C-PROGRAMMING

~by Gaurang Dalal

Maker of C-Programming Language: *Dennies Ritchie* at *AT & T Bells Labs, USA*.

- **Features of C:**
  - Middle Level language, used as both High and Low level Language., hence it is called as Mid-Level Language.
  - Flexibility.
  - Small size (32 Keywords).
  - Modular Design.
  - Powerful.
  - Portability (Platform Independent).
  - Bit Engineering.
  - Use of pointers (*ptr).
  - Efficiency.

STRUCTURE OF C PROGRAMMING

Documentation Section.

Link Section.

Macro Definition Section.

Global Declaration Section.

Function Section

```
main()
{
   Local Declaration
   Executable Statements
}
```

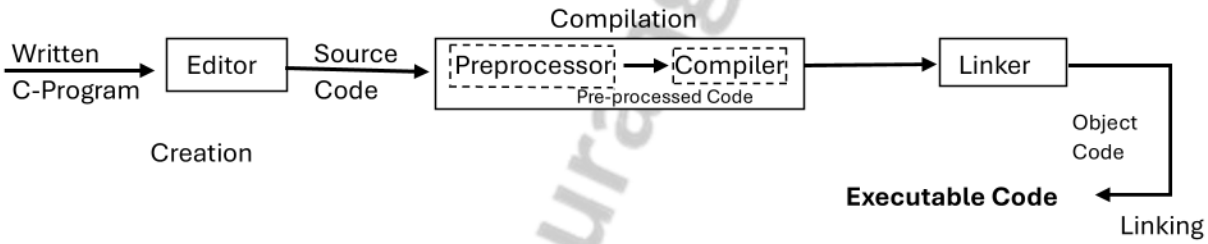
Sub Program Section

Function 1
Function 2
⋮

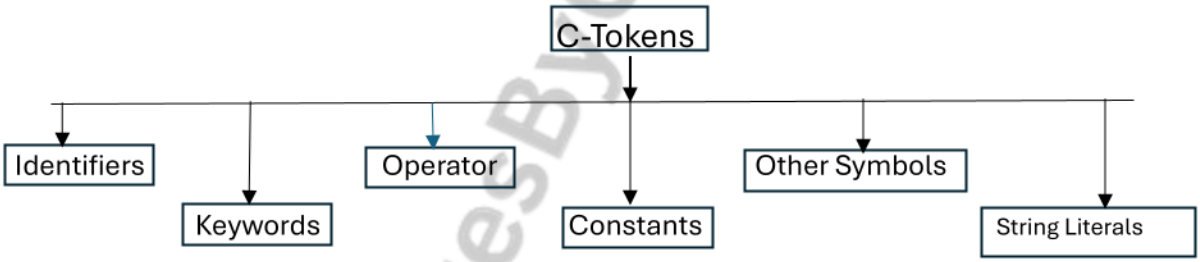
C Hello World Program

```c
#include<stdio.h>
void main ()
{
      printf("Hello World");
}
```

- **C-Program Development Step**



- **C-Tokens:**




```c
void main()
{
          int num = 10;
          printf("num = %d", num);
}
```

- **C Character Set**

  Can be used as UPPERCASE, lowercase, digits and symbols.

- **Keywords:**
  - Are the reserved words that are predefined in language, having specific meaning., cannot be changed, must written in lowercase.

- **32 Keywords:**

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

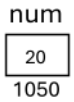
- **Identifiers:**

  Identifiers are the user defined names given to program elements like variable, function and symbolic function.

  e.g., *sum, roll_number, PI, radius1,etc.,*

- **Rules**:
  1. Should contains alphabets from **uppercase** to **lowercase**, also **digits** from *0* to *9* and *(_)* **underscore** are allowed.
  2. **Must start** with *alphabet* or *underscore* only.
  3. **No** special *symbols* are **allowed**.
  4. *Keywords* **should not be** used as *identifiers*.
  5. C is *case sensitive* language.

- **Variables:** A Variable is a name given to the memory location where data is stored.



  In the above example, 'num' is a variable which stores a value '20' at memory location '1050'.

  Maximum length is 31 characters.

- Some examples of valid and invalid tags.

Valid Tags	Invalid Tags
a	4abc
Roll_no	“a”
num2	Roll-no
string	num-2
String	-num-2
Integer	emp salary
Add	~~Double *~~
Add2	interest-rate
g67	

Note: We can use ***Double*** as valid tag, but it is **not** recommended.

