Introduction

- ✓ C is a general purpose language which is very closely associated with UNIX for which it was developed in Bell Laboratories.
- ✓ Most of the programs of UNIX are written and run with the help of 'C'.
- ✓ Many of the important ideas of 'c' stem are from BCPL by Martin Richards.
- ✓ In 1972, Dennies Ritchie at Bell Laboratories wrote C Language which caused a revolution in computing world .
- ✓ From beginning C was intended to be useful for busy programmers to get things done easily because C is powerful,dominant and supple language.

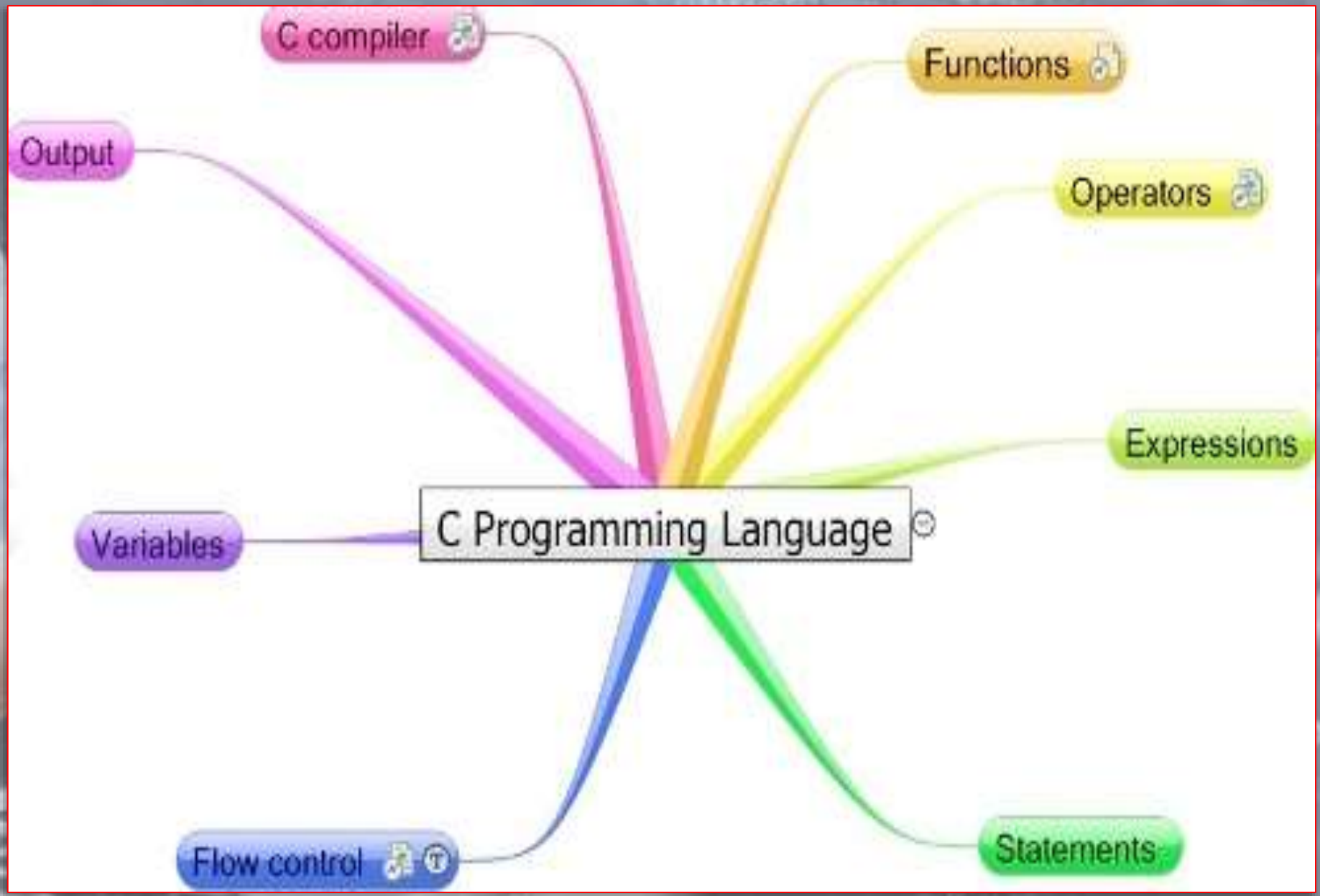
Why Name 'C' was given to this language?

