Data types. Type of data, 'C' support various data types. each type may have predefined memory requirement, operators, variables, Storage representation.

① Primary — const, int, float, double

② Userdefined — typedef, enum.

③ Derived — arrays, pointer, structures, union

④ Empty — void → has no value
→ doesn't return any value

① Primary Datatype.

int - 2bytes (-32768 to 32767) 16-bit compiler — 16th bit - sign indication. %d or %i
32-bit compiler -2147483648 to 2147483647

char - 1byte (-128 to 127) %c Ex: char s = 'n';

float - 4bytes (3.4E-38 to 3.4E+38) %f or %g Ex: float f = 29.77;

double - 8bytes (1.7E-308 to 1.7E+308) %lf Ex: double d = 29107511 14

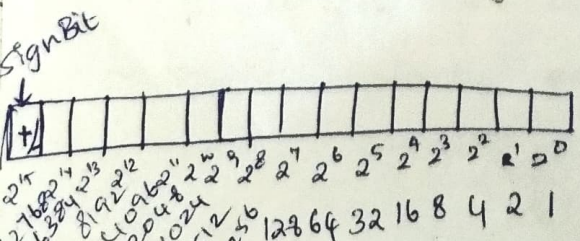
Primary Datatype is dirided into:

Integer — signed / unsigned
[int, short int, long int]

char — char / signed char (-128 to 127) / unsigned char (0 and 255)

Float — float / double / long double

Refer Balaguruswamy Book for reference to various Datatypes



$2^0 = 1$ (0,1)

$2^{15} = 32,768$ (signed)

$2^{16} = 65536$ (unsigned)

0-00 1-01 2-10 3-11

0-000 1- 2- 3- 4- 5- 6- 7-

Operators & Expressions.

Operators are used in programs to manipulate data & variables. (Mathematical / logical manipulations).

* Data item that operators act upon are called operands.

* Unary operator, Binary operator.

Ex: a+b

a, b - operands

+ - operator.

* Types.

(i) Arithmetic
(ii) Relational
(iii) Logical
(iv) Assignment
(v) Increment & Decrement
(vi) Conditional
(vii) Bitwise
(viii) Special

(1) Arithmetic operator.

+	Addition / Unary plus	2+3 = 5
-	Subtraction / Unary Minus	3-2 = 1
*	Multiplication	3*2 = 6
/	Division	6/3 = 2
%	Modulo Division	7%3 = 1

* Classification:

① Unary arithmetic - +x, -y

② Binary arithmetic - x+y

③ Integer arithmetic - a = 5, b = 4

2 operands should be integers.

Ex: a/b = 5/4 = 1

Here the decimal part is truncated.

④ Real arithmetic - operands are real.

Ex: x = 6.0/7.0 = 0.857143.

Mixed-mode Arithmetic: one operand is real & other integer. If any of the operand is real, result is real

Ex: 2/5 = 0

Involves Real as 1 operand:
- 5.0/2 = 2.5
- 5/2.0 = 2.5
- 5.0/2.0 = 2.5

→ Result will be Real.

Example.

```
#include <stdio.h>
#include <conio.h>
Void main()
{
    /*Local definitions*/
    int a,b,c,d;
    int sum, sub, mul, rem;
    float div;
    /*stmts*/
    clrscr();
    printf("Enter values of b,c,d:");
    scanf("%d %d %d", &b, &c, &d);
    Sum = b+c;
    Sub = b-c;
    mul = b*c;
    div = b/c;
    rem = b%d;
    Printf("\n Sum = %d, Sub = %d, mul = %d, div = %f",
                    Sum, Sub, mul, div);
    getch();
}
```

Output: Enter values of b, c, d: 3, 5, 10

Sum = 8 Sub = -2 mul = 15 div = 0.3

3/10

Relational Operators.

Used to compare 2/more operands. operands - variables, constants or expressions.

Ex: Compare age of 2 persons.

			RESULT
<	less than	2 < 9	1 (Return value;
>	Greater than	2 > 9	0 False
<=	less than or equal	2 <= 2	1 True
>=	Greater than or equal to	2 >= 3	0
==	Equal to	2 == 3	0
!=	Not Equal to	2 != 3	1

Relational operator complements:

> is complement of <=

< is complement of >=

== is complement of !=

Ex: ! (x < y) ⇒ x >= y.

! (x > y) ⇒ x <= y.

③ Logical operators.

* Used when we want to test more than one cndtn & take a decision.

&& logical AND (exp1) && (exp2).

|| logical OR ! (exp) ⇒ if(! ($c_1 < c_2$))

! logical NOT.

(A) Assignment operator.

* Assign a value to an variable, value of a variable to another variable.

Syntax: Variable = expression (or) value;

Example: x = 10;

y = a+b;

(a) Compound Assignment - assign a value to a variable

Shorthand operator

x += y	x = x+y
x -= y	x = x-y.

(b) Nested / Multiple Assignments.

Var1 = Var2 = ... = Varn = Single Variable or expression;

i = j = k = 1;

x = y = z = (i+j+k);

(5) Increment & Decrement operators (Unary Operators).

++ → Add 1 to variable.

-- → Sub 1 from variable

Hence it is called

++x	pre increment	
--x	pre decrement	
x++	post increment	11, 11, 10, 10
y--	post decrement.	

Example: a = 10

		Now value of a is
a++	10	10+1 = 11
a--	11	11-1 = 10
--a	9	10-1 = 9
++a	10	9+1 = 10

Conditional / Ternary operator.

checks the condition & executes the statement depending on the condition.

Syntax: condition ? exp1 : exp2; ← ternary operator

↓

True exp1 is evaluated

False exp2 is evaluated.

```
Example: main()
{
    int a=5, b=3, big;
    big = a>b ? a : b;
    printf("Big is ... %d", big);
}
```

O/P : Big is ... 5

④ Bitwise operator. used to manipulate data at bit level. It operates on integers only. It may not be applied to float/real

a = 5 & 6;

a = 4

5 - 0101

6 - 0110

0 1 0 0

8 4 2 1

&	Bitwise AND	5 & 6 → 0100
|	Bitwise OR	
^	Bitwise XOR	
<<	Shift left	a = 7; a << 2. → 10000
>>	Shift right.	14 a >> 2
~	one's complement.	

7 << 1 → 1110

a << 2 → 10000

a >> 2

Example: Bitwise AND (&).

x = 7 = 0000 0111

y = 8 = 0000 1000

x & y = 0000 0000.

&	0	1
0	0	0
1	0	1

Bitwise OR

|	0	1
0	0	1
1	1	1

Bitwise XOR

^	0	1
0	0	1
1	1	0

x	y	x^y
0	0	0
0	1	1
1	0	1
1	1	0

1000 ^ 0100 → 1100

⑧ Special operator.

Comma operators ,

Sizeof operators sizeof

Address of variable ← & * → value of a variable

pointer operators

. --> Member selection operator.

↳ Access the elements from the structure.

(i) Comma operator. used to seperate the stmt elements such as variables, constants or expression.

Link the related expressions together.

Ex: Val=(a = 3, b = 9, c = 77, a+c); 3 is assigned to a

9 is assigned to b

77 is assigned to c

a+c = 80.

(ii) Sizeof() is a unary operator, returns length in bytes of the specified variable. Helps in finding the bytes of a variable.

Syntax: sizeof(var);

Example:

```
main()
{
    int a;
    printf("size of a is %d", sizeof(a));
}
```

<u>O/P</u>: size of a is 2.