Token

Structured Programming Language (CSE-1271)

Course Instructor : Mohammed Mamun Hossain
Assistant Professor, Dept. of CSE, BAUST

Outline

1. Simple C Program
2. Token (identifier, variable, constant)
3. Basic Data type of C Program
4. Statements
5. Expressions
6. Errors

Tokens

Computers aren't very smart. Sure, they can do a lot of math or help you search the Internet. But, if you asked a computer to vacuum the house for you, could it do it? If you asked a computer to draw a picture of a bird for you, would it? A computer would have no idea about what you're saying.

Because computers don't understand English, you have to give them instructions in special computer languages that computers can understand.

```
#include<stdio.h>

int main()
{
    printf("Hellow World\n");

    return 0;
}
```

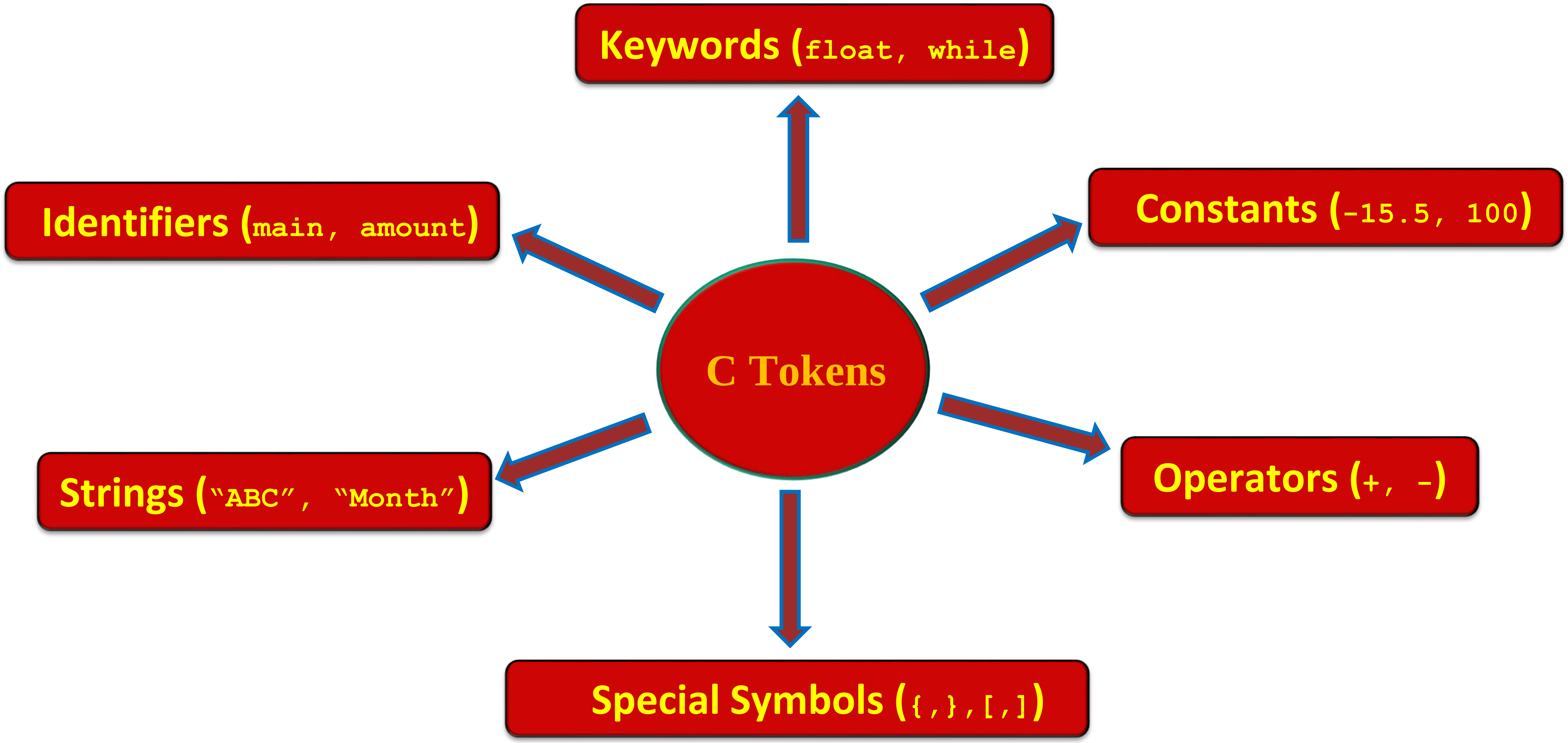
In a passage of text, individual words and punctuations marks are called tokens

In C program, the smallest individual units are known as C tokens.

