CONSTANTS, VARIABLES, KEYWORDS, VARIABLES, TOKENS, DATA TYPES, ……

IN

C

MOHD ADIL

ASSISTANT PROFESSOR

CHARACTER SET

- The characters that can be used to form words, numbers and expressions

The characters in C are:

- Letters
- Digits
- Special characters
- White spaces

CHARACTER SET

Letters	Digits
• Uppercase A.....Z	• All decimal digits 0.....9
• Lowercase a.....z	

, comma	" quotation mark	_ underscore
. period	! exclamation mark	$ dollar sign
; semicolon	| vertical bar	% percent sign
: colon	/ slash	& ampersand
? question mark	\ backslash	^ caret
' apostrophe	~ tilde	* asterisk

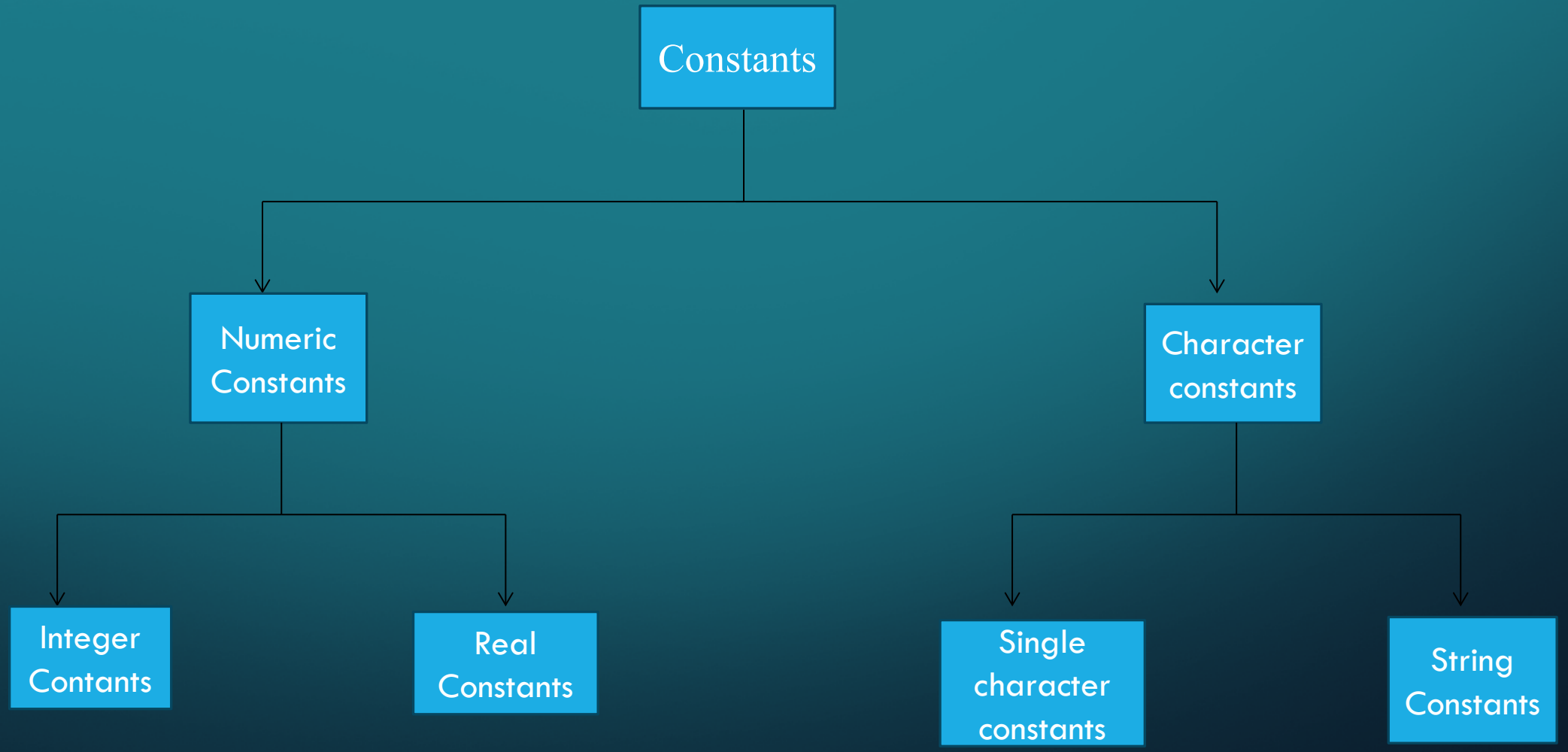
CONSTANTS

- A **constant** is something that does **not change** and remains **fixed or stable** over time
- Constants refer to fixed values that do not change during the execution of program.
- Constants are also called literals

Example:

- Number of days in a week (**7 days**)

- **C language has several types of constants**



Integer constant

1. Integer constants refer to a sequence of digits.
2. There are following three types of integers:-

I. **Decimal Integers** –

- Decimal integer consists of a set of digits , 0 through 9, preceded by an optional - or + sign
- Examples :- 123 -321 0 654321 +78
- Embedded spaces, commas, and non-digit characters are not permitted between digits.
- For example, 18 750 20,000 $1000 are illegal numbers.

