Structured Programming Methodology

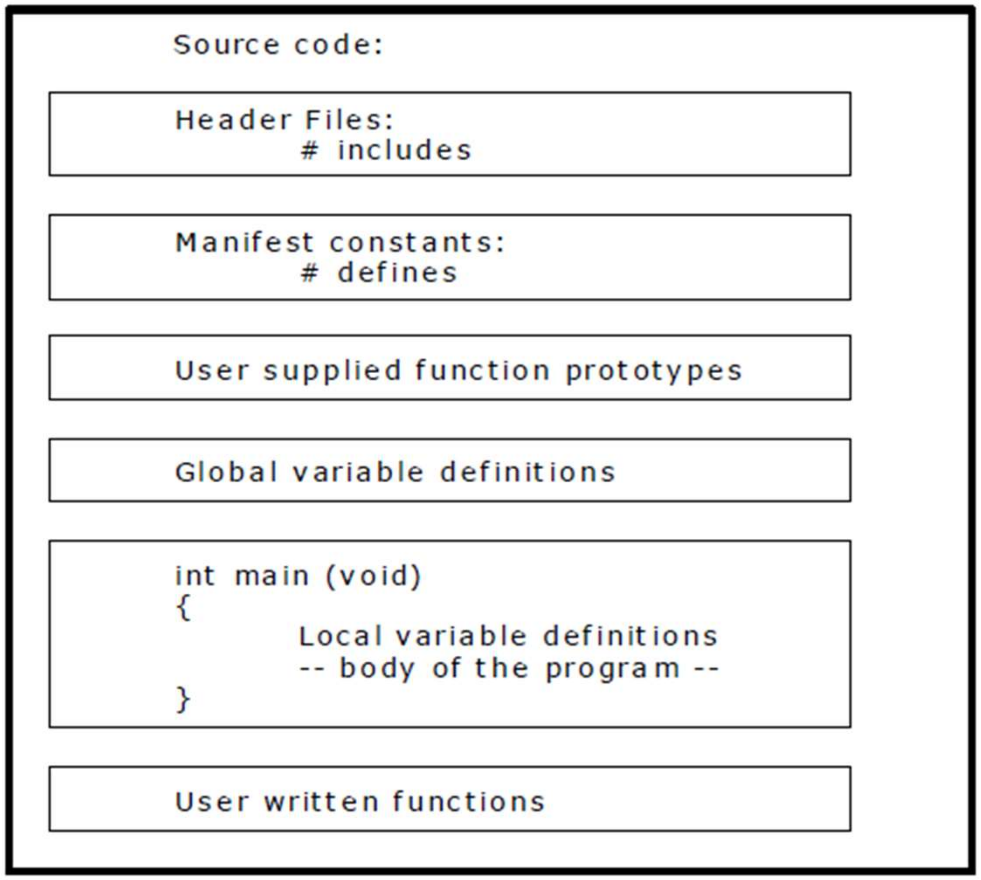
Topics for today

- Understanding concept and importance of header file / package / namespaces
- Data & Operators: Data Types, Identifier, Constants and Variables

Understanding Header Files

- Header files contain definitions of functions and macros.
- Used to share declarations between multiple source files.
- Example in C: #include <stdio.h>
- Ensures code reusability and modularity.

Simple C program structure


Source code:
Header Files:
# includes
Manifest constants:
# defines
User supplied function prototypes
Global variable definitions
int main (void)
{
Local variable definitions
-- body of the program --
}
User written functions

Simple C program structure

Header Files (.h):

Header files contains declaration information for function or constants that are referred in programs. They are used to keep source-file size to a minimum and to reduce the amount of redundant information that must be coded.

# includes:

An include directive tells the preprocessor to include the contents of the specified file at the point in the program. Path names must either be enclosed by double quotes or angle brackets.