Look the Tokens

Look C Tokens

C Tokens



Keywords (float, while)
Identifiers (main, amount)
Constants (-15.5, 100)
C Tokens
Strings ("ABC", "Month")
Operators (+, -)
Special Symbols ({,},[,])

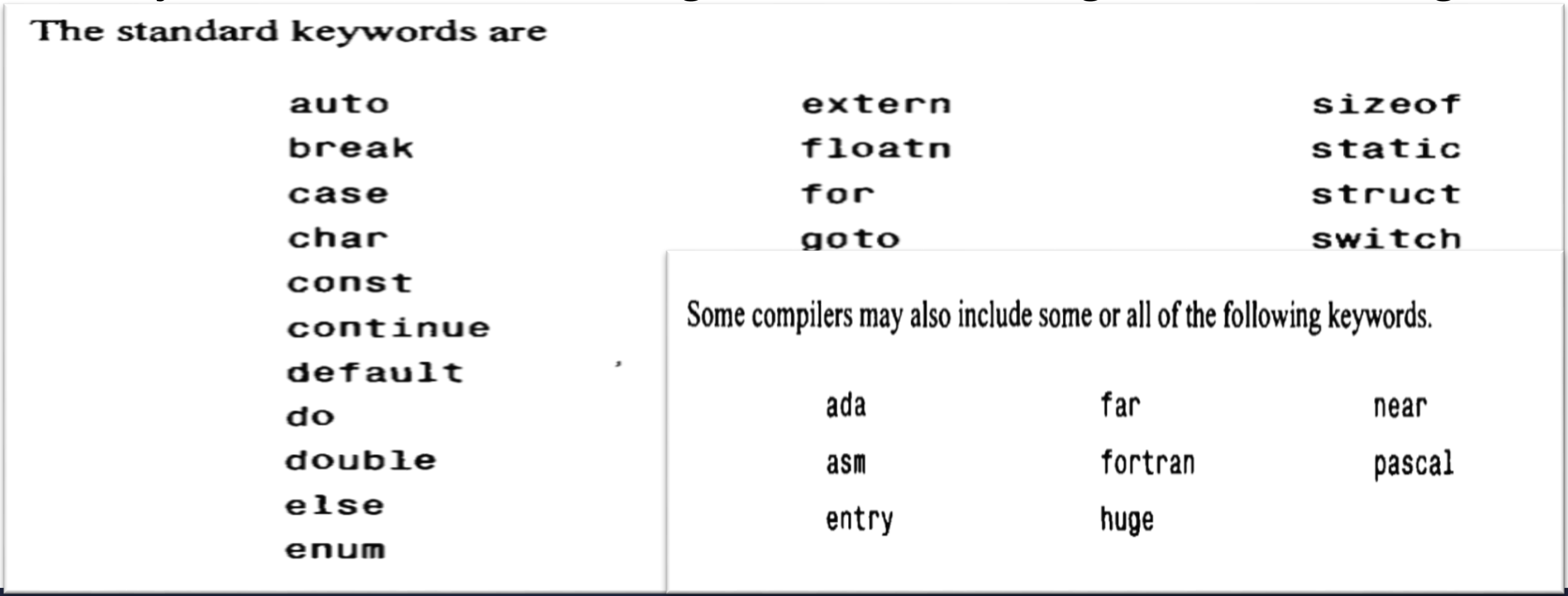
Keywords & Identifier

Every C word is classified as

either a *keyword*

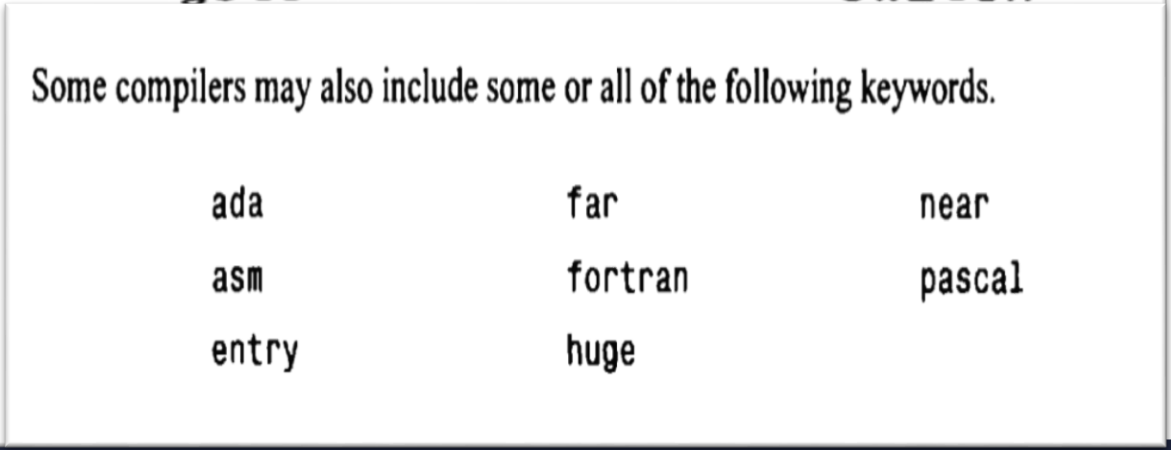
Or an *identifier*

All keywords have fixed meanings and these meanings cannot be changed.