II. **Octal Integers-**

- An octal integer constant consists of any combination of digits from the 0 through 7 , with leading 0. Some example of octal integer are: 037 0 0435 0551

III. Hexadecimal Integer –

- A sequence of digits preceded by 0x or OX is considered as hexadecimal integers.
- They may also include alphabets A through F or a through f, the letter A through F represent the numbers 10 through 15.
- Examples:- 0X2 0x9F 0Xbcd 0x

REAL CONSTANT

INTEGER NUMBER ARE INADEQUATE TO REPRESENT QUANTITES THAT VERY CONTINUOUSLY ,SUCH AS DISTANCES, HEIGHTS,TEMPERTURES,PRICES AND SOON .. SUCH NUMBERS ARE CALLED REAL CONSTANTS ..

Ex=> 0.0083 -0.73 435.36 +247.0

The mantissa is either a real number expressed in exponential notation an integer. The letter separating the mantissa and the exponent can be written in either lowercase or upper case .

Ex=> 0.65e4 12e-2 1.5e+5

Example of numeric Constants

Constant	Vaild?	Remarks
698	yes	Represent integer
25,000	no	Comma is not allowed
+5.0E3	yes	C support unary plus
3.5e-5	yes	
7.1e 4	no	No white space is permitted
-4.5e-2	yes	
1.5E+2.5	no	Exponent must be an integer
$255	no	$symbol is not permitted
0X7B	yes	Hexadecimal integer

Single Character Constant

- A single character constant (or simply character constant) contains a single character enclosed within a pair of single quote marks.
- Ex=> ’5’ ‘X’ ‘;’ ‘ ‘
- Note that character constant ‘5’ is not the same as the number 5

String Constants

- A string constant is aa sequence of characters enclosed in ***double quotes.***
- The characters may be letters, numbers, special character and blank space .
- ***Example are:***
- “Hello!” “1987” “Well done” “?.....!” “5+3” “X”

- Note that a character constant (e.g., ‘X’) is not equivalent to the single character string constant (e.g., “X”).

Backslash Character Constants

- C language supports some special backslash character constants that are used in output functions.
- ***Example :***
- '\n' stands for newline character.
- '\0' stands for null character.

- A list of such backslash character constants is given in table

Constant	Meaning
'\a'	Audible alert
'\b'	Back space
'\f'	Form feed
'\n'	New line
'\r'	Carriage return
'\t'	Horizontal tab
'\v'	Vertical tab
'\''	Single quote
'\"'	Double quote
'\?'	Question mark
'\\'	backslash
'\0'	null

These characters combinations are known as escape sequences.

Keywords in C

auto	double	int	struct
break	else	long	switch
case	enum	register	typedef
char	extern	return	union
const	float	short	void
continue	for	signed	unsigned
default	goto	sizeof	volatile
do	if	static	while

In c there are 32 keywords

Variables

- Variables are the names of memory locations where we store data.
- In simple terms variable are name that points to some memory locations.

Declare variables

- **Declaration:** announcing the properties of variable to the compiler.

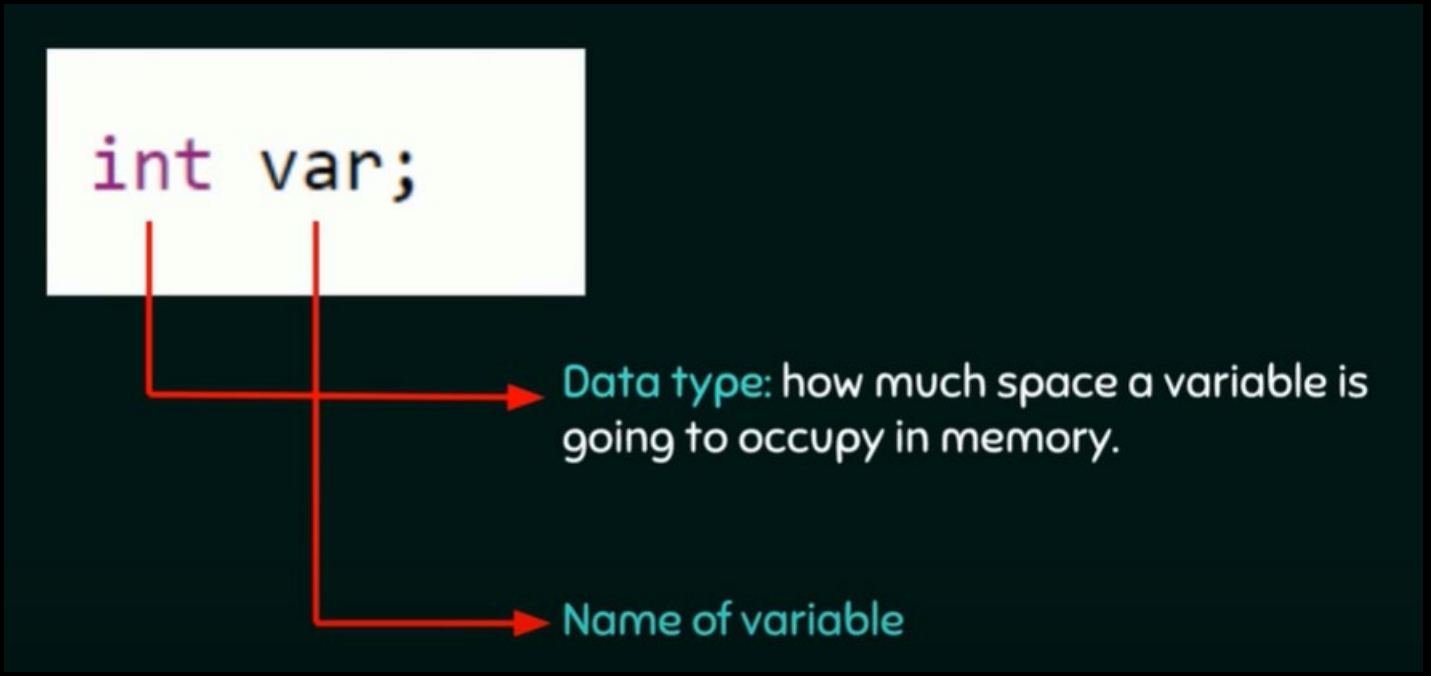
- **Properties**
- **Size of the variable**
- **Name of the variable**

```
int var;
```

