CPE100
Computer Programming for Engineers

Lesson 2
C Constructs

Assc. Prof. Dr. Natasha Dejdumrong

KING MONGKUT'S UNIVERSITY
OF TECHNOLOGY THONBURI

Outline

- ★ Introduction
- ★ Character Set
- ★ Tokens
- ★ Keywords
- ★ Identifiers
- ★ Constants
- ★ Variables

Introduction

- ★ Every language has a set of symbols that are used to create tokens or words.
- ★ These tokens are used to create statements.
- ★ A statement is a meaningful dialog used to create a program.
- ★ Every programming language like C has a set of characters that are used to create tokens which in turn are used to create meaningful statements.
- ★ These statements constitute a program. Note that certain rules are to be followed to create tokens and compose statements using these tokens. These rules are called Syntax Rules or Grammar.

Character Set

- ★ The character set of C comprises of wide range of symbols that can be used to create tokens,
- ★ These symbols are available on all modern personal computers.
- ★ These characters can be grouped into the following categories:
  - ★ Letters
  - ★ Digits
  - ★ Special characters
  - ★ White Space

Letters	Digits	White Spaces
Upper Case A to Z	0 to 9	Blank Space
Lower Case a to z		Tab
		Carriage Return
		Line Feed

Character Set

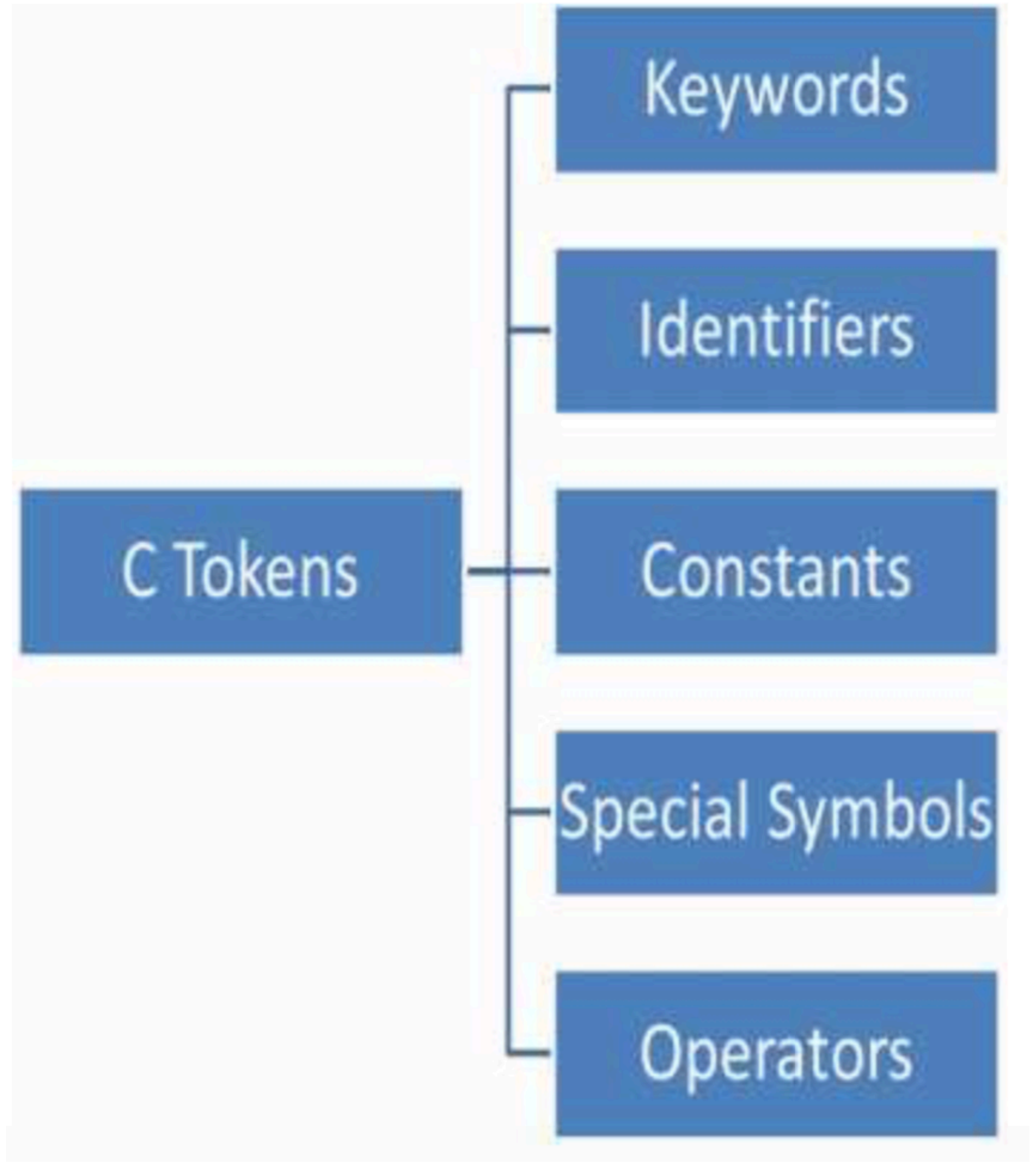
Letters	Digits	White Spaces
Upper Case A to Z	0 to 9	Blank Space
Lower Case a to z		Tab
		Carriage Return
		Line Feed

