Module2

Programming Features: Constants, Variables, and Datatypes, Operators, and Expressions, Control statements, Iterations

Constants, Variables and Data Types

- Programming language is designed to help process certain kinds of data consisting of numbers, characters and strings for providing useful output known as ***information***.
- ***Program instructions*** are formed using certain symbols and words according to some rigid rules known as syntax rules (or grammar).
- C Characters that can be used to form words, numbers and expressions are grouped in
  - **Letters**
  - **Digits**
  - **Special Characters**
  - **While Spaces**
- Compiler ignores white spaces unless they are part of a string constant
- **White spaces** must be used to **separate words**, but are prohibited between the **characters of keywords and identifiers**.

Character Set

Letters		*Digits*
Uppercase A.....Z		All decimal digits 0 9
Lowercase a.....z		
	Special Characters	
, comma		& ampersand
. period		^ caret
; semicolon		* asterisk
: colon		– minus sign
? question mark		+ plus sign
' apostrophe		< opening angle bracket
" quotation mark		(or less than sign)
! exclamation mark		> closing angle bracket
| vertical bar		(or greater than sign)
/ slash		(left parenthesis
\ backslash		) right parenthesis
~ tilde		[left bracket
_ under score		] right bracket
$ dollar sign		{ left brace
% percent sign		} right brace
		# number sign
	White Spaces	
	Blank space	
	Horizontal tab	
	Carriage return	
	New line	
	Form feed	

Table: C Character Set

Trigraph Characters

- Many non-English keyboards do not support all C character set. e.g. Japanese keyboard, German keyboard, French keyboard.
- ANSI C introduces the concept of "***trigraph***" sequences to provide a way to enter characters not available in some keyboards.
- Each trigraph sequence consists of three characters ***(two question marks followed by another character)***

Trigraph sequence	*Translation*
??=	# number sign
??(	[left bracket
??)	] right bracket
??<	{ left brace
??>	} right brace
??!	| vetical bar
??/	\ back slash
??/	^ caret
??-	~ tilde

Table: ANSI C Trigraph Sequences

C Tokens

- Individual words and punctuation marks are called ***tokens***.
- In ***C program*** the ***smallest individual units*** are called ***C tokens***.
- C has ***6 types of tokens***.