Variables

Definition of variable

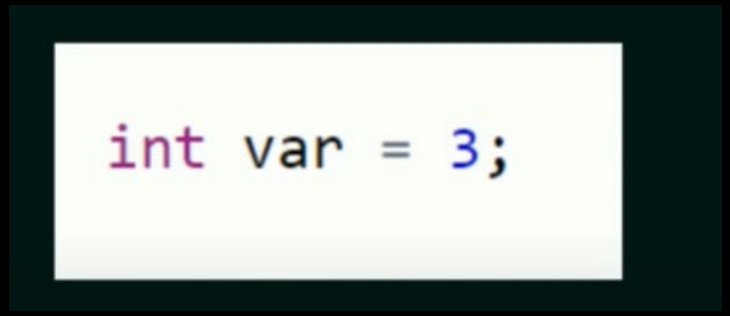
Definition: allocating memory to a variable.



Variables

Initialization of variable

initializing: assigning some constants value to a variable.



Rules for giving name of variable

Rule 1: don’t start variable name with digits.

example:

Bca **1Bca**

Rules for giving name of variable

Rule 2: beginning with underscore is valid but not recommended.

-(underscore) is treated as letter

example:

_Bca

Rules for giving name of variable

Rule 3: C language is case sensitive. Uppercase letters are different from lowercase letters.

example:

Var, var, vAr, vaR, vAR, VAR all are different.

Rules for giving name of variable

Rule 4: Special characters(@,#,%, ^, &, …) not allowed in the name of variable.

example:

%bca ,^bca

Rules for giving name of variable

Rule 5: blank spaces or white spaces not allowed.

example:

Int my var

But

Int my_var is vaild

Rules for giving name of variable

Rule 6: Don’t use keywords to naming the variable

example:

Int, float, char , if, while,

Example of variable

Variable name	Valid?	Remark
First_tag	valid	
char	Not valid	Char is a keyword
Price$	Not valid	Dollor sign is illegal
Average_number	valid	
Group one	Not valid	Blank space is not permitted

Identifiers

- Identifiers: in C can be composed of letters, digits, and the underscore character.
- Sequence of characters that make up a name.
- An identifier should give some indication of its meaning.
- A identifiers can be variable name function name
- array name

