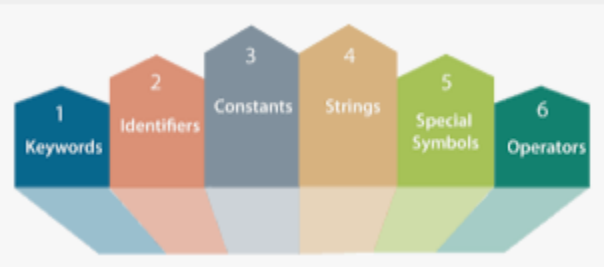
Variables and Constants: Character Set, Identifiers and Keywords, Rules for Forming Identifiers, Qualifiers, Variables, Declaring Variables, Initializing Variables, Constants, Types of Constants

==========

C tokens-

C Tokens are the smallest building block or smallest unit of a **C program**. The compiler breaks a **program** into the smallest possible units called a token. It includes keywords, identifiers, constants, strings, operators.



==========

Character Set

character:- It denotes any alphabet, digit or special symbol used to represent information.

Use:- These characters can be combined to form variables. C uses constants, variables, operators, keywords and expressions as building blocks to form a basic C program.

Character set:- The character set is the fundamental raw material of any language and they are used to represent information. Like natural languages, computer language will also have well defined character set, which is useful to build the programs.

==========

The characters in C are grouped into the following **two** categories:

1. **Source character set**
    a. Alphabets

b. Digits

c. Special Characters

d. White Spaces

Execution character set

a. Escape Sequence

==========

Source character set

ALPHABETS

Uppercase letters	A-Z
Lowercase letters	a-z

DIGITS 0, 1, 2, 3, 4, 5, 6, 7, 8, 9

SPECIAL CHARACTERS

~	tilde	%	percent sign	|	vertical bar	@	at symbol	+	plus sign	<	less than
_	underscore	-	minus sign	>	greater than	^	caret	#	number sign	=	equal to
&	ampersand	$	dollar sign	/	slash	(	left parenthesis	*	asterisk	\	back slash

WHITESPACE CHARACTERS

\b	blank space	\t	horizontal tab	\v	vertical tab	\r	carriage return	\f	form feed	\n	new line

==========

Execution character set: Escape Sequence

Certain ASCII characters are unprintable, which means they are not displayed on the screen or printer. Those characters perform other functions aside from displaying text. Examples are backspacing, moving to a newline, or ringing a bell.

They are used in output statements. Escape sequence usually consists of a backslash and a letter or a combination of digits. An escape sequence is considered as a single character but a valid character constant.

Backslash character constants

Character	ASCII value	Escape Sequence	Result
Null	000	\0	Null
Alarm (bell)	007	\a	Beep Sound
Back space	008	\b	Moves previous position
Horizontal tab	009	\t	Moves next horizontal tab
New line	010	\n	Moves next Line

==========

Identifiers

C identifiers represent the name in the C program, for example, variables, functions, arrays, structures, unions, labels, etc. An identifier can be composed of letters such as uppercase, lowercase letters, underscore, digits, but the starting letter should be either an alphabet or an underscore. If the identifier is not used in the external linkage, then it is called as an internal identifier. If the identifier is used in the external linkage, then it is called as an external identifier.

We can say that an identifier is a collection of alphanumeric characters that begins either with an alphabetical character or an underscore, which are used to represent various programming elements such as variables, functions, arrays, structures, unions, labels, etc. There are 52 alphabetical characters (uppercase and lowercase), underscore character, and ten numerical digits (0-9) that represent the identifiers. There is a total of 63 alphanumerical characters that represent the identifiers.

==========

Rules for constructing C identifiers

- The first character of an identifier should be either an alphabet or an underscore, and then it can be followed by any of the character, digit, or underscore.
- It should not begin with any numerical digit.

- In identifiers, both uppercase and lowercase letters are distinct. Therefore, we can say that identifiers are case sensitive.
- Commas or blank spaces cannot be specified within an identifier.
- Keywords cannot be represented as an identifier.
- The length of the identifiers should not be more than 31 characters.
- Identifiers should be written in such a way that it is meaningful, short, and easy to read.

Example of valid identifiers

1. total, sum, average, _m _, sum_1, etc.

==========

Keywords

1. Keywords are those words whose meaning is already defined by Compiler
2. Cannot be used as **Variable Name**
3. There are **32 Keywords** in C
4. C Keywords are also called as **Reserved words** .

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

==========

Qualifiers

Qualifiers are keywords which are used to modify the properties of a variable are called qualifiers.

Types of qualifiers:

There are two types of qualifiers available in C language. They are,

1. const
2. volatile

==========

1. const keyword:

- Constants are also like normal variables. But, only difference is, their values can't be modified by the program once they are defined.
- They refer to fixed values. They are also called as literals.
- They may be belonging to any of the data type.

  Syntax:
  const data_type variable_name; (or) const data_type *variable_name;

==========

2. volatile keyword:

- When a variable is defined as volatile, the program may not change the value of the variable explicitly.
- But, these variable values might keep on changing without any explicit assignment by the program. These types of qualifiers are called volatile.
- For example, if global variable's address is passed to clock routine of the operating system to store the system time, the value in this address keep on changing without any assignment by the program. These variables are named as volatile variable.