- Many of the ideas of C language were derived and taken from 'B' language.
- BCPL and CPL are previous versions of 'B' language.
- As many features came from B it was named as 'C'.

ABOUT "C"

✓ ***C programming language - Structured and disciplined approach to program design.***

- C is a structured programming language
- C supports functions that enables easy maintainability of code, by breaking large file into smaller modules
- Comments in C provides easy readability
- C is a powerful language.
- C programs built from
  - Variable and type declarations
  - Functions
  - Statements
  - Expressions


C compiler
Functions
Output
Operators
Expressions
C Programming Language
Variables
Flow control
Statements

Structure Of *"C"* Programs

- Before going and reading the structure of C programs we need to have a basic knowledge of the following:

1. C's Character Set
2. C's Keywords
3. The General Structure of a 'C' Program
4. How To End A Statement
5. Free Format Language
6. Header Files & Library Functions

C's Character Set

C does not use every character set and key found on
modern computers . The only characters that C - Language uses for its programs are as follows:

- ✓ A-Z all alphabets
- ✓ a-z all alphabets
- ✓ 0-9
- ✓ # % & ! _ {} [] () $$$$ &&&& |
- ✓ space . , : ; ' $ "
- ✓ + - / * =

The keywords

- ✓ "**Keywords**" are words that have special meaning to the **C** compiler.
- ✓ Their meaning cannot be changed at any instance.
- ✓ Serve as basic building blocks for program statements.
- ✓ All keywords are written in **only lowercase.**

Basic Structure Of "C" Programs

```
#include<stdio.h>
#include<conio.h>

void main()
{
   -- other statements
}
```

Header Files

Entry Point Of Program

Indicates Starting of Program

Header files

- ✓ The files that are specified in the include section is called as Header File.
- ✓ These are precompiled files that has some functions defined in them.
- ✓ We can call those functions in our program by supplying parameters.
- ✓ Header file is given an extension .h .
- ✓ C Source file is given an extension .c .

Main function

- ✓ This is the *"Entry Point"* of a program.
- ✓ When a file is executed, the start point is the main function.
- ✓ From main function the flow goes as per the programmers choice.
- ✓ There may or may not be other functions written by user in a program.
- ✓ Main function is compulsory for any C program.

Running a ‘C’ Program

- Type a program.
- Save it.
- Compile the program – This will generate an .exe file (executable)
- Run the program (Actually the exe created out of compilation will run and not the .c file)
- In different compiler we have different option for compiling and running.

“C” language TOKENS

- ✓ The smallest individual units in a C program are known as tokens. In a C source program, the basic element recognized by the compiler is the "token." A token is source-program text that the compiler does not break down into component elements.
- ✓ C has 6 different types of tokens viz.

1. Keywords [e.g. float, int, while]
2. Identifiers [e.g. main, amount]
3. Constants [e.g. -25.6, 100]
4. Strings [e.g. “SMIT”, “year”]
5. Special Symbols [e.g. {, }, [,]]
6. Operators [e.g. +, -, *]

- ✓ C - programs are written using these tokens and the general syntax.