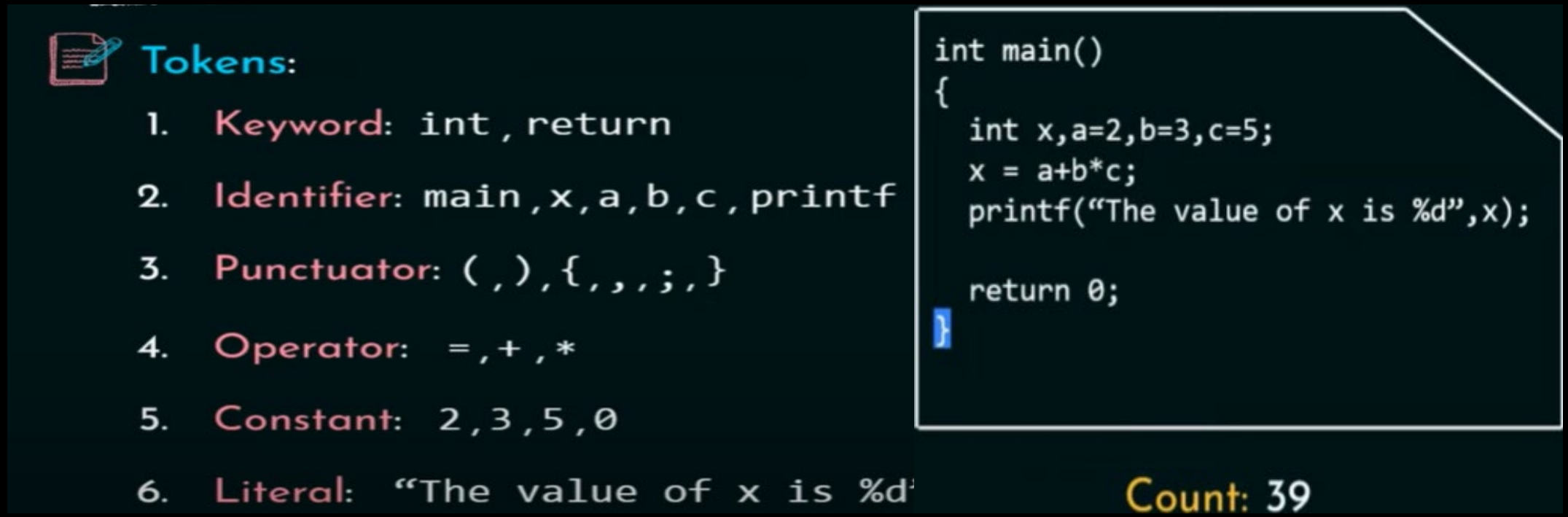
```
int a1;

int A1;

float _b3;
```

TOKENS

- In a c program the smallest individual units are known as c tokens.
- C has six type of tokens

1. Keyword
2. Constants
3. String
4. Operator
5. Identifiers
6. Special Symbols

TOKENS


Tokens:
1. Keyword: int , return
2. Identifier: main , x , a , b , c , printf
3. Punctuator: (,) , { , , , ; , }
4. Operator: = , + , *
5. Constant: 2 , 3 , 5 , 0
6. Literal: “The value of x is %d
int main()
{
int x,a=2,b=3,c=5;
x = a+b*c;
printf(“The value of x is %d”,x);
return 0;
}
Count: 39

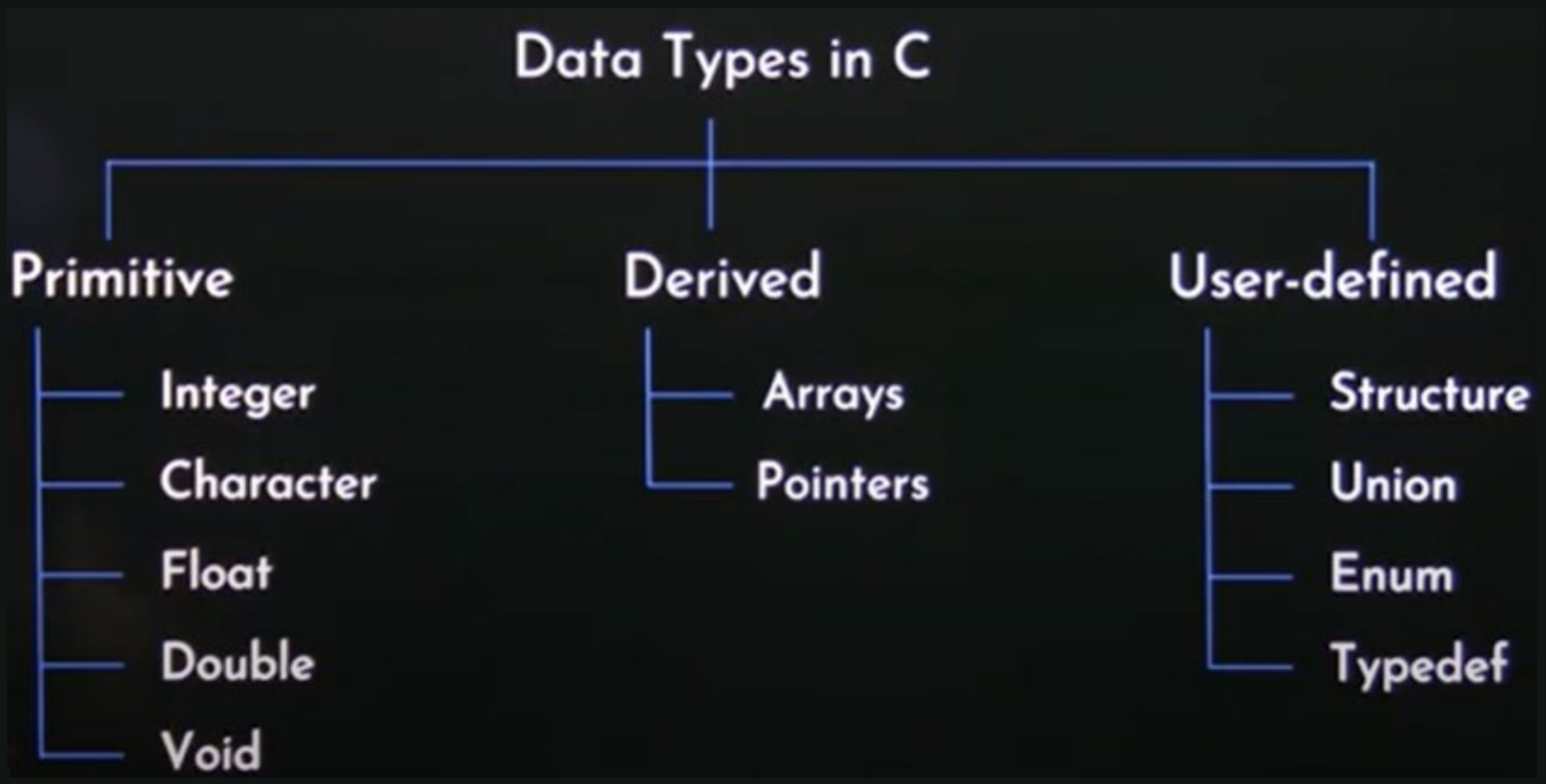
DATA TYPES IN C

- A data type specifies the kind of data a variable can hold or store.