The standard keywords are

auto	extern	sizeof
break	floatn	static
case	for	struct
char	goto	switch
const		
continue		
default		
do		
double		
else		
enum		



Some compilers may also include some or all of the following keywords.

ada	far	near
asm	fortran	pascal
entry	huge	

Keywords & Identifier

Identifiers are names that are given to various program elements, such as

- ❖ variables,
- ❖ functions and
- ❖ arrays.

These are user defined names. Rules for identifiers given bellow:

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.

Check valid identifier

number	✓	1stBoy	✗
addition2value	✓	Good-girl	✗
me123	✓	this_is_long_name	✓
_price1	✓	UnIvErSiTy	✓

Why invalid?

`4th`	✗	The first character must be letter *
`"x"`	✗	Illegal characters (")
`Order-no`	✗	Illegal characters (-)
`My variable`	✗	Illegal characters (blank space)

Variables (Identifiers)

- Variables consist of letters and digits, in any order, except that the first character must be a letter.
- Both upper-and lowercase letters are permitted, though common usage favors the use of lowercase letters for most types of variables.
- Upper- and lowercase letters are not interchangeable (i.e., an uppercase letter is not equivalent to the corresponding lowercase letter.)
- The underscore character (_) can also be included, and is considered to be a letter.
- An underscore is often used in the middle of an variable.
- A variable may also begin with an underscore, though this is rarely done in practice.
- Case-sensitive, **COUNT** and **count** are not same.

Variables Declaration

- ❖ Variable is a named memory location that can hold various values.
- ❖ All variables must be declared before they can be used.
- ❖ When we declare a variable, we tell the compiler what type of variable is being used.
- ❖ A declaration associates a group of variables with a specific data type.

How to Declare Variables?

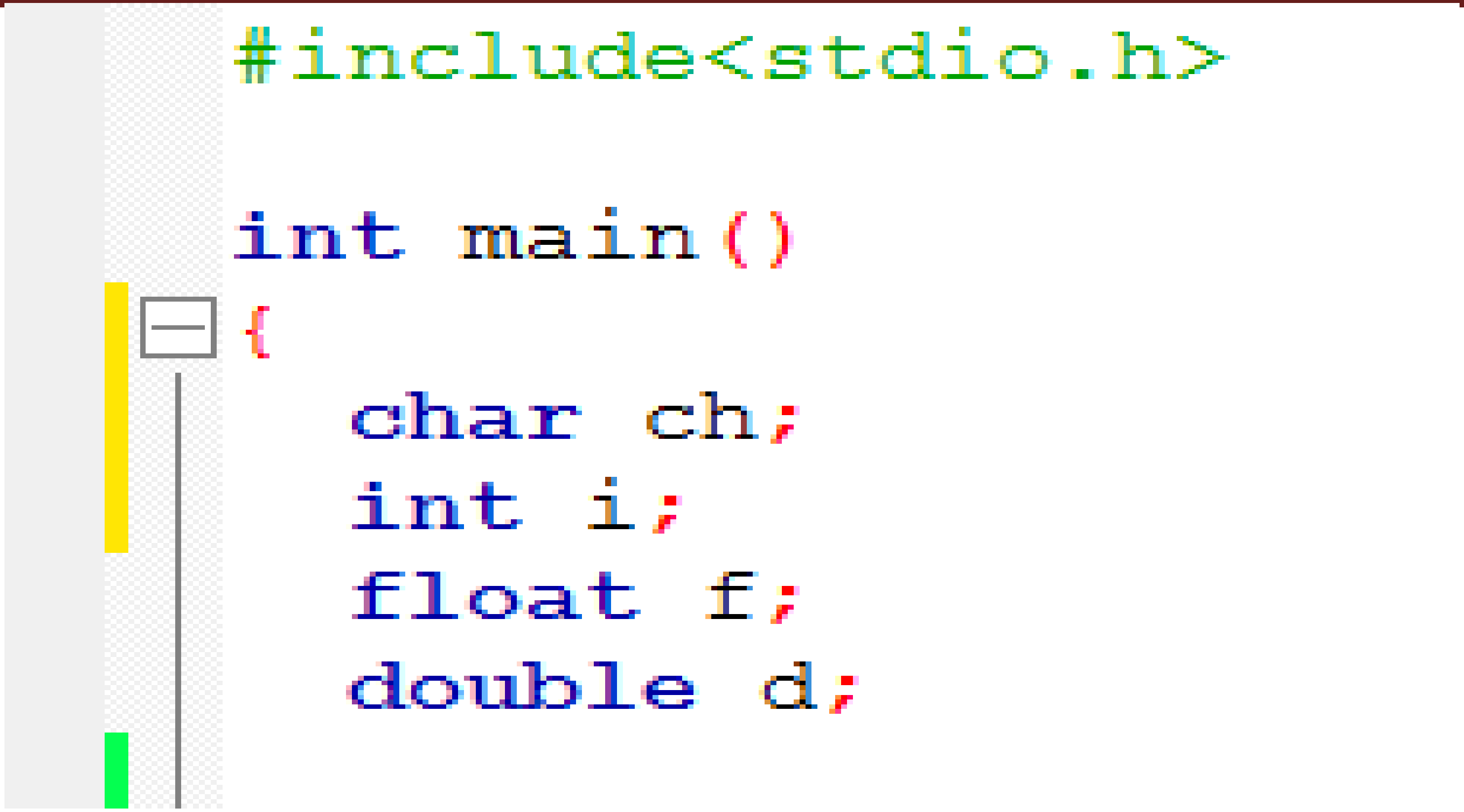
- ✓ To declare a variable, use this general form:

type var-name;

- ✓ In C, a variable declaration is a statement and it must end in a semicolon (;).