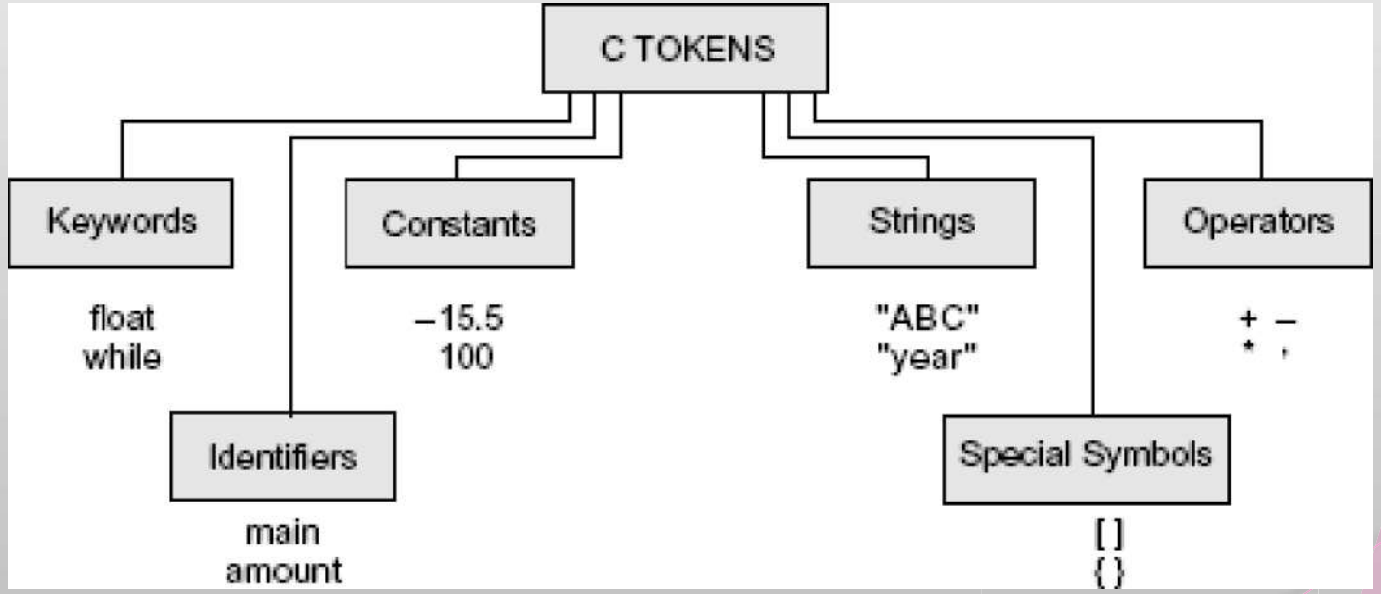


Fig: C tokens and Examples

C Keywords and Identifiers

- Every C word is classified as either a ***keyword*** or an ***identifier***.
- Keywords have fixed meanings and these meaning cannot be changed.
- Keywords serve as basic building blocks for program statements.
- ***All keywords must be written in lowercase***
- ***Identifiers*** refer to ***name of variables, functions, and arrays***.
- ***Identifiers*** are ***user-defined*** names and consist of a sequence of ***letters and digits***, with a letter as a first character
- Both ***uppercase and lowercase letters*** are ***permitted***
- ***Underscore character*** is also ***permitted***, act as a link between two words in ***long identifiers***.

float a; int A; double my_data;

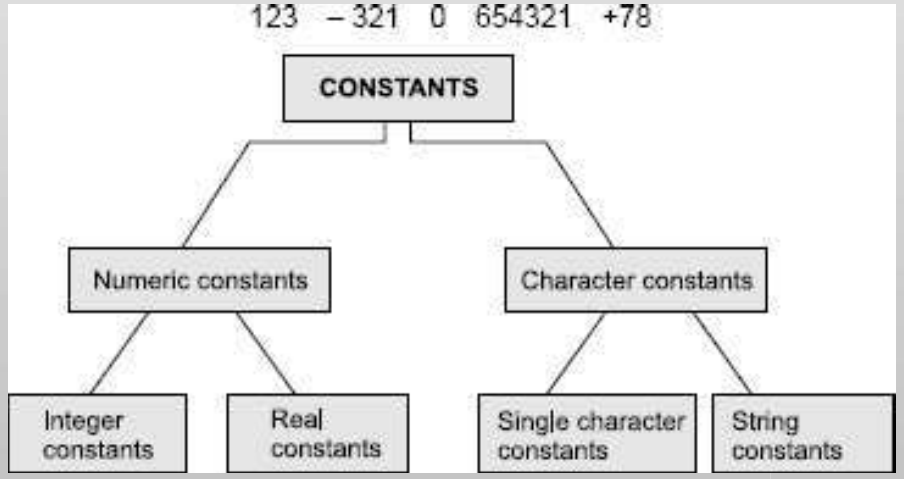
auto	double	int	struct
break	else	long	switch
case	enum	register	typedef
char	extern	return	union
const	float	short	unsigned
continue	for	signed	void
default	goto	sizeof	volatile
do	if	static	while

Rules for Identifiers

1. First character must be an alphabet (or underscore)
2. Must consist of only letters, digits or underscore.
3. Only first 31 characters are significant.
4. Cannot use a keyword.
5. Must not contain white space.

C Constants

- Constants refer to fixed values that do not change during execution of a program.
- Two types of constants ***Numeric constants*** and ***Character constants***
- ***Numeric Constants*** are subdivided into ***Integer constants*** and ***Real constants***
- ***Character Constants*** are subdivided into ***Single character constants*** and ***String constants***



