C LANGUAGE

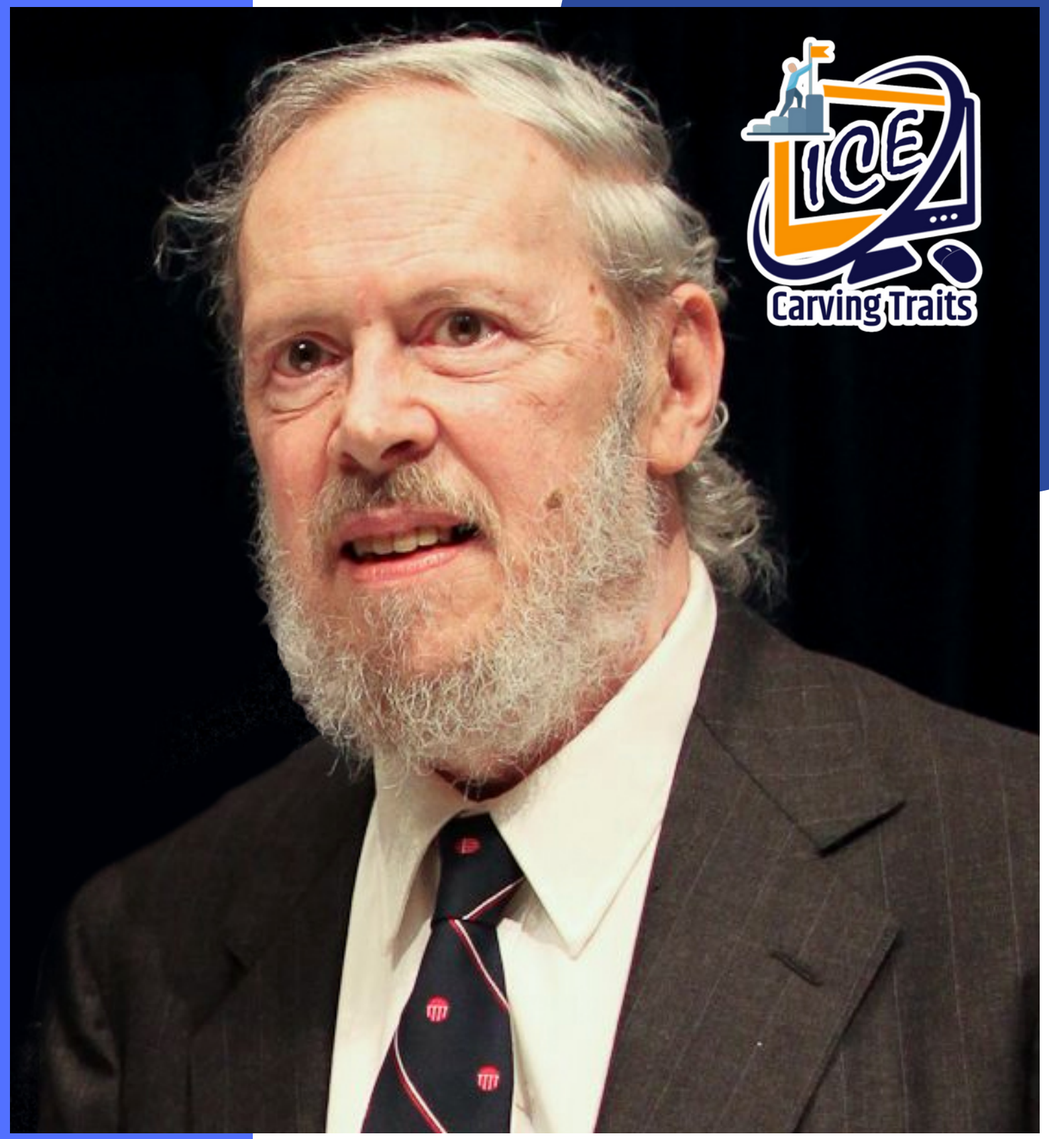
BY AADESH LOKHANDE

INTRODUCTION

WHAT IS C LANGUAGE?