# defines:

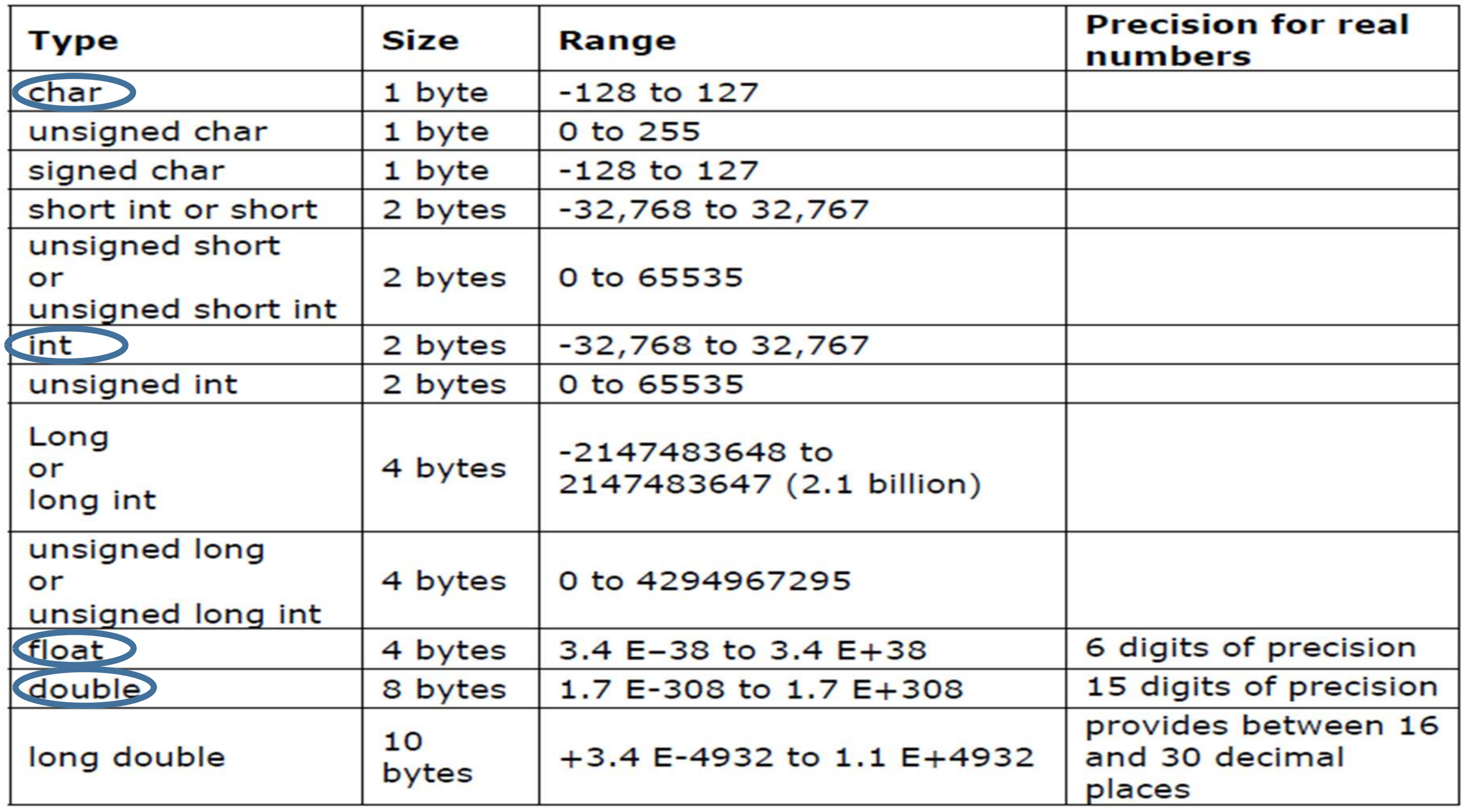
ANSI C allows you to declare ***constants.*** The # define directive is used to tell the preprocessor to perform a search-and-replace operation.
Example:

```
# define Pi 3.14159

# define Tax-rate 0.0735
```

In the example above, the preprocessor will search through the source file and replace every instance of the token Pi with 3.14159