Special Characters		
, Comma	. Period	; Semicolon
: Colon	? Question Mark	‘ Apostrophe
“ Quotation Mark	! Exclamation Mark	| Vertical Bar
/ Slash	\ Backslash	~ Tilde
_ Underscore	$ Dollar Sign	% Percent Sign
# Number Sign	& Ampersand	^ Caret
* Asterisk	- Minus Sign	+ Plus Sign
< Opening Angle	> Closing Angle	(Left Parenthesis
) Right Parenthesis	[Left Bracket	] Right Bracket
{ Left Brace	} Right Brace	

Token

- ★ In C language the smallest unit of a program is called Token.
- ★ Tokens are created using the C character set.
- ★ These tokens are delimited from each other by white spaces just like words in English language are delimited by white spaces.
- ★ The Syntax Rule of the C language dictates how to create tokens in C and how to use these tokens to create meaningful statements.

C Token


C Tokens
Keywords
Identifiers
Constants
Special Symbols
Operators

Keyword

- ★ Keywords are those tokens or words whose meaning are already defined within the C language and cannot be changed.
- ★ As an analogy, in English language the meaning of verbs, adjectives, common nouns, and so on is pre-defined and can't be changed. These tokens of English language are its keywords.
- ★ Similarly, in C language some tokens are reserved and have some pre-defined meaning. These keywords serve as the basic building blocks for program statements.
- ★ The number of keywords present in C language is 32 as per ANSI standard.

32 Keywords in C

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

Identifier

- ★ Identifiers are user-defined tokens used to name variables, functions, etc.
- ★ An identifier consists of a sequence of letters and digits. Underscore is also used.
- ★ However, the first character of the identifier should a letter or an underscore.
- ★ Though there is logically no restriction on the length of the number of characters in an identifier, however for some compilers only 31 characters are significant to differentiate between 2 or more identifiers. It should be noted that identifier is case sensitive. An identifier "test" is different from another identifier "teSt".

Identifier

Identifier Name	Valid	Reason
First_name	Yes	
extern	No	extern is a keyword
Money$	No	Only Alphabets, Digits and Underscore are valid
extern_name	Yes	
First name	No	No space is allowed

Constant

- Constants are those tokens in C language whose value is fixed and does not change during the execution of a program. There are various types of constants in C.
  - Integer Constant
  - Real Constant
  - Character Constant
  - String Constant
  - Backslash Constant

Integer Constant

★ These constants refer to a sequence of digits. Any representation can be used to specify an integer constant.

Representation	**Value**
Decimal	1234
Hexadecimal	0x4D2
Octal	02322

Integer Constant

Real Constant

- ★ These constants refer to those numbers which contain fractional parts like 12.34. Such numbers are called real or floating point numbers.
- ★ A real constant can also be expressed in exponential or scientific notation.
- ★ As an example, 12.34 can also be expressed as 1234E-2.
- ★ In this form, the real number is expressed in terms of mantissa and exponent.