Importance of Data types

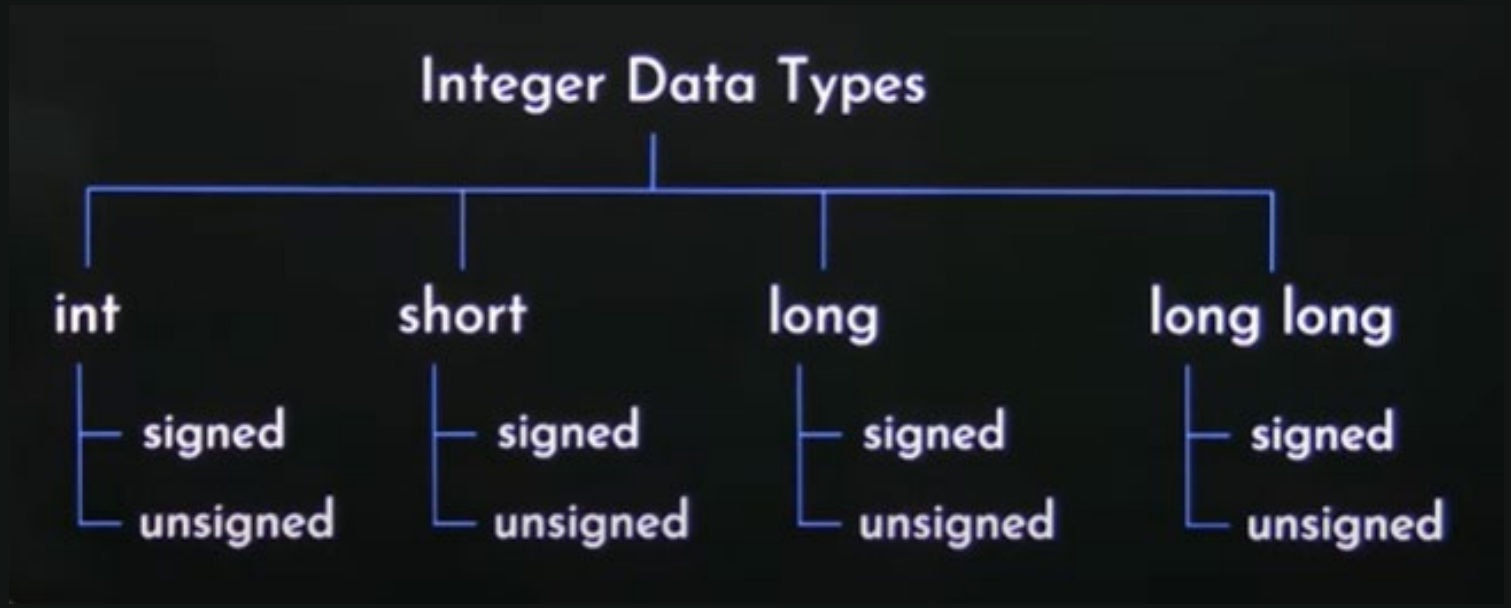
- **Memory Allocation: Different data require different amounts of memory.**
- **Knowing the data type of a variable helps the compiler allocate the correct amount of memory.**
- **Operations: A data type tells the set of operations that can be performed on data.**
- **Type Safety: using data types ensure only expected data types being manipulated.**

DATA TYPES IN C



INTEGER DATA TYPES IN C

- **Represents integers in decimal, octal, and hexadecimal format.**



UNSIGNED INTEGER TYPES

- By default, **integer data types** are **signed** means they can represent both **negative** and **positive numbers**
- **Unsigned integer data types** represents **positive values** only.

Unsigned types	size	Range
Unsigned int	4 bytes/32bits	0 to 4294967295
Unsigned short	2 bytes/16bits	0 to 65535
Unsigned long	4 bytes/32 bits	0 to 4294967295
Unsigned long long	8 bytes/64bits	0 to 2^{64}-1

- The formula to calculate the range is: 2^n-1 where n is size of data type in bits.

SIGNED INTEGER TYPES

- **By default, integer data types are signed means they can represent both negative and positive numbers.**

signed types	size	Range
int	4 bytes/32bits	-2147483648 to +2147483647
short	2 bytes/16bits	-32768 to +32767
long	4 bytes/32 bits	-2147483648 to +2147483647
long long	8 bytes/64bits	-2^{63} to $+2^{63}-1$

- **The formula to calculate the signed range is: -2^n-1 to $+(2^{n-1}$ -1) where n is size of data type in bits.**
- **Note: I have not written the signed keyword before the data type because by default they are signed.**

FLOAT DATA TYPE

- Used to represent a **decimal** or an **exponential** number.
- It represented by **float.**
- **It provides** 6 digits of precision**.**

Data type	size	Range
float	4 bytes	1.2E-38 to 3.4E+38

DOUBLE DATA TYPE

- Used to represent a **decimal** or an **exponential** number.
- It represented by **double.**
- **It provides 14** digits of precision.