Keywords in Ansi “C”

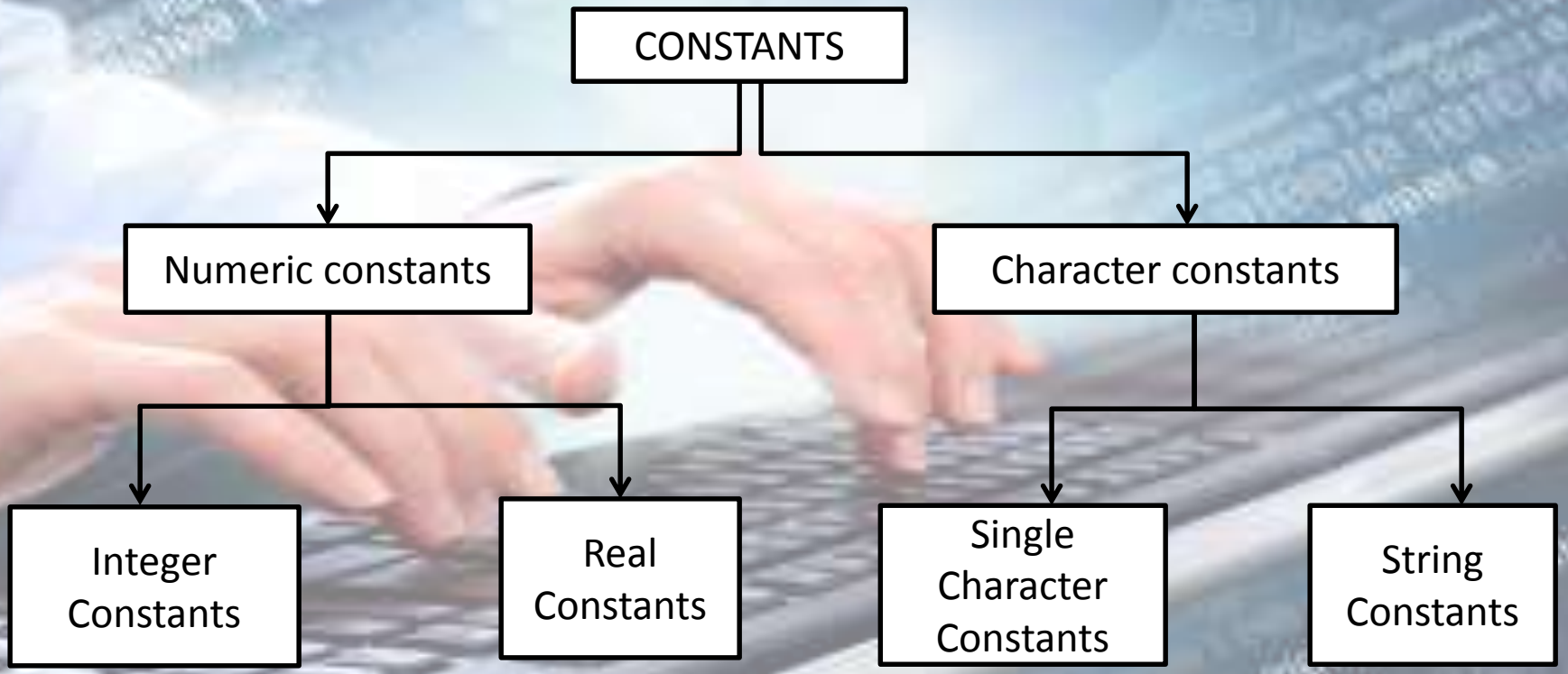
auto	double	register	switch
break	else	return	typedef
case	enum	short	union
char	etern	signed	unsigned
const	float	sizeof	void
continue	for	static	volatile
default	goto	struct	while
do	if	int	long

The Identifiers

- ✓ They are programmer-chosen names to represent parts of the program: variables, functions, etc.
- ✓ Cannot use C keywords as identifiers
- ✓ Must begin with alpha character or _, followed by alpha, numeric, or _
- ✓ Upper- and lower-case characters are important (case-sensitive)
- ✓ Must consist of only letters, digits or underscore (_).
- ✓ Only first 31 characters are significant.
- ✓ Must NOT contain spaces ( ).

Constants

- **Constants** in C are the fixed values that do not change during the execution of a program.