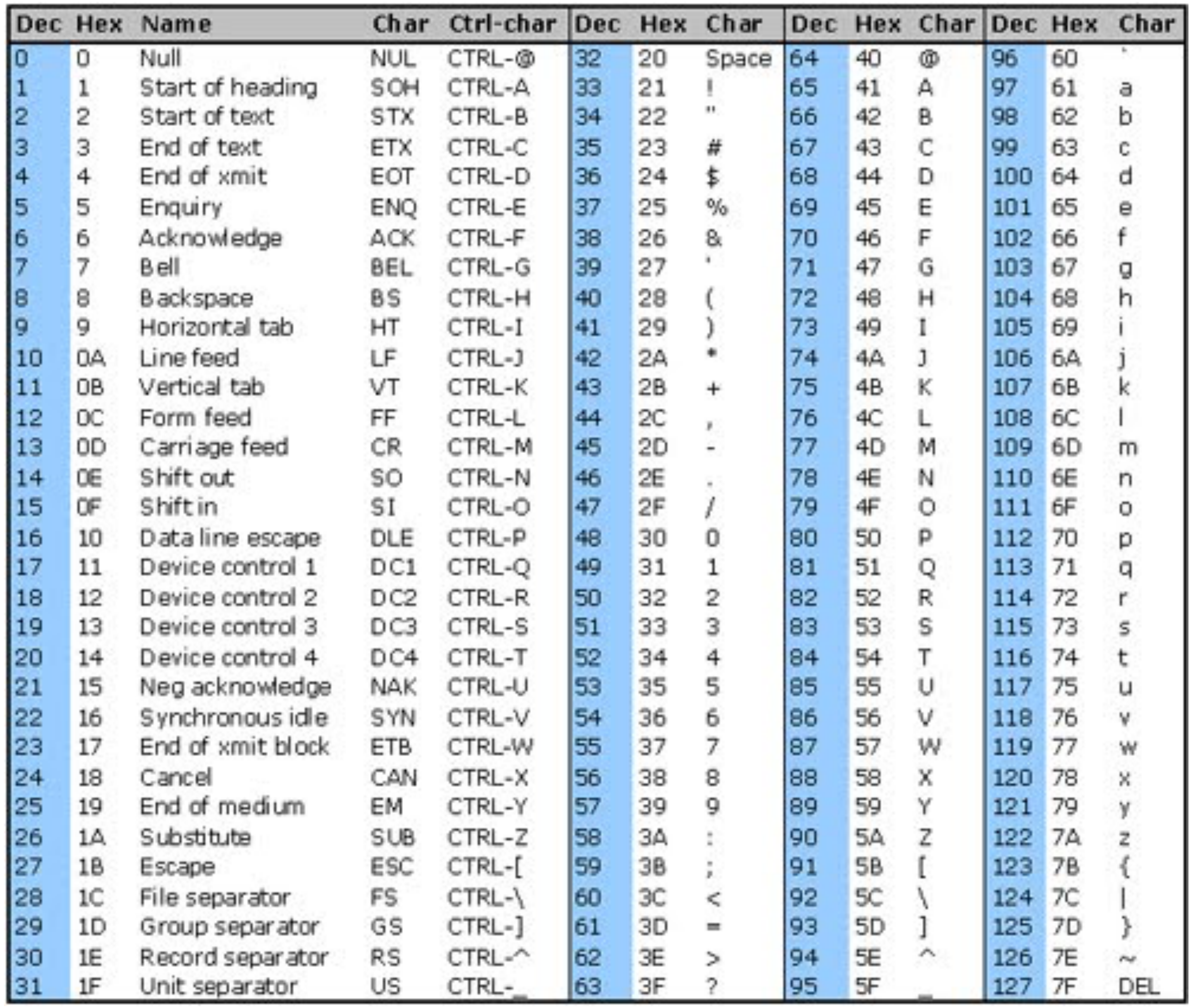
  The mantissa is either a real number or an integer and exponent is an integer number. Both can have an optional plus or minus sign. The letter E separating the mantissa and the exponent can also be written in lowercase.

Real Constant

Character Constant

- ★ These constants refer to those tokens which contain a single character enclosed within a pair of single quotes like **’W’**.
- ★ The character may be a letter, number, special character or blank space. As an example, **’7’** is a character constant while as 7 is an integer constant. Similarly, 7.0 can be treated as real constant.
- ★ Character constants are printable characters which are stored in memory as their ASCII code (American Standard Code for Information Interchange).