Data type	size	Range
double	8 bytes	1.7E-308 to 1.7E+308

CHARACTER DATA TYPE

- It used to represent a single character.
- It represented by data type char.
- Size is typically 1 byte (8 bits).
- It can be signed or unsigned

Data type	size	Range
char	1 bytes	-128 to +127
signed char	1 byte	-128 to +127
Unsigned char	1 byte	0 to 255

VOID DATA TYPE

- It indicates absence of something
- E.g. absence of a return value , absence of parameters, etc.
- void as the return type of a function indicates that the function will not return a value.

FORMAT SPECIFIERS

- **Format specifiers are special codes used to display or receive different types of data.**

```c
#include <stdio.h>

int main()
{
    int age;
    printf("Enter your age: ");
    scanf("%d",&age);
    printf("You are %d years old.", age);
    return 0;
}
return  0;
```

%d tells the scanf function to read an integer.

%d tells the printf functic to display an integer.

COMMONLY USED FORMAT SPECIFIERS

Type	Format Specifier
int	%d ,%i
float	%f and %e
char	%c

%d => integer values

%f => decimal values

%e => exponential

%c => character

COMMONLY USED FORMAT SPECIFIERS

Type	Format Specifier
int	%d ,%i
float	%f and %e
char	%c

%d => integer values

%f => decimal values

%e => exponential

%c => character

There is no difference between %d and %i when used in printf

COMMONLY USED FORMAT SPECIFIERS

Type	Format Specifier
int	%d ,%i
float	%f and %e
char	%c

%d => integer values

%f => decimal values

%e => exponential

%c => character

There is no difference between %d and %i when used in printf

ACCEPTING INTEGERS

```
#include<stdio.h>

int main(){

Int a ,b;

printf("enter integers number:");

Scanf("%i %i , &a, &b);

printf("user input is %d and %d", a, b);

return 0;
```

Output:
Enter integer number: 0786 0xf54
User input is 62 3924

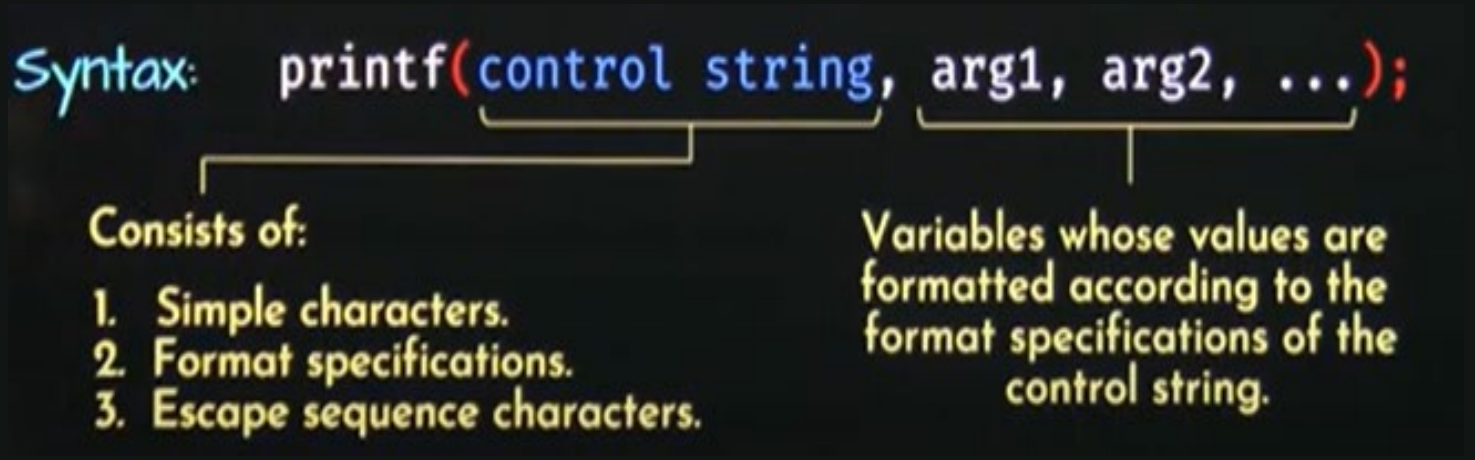
ACCEPTING INTEGERS

So in the previous code

- **%d in scanf accepts only decimal numbers.**
- **%i in scan auto-detects the base: decimal,octal (0….) or hexadecimal (0x….)**
- **Inside the variable the value is always stored as an integer in decimal form.**
- **%d in printf always prints the value in decimal format.**
- **So, reading with %i and printing with %d converts octal/hex inputs into their decimal equivalents.**
- **Example input 076 (octal)-> output 62 decimal**