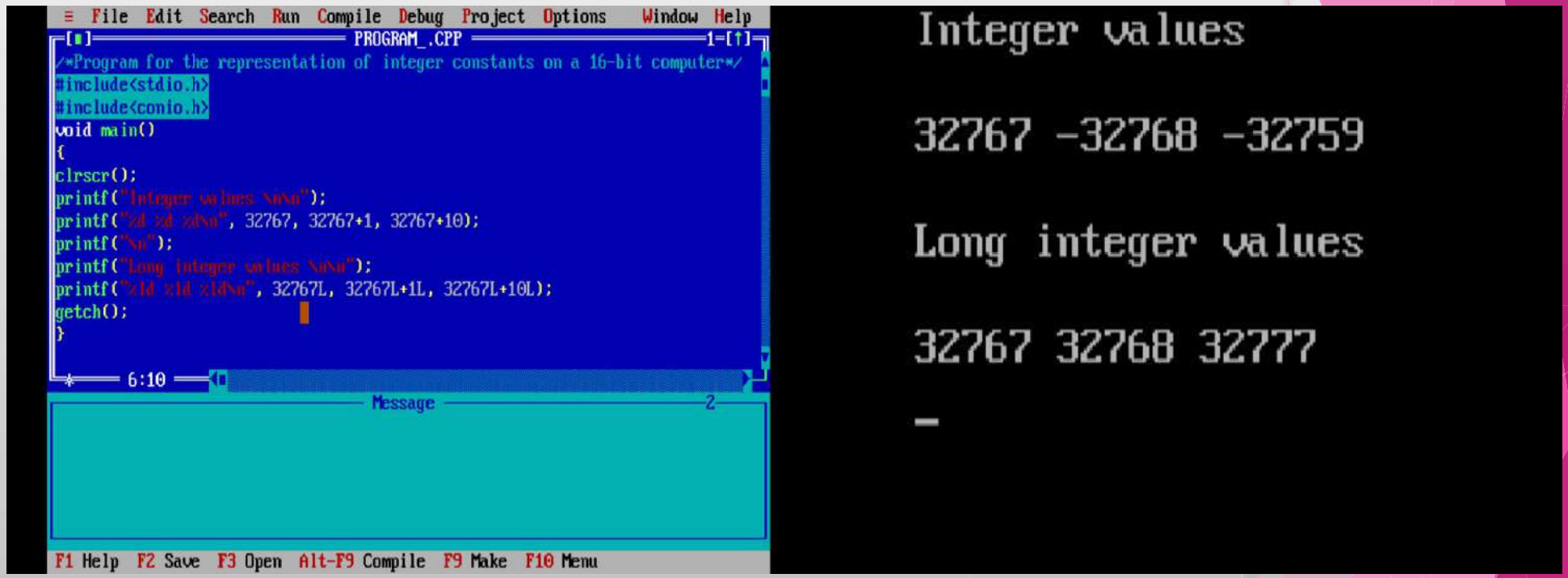
Integer Constants

- ***An Integer constant*** refer to a sequence of digits.
- ***Three types*** of integers, namely, ***decimal integer, octal integer, and hexadecimal integer***
- ***Decimal integers*** consists of a set of digits, ***0*** to ***9,*** preceded by an optional **-** or **+** ***sign***. e,g. -123 21 654321 +78
- Embedded spaces, commas, and non-digit characters are not permitted between digits. e.g. 15 7250 20,000 $249
- ***Octal integer*** constant consists of any combination of digits from the ***set 0*** to ***7, with a leading 0***. e.g. 037 0 0435
- ***Sequence*** of digits preceded by ***0x*** or ***0X*** is considered ***hexadecimal integer***. Include alphabets from ***A*** to ***F***. e.g. 0X2 0X9F 0Xbcd 0x
- Largest integer value stored is machine dependent. e.g. 32767 on 16-bit machines and 2,147,483,647 on 32 bit machines.
- ***Qualifiers*** such as ***U, L, and UL*** can be appended to store larger integer constants. e.g.

56789U	or	56789u	(unsigned integer)
987612347UL	or	987612347ul	(unsigned long integer)

Integer Constants contd.

- Problem: Write a program for the representation of integer constants on a 16-bit computer.