ASCII Code



Dec	Hex	Name	Char	Ctrl-char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	0	Null	NUL	CTRL-@	32	20	Space	64	40	@	96	60	`
1	1	Start of heading	SOH	CTRL-A	33	21	!	65	41	A	97	61	a
2	2	Start of text	STX	CTRL-B	34	22	"	66	42	B	98	62	b
3	3	End of text	ETX	CTRL-C	35	23	#	67	43	C	99	63	c
4	4	End of xmit	EOT	CTRL-D	36	24	$	68	44	D	100	64	d
5	5	Enquiry	ENQ	CTRL-E	37	25	%	69	45	E	101	65	e
6	6	Acknowledge	ACK	CTRL-F	38	26	&	70	46	F	102	66	f
7	7	Bell	BEL	CTRL-G	39	27	'	71	47	G	103	67	g
8	8	Backspace	BS	CTRL-H	40	28	(	72	48	H	104	68	h
9	9	Horizontal tab	HT	CTRL-I	41	29	)	73	49	I	105	69	i
10	0A	Line feed	LF	CTRL-J	42	2A	*	74	4A	J	106	6A	j
11	0B	Vertical tab	VT	CTRL-K	43	2B	+	75	4B	K	107	6B	k
12	0C	Form feed	FF	CTRL-L	44	2C	,	76	4C	L	108	6C	l
13	0D	Carriage feed	CR	CTRL-M	45	2D	-	77	4D	M	109	6D	m
14	0E	Shift out	SO	CTRL-N	46	2E	.	78	4E	N	110	6E	n
15	0F	Shift in	SI	CTRL-O	47	2F	/	79	4F	O	111	6F	o
16	10	Data line escape	DLE	CTRL-P	48	30	0	80	50	P	112	70	p
17	11	Device control 1	DC1	CTRL-Q	49	31	1	81	51	Q	113	71	q
18	12	Device control 2	DC2	CTRL-R	50	32	2	82	52	R	114	72	r
19	13	Device control 3	DC3	CTRL-S	51	33	3	83	53	S	115	73	s
20	14	Device control 4	DC4	CTRL-T	52	34	4	84	54	T	116	74	t
21	15	Neg acknowledge	NAK	CTRL-U	53	35	5	85	55	U	117	75	u
22	16	Synchronous idle	SYN	CTRL-V	54	36	6	86	56	V	118	76	v
23	17	End of xmit block	ETB	CTRL-W	55	37	7	87	57	W	119	77	w
24	18	Cancel	CAN	CTRL-X	56	38	8	88	58	X	120	78	x
25	19	End of medium	EM	CTRL-Y	57	39	9	89	59	Y	121	79	y
26	1A	Substitute	SUB	CTRL-Z	58	3A	:	90	5A	Z	122	7A	z
27	1B	Escape	ESC	CTRL-[	59	3B	;	91	5B	[	123	7B	{
28	1C	File separator	FS	CTRL-\	60	3C	<	92	5C	\	124	7C	|
29	1D	Group separator	GS	CTRL-]	61	3D	=	93	5D	]	125	7D	}
30	1E	Record separator	RS	CTRL-^	62	3E	>	94	5E	^	126	7E	~
31	1F	Unit separator	US	CTRL-_	63	3F	?	95	5F	_	127	7F	DEL

String Constant

- ★ These constants refer to those tokens which contain a string of characters enclosed within a pair of double quotes.
- ★ The characters may be letters, numbers, special characters and blank space.
- ★ Examples include **"Hello World!"**, **"123"**, and so on.
- ★ It should be noted "123" is a string constant while 123 is integer constant.

Backslash Constant

- These constants refer to those special character constants which contain a backslash and some character enclosed within a pair of single quotes.
- These constants are used in output functions and are known as escape sequences..

Backslash Constant

Backslash Character	Meaning
\a	alert (bell) character
\b	backspace
\f	form feed
\n	newline
\r	carriage return
\t	horizontal tab
\v	vertical tab
\\	backslash
\?	question mark
\'	single quote
\"	double quote

Variables

- ★ A Variable represents some memory location which is used to store some value.
- ★ Unlike constants, a variable may take different values at during times during execution.
- ★ In other words, the value within a variable can be changed; hence the name.
- ★ A Variable-name is an identifier token that follows the syntax rules for creating identifier, however programmers choose variable names in such way so as to reflect its function or nature in program.

Lesson 3
Variables and Data Types

Assc. Prof. Dr. Natasha Dejdumrong

Outline

- ★ Introduction
- ★ Variables and Data Types
- ★ Classification of Data Types
- ★ Integer Data Type
- ★ Floating Point Data Type
- ★ Character Data Type
- ★ Declaration of a Variable
- ★ Initialization of a Variable
- ★ User-Defined Data Types

Introduction

- ★ A computer processes data according to the instructions specified in a program and produces the output.
- ★ This processing is done by Central Processing Unit (CPU) of the computer which performs arithmetic operations such as addition, subtraction, multiplication and division, and logical operations such as comparing two numbers. numbers like
- ★ The data and programs are stored in primary memory (RAM) of a computer where from it is fetched by CPU.
- ★ This computer memory is made up of cells which are capable of holding information in the form of binary digits 0 and 1 (bits). This is the way information is stored in a memory just like we memorize numbers using digits 0 to 9.

Introduction

Introduction

- ★ Every such cell of computer memory has a unique memory address. In order to instruct CPU to process some data located in its memory we have to specify the address of that location.
- ★ However, it is very difficult for humans to remember the memory addresses in Declaration of a Variable 316976.
- ★ So instead of using some number to address any memory location which contains data, we label that location by a Variable.
- ★ This means using variables a programmer no more needs to be concerned about the actual location in memory where some data is stored, rather he/she can simply use the label i.e. Variable name, to access it.