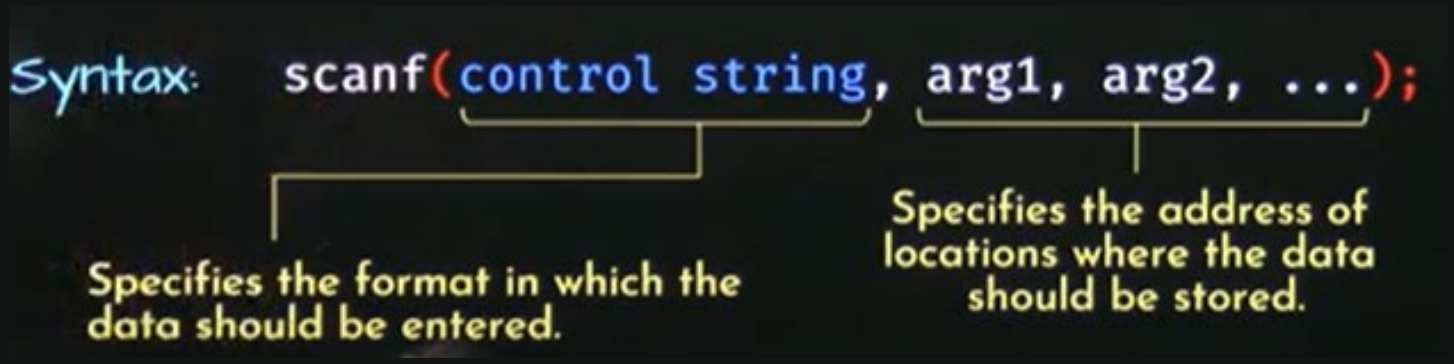
PRINTF FUNCTION

- **It's a predefined function**
- **Allows printing text and values to the console.**
- **Declaration available in stdio.h header file.**

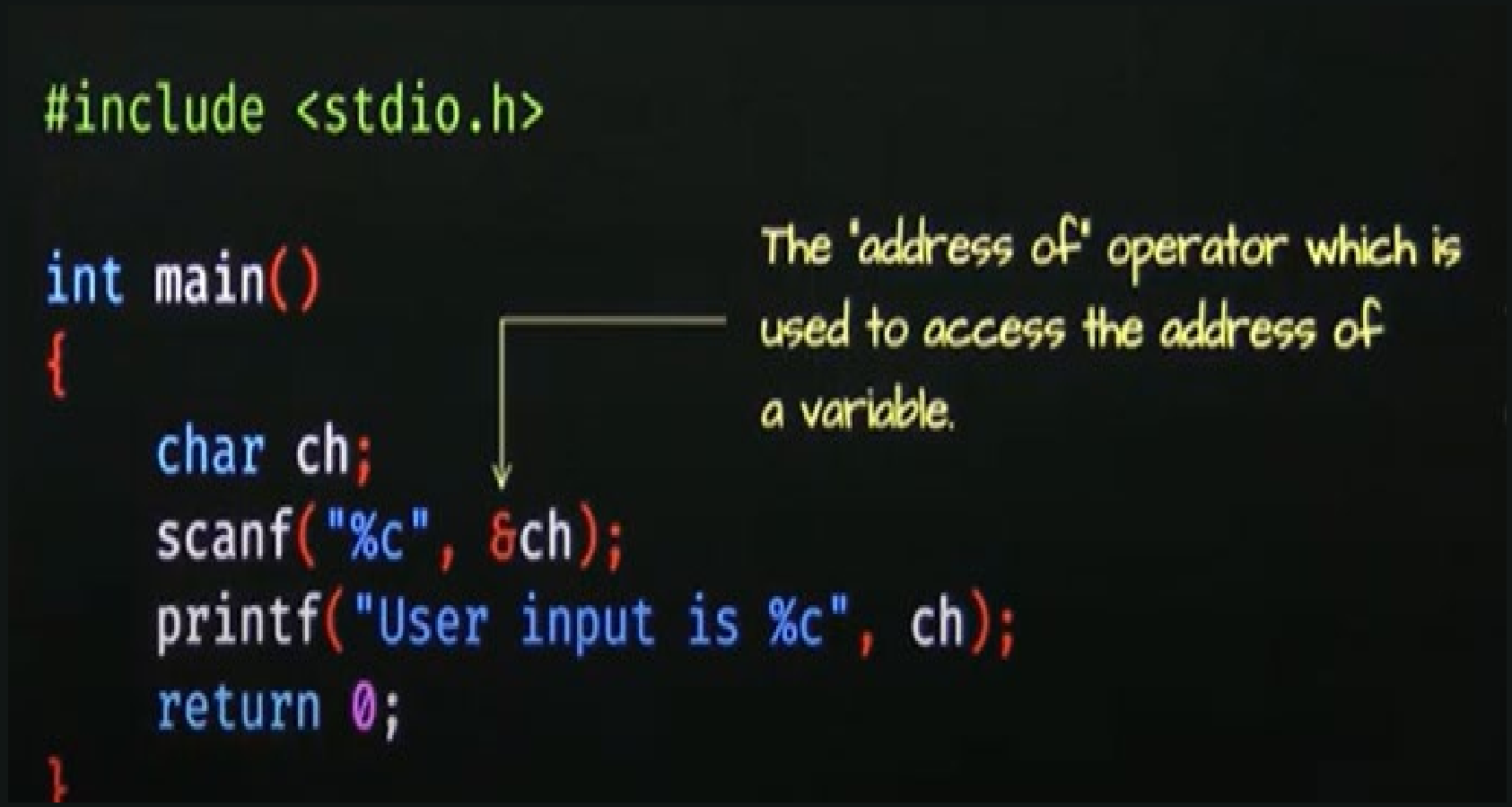


SCANF FUNCTION

- It’s a predefined function
- Receives data arranged in a specific format.
- Declaration available in stdio.h header file.
- Example format 15.76 10 Alice.