Let see the example…

Variables Declaration



```
#include<stdio.h>

int main()
{
    char ch;
    int i;
    float f;
    double d;
```

- ❖ Inside a function (local variable)
- ❖ Outside all functions (global variable)
- ❖ As function’s parameter

Constants

- There are four basic types of constants in C.

1. integer constants,
2. floating-point constants,
3. character constants and
4. string constants.

Numeric type constant

Integer Constants

- An integer constant is an integer-valued number.
- Thus it consists of a sequence of digits.
- Integer constants can be written in three different number systems: decimal (base 10), octal (base 8) and hexadecimal (base 16).

Several valid decimal integer constants are shown below:

```
0      1      743   5280   32767      9999
```

Integer Constants

The following decimal integer constants are written incorrectly for the reasons stated.

`12,245`	illegal character (,).
`36.0`	illegal character (.).
`10 20 30`	illegal character (blank space).
`123-45-6789`	illegal character (-)
`0900`	the first digit cannot be a zero.

Floating Point Constants

- A floating-point constant is a base-10 number that contains either a decimal point or an exponent (or both).

- Several valid floating-point constants are shown below:

```
0.          1.          0.2         827.602
50000.      0.000743    12.3        315.0066
2 E-8       0.006e-3    1.6667E+8   .12121212e12
```

Character Constants

- A character constant is a single character, enclosed in apostrophes (i.e., single quotation marks).
- Several character constants are shown below.

```
'A'   'X'      '3'   '?'   ' '
```

Notice that the last constant consists of a blank space, enclosed in apostrophes.

Character Constants

Character constants have integer values that are determined by the computer's particular character set.

Some examples are:

Constant	Value	Constant	Value
'A'	65	‘%’	37
‘Z’	90	‘0'	48
‘a’	97	‘ ‘	32
‘z’	122	'7'	54

