
%f	Floating-point number (float)
%lf	Floating-point number (double)
%c	Single character
%s	String (character array)
%u	Unsigned integer
%x	Unsigned hexadecimal integer (lowercase letters)
%X	Unsigned hexadecimal integer (uppercase letters)
%o	Unsigned octal integer