- **Constants:** Constants are fixed values or literals that *do not change* in a program.

    - Integer Constant.
    - Floating Point Constant.
    - Character Constant.
    - String Literals.
    - Enumeration Constants.
    - Symbolic Constants.

---

1. **Integer Constants.**

→ Numeric values that are whole numbers does not contain fractions, specified in three types, decimal numbers (base 10), octal numbers (base 8) and hexadecimal numbers (base 16). These are signed or unsigned.

2. **Floating point Constants.**

→ Real numbers have a decimal point or exponential. These are signed or unsigned.

- **Escape Sequences**

  Also called as backslash character constant.

  1. \a: Alert or Beep.
  2. \b: Backspace. (Moves cursor to 1 position left.)
  3. \n: New Line. (Moves cursor to start of next line.)
  4. \r: Carriage Return. (It takes cursor to the beginning of same line.)
  5. \t: Horizontal Tab. (Moves cursor to next tab stop.)
  6. \v: Vertical Tab. (Moves cursor to next vertical tab stop.)
  7. \0: Null character. (It is used to terminate string.)
  8. \\: Backslash. (Use to display backslash.)
  9. \': Single Quote. (Use to display single quote.)
  10. \" : Double Quote. (Use to display double quote.)
  11. \?: Question mark. (Use to display question mark.)
  12. \ooo: Octal number. (Each 'o' represents an octal number.)
  13. \xhh: Hexadecimal. (Each 'h' represents a hexadecimal number.)

- **Enumeration constant:**

  Enumeration constant is used to create a list.
  e.g., enum color{red=10, green=20, blue=30};
  enum day{Sunday, Monday, Tuesday};

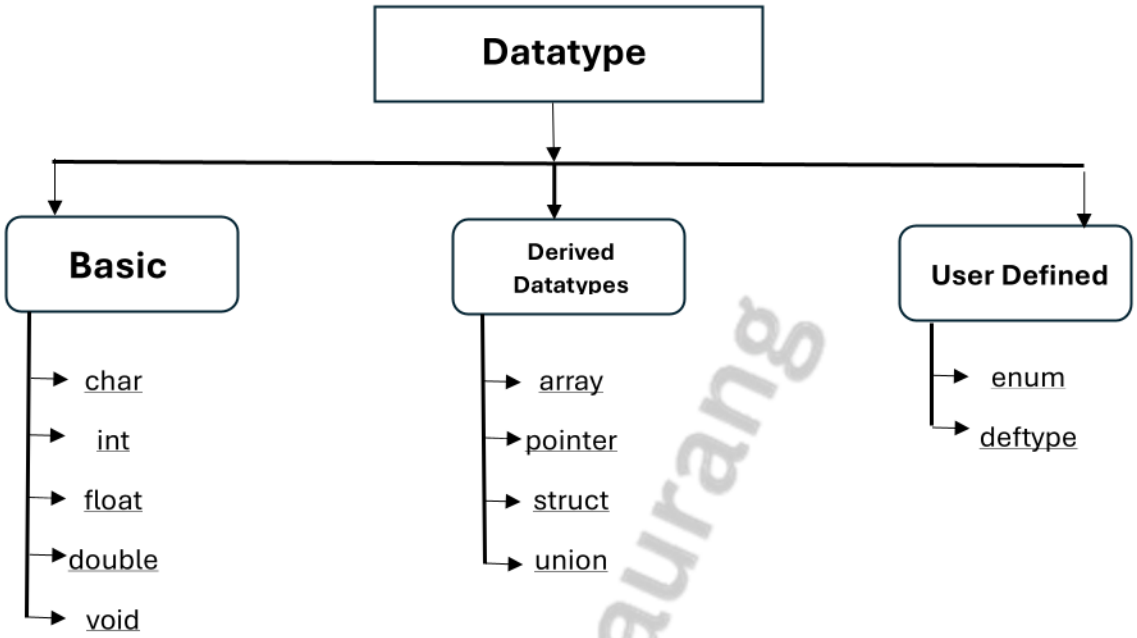
- **Symbolic Constant:**

  Symbolic constants are also called as ***Macro.*** It defines a symbol which represents a value. A symbolic constant is defined by special preprocessor instruction which is ***'#define'.***

  For e.g., **#define** PI 3.142

PTO

- **Datatypes**



Datatype	Description	Size
char	A single character	1 byte
int	An integer number	2/4 bytes
float	Single precision floating point number	4 bytes
double	Double precision floating point number	8 bytes
void	Empty database	0

- **sizeof():** It gives the size of specified data types in bytes.