See the ASCII character set…

Character Constants

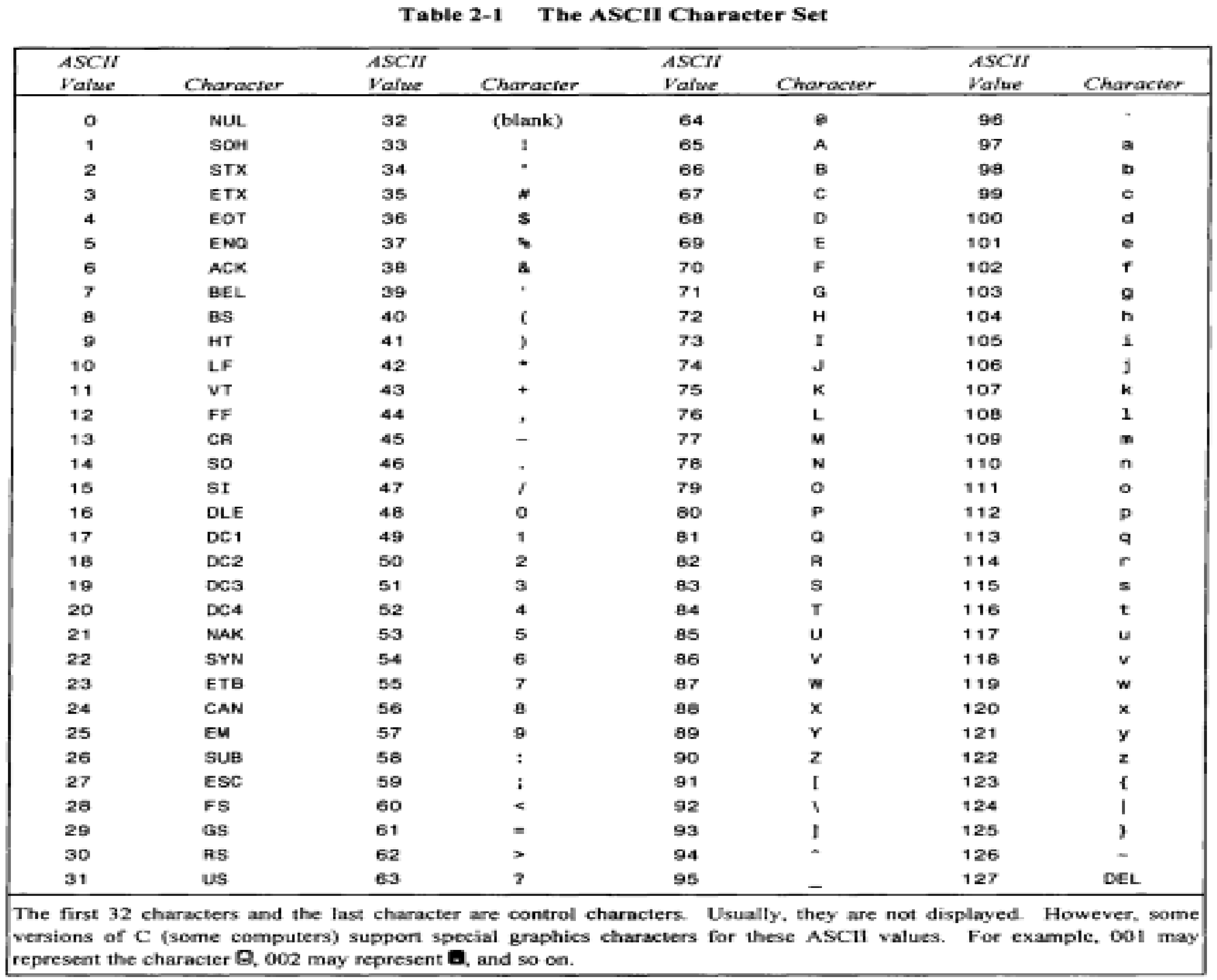


Table 2-1 The ASCII Character Set

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

The first 32 characters and the last character are control characters. Usually, they are not displayed. However, some versions of C (some computers) support special graphics characters for these ASCII values. For example, 001 may represent the character ☺, 002 may represent ☻, and so on.

String Constants

- A string constant consists of any number of consecutive characters (including none), enclosed in (double) quotation marks.
- Several string constants are shown below:

```
"green"                 "Washington,  D.C. 20005H"
"270-32-3456"           "$19.95"
"CORRECT ANSWER IS:"    "2* ( I+3)/J‶
"Line 1\nLine 2"        "  "
```

Symbolic Constants

- A symbolic constant is a name that substitutes for a sequence of characters.
- The characters may represent a numeric constant, a character constant or a string constant.
- Thus, a symbolic constant allows a name to appear in place of a numeric constant, a character constant or a string.
- When a program is compiled, each occurrence of a symbolic constant is replaced by its corresponding character sequence.
- Symbolic constants are usually defined at the beginning of a program.
- The symbolic constants may then appear later in the program in place of the numeric constants, character constants, etc. that the symbolic constants represent.
- A symbolic constant is defined by writing

```
#define   name   value/text
```

Escape Sequences

- Certain nonprinting characters, as well as the backslash (\) and the apostrophe (‘), can be expressed in terms of escape sequences.

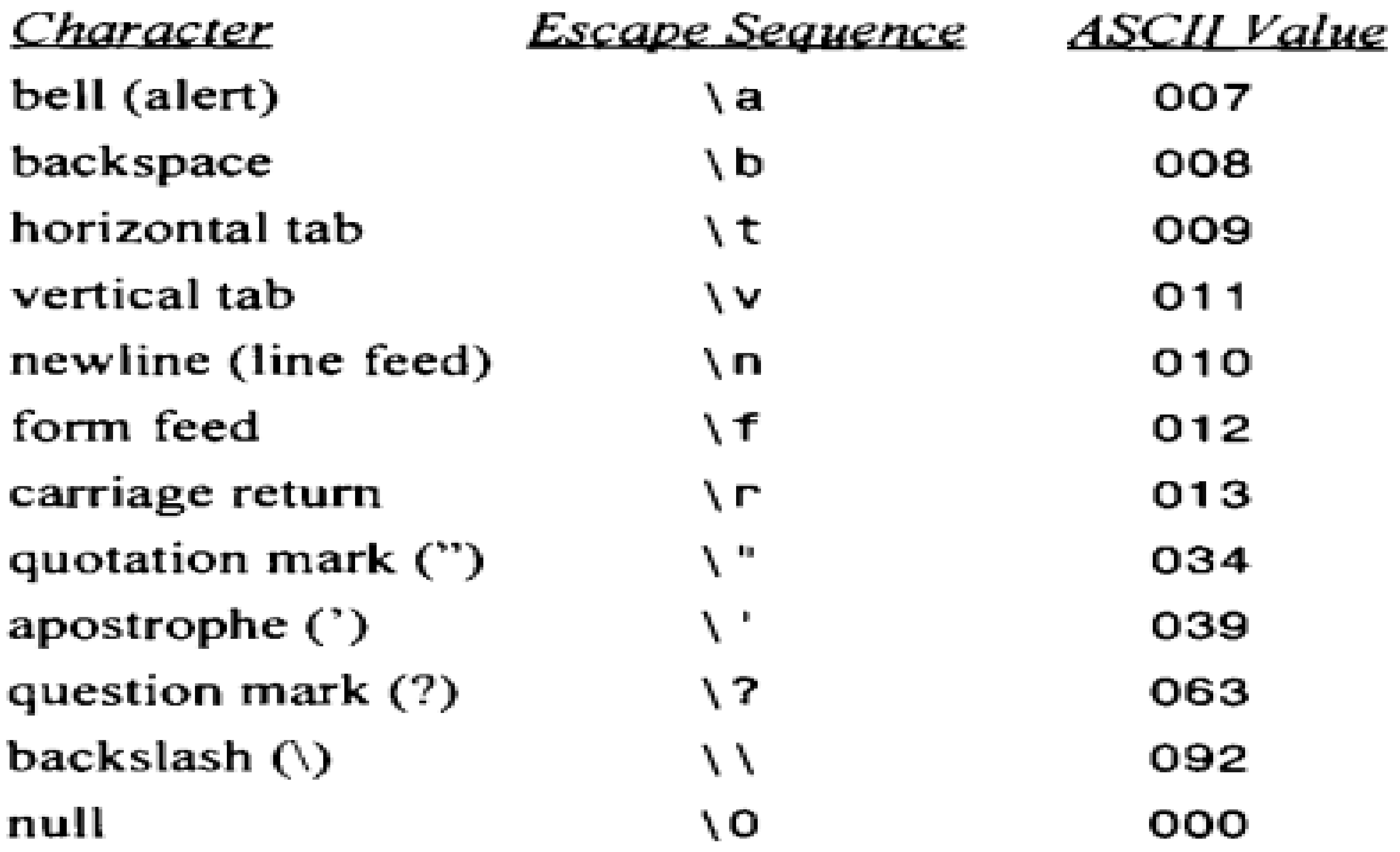
- An escape sequence always begins with a backward slash and is followed by one or more special characters.