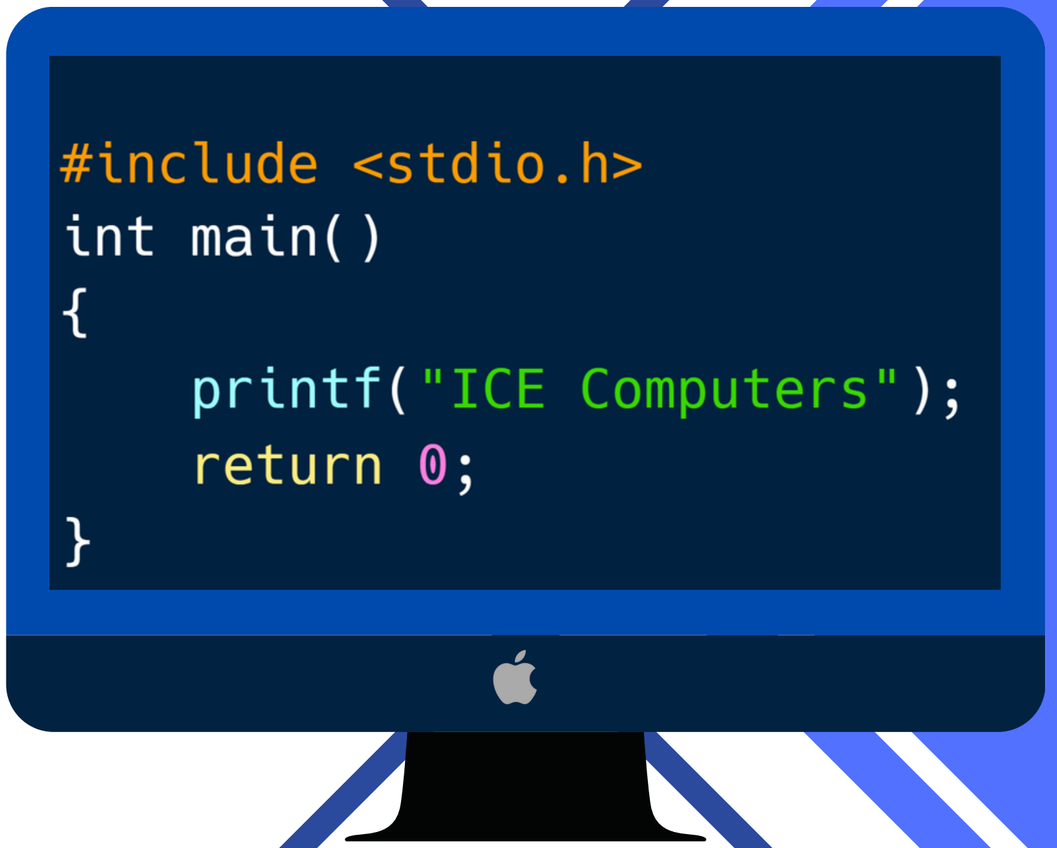
C programming is a general-purpose, procedural programming language developed in 1972 by Dennis M. Ritchie at the Bell Telephone Laboratories to develop the UNIX operating system.

C is the most widely used computer language. It keeps fluctuating at number one scale of popularity along with Java programming language, which is also equally popular and most widely used among modern software programmers.



WHY TO LEARN C PROGRAMMING?

1. Easy to learn
2. Structured language
3. It produces efficient programs
4. It can handle low-level activities
5. Ready compiled on a variety of computer platforms



FACTS ABOUT C

1. C was invented to write an operating system called UNIX.
2. C is a successor of B language which was introduced around the early 1970s.
3. The language was formalized in 1988 by the American National Standard Institute (ANSI).
4. The UNIX OS was totally written in C.
5. Today C is the most widely used and popular System Programming Language.
6. Most of the state-of-the-art software have been implemented using C.

STRUCTURE OF C PROGRAMMING

#include	-	Preprocessing command
stdio	-	Library
.h	-	Library extension
int	-	Return type
main()	-	Function (execution starting point)
{}	-	Scope
printf()	-	String print
"hello world"	-	String
;	-	Semicolon (full stop)
return	-	Keyword (values return)
0	-	Value

```
#include <stdio.h>
int main()
{
    printf("Hello World!");
    return 0;
}
```

Carving Traits

COMMENTS IN C PROGRAMMING

Carving Traits

In C programming, comments are annotations within the code that are ignored by the compiler during compilation. They are used to provide explanations, notes, or disable code temporarily. Comments help programmers understand the code and make it more readable.

C supports two types of comments:

1) Single-line Comments:
Denoted by //. Anything following // on the same line is treated as a comment.