Data types

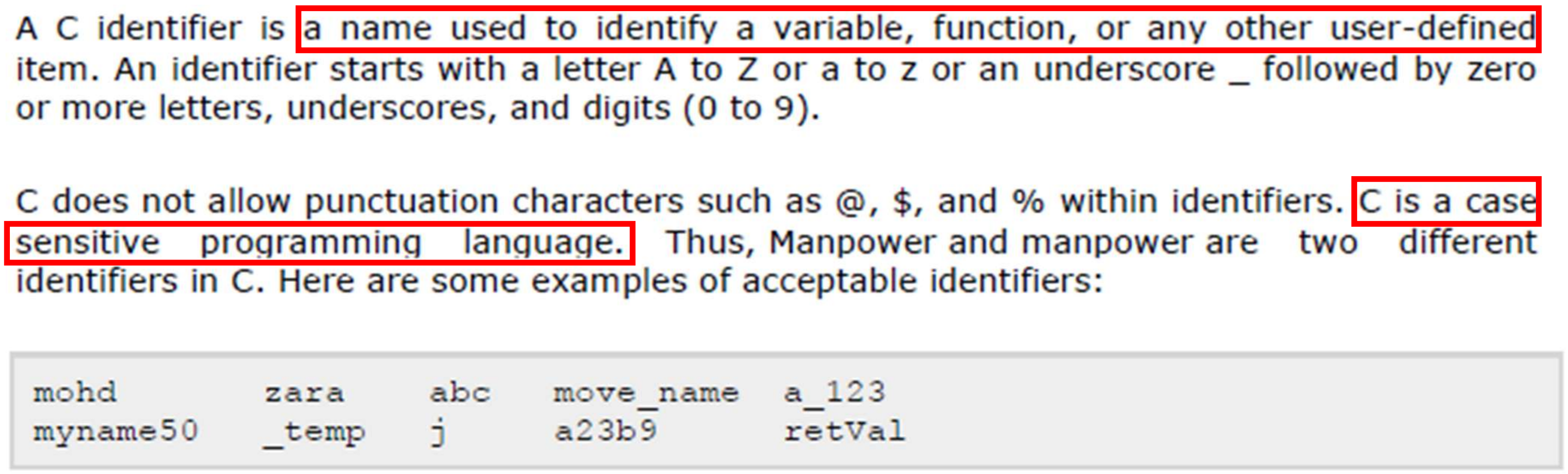


Type	Size	Range	Precision for real numbers
char	1 byte	-128 to 127	
unsigned char	1 byte	0 to 255	
signed char	1 byte	-128 to 127	
short int or short	2 bytes	-32,768 to 32,767	
unsigned short or unsigned short int	2 bytes	0 to 65535	
int	2 bytes	-32,768 to 32,767	
unsigned int	2 bytes	0 to 65535	
Long or long int	4 bytes	-2147483648 to 2147483647 (2.1 billion)	
unsigned long or unsigned long int	4 bytes	0 to 4294967295	
float	4 bytes	3.4 E–38 to 3.4 E+38	6 digits of precision
double	8 bytes	1.7 E-308 to 1.7 E+308	15 digits of precision
long double	10 bytes	+3.4 E-4932 to 1.1 E+4932	provides between 16 and 30 decimal places

Data types

- int age = 25;
- float salary = 50000.50;
- char grade = 'A';
- boolean isPassed = true;

