Characters Set in C

Characters: The raw materials used in representation of computer program is called characters.

Character Set is the set of all possible characters used to represent / write a computer program.

C-programming language has a well defined character set to represent information.
The character set of c-programming language includes:

a) Alphabets: A - Z ~~or~~ and a - z
b) Digits: 0 - 9
c) Special symbols: { }, ( ), [ ]. &, ?, $, ;, :, @, +, -, *, %, #, ., !, <, >, _, ^, /, ,, " ", =
d) other symbols: blank space, tab, new line, carriage return.

- The character set is used to create variables, keywords, identifiers and constants.

All characters used in C-language represent ASCII codes. ie, American Standard Code for Information Interchange

- It has 128 characters represented from 0 to 127

Eg: A - Z : 65 - 90 0 - 9 : 48 - 57
a - z : 97 - 12~~7~~2

Date.   No.

Identifiers and KeyWords:

A) Keywords:

- Also known as reserved words or pre-defined words.

- Keywords are those words whose meaning is prefined and it is the basic building block for writing instruction in C- program language.
- There are 32 keywords in C.

Eg: int, float, break, goto, if, for, continue, etc.

- Every keyword has their own meaning.
- Keywords cannot be identifiers, variables, etc.
- Meanings of keywords cannot be change while writing a program.

B): Identifiers:

- Identifiers are user-defined names for functions, variables, labels in the program.

- Identifiers can be letters, numbers or underscore but they cannot have symbols.
- They can also not starts with numbers but can begin with underscore and alphabet.
- Identifiers can't be keywords.
- Length of identifier in ANSI = 31

*) Differences between ~~Ko~~ Identifiers and Keywords:

Identifiers	KeyWords.
Identifiers are user-defined.	Keywords are pre-defined.
It can have both upper case, lowercase, numbers, and underscore.	It can only be written in lower case.
Identifiers can't be keywords.	Keywords can't be identifiers.
Meanings of keywords can't be changed while running a program	Meanings of identifiers can be changed while writing a program.

Datatypes in C:

Datatypes in C is used to define the type of data that you are going to ~~use~~ store in a variable

- It tells us about how much storage/memory to be allocated to a variable.

The datatypes in C are as follows:

Date. No.

Primary datatypes

→ Also known as fundamental or built in datatypes.

Eg: - int
- float
- char
- double
- void

Derived datatypes

- datatypes derived from primary datatypes

Eg: - array
- structure
- union
- pointers.

User-defined datatypes.

- datatypes ~~are~~ that are defined by the user.

Eg:- typedef
- Enumerated datatype

typedef → gives name to datatype

Eg: typedef int Jenny;
Jenny a;

(a): Primary datatypes:

→ It is fundamental or built-in datatypes

(i): int:

- It stores integer values

- Based on qualifiers, int can also be of two types:

⇒ Size qualifiers:
short and long

⇒ Signed qualifier
Signed and unsigned

On a 16-bit machine,
Range of signed int: -32768 to 32767 (default)
Range of unsigned int: 0 to 65535

We know; 1 byte = 8 bits.

short int : stores short space : 1 byte : 8 bits.

int : stores interior : 2 bytes : 16 ~~byter~~ its

long int : stores longes space : $\overset{4}{8}$ bytes : 32 ~~by~~ bits

Given, -2 50, 0, +2100, 88888888, 4,442, -31·8,

Here,

valid integers : invalid integers.

-250, 0, +2100

4,442 - comma use

-31·8 - decimal

-88888888 → out f range

Assigning : int a;

Format specifies ~~: %d~~ for int or signed int = %d

unsigned int = %u

long int = %ld

unsigned long int = %lu.

unsigned short int = %hu

signed short int = %hi

(ii): char:

- It stores a single character.
- Occupies 1 byte.
- Based on ~~si~~ qualifiers, char is of two types:

Signed char	Unsigned char
Range: -128 to 127	Range: 0 to 255

format specifies for char: %. c

Assignment: char a;
eg: printf ("%.c", 98) => b
↳(ASCII code)

(iii): float:

- It is used to store decimal values
→ Occupies 4 bytes of space ie, 32 bits.
Range: $-3.4e38$ to $+3.4e^{38}$
- float takes 6 digits of precision
Format specifies = %.f
Assignment: float a, a=10.0
printf ("%.f"; a) => 10.000000

(iv): double:

- It also stores decimal values with greates precision that float. ~~Ie~~,

Double	Long-double
→ 8 bytes	→ 10 bytes
→ Digits of precision: 14	→ Digits of precision: 15, 16, 33
format specifies: %.lf	Format specifier: %.Lf

(v): void:

→ It is ~~specif~~ special datatype that doesn't return any value.
- So, it is only used to define a function and not used with variables.