- For example, a line feed (LF), which is referred to as a newline in C, can be represented as \n.

- Such escape sequences always represent single characters, even though they are written in terms of two or more characters.

Let see the example…

Escape Sequences



Character	*Escape Sequence*	*ASCII Value*
bell (alert)	\a	007
backspace	\b	008
horizontal tab	\t	009
vertical tab	\v	011
newline (line feed)	\n	010
form feed	\f	012
carriage return	\r	013
quotation mark (")	\"	034
apostrophe (')	\'	039
question mark (?)	\?	063
backslash (\\)	\\\\	092
null	\0	000

Data types

ANSI C supports classes of data types.

1. Primary (or fundamental) data types (`int, char, float, double, void`)
2. Derived data types (`array, functions, structures, pointers`)
3. User-defined data types (`typedef int unit`)

Type	Keyword	format Specifier	Memory Requirements
Character data	char	**%c**	1 Byte
Signed whole numbers	int	**%d**	2 or 4 Byte
Floating-point numbers	float	**%f**	4 Byte
Double-precision floating-point number	double	**%lf**	8 Byte
valueless	void		---

Statements

- ❖ A statement causes the computer to carry out some action.
- ❖ There are three different classes of statements in C.
  1. expression statements,
  2. compound statements and
  3. control statements.

An expression statement consists of an expression followed by a semicolon. Several expression statements are shown below.

```
a = 3;
c = a+b;
i++;
printf("Area = %f”, area);
;
```

Assign Value to the Variable

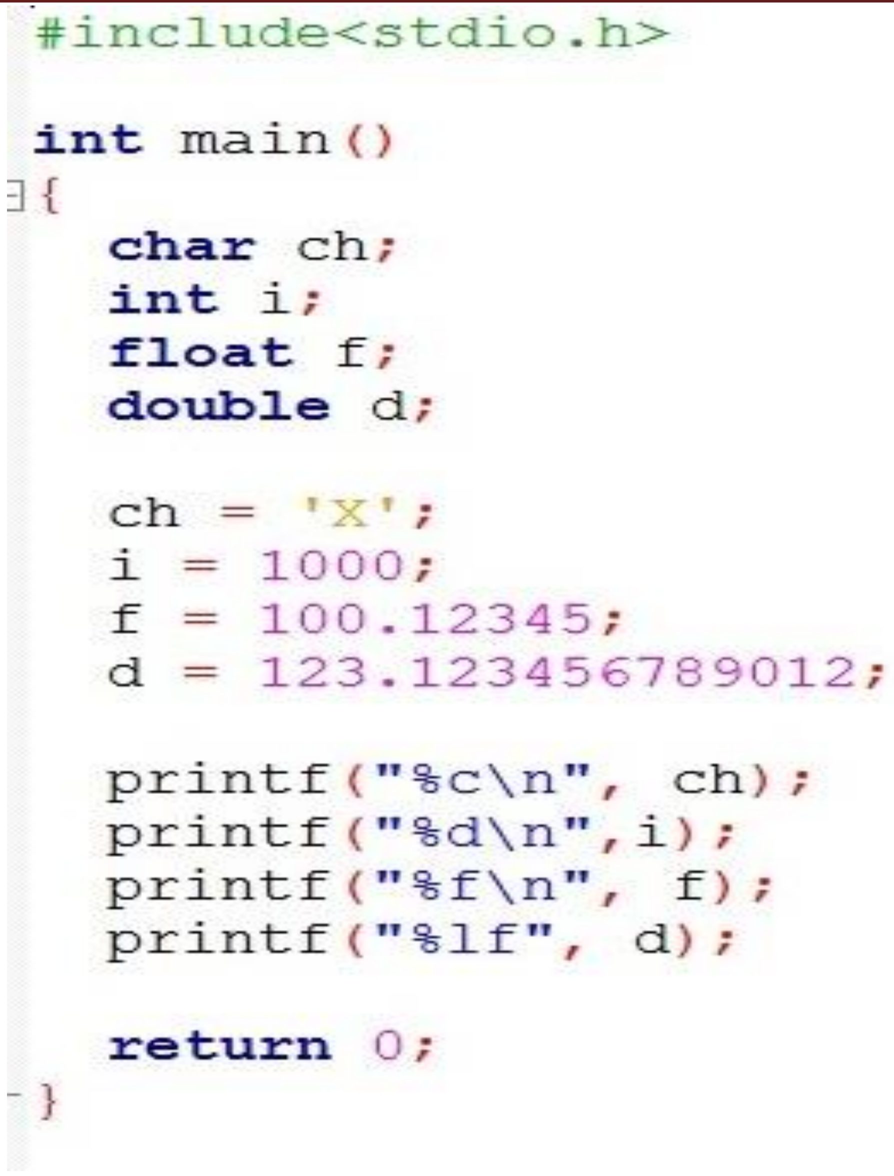
To assign a value to a variable, put its name to the left of an equal sign (=).