Introduction

Introduction

- ★ In addition, the CPU does not access memory by individual bits; instead it accesses data in a collection of bits, typically 8 bits, 16 bit, 32 bit or 64 bit.
- ★ Furthermore, the data within that memory location may be of any type like integer, floating point, character, and so on.
- ★ To further qualify that memory location, we associate the type of data it contains, which also includes the width of data.
- ★ C programming language provides a rich set of Data Types which in association with variables can be used to label some memory location, specify the type of data it contains and width of that data.

Variables and Data Types

- ★ Variables are a way of reserving memory to hold some data and assign names to them so that we don't have to remember the numbers like 316976 and instead we can use the memory location by simply referring to the variable name.
- ★ Every variable is mapped to a unique memory address.
- ★ However, this is not enough. A variable can contain different types of data and each type may have different space requirement.
- ★ Thus as mentioned earlier we also specify the data type of a variable which signifies the type of data it contains and the space it requires to hold it.

Variables and Data Types

- ★ Having a variety of data types at the disposal of programmer allows him to make appropriate data type selection as per the nature of computation and type of machine.
- ★ In addition, the operations legitimate for one type of data may be invalid for some other type.
- ★ Having specified the data type of a variable allows the compiler to make compile time type checking in order to avoid run time failures.

Classification of Data Types

- ★ The wide variety of data types supported by C language can be classified into 3 classes:
  - ★ 1. Primary Data Types
  - ★ 2. User-defined Data Types.
  - ★ 3. Derived Data Types.

C supports 4 primary data types which include **integer, character, floating point** and **double-precision floating point**.

C also supports creating user-defined data types using keywords **typedef** and **enum.**

Integer Data Type

★ The Integer data type is used to specify a variable that will contain only integer values,
i.e. any positive or negative number without fractional part.

★ The keyword used by C language to specify some variable as integer is int.

★ The width of memory allocated to an integer variable is generally one word.

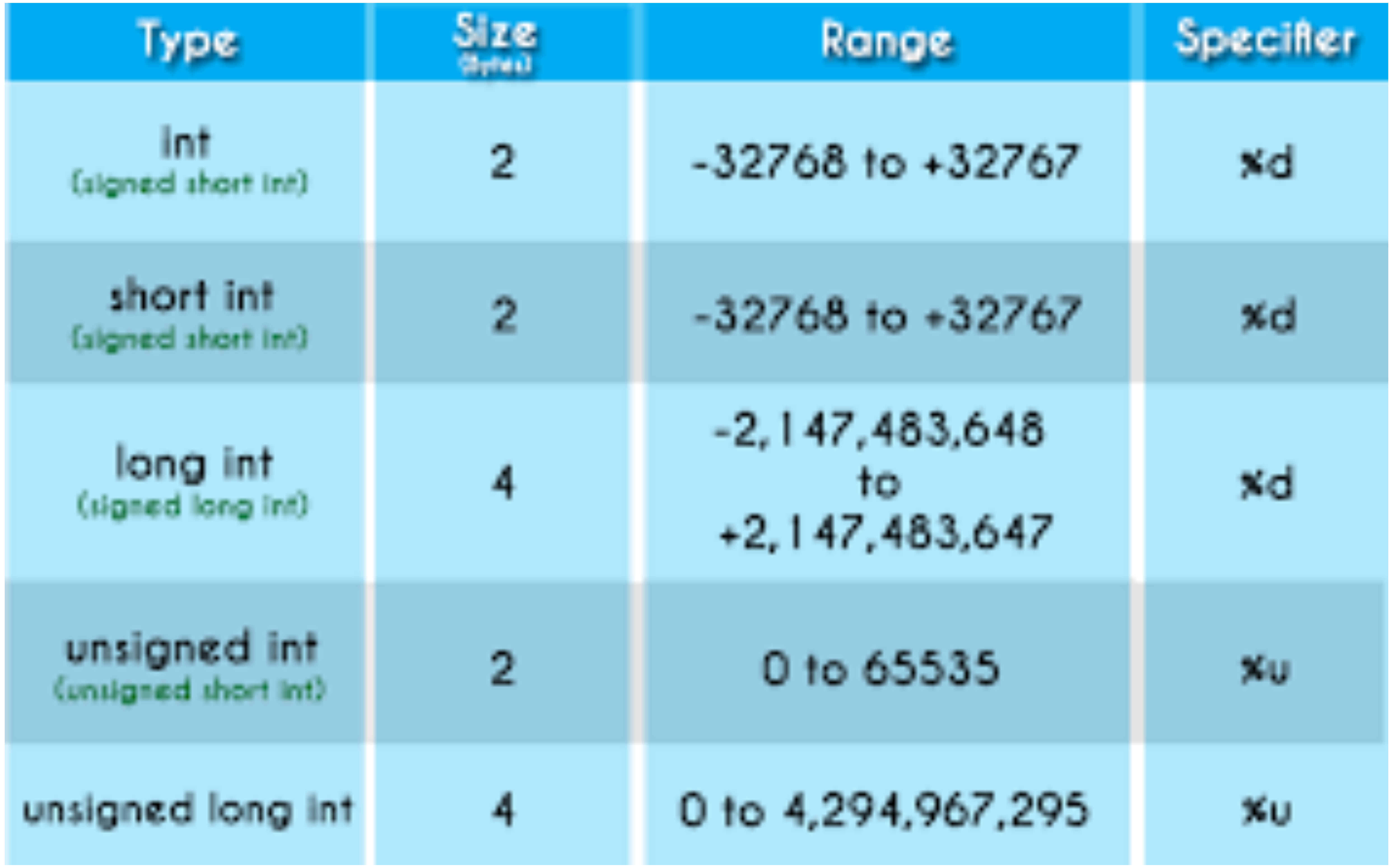
★ However, the word length varies from machine to machine.
For a 16-bit machine, the word length is 16-bits.