(b): Secondary datatype: /Derived datatypes

Variables in C

Variables are the name given to a memory location where we store values, characters, etc. while writing a program

x) Declaration of variable:

+) Format: datatype variablename;

Eg: int a;

+) Initialization: a = 10;

We can declare and initialize variables together

Eg: int a = 10;

More than one variables can also be initialized and declared together.

Eg int a = 10, b = 14;

- A variable can only be initialized after declaring it.

x) Rules:

i) Name must start with uppercase or lowercase or underscore

ii) Can't being with numbers ~~keywords~~ or have special symbols

iii) Keywords can't be variables.

Constants in C

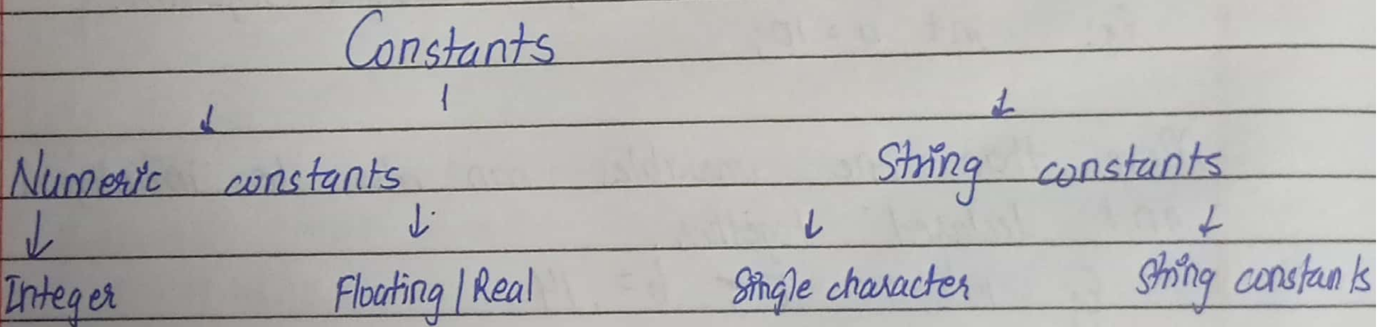
Constants are the fixed values that remain same throughout the execution of the program.

*) Assignment of symbolic Constant:

define PI 3·14

*) Rules for Assigning symbolic constants:

i) Generally, it is declared above main function.
ii) Capital letters is used to define.
iii) We don't use semicolon or equals to sign.



*) Integer constant:

It is of three types:

i) decimal: 10, 5, 1 (default)
ii) octal constant: 05, 07, 00 (proceeds with 0)
iii) hexadecimal constant: 0x / 0X AF (proceeds with 0x / 0X)

Assignment: const int a=10;

x) Rules: i) we can use '+' and '-' sign
ii) we can use spaces or special characters.

)Floating) Real Numeric Constant:

The numeric constant that have fractional value ie, decimal point are called floating / real constants.

Assignment: const float a = 10.11;

x) Single Character Constant:

- The single character enclosed with single quotation marks.
- Characters are stored in form of ASCII codes.

Assignment: const char a = 'a'

```
Eg: printf ("%d", 'a'); => 97
    printf ("%c", 97);  => a
```

Note: $5 \neq$ '5'

Date No.

x) String constant:

- The string constant are the sequence of character enclosed within double quotation marks.

Assignment: const char a = "Hello"

Note: (i): "a" = string constant → assigns value equal to ASCII

(ii) : 'a' = single character → string constant not equal to ASCII

When compiler reads string constant, it reads the first character and adds zero at the end indicating the end of string.

Eg: "Jenny" $\xrightarrow{\text{compiler}}$ "Jenny\0"

$\underset{5}{\downarrow}$ $\quad$ $\underset{6 \text{ characters.}}{\llcorner}$