Identifiers

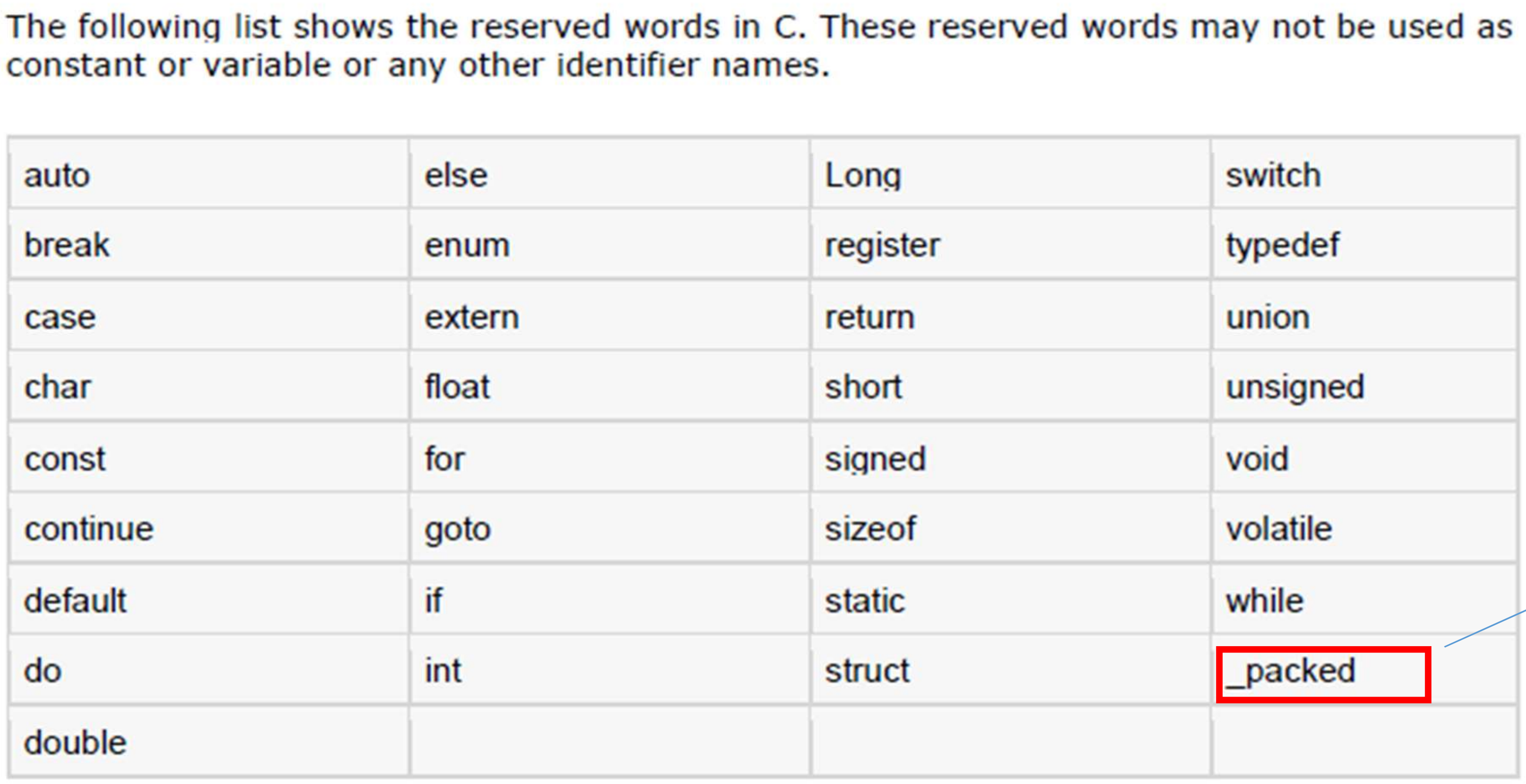


A C identifier is a name used to identify a variable, function, or any other user-defined item. An identifier starts with a letter A to Z or a to z or an underscore _ followed by zero or more letters, underscores, and digits (0 to 9).

C does not allow punctuation characters such as @, $, and % within identifiers. C is a case sensitive programming language. Thus, Manpower and manpower are two different identifiers in C. Here are some examples of acceptable identifiers:

```
mohd        zara    abc    move_name  a_123
myname50    _temp   j      a23b9      retVal
```