Real Constants

- Integer numbers are inadequate to represent distances, heights, temperatures, prices.
- These quantities contain fractional parts and are known as ***real or floating point*** constants with ***decimal notation***. e.g. 215.0 0.95 -.71 +.5
- Real number also expressed in ***exponential (or scientific)*** notation. e.g. value 215.65 may be written as 2.1565e2 (here e2 means multiply by 10^2).
- General notation is :

mantissa e exponent

- ***mantissa*** is either real number expressed in ***decimal notation*** or an integer.
- ***exponent*** is an integer number with an ***optional plus or minus sign***.
- Letter **e** separating **mantissa** and **exponent** is written either in lowercase or uppercase.
- Represent real number in floating point form. e.g. 0.65e4 12e-2 1.5e+5
- Exponential notations useful for representing numbers that are either very large or very small. e.g. 7.5E9 -3.68E-7

Real Constants contd.

- Floating-point constants are normally represented as double-precision quantities.
- The suffixes f or F may be used to force single-precision and l or L to extend double-precision.

Constant	*Valid?*	*Remarks*
698354L	Yes	Represents long integer
25,000	No	Comma is not allowed
+5.0E3	Yes	(ANSI C supports unary plus)
3.5e-5	Yes	
7.1e 4	No	No white space is permitted
-4.5e-2	Yes	
1.5E+2.5	No	Exponent must be an integer
$255	No	$ symbol is not permitted
0X7B	Yes	Hexadecimal integer

Single Character Constants

- Single character constant contains a single character enclosed within a pair of ***single*** quote marks. e.g. '5' 'X' ';' ' '
- **Note: Character constant *'5'* is not same as number *5***
- Character constants have integer values known as ASCII values. e.g. statement **printf("%d", 'a');** would print number **97.**
- Statement **printf("%c", 97);** would print letter 'a'.

ASCII Value	Character	ASCII Value	Character	ASCII Value	Character	ASCII Value	Character
000	NUL	027	ESC	054	6	081	Q
001	SOH	028	FS	055	7	082	R
002	STX	029	GS	056	8	083	S
003	ETX	030	RS	057	9	084	T
004	EOT	031	US	058	:	085	U
005	ENQ	032	blank	059	;	086	V
006	ACK	033	!	060	<	087	W
007	BEL	034	"	061	=	088	X
008	BS	035	#	062	>	089	Y
009	HT	036	$	063	?	090	Z
010	LF	037	%	064	@	091	[
011	VT	038	&	065	A	092	\
012	FF	039	'	066	B	093	]
013	CR	040	(	067	C	094	↑
014	SO	041	)	068	D	095	-
015	SI	042	*	069	E	096	←
016	DLE	043	+	070	F	097	a
017	DC1	044	,	071	G	098	b
018	DC2	045	–	072	H	099	c
019	DC3	046	.	073	I	100	d
020	DC4	047	/	074	J	101	e
021	NAK	048	0	075	K	102	f
022	SYN	049	1	076	L	103	g
023	ETB	050	2	077	M	104	h
024	CAN	051	3	078	N	105	i
025	EM	052	4	079	O	106	j
026	SUB	053	5	080	P	107	k

ASCII Value	Character	ASCII Value	Character	ASCII Value	Character	ASCII Value	Character
108	l	113	q	118	v	123	{
109	m	114	r	119	w	124	|
110	n	115	s	120	x	125	}
111	o	116	t	121	y	126	~
112	p	117	u	122	z	127	DEL

Note The first 32 characters and the last character are control characters; they cannot be printed.

String Constants

- ***String constant*** is a sequence of characters enclosed in ***double*** **quotes**.
- Characters may be letters, numbers, special characters and blank space. e.g.

"Hello!" "1987" "WELL DONE" "?...!" "5+3" "X"

- **Note: Character constant (e.g., 'X') is not equivalent to the single character string constant ("X").**

- Single character string constant does not have equivalent integer value while a character constant has an integer value.
- Character strings are often used in programs to build meaningful programs.

Backslash Character Constants

- C supports some special backslash character constants that are used in output functions. e.g. ***'\n' stands for newline character.***
- Note that each represents one character, although they consist of two characters and are known as ***escape sequences***.