2) Multi-line Comments:
Enclosed between /* and */. They can span multiple lines and are often used for longer explanations or for temporarily excluding blocks of code from execution.

```
// This is a single-line comment

/*
   This is a multi-line comment.
   It can span multiple lines.
*/

int main()
{
    // This is another single-line comment
    return 0;
}
```

TOKENS

In C programming, tokens are the fundamental components of the code, including keywords, variable names, constants, operators, and punctuation. They serve as the building blocks that compilers use to interpret and process the code.

```
#include <stdio.h>
int main()
{
    printf("Hello World!");
    return 0;
}
```

Carving Traits

TOKENS

- Keywords: Reserved words with special meanings. Example: if, while, int.
- Identifiers: Names for variables, functions, etc. Example: count, main.
- Constants: Fixed values. Example: 42, 3.14, 'A'.
- Operators: Symbols for operations. Example: +, =, &&.
- Punctuation: Structural symbols. Example: ;, {, (.
- Comments: Explanatory notes (ignored by compiler). Example: // This is a comment.
- Preprocessor Directives: Instructions starting with #. Example: #include <stdio.h>.

Tokens in the code:

- Keywords: int, return
- Identifier: main, num
- Constants: 5, 0
- Operators: = ()
- Punctuation: { } ; ( ) < >
- Comments: // This is a comment
- Preprocessor Directive: #include, stdio, .h

```
#include <stdio.h>
int main()
{
    int num = 5;
    printf("Value: %d", num);
    return 0;
}
```

Carving Traits

KEYWORDS

Carving Traits

In C programming, keywords are reserved words that have predefined meanings and specific roles in the language. They cannot be used as identifiers because they are already assigned special purposes by the language. Keywords play a crucial role in defining the structure and behavior of C programs.

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

ESCAPE SEQUENCE

Carving Traits

Escape sequences help you insert characters with special meanings into your strings and characters, enhancing their readability and functionality.

For example:

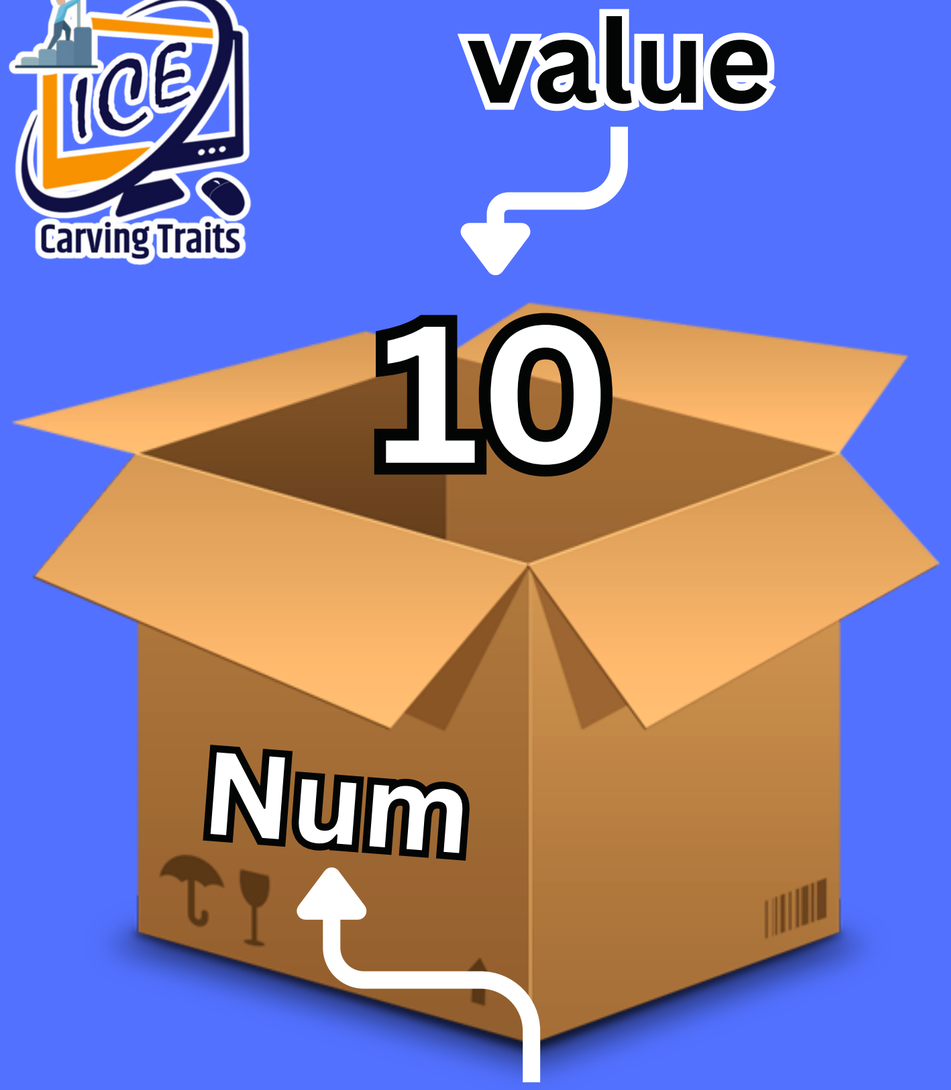
- \n represents a newline character.
- \t represents a tab character.
- \" represents a double quote character.
- \\ represents a literal backslash.