minimum value of -32768 and a maximum value of 32767 (-2^{15} to $+2^{15}-1$).

Integer Data Type

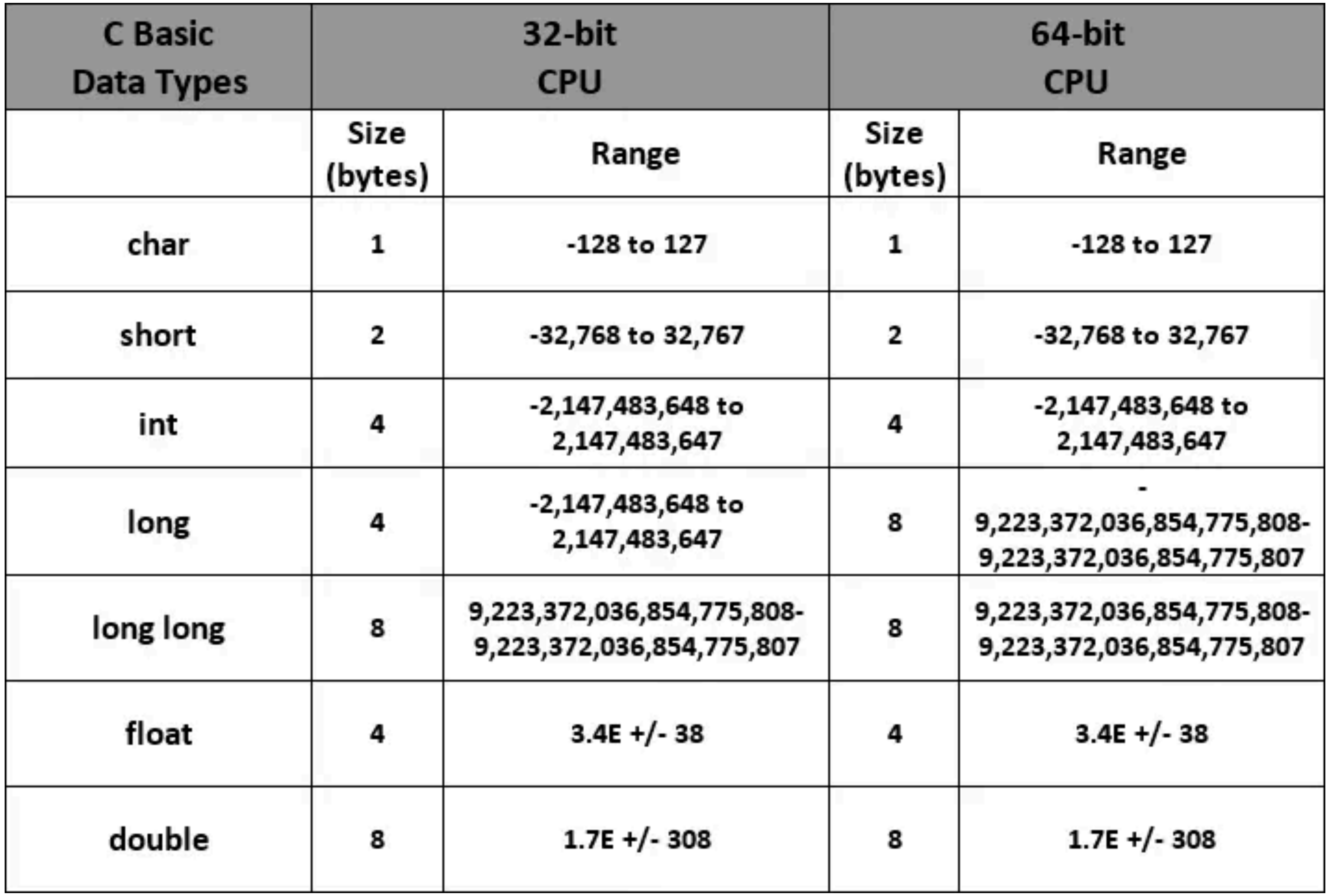
Data Type	**Width (in bits)**	**Range**
int or **signed int**	16	-32768 to 32767
unsigned int	16	0 to 65535
short int or **signed short int**	8	-128 to 127
unsigned short int	8	0 to 255
signed long int or **long int**	32	-2147483648 to 2147483647
unsigned long int	32	0 to 4294967295