Put the value you want to give the variable to the right of the equal sign.

It is a statement, so end with a ‘;’

variable = value ;

Assign Value to the Variable



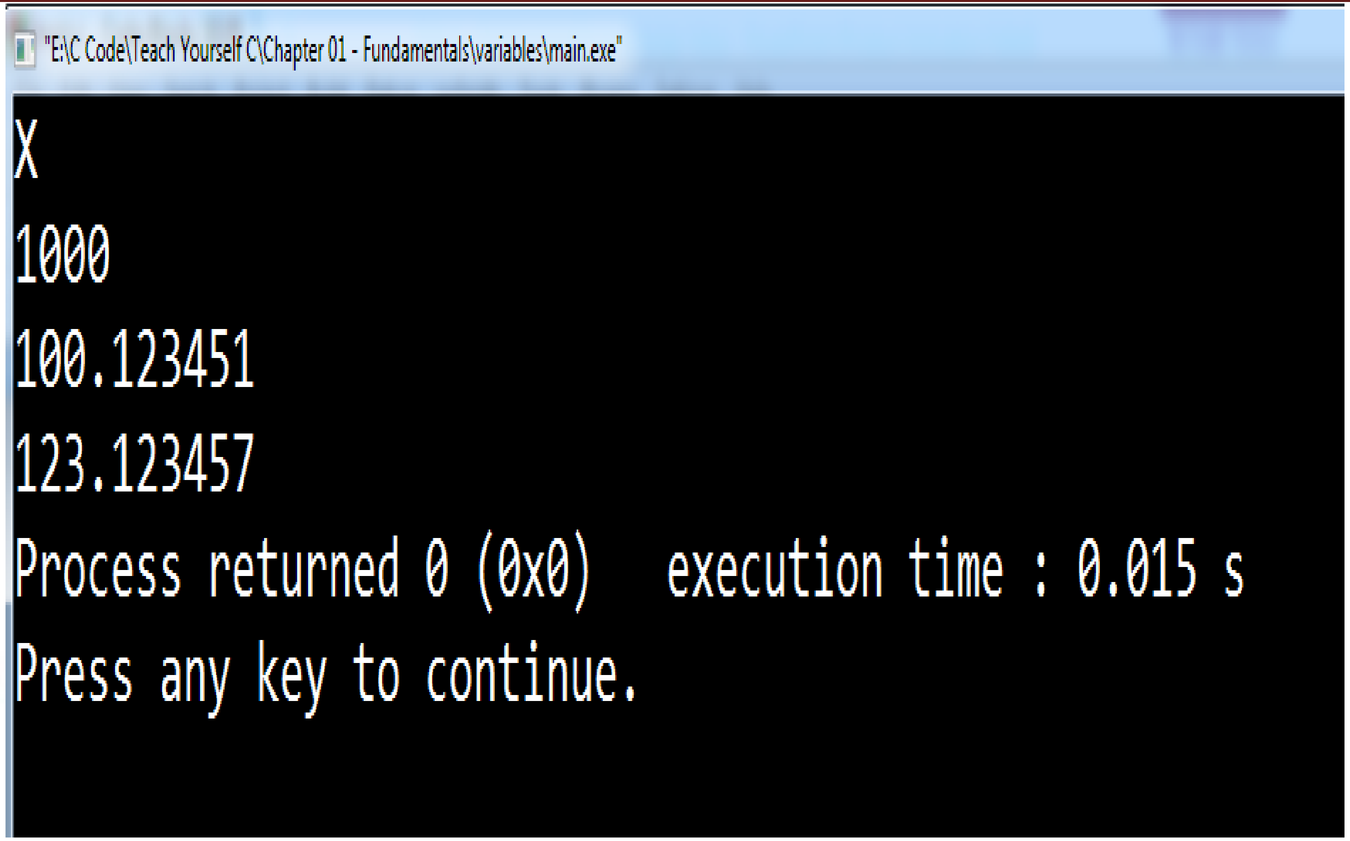
```c
#include<stdio.h>

int main()
{
    char ch;
    int i;
    float f;
    double d;

    ch = 'X';
    i = 1000;
    f = 100.12345;
    d = 123.123456789012;

    printf("%c\n", ch);
    printf("%d\n",i);
    printf("%f\n", f);
    printf("%lf", d);

    return 0;
}
```