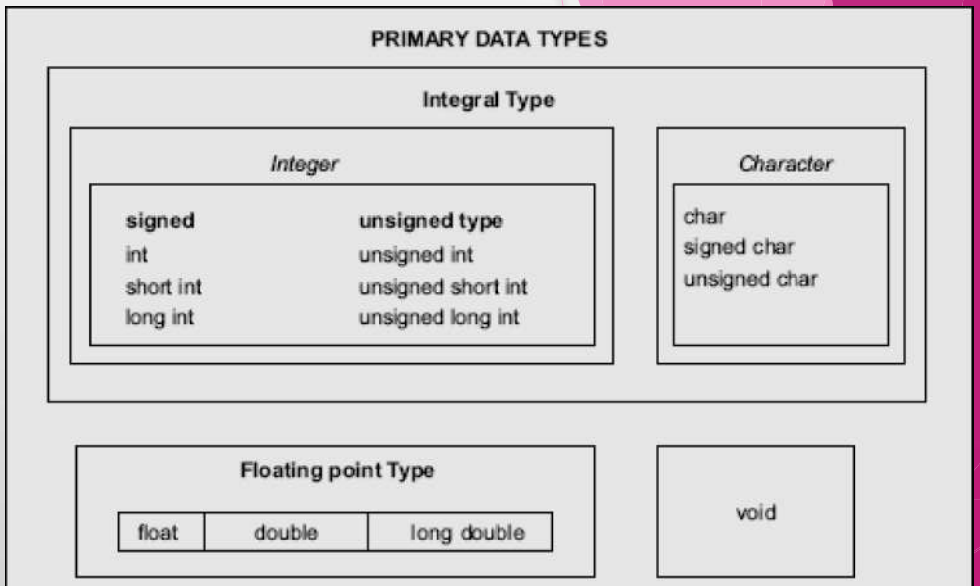
Constant	*Meaning*
'\a'	audible alert (bell)
'\b'	back space
'\f'	form feed
'\n'	new line
'\r'	carriage return
'\t'	horizontal tab
'\v'	vertical tab
'\''	single quote
'\"'	double quote
'\?'	question mark
'\\'	backslash
'\0'	null

Data Types

- C supports various data types.
- ***Data types*** variety allow the programmer to select appropriate data type needed for a particular application.
- ***ANSI C*** supports three classes of data types:

1. Primary (or fundamental) data types
2. Derived data types
3. User-defined data types

- All C compilers support five fundamental data types: ***integer (int), character (char), floating point (float), double-precision floating point (double) and void.***



Data type	Range of values
char	–128 to 127
int	–32,768 to 32,767
float	3.4e–38 to 3.4e+e38
double	1.7e–308 to 1.7e+308

Integer Types

- Integers are whole numbers with a range of values supported by a particular machine.
- Integers occupy one word of storage, and since the word sizes of machines vary (typically, 16 or 32 bits) the size of an integer that can be stored depends on the computer
- ***16 bit word length corresponds to range 0 to 65535 for unsigned and -32767 to +32768 (-2^{15} to $+2^{15}$ -1) for signed***
- ***signed int*** use 1 bit for sign and 15 bits for the magnitude of number.
- ***32 bit word length corresponds to range 0 to 4294967295 for unsigned and -2147483648 to +2147483647 (-2^{32} to $+2^{32}$ -1) for signed.***
- ***short int***, ***int***, and ***long int***, in ***signed*** and ***unsigned*** form are used to control number range
- ***short int*** represents fairly small integer values and require half amount of storage as a regular ***int***.
- ***long and unsigned*** integers to increase the range of values.

Integer Types contd.

Type	*Size (bits)*	*Range*
char or signed char	8	–128 to 127
unsigned char	8	0 to 255
int or signed int	16	–32,768 to 32,767
unsigned int	16	0 to 65535
short int or signed short int	8	–128 to 127
unsigned short int	8	0 to 255
long int or signed long int	32	–2,147,483,648 to 2,147,483,647
unsigned long int	32	0 to 4,294,967,295
float	32	3.4E – 38 to 3.4E + 38
double	64	1.7E – 308 to 1.7E + 308
long double	80	3.4E – 4932 to 1.1E + 4932

Floating Point Types

- Floating point (or real) numbers are stored in 32 bits (on all 16 bit and 32 bit machines) with 6 digits of precision.
- ***float*** keyword is used to define floating point numbers.
- ***double*** data type can be used to define the number when ***float*** is not sufficient
- ***double*** uses 64 bits giving precision of 14 digits.
- ***long double*** uses 80 bits of precision.
- Floating point numbers are represented within the computer's memory using IEEE standards

Void and Character Types

- ***void*** types has no values, used to specify the type of functions
- Functions that does not return any value to the calling is known as ***void***
- Also play the role of a generic type. e.g.: **void main(); void sum(int a, int b);**
- Single character can be defined as a character(char) type data.
- Characters are usually stored in 8bits (1 byte)
- ***signed char*** have values from ***-128 to 127***
- ***Unsigned char*** have values between ***0 to 255***.