Integer Data Type



Type	Size (Bytes)	Range	Specifier
int (signed short int)	2	-32768 to +32767	%d
short int (signed short int)	2	-32768 to +32767	%d
long int (signed long int)	4	-2,147,483,648 to +2,147,483,647	%d
unsigned int (unsigned short int)	2	0 to 65535	%u
unsigned long int	4	0 to 4,294,967,295	%u

Integer Data Type



C Basic Data Types	32-bit CPU		64-bit CPU	
	Size (bytes)	Range	Size (bytes)	Range
char	1	-128 to 127	1	-128 to 127
short	2	-32,768 to 32,767	2	-32,768 to 32,767
int	4	-2,147,483,648 to 2,147,483,647	4	-2,147,483,648 to 2,147,483,647
long	4	-2,147,483,648 to 2,147,483,647	8	- 9,223,372,036,854,775,808- 9,223,372,036,854,775,807
long long	8	9,223,372,036,854,775,808- 9,223,372,036,854,775,807	8	9,223,372,036,854,775,808- 9,223,372,036,854,775,807
float	4	3.4E +/- 38	4	3.4E +/- 38
double	8	1.7E +/- 308	8	1.7E +/- 308

Floating-Point Data Type

Data Type	Width (in bits)	Range
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

Character Data Type

Data Type	Width (in bits)	Range
char or **signed char**	8	-128 to 127
unsigned char	8	0 to 255

Declaration of a Variable

Data-type Variable-name;

```
int x;

float y;

int abc;

float def;
```

abc is the variable name

Data-type Variable-name1[, … Variable-nameN];

```
int x, abc;

float y, def;
```

Declaration of a Variable

Declaration	Status
int x, y, z, abc, def, pgdca, qwert;	Valid

float x;
 	
int y;
	Valid

char c;	

float x, y;
 	
int x, y;
	Invalid; because a variable can't be of two types

int x; int x, y, z;	Invalid; because a single variable can't be declared twice
int 123x;	Invalid variable-name

Initialization of a Variable

- ★ After a variable is declared it may be required to initialize it. Initialization of a variable means assigning some value to it.
- ★ However, technically there is no problem with using a variable without initializing it.
- ★ But logically it is wrong. When a variable is declared but not initialized it contains some random bogus but valid value.
- ★ As an example, if a variable x is declared as int, then x may contain say 1278 value before its initialization. The value is bogus but valid.

Initialization of a Variable

1. First, during declaration we can assign some value to it. As an example,

   ```
   int x = 1234;
   ```

   This way variable *x* is initialized with value 1234 when it is declared as integer.