VARIABLE

Carving Traits

A variable is a named storage location in memory used to hold data. It must be declared with a type and can store values for calculations and manipulation.

```
int num = 10;
```



DATA TYPE

1. Data type is an attribute of data which tells the C compiler, which type of data a variable is holding.
2. It can be of type integer, float(decimal), character , Boolean (true/false) etc.
3. Formally we use data types to specify the type of data our variables are holding.

Four main types of primary/basic data types are:

DataType	Format Specifier	Values
int	%d	12, 23, 43, 0, -12, -23, ...
float	%f	3.14, 7.5, 5.9, 1.5, ...
char	%c	‘a’, ‘A’, ‘@’, ‘9’, ...

IDENTIFIER

Identifiers are names used to identify variables, functions, arrays, and other user-defined entities.

Here are the rules for creating valid identifiers in C:

1. We can’t start identifier name with number
2. we can use underscore
3. we can’t user keywords as a identifier
4. identifiers are case sensitive

OPERATOR

Carving Traits

Operators are symbols that perform operations on one or more operands to produce a result. They are fundamental for performing various computations and manipulations within a program. C provides a wide range of operators that can be classified into several categories:

Arithmetic Operators	Relational Operators
Logical Operators	Assignment Operators
Increment and Decrement Operators	Conditional (Ternary) Operator

ARITHMETIC OPERATOR

Carving Traits

Operator	Description	Example
+	Addition	int sum = a + b;
-	Subtraction	int diff = a - b;
*	Multiplication	int product = a * b;
/	Division	float quotient = a / b;
%	Modulo (Remainder)	int remainder = a % b;

LOGICAL OPERATOR

Carving Traits

OPERATOR	DESCRIPTION	EXAMPLE
&&	Logical AND	if (a && b)
||	Logical OR	if (a || b)
!	Logical NOT	if (!condition)

LOGICAL AND

INPUT 1	INPUT 2	OUTPUT
0	0	0
0	1	0
1	0	0
1	1	1

LOGICAL OR

INPUT 1	INPUT 2	OUTPUT
0	0	0
0	1	1
1	0	1
1	1	1

LOGICAL NOT

INPUT	OUTPUT
0	1
1	0

RELATIONAL OPERATOR

Carving Traits

Operator	Description	Example
<	Less than	if (a < b)
>	Greater than	if (a > b)
<=	Less than or equal to	if (a <= b)
>=	Greater than or equal to	if (a >= b)
==	Equal to	if (a == b)
!=	Not equal to	if (a != b)

ASSIGNMENT OPERATOR

Carving Traits

Operator	Description	Example
=	Simple assignment	x = 10;
+=	Addition assignment	x += 5; (equivalent to x = x + 5;)
-=	Subtraction assignment	x -= 3; (equivalent to x = x - 3;)
*=	Multiplication assignment	x *= 2; (equivalent to x = x * 2;)
/=	Division assignment	x /= 4; (equivalent to x = x / 4;)
%=	Modulo assignment	x %= 3; (equivalent to x = x % 3;)

INCREMENT AND DECREMENT OPERATORS

Carving Traits

Operator	Description	Example
++	Increment (add one)	x++; (post-increment)
--	Decrement (subtract one)	y--; (post-decrement)
++x	Pre-increment	++x; (increment before use)
--y	Pre-decrement	--y; (decrement before use)

CONDITIONAL (TERNARY) OPERATOR

Syntax:

condition ? trueStatement : falseStatement

Example:

```
10 < 20 ? printf(”Hello”) : printf(”Bye”);        // Hello
10 > 20 ? printf(”Hello”) : printf(”Bye”);        // Bye
```

THANK YOU

WELCOME TO THE CODING WORLD

BECAUSE

CODE NEVER LIE