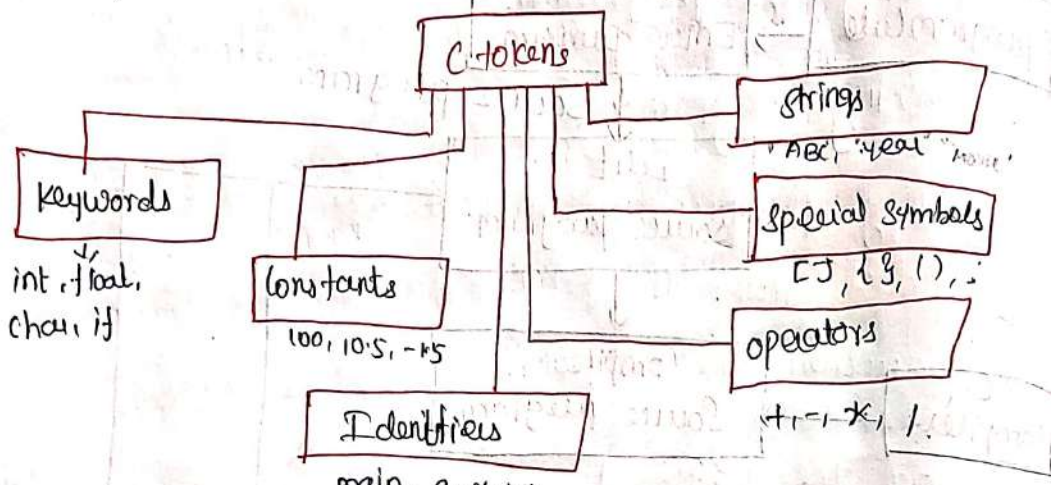
Constants, Variables, Keywords, Identifiers, Delimiters:

C - tokens:

↳ Smallest individual units are known as C tokens.

6 - types of tokens:



Keyword:

Example:

```
int main ()
{
printf ("Hello word");
return 0;
}
```

Keyword:

1. Predefined reserved word.
2. Fixed Meaning
3. Can not be used or identified
4. 32 keywords in C
5. lower case.

Difference:

Keyword:	Identifier:
→ predefined.	→ user defined
→ lower case	→ both lowercase & uppercase
→ Alphabetical characters	→ Alpha numeric char & underscore.

- bool.
- complex.

Identifier:

→ name of variables, functions & arrays.

→ These are user-defined names & consist of a sequence of letter and digits, with a letter as the 1st char.

→ Both upper & lower case letters are permitted.

Rules for identifier:

→ 1st char. must be an alphabet (or underscore)

→ It must consist only letters, digits or underscore.

→ only 1st 31 characters are significant in an identifier.

→ It can't be same as a keyword.

→ It must not contain white space.

→ case sensitive

Constant

Is fixed values that do not change during the execution of a pgm.

C- Support several Constants.

1. Numerical Constant — only number
   - Integer
   - Real
2. Character Constant
   - Single char.
   - String.

Integer Constant:

An integer constant refers to a sequence of digits.

There are 3 types:

1. Decimal integer → 10_{10} → 0 to 9
   ↳ no spaces, commas, non digit chars not permitted.
2. octal integer → 8 - 0 to 7 → 023, 0548.
3. Hexa decimal integer → 0 to 9 0X 72
   A to F 0X 7A.

Valid	invalid
54 00	5400x
20 343	20343x
+627	$627 x
- 267	£ 627x

Y: 0 → octal
Y. X → hexa decimal
Y. X

Real Constant:

It represent real numbers (or) points

1.23 , 4.567.

1.2345 ⇒ 1.2345 x 10^2
1.2345 E2
1.2345 e2

Valid	Invalid
+1.23	1.23e.4
-1.23	1.23e 4.5
4.2E2	1.23e.5
4.5e+2	
4.5E -2	

Single character constants:

→ Simply char. constant.

→ A single char. enclose within a pair of "Single quote Marks". (' ').

Ex: '5', 'x'

char c = 'x';

char e = '8';

String constants:

→ A sequence of char. enclosed in doub quote (" ").

→ These char may be letters, numbers, special char. & blank spaces.

Ex: "Hello", "1987" "well done"

Backslash Char: (o/p fun).

Constant	Meaning:
'\a'	→ audible alert (bell)
'\b'	→ back space.
'\f'	→ form feed.
'\n'	→ newline
'\r'	→ carriage return
'\t'	→ horizontal tab
'\v'	→ vertical tab
'\'	→ single quote

\" → double quote → printf("\"Deepika\"");

\\ → backslash.

\0 → null.

Variables: → named memory location.

→ A data name is used for storing a data value. Its value can be changed during the execution of a pgm. In other words, a variable can be assigned diff. values at diff. times during the execution of a pgm.

datatype variable_name;

int a, b, c

a, b, c → variable list

Ex:

data type ← int a; → variable

float b;

char, double.

Example of valid variable:

John value T_raise

Mark sum1 distance

invalid: 123, (area), %a, 25th

Rules for declaring variables:

→ They must begin with a "character" without space.

→ They can also begin with "underscore" (_)

→ The variable may be combination of "uppercase" and "lowercase" char. eg sum, SUM

→ The variable should not be C keyword.

Ex: of variable:

height, average, _value, total_amount

Initialization of Variable:

int num = 100;

float Pi = 3.14

int a=5, b=6, c=7;

Declaration of variable:

Datatype: Variablename;

int a;

Data type var_name1, var_name2, var_name3.... ;

int a, b, c;

Delimiters:

C - uses special kind of symbols which are called Delimiters.

They are given

colon : → used to define a label

semi colon ; → used as the end of the statement

parentheses ( ) → used in expressions and fun.

square Braces [ ] → used to declare an array

Curly Braces { } → used to provide the scope of a set of statements

Hash # → pre-processor directive.

Comma , → variable separator.

void main()
{ int a,b ,x=10, y=

Eg:

```
#include <stdio.h>
#include <conio.h>
void main ()
{
    clrscr ();
    print ("print two names : Ram, Sita");
    getch ();
}
```

Ib :→ Printf (" Hellow \b world\b . (Hellow world).

II → current line starting position: → Print f (" Hellow word \r world ");
o/p: world.

III → (space) . 8-space.
Print f (" Hellow\t world"); → o/p: Hello ...... world.
(" Hello \n \t \t o

IV → cursor odd same posistion: (Print f : (Hello \v world).
Hello
world.