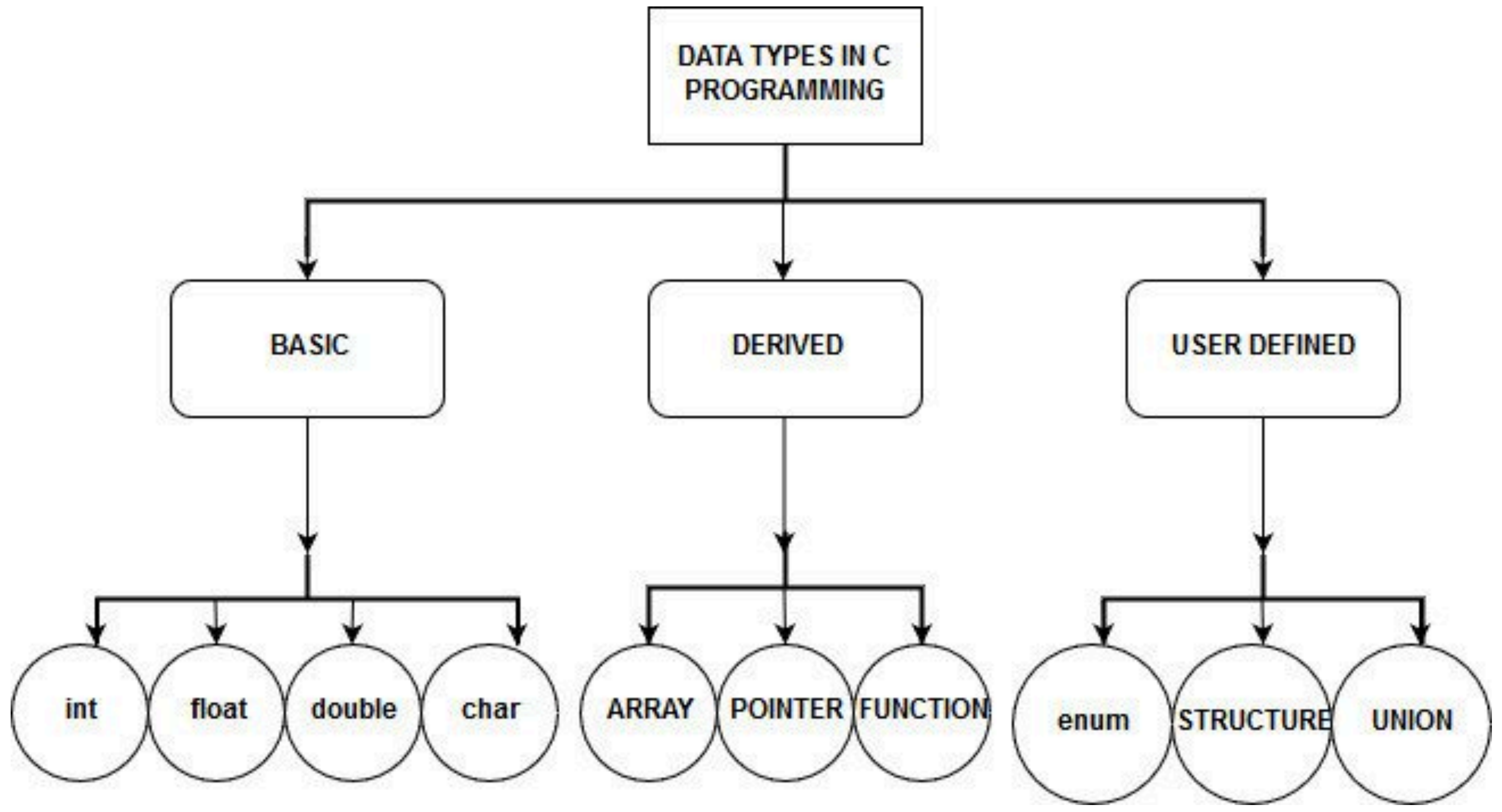
2. Second, after the variable is declared we can assign it some value in a separate statement. As an example,

   ```
   int x;

   x = 1234;
   ```

User-Defined Data Types

★ C language allows a programmer to create user defined data types. It provides 2 keywords to do so; typedef and enum.



typedef

★ typedef allows a programmer to define an identifier that would represent an existing data type. Therefore, the identifier so defined can be later used to declare variables.

```
typedef data-type identifier;
```

★ The existing data type can belong to any class of data types; primary, derived and even user-defined. It should be noted that the user-defined data type so created does not actually create a new data type, rather it creates a new name for the data type. As an example,

```
typedef int rollno;
```

typedef

- This will create another name, rollno, for int data type. Therefore, the new name can also be used to declare variables and it is as good as the existing data type. It is worth mentioning here that int keyword still exists and thus can be used to declare variables.

Hence, the statement

```
rollno student1, student2;
```

is as good as statement

```
int student1, student2;
```

typedef

★ One may argue what for can this be used?

★ The answer is using this mechanism the readability of a program is enhanced.

enum

★ enum is an enumerated data type which can be used to declare variables that can have one of the values enclosed within the braces. The general syntax to do so is as follows:

enum *identifier* *{ value1,* *value2,* *....* *valuen}* **;**

★ Now, the identifier can be used to declare variables and these so declared variables can only hold one of the values within the curly braces. As an example,

enum weekday { mon, tue, wed, thu, fri };

enum

★

```
enum weekday { mon, tue, wed, thu, fri };
```

```
enum weekday { mon=1, tue, wed, thu, fri };
```

Questions and Answers