ES-101\ES-102
Programming In C

Unit - 1

Overview of C

Welcome
In
The
World
Of
‘C’

Course Materials

- Text Book:
  - **Programming In ANSI C** (5th Edition) by E Balagurusamy
  - **C – The Complete Reference** by Herbert Schildt
  - **Let Us C- Yashwant Kanetkar**
- Additional Support:
  - Class Lecture Slides

Course Outline:

- Introduction to C programming
- Constants, Variable & Data Types
- C Operators & Expressions
- Managing I/O Operations in C
- C Control Structure
- C Arrays
- C Characters and Strings
- C Functions
- C Pointers
- Structure and Union in C

Today’s Outline:

- Introduction to Programming
- History of C
- Why teach C
- What is C used for?
- Hello World! C Program
- Basic Structure of C Programs
- Programming Style in C
- Executing A ‘C’ Program
- Process for Executing A ‘C’ Program
- Rules to Remember.

Introduction to Programming

- What is a program?
  - A **computer program** is a set of instructions that tell a computer what to do.

- What is programming language?
  - A **programming language** is a formal computer language designed to communicate and give instructions to a machine, particularly a computer.

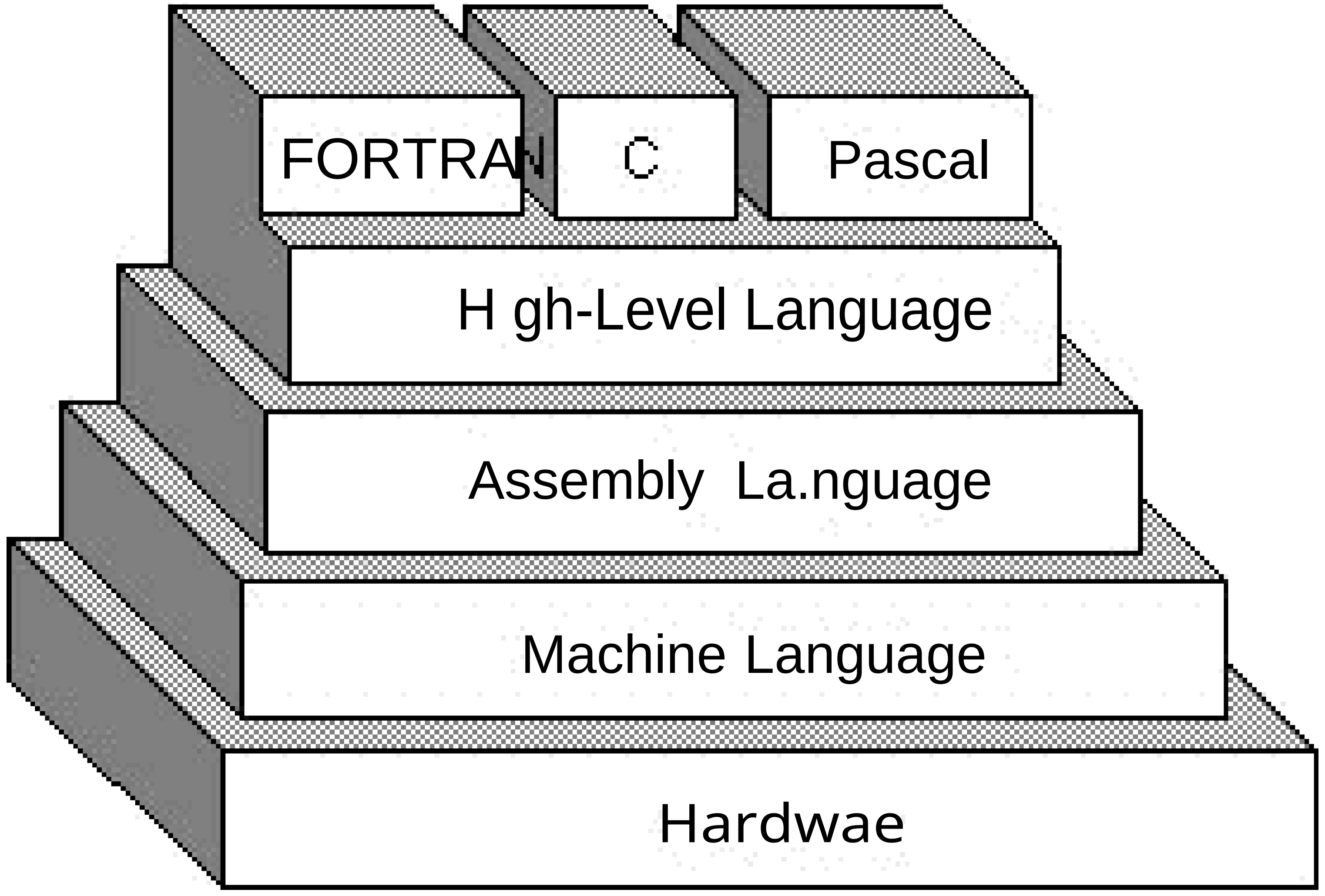
- What is Programming C?
  - **C** is a high-level and general purpose **programming** language.

Types of languages

(I) Lower level languages:-
Languages which are very near to machine…. I.e. machine language, Assembly language.

(II) Higher level languages:-
Languages which are very near to programmer rather than to machine….

I.e. C++,Visual C++,Visual basic,Java.