Data type	*Keyword equivalent*
Character	char
Unsigned character	unsigned char
Signed character	signed char
Signed integer	signed int (or int)
Signed short integer	signed short int (or short int or short)
Signed long integer	signed long int (or long int or long)
Unsigned integer	unsigned int (or unsigned)
Unsigned short integer	unsigned short int (or unsigned short)
Unsigned long integer	unsigned long int (or unsigned long)
Floating point	float
Double-precision floating point	double
Extended double-precision floating point	long double

Variables

- ***Variable*** a data name used to store value.
- ***Unlike constants*** variable take different values at different times during execution.
- ***Variable name*** chosen that reflects the function or nature of program. ***e.g. Average, height, Total, Counter_1, class_strength***
- ***Variable names*** consists of letters, digits, and underscore (_) character subject to following conditions:

1. ***Must begin with a letter.*** Some systems permit ***underscore as the first character.***
2. ***ANSI standard recognizes*** a ***length of 31 characters***. (In ***C99***, ***at least 63 characters*** are ***significant***.)
3. ***Uppercase*** and ***lowercase*** are ***significant***. That is, the variable **Total** is not the same as **total** or **TOTAL**.
4. ***Should not be keyword.***
5. ***White spaces not allowed.***

Variable name	*Valid ?*	*Remark*
First_tag	Valid	
char	Not valid	char is a keyword
Price$	Not valid	Dollar sign is illegal
group one	Not valid	Blank space is not permitted
average_number	Valid	
int_type	Valid	Keyword may be part of a name

Variables Declaration

- ***Variable declaration*** must after assigning suitable variable name, and at the beginning of a statement block before they are used in a program.
- ***Declaration do two things:***

1. ***Inform compiler what is the name of the variable***
2. ***Specifies what type of data variable will hold***

- ***Primary Type Declaration:*** variable can store a value of any data type.

data-type v1,v2,…vn;

- Variables are separated by commas and must end with semicolon. e.g.

int count;

int number, total;

double ratio;

- **Qualifiers short, long, or unsigned** are used along with data types

Default Values of Constants

- Default integer constants representing ***int data types*** can be overwritten by specifying unsigned or long after the number (appending U or L)

Literal	Type	Value
+111	int	111
–222	int	–222
45678U	unsigned int	45,678
–56789L	long int	–56,789
987654UL	unsigned long int	9,87,654

- Default floating constants representing ***double data types***, and can be appended by specifying ***f or F*** for ***float*** and letter ***l*** or ***L*** for ***long double***

Literal	Type	Value
0.	double	0.0
.0	double	0.0
12.0	double	12.0
1.234	double	1.234
–1.2f	float	–1.2
1.23456789L	long double	1.23456789

User-Defined Type Declaration

- C language support ***type definition*** feature that allows user to define an identifier used to represent existing data type.
- User defined data type identifier used to declare variables. e.g.

  typedef type identifier;
- Here ***type*** refers existing data type and ***"identifier"*** *refers to the* "**new**" name given to the data type.
- ***typedef*** cannot create ***new data type.*** *e.g.*

  typedef int *units;*

  typedef float *marks;*
- ***units*** and ***marks*** can be used to define new variable which fall under same data type as original datatype. e.g.

  units batch1, batch;

  marks name1[50], name2[50]
- Main advantage of ***typedef*** is to increase program readability.
- ***Enumerated*** data type is another user-defined data type used to declare variable having one of the enclosed values.

  enum identifier{value1,value2,…valuen};

  enum identifier v1, v2, … vn;
- Enumerated variables can have only one value. e.g.

  enum day {Monday, Tuesday, … Sunday};

  enum day week_st, week_end;