- **Typecasting:** Typecasting means converting one database into another.

  There are 2 types of typecasting:

  1. Implicit typecasting: Also called as automatic type conversion.

     e.g.,
     ```
     int num = 20;
     float f = num;
     ```

2. Explicit typecasting: Explicit typecasting is performed to convert one type to another using a special operator called typecast operator. C provides a unary operator for explicit type conversion called cast operator.

   Syntax: `(type_name)expression;`
   Example:
   ```
   float ans;
   int a,b;
   ans = (float)a/b;
   ```

- **Operator and Expressions:**

  - **Operator Precedence (Hierarchy) Associativity:**

    If an expression contains more than one operator, some rules are needed to specify the order in which operations are performed.
    Precedence gives priority to operators with respect to other operators and expressions may contain more than one operator having same priority. Here the associativity specifies the order of evaluation of operators having the same priority.

  - **Types of Operators:**

    1. *Arithmetic operator.*
    2. *Relational operator.*
    3. *Logical operator.*
    4. *Assignment operator.*
    5. *Increment and Decrement operator.*
    6. *Conditional operator.*
    7. *Bitwise Operator.*
    8. *Other Operator*

PTO

- Arithmetic Operators:
  e.g., : *5/2+4-6*2+25/5-5%4*

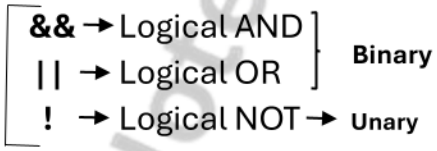
  ans: 2+4-6*2+25/5-5%4
  2+4-12+25/5-5%4
  2+4-12+5-5%4
  2+4-12+5-1
  6-12+4
  -1-1
  -2

  e.g., : 5-2*3/4+2

  ans: 5-6/4+2
  5-1+2
  4+2
  6

Relational Operator: (<, <=, >,=>, ==, !=).

Logical Operator:



The Logical operators are used to combine two or more expressions (usually rational), and the entire expression is called logical expression which evaluates to 'true' or 'false'.

The result of logical '**&&**' and '**||**' operations for different combination of two operands is given in following table:

OP1	OP2	OP && OP2	OP1 || OP2
T	T	T	T
T	F	F	T
F	T	F	T
F	F	F	F

Examples: `(mark => 60) && (mark < 70)`
`(age => 25) | | (salary > 10,000)`

`!(5<10)` → output: F

- **Increment and Decrement Operator:**
    1. Prefix: The operator is written before operand; the decrement or increment is done before the operand is used in an expression.
        e.g., ++a, ++b, --a;

    2. Postfix: The operator is written after operand; the decrement or increment is done after the operand value is used in an expression.
        e.g., a++, y--;

- Bitwise Operator: C provides six operators for manipulation of data at bit level.

Operator	Operation
&	Bitwise AND
|	Bitwise OR
^	Bitwise XOR
<<	Left Shift
>>	Right Shift
~	One's Compliment (unary)

PTO

e.g., let a = 13, b = 7 then find out: **a & b**, **a | b**, **a ^ b**, **~a**, **a<<3**, **a>>3 .**

Given: a = 13, b = 7; So, 13 in binary is **1101** and 7 in binary is **0111**.

a = 13: 0000 0000 0000 1101
b = 7 : 0000 0000 0000 0111

a & b: 0000 0000 0000 0101
a | b: 0000 0000 0000 1111
a ^ b: 0000 0000 0000 1010
~a: 1111 1111 1111 0010

a<<3: 0000 0000 0110 1000
a>>3: 0001 1100 0000 0001

- Shorthand operator: It enables writing simple operation in short form.
  e.g., x+=y; → x = x+y;
  m/3=3; → m=m/3;

- Conditional operator: This is the only ternary operator (3 operands)
  The pair *'?:'* is used to construct conditional expression.

  Syntax: `expression_1 ? expression_2 : expression 3;`

  e.g.,
  ```
  int a = 10, b = 15;
  max = (a>b)? a : b;
  ```
  Hera Pheri of order of variables is not allowed

END OF CHAPTER