Constants Examples

- ***Integer Constants***
  - Refers to sequence of digits such as decimal integer, octal integer and hexadecimal integer.
  - Some of the examples are 112, 0551, 56579u, 0X2 etc.
- ***Real Constants***
  - The floating point constants such as 0.0083, -0.78, +67.89 etc.
- ***Single Character Constants***
  - A single char const contains a single character enclosed within pair of *single quotes* [‘ ’]. For example, ‘8’, ‘a’ , ‘i’ etc.
- ***String Constants***
  - A string constant is a sequence of characters enclosed in double quotes [“ ”]; For example, “0211”, “Stack Overflow” etc.

DECLARATIONS

- ✓ Constants and variables must be declared before they can be used.
- ✓ A constant declaration specifies the type, the name and the value of the constant.
- ✓ any attempt to alter the value of a variable defined
- ✓ as constant results in an error message by the compiler
- ✓ A variable declaration specifies the type, the name and possibly the initial value of the variable.

- ✓ When you declare a constant or a variable, the compiler:

  - ❖ Reserves a memory location in which to store the value of the constant or variable.
  - ❖ Associates the name of the constant or variable with the memory location.

What Are Variables in C?

- A **Variable** is a data name that is used to store any data value.
- ***Variables*** are used to store values that can be changed during the program execution.
- **Variables** in C have the same meaning as variables in algebra. That is, they represent some unknown, or variable, value.

$$x = a + b$$

$$z + 2 = 3(y - 5)$$

- Remember that variables in algebra are represented by a single alphabetic character.

Naming Variables

- ✓Variables in C may be given representations containing multiple characters. But there are rules for these representations.
- ✓Variable names in C :
  - May only consist of letters, digits, and underscores
  - May be as long as you like, but only the first 31 characters are significant
  - May not begin with a number
  - May not be a C **reserved word (keyword)**
  - Should start with a letter or an underscore(_)
  - Can contain letters, numbers or underscore.
  - No other special characters are allowed including space.

Case Sensitivity

- ✓ C is a **case sensitive language.**
- ✓ It matters whether an **identifier**, such as a variable name, is uppercase or lowercase.
- ✓ Example:

  area

  Area

  AREA

  ArEa

  are all seen as <u>different</u> variables by the compiler.

Declaring Variables

- ✓ Before using a variable, you must give the compiler some information about the variable; i.e., you must declare it.
- ✓ The declaration statement includes the data type of the variable.
- ✓ Examples of variable declarations:

```
int  length ;
float  area ;
```

- ✓ Variables are not automatically initialized. For example, after declaration

```
int sum;
```

the value of the variable sum can be anything (garbage).

- ✓ Thus, it is good practice to initialize variables when they are declared.
- ✓ Once a value has been placed in a variable it stays there until the program alters it.

Data types in 'ansi c'

- There are three classes of data types here::

- Primitive data types
  - int, float, double, char
- Aggregate OR derived data types
  - Arrays come under this category
  - Arrays can contain collection of int or float or char or double data
- User defined data types
  - Structures and enum fall under this category.

Data Types- different attributes

Type	Size	Representation	Minimum range	Maximum range
char, signed char	8 bits	ASCII	-128	127
unsigned char	bool 8 bits	ASCII	0	255
short, signed short	16 bits	2's complement	-32768	32767
unsigned short	16 bits	Binary	0	65535
int, signed int	16 bits	2's complement	-32768	32767
unsigned int	16 bits	Binary	0	65535
long, signed long	32 bits	2's complement	-2,147,483,648	2,147,483,647
unsigned long	32 bits	Binary	0	4,294,967,295
float	32 bits	IEEE 32-bit	1.175495e-38	3.4028235e+38
double	32 bits	IEEE 32-bit	1.175495e-38	3.4028235e+38
long double	32 bits	IEEE 32-bit	1.175495e-38	3.4028235e+38

Example of “C” Program

```
/*      HELLO.C -- Hello, world */

#include <stdio.h>

Void main()
{
        printf("Hello, world\n");

        Getch();
}
```