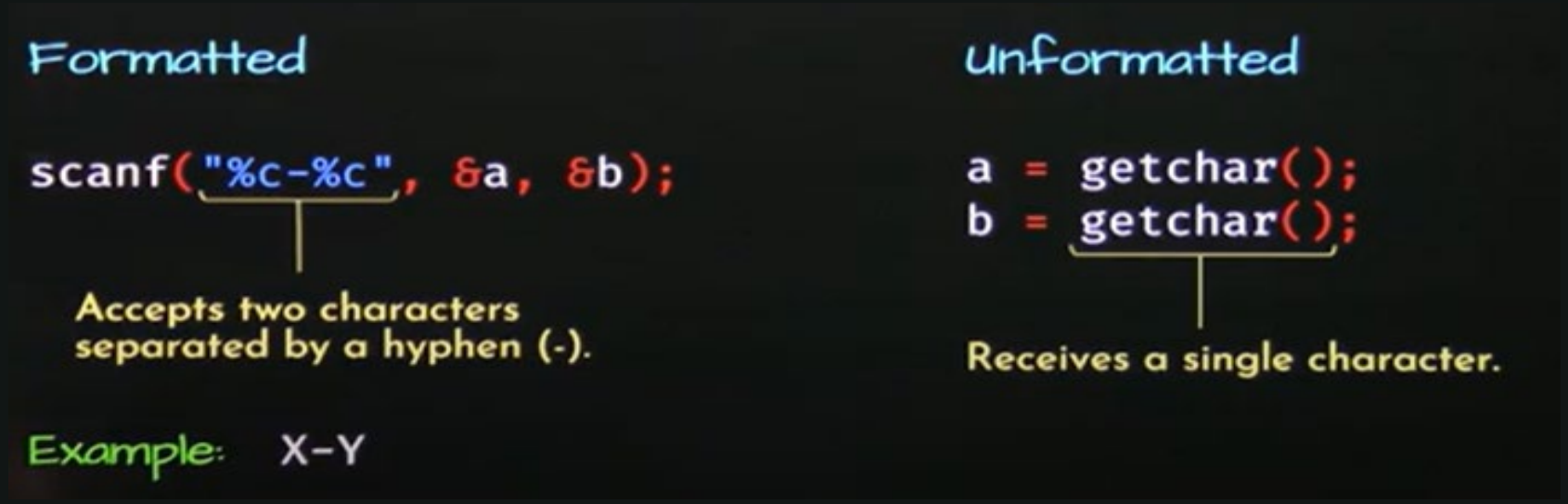


SCANF FUNCTION



FORMATTED VS UNFORMATTED INPUT

- **Scanf stands for scan formatted.**
- **Capable to receive input in a specific format.**



Operators And Expressions

- An operator is a symbol that tells the computer to perform certain mathematical or logical manipulations.
- Operators are used in programs to manipulate data and variables.
- Operator perform action on the operands which are the data items and variables that take part in the operation.

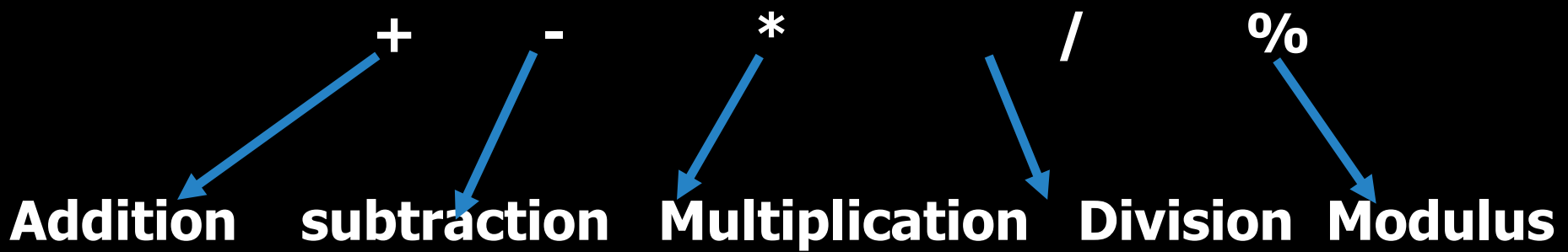
TYPES OF OPERATORS IN C

Name of operators	operators
Arithmetic operators	+ , - , * , / , %
Increment/Decrement	++, --
Relation operators	== , != , <= , >= , <, >
Logical operators	&& , || , !
Bitwise operators	& , ^ , | , ~ , >> , <<
Assignment operators	=, += ,*=, /= ,%=, <<= ,>>= ,&= ,^= |=
Other operators	?: & * sizeof() ,

TYPES OF OPERATORS IN C

Name of operators	operators
Arithmetic operators	+ , - , * , / , %
Increment/Decrement	++, --
Relation operators	== , != , <= , >= , <, >
Logical operators	&& , || , !
Bitwise operators	& , ^ , | , ~ , >> , <<
Assignment operators	=, += ,*=, /= ,%=, <<= ,>>= ,&= ,^= |=
Other operators	?: & * sizeof() ,

ARITHMETIC OPERATORS



All are binary operator => means two operands are required to perform operation

For example: A + B

op1 op2