"E:\C Code\Teach Yourself C\Chapter 01 - Fundamentals\variables\main.exe"

```
X
1000
100.123451
123.123457
Process returned 0 (0x0)   execution time : 0.015 s
Press any key to continue.
```

Expressions

An expression is a combination of operators and operands.

C expressions follow the rule of algebra:

Expression	Operator
Arithmetic Expression	+, -, *, /, %
Logical Expression	AND, OR, NOT
Relational	==, !=, <, >, <=, >=

```
a + b
x = y
t = u + v
x <= y
++j
```

expressions

```
a = 6;
c = a + b;
++j;
```

expression statements

Errors

A C program may have one or more of four types of errors:

- ✓Syntax errors (Compiler errors or Compile-time errors)
- ✓Linker Errors
- ✓Runtime errors
- ✓Logic errors

Syntax Errors

Syntax errors represent *grammar errors* in the use of the programming language. Common examples are:

- ✓Misspelled variable and function names
- ✓Missing semicolons
- ✓Unmatched parentheses, square brackets, and curly braces
- ✓Using a variable that has not been declared
- ✓Incorrect format in selection and loop statements

Linker Errors

- Linker errors are generated when the linker encounters what looks like a function call; but it cannot find a function with that name.

- This is usually caused by misspelling a C standard function (like main) or not including the header file for a function.

Runtime Errors

- A type of error that occurs during the execution of a program is known as run-time error.
- Runtime errors may crash your program when you run it. Runtime errors occur when a program with no syntax errors directs the computer to execute an illegal operation.
- Common examples are:
  - ✓ Trying to divide by a variable that contains a value of zero
  - ✓ Trying to open a file that does not exist

Logic Errors

- ❖ Logic errors occur when a programmer implements the algorithm for solving a problem incorrectly. A statement with logical error may produce unexpected and wrong results in the program. Common examples are:
  - ➢ Multiplying when you should be dividing
  - ➢ Adding when you should be subtracting
  - ➢ Opening and using data from the wrong file
  - ➢ Displaying the wrong message

- ❖ Logic errors are the hardest to find and fix because:
  - ➢ The compiler does not detect these errors
  - ➢ There is no indication of error when the program is executed.
  - ➢ The program may produce correct results for some input data and wrong results for other input data.

Summery

- Token (Keyword, Identifier, Constants, Operator, String)
- Variable declaration
- Data Type
- Statement & Expression
- Show the value of variables
- Error

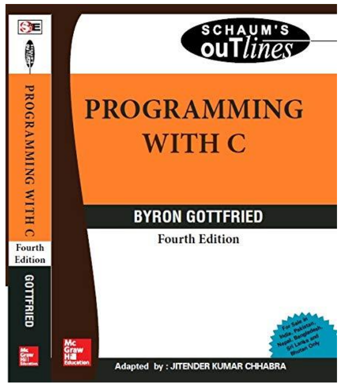
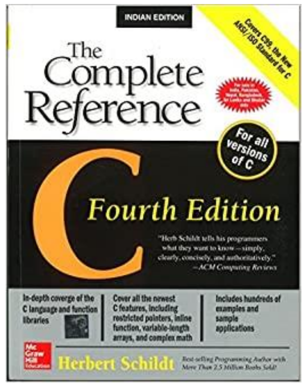
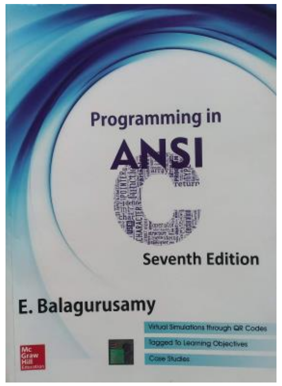
Thank You.

Questions and Answer

References

Books:

1. Programming With C. *By Byron Gottfried*
2. The Complete Reference C. *By Herbert Shield*
3. Programming in ANSI C *By E. Balagurusamy*
4. Teach yourself C. *By Herbert Shield*

Web:

1. www.wikbooks.org

and other slide, books and web search.