  Syntax:
  volatile data_type variable_name; (or) volatile data_type *variable_name;

==========

Variables

- C variable is a named location in a memory where a program can manipulate the data. This location is used to hold the value of the variable.
- The value of the C variable may get change in the program.
- C variable might be belonging to any of the data type like int, float, char etc.

Rules for naming C variable:

1. Variable name must begin with letter or underscore.
2. Variables are case sensitive

3. They can be constructed with digits, letters.
4. No special symbols are allowed other than underscore.
5. sum, height, _value are some examples for variable name

==========

Declaring & initializing C variable:

- Variables should be declared in the C program before to use.
- Memory space is not allocated for a variable while declaration. It happens only on variable definition.
- Variable initialization means assigning a value to the variable.

Type	Syntax
Variable declaration	data_type variable_name; Example: int x, y, z; char flat, ch;
Variable initialization	data_type variable_name = value; Example: int x = 50, y = 30; char flag = 'x', ch='l';

==========

There are three types of variables in C program They are,

1. Local variable
2. Global variable
3. Environment variable

program for local variable in C:

- The scope of local variables will be within the function only.
- These variables are declared within the function and can't be accessed outside the function.
- In the below example, m and n variables are having scope within the main function only. These are not visible to test function.
- Like wise, a and b variables are having scope within the test function only. These are not visible to main function.

#include<stdio.h>

```
void test();

int main()
{
  int m = 22, n = 44;
      // m, n are local variables of main function
      /*m and n variables are having scope
      within this main function only.
      These are not visible to test funtion.*/
      /* If you try to access a and b in this function,
      you will get 'a' undeclared and 'b' undeclared error */
  printf("\nvalues : m = %d and n = %d", m, n);
  test();
}

void test()
{
  int a = 50, b = 80;
      // a, b are local variables of test function
      /*a and b variables are having scope
      within this test function only.
      These are not visible to main function.*/
      /* If you try to access m and n in this function,
      you will get 'm' undeclared and 'n' undeclared
      error */
  printf("\nvalues : a = %d and b = %d", a, b);
}
```

==========

Example program for global variable in C:

- The scope of global variables will be throughout the program. These variables can be accessed from anywhere in the program.
- This variable is defined outside the main function. So that, this variable is visible to main function and all other sub functions.

```
#include<stdio.h>
void test();int m = 22, n = 44;
int a = 50, b = 80;

int main()
{
  printf("All variables are accessed from main function");
  printf("\nvalues: m=%d:n=%d:a=%d:b=%d", m,n,a,b);
  test();
}

void test()
{
  printf("\n\nAll variables are accessed from" \
  " test function");
  printf("\nvalues: m=%d:n=%d:a=%d:b=%d", m,n,a,b);
}
```

==========

Environment variables in C:

- Environment variable is a variable that will be available for all C applications and C programs.
- We can access these variables from anywhere in a C program without declaring and initializing in an application or C program.
- The inbuilt functions which are used to access, modify and set these environment variables are called environment functions.

- There are 3 functions which are used to access, modify and assign an environment variable in C. They are,

1. setenv()
2. getenv()
3. putenv()

==========

Constants

- Constants refer to fixed values that the program may not alter during its execution. These fixed values are also called **literals**.
- Constants can be of any of the basic data types like *an integer constant, a floating constant, a character constant, or a string literal*. There are enumeration constants as well.
- Constants are treated just like regular variables except that their values cannot be modified after their definition.

Types: Integer, Float, Character, String

==========

Integer constants- integer valued number

Rules:

- An integer literal can be a decimal, octal, or hexadecimal constant.
- An integer literal can also have a suffix that is a combination of U and L, for unsigned and long, respectively. The suffix can be uppercase or lowercase and can be in any order.
- must have at least one digit.
- Must not have decimal point.
- May be either positive or negative
- No comma, blank space allowed
- Stores 2 bytes of memory

Eg. 0, 1, 455, -763

==========

Floating point constants-

1. A floating-point literal has an integer part, a decimal point, a fractional part, and an exponent part. We can represent floating point literals either in decimal form or exponential form.
2. The signed exponent is introduced by e or E.
3. Must have at least one digit.
4. Must have a decimal point.
5. Either positive or negative , default is positive
6. No commas or blank spaces allowed.
7. Occupies 4 bytes memory space.

Eg. +225.75, 35.00

==========

Character Constants

- Character literals are enclosed in single quotes, e.g., 'x' can be stored in a simple variable of **char** type.
- A character literal can be a plain character (e.g., 'x'), an escape sequence (e.g., '\t').
- There are certain characters in C that represent special meaning when preceded by a backslash for example, newline (\n) or tab (\t).
- Occupies 1 byte

String Literals

String literals or constants are enclosed in double quotes "". A string contains characters that are similar to character literals: plain characters, escape sequences, and universal characters.

You can break a long line into multiple lines using string literals and separating them using white spaces.

Here are some examples of string literals. All the three forms are identical strings.

```
"hello, dear"

"hello, \

dear"

"hello, " "d" "ear"
```

Compiler inserts a NULL or ‘\0’ at the end of every string.