Keywords



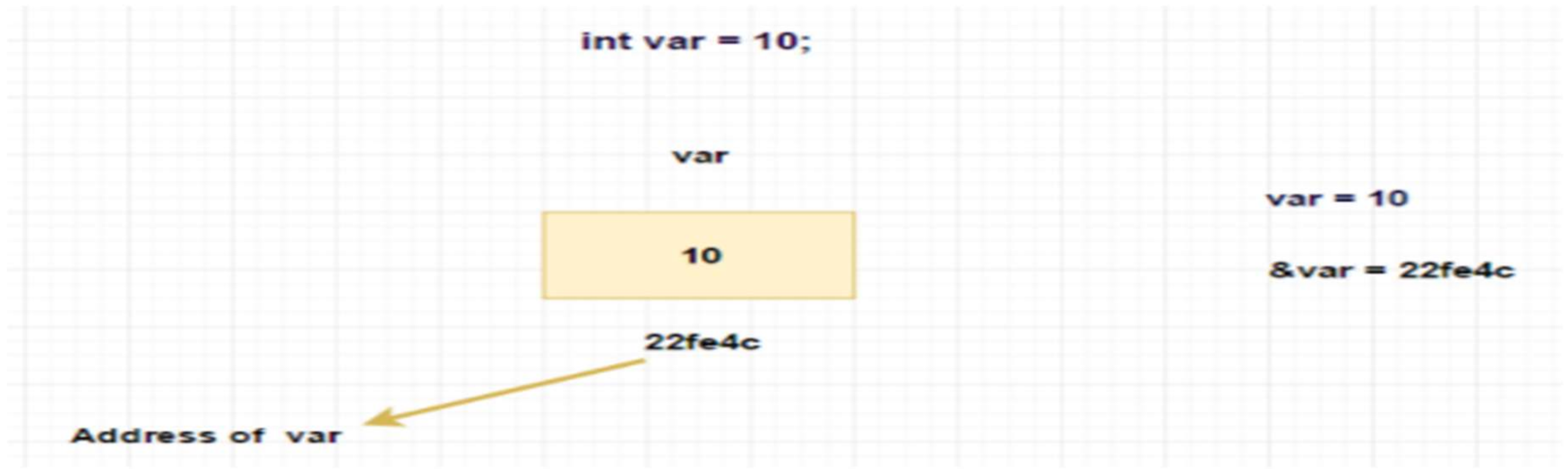
The following list shows the reserved words in C. These reserved words may not be used as constant or variable or any other identifier names.

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

Compiler specific

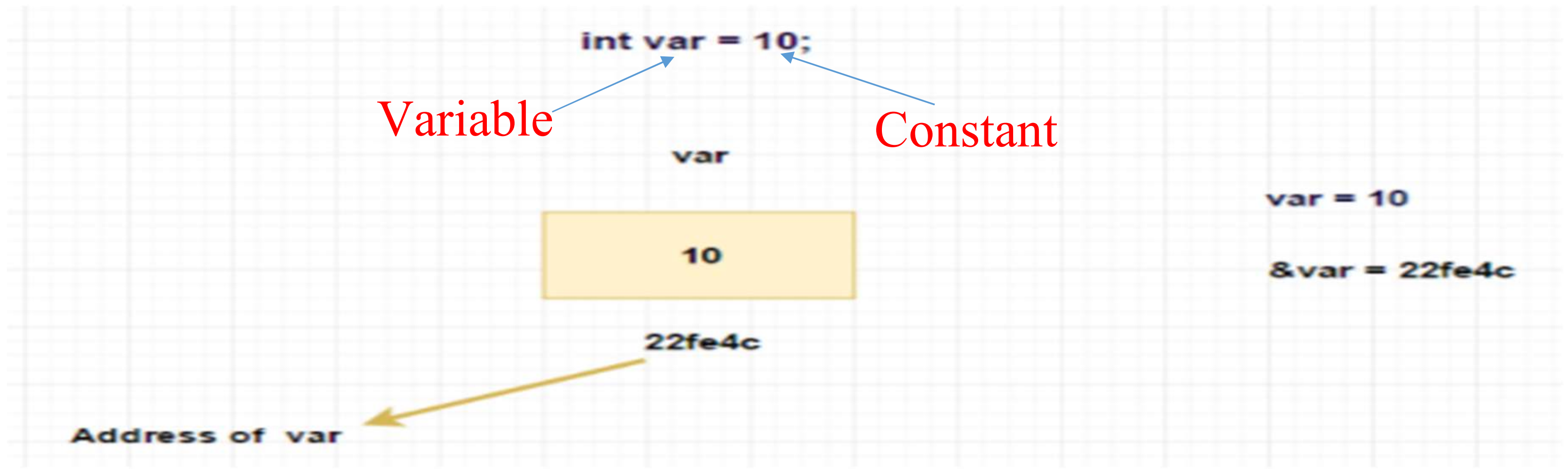
- Constants and variables

A constant is an entity that doesn’t change whereas a variable is an entity that may change.



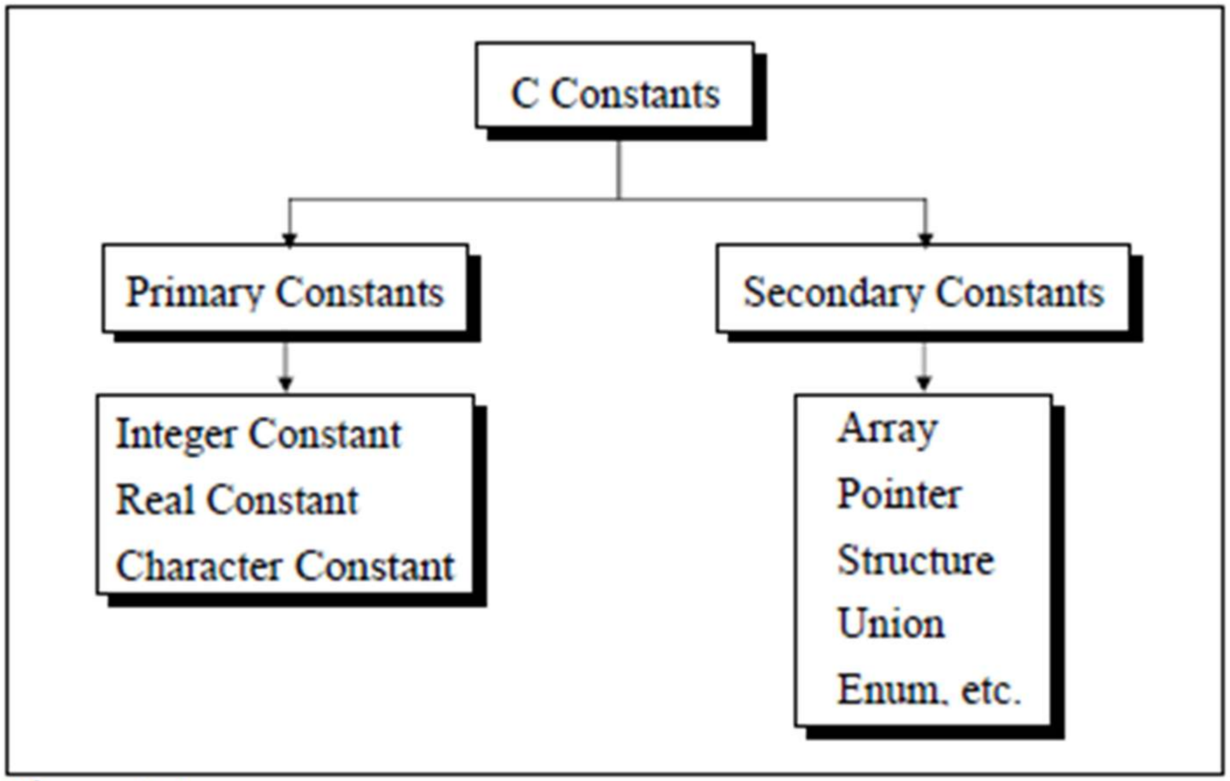
- Constants and variables

A constant is an entity that doesn’t change whereas a variable is an entity that may change.



Since the location whose name is var can hold different values at different times, var is known as a variable. As against this, 10 or 5 do not change, hence are known as constants.

- Types of Constants



- Rules for Constructing Integer Constants

(a) An integer constant must have at least one digit.
(b) It must not have a decimal point.
(c) It can be either positive or negative.
(d) If no sign precedes an integer constant it is assumed to be positive.
(e) No commas or blanks are allowed within an integer constant.
(f) The allowable range for integer constants is -32768 to 32767.

- ## Rules for Constructing Character Constants

(a) A character constant is a single alphabet, a single digit or a single special symbol enclosed within single inverted commas. Both the inverted commas should point to the left.

For example, ’A’ is a valid character constant whereas ‘A’ is not.

(b) The maximum length of a character constant can be 1 character.

Ex.:

'A'

'I'

'5'

'='

- Rules for Constructing Variable Names

(a) A variable name is any combination of 1 to 31 alphabets, digits or underscores. Some compilers allow variable names whose length could be up to 247 characters. Still, it would be safer to stick to the rule of 31 characters. Do not create unnecessarily long variable names as it adds to your typing effort.
(b) The first character in the variable name must be an alphabet or underscore.
(c) No commas or blanks are allowed within a variable name.
(d) No special symbol other than an underscore (as in gross_sal) can be used in a variable name.

Ex.: si_int
m_hra
pop_e_89