FORTRAN
C
Pascal
H gh-Level Language
Assembly La.nguage
Machine Language
Hardwae

History of C

- The root of all modern languages is **ALGOL**, introduced in 1960s.
- In 1967, Martin Rachards developed **BCPL**.
- In 1970, Ken Thompson created **B** by using BCPL features.
- In 1972, **C** was evolved from ALGOL, BCPL and B by **Dennis Ritchie** at the **Bell Laboratories** on **DEC PDP-11** machine. Which referred as "Traditional C“.
- In 1978, introduced K&R C (Kerningham and Dennis Ritchie).
- In 1989, ANSI approved a version of C known as **ANSI C**.
- In 1990, ISO also approved this version referred as **C89**.
- In 1999, Another enhanced version of C is introduced **C99**.

Why teach C / Importance of C

- C is ***small*** (only 32 keywords).
- C has rich set of ***built-in functions*** and support variety of ***data types*** & **operators**.
- C is ***highly portable*** (Machine independent).
- C is ***structured***.
- C has ***ability to extend*** itself.
- C is ***stable*** (the language doesn't change much).
- C is ***quick running*** (code written in c is efficient & fast).
- C is the ***basis for many other languages*** (C++, C#, Java, Perl etc).
- C is a ***Programmers Language***.

- ***It may not feel like it but C is one of the easiest language to learn.***

What is C used for?

- C is most likely an evergreen language.
- Initially, C widely known as the development language of the UNIX operating system but today virtually all new major operating systems are written in C and/or C++.

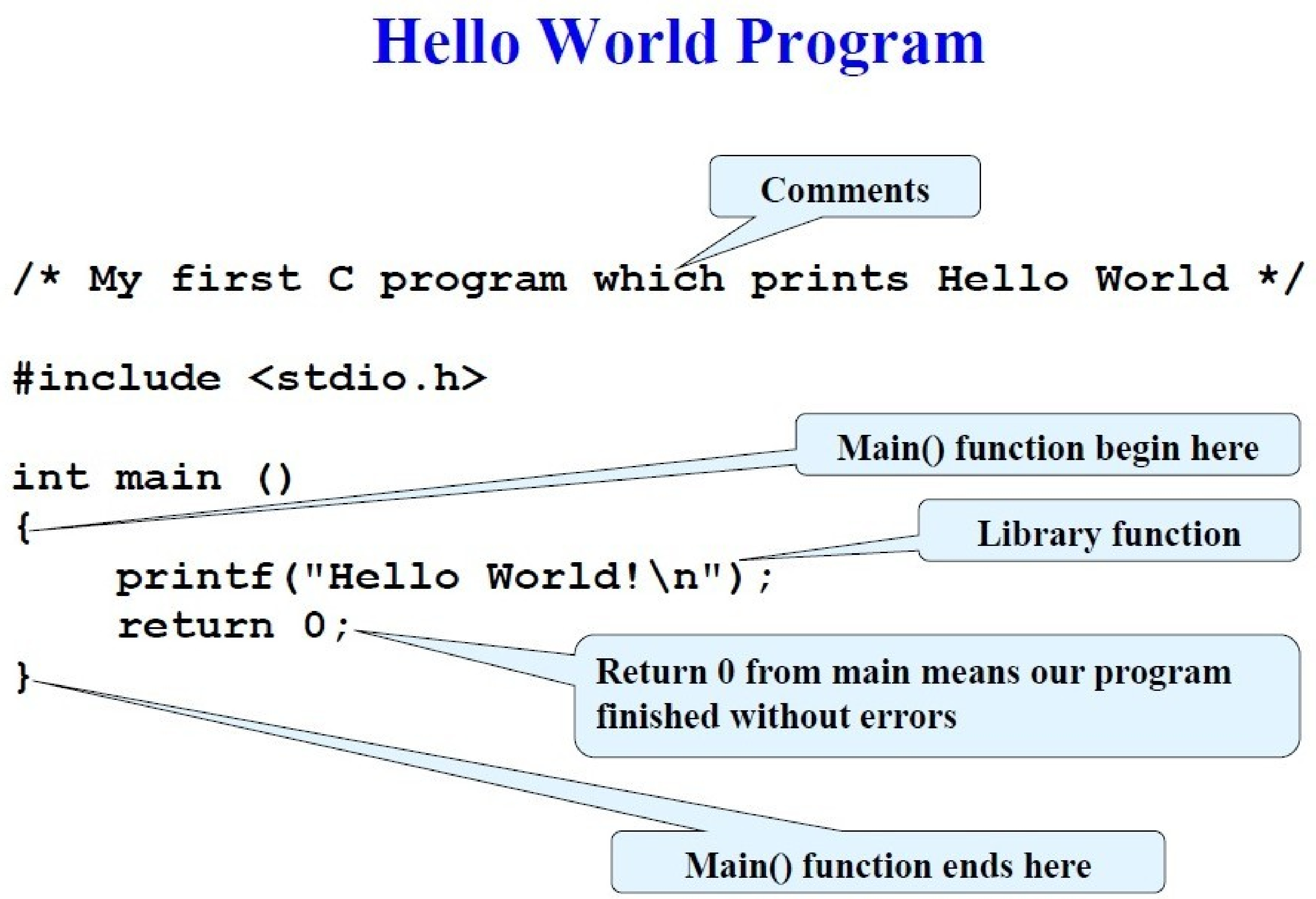
  - **Systems programming:** OSes, like Linux.

  - **Microcontrollers:** Automobiles and Airplanes.

  - **Embedded processors:** Phones, Portable Electronics etc.

  - **DSP processors:** Digital Audio.

Hello World! Program



Hello World Program

```
/* My first C program which prints Hello World */

#include <stdio.h>

int main ()
{
    printf("Hello World!\n");
    return 0;
}
```

Comments
Main() function begin here
Library function
Return 0 from main means our program finished without errors
Main() function ends here

Format of simple C programs

```
main()
{
…………
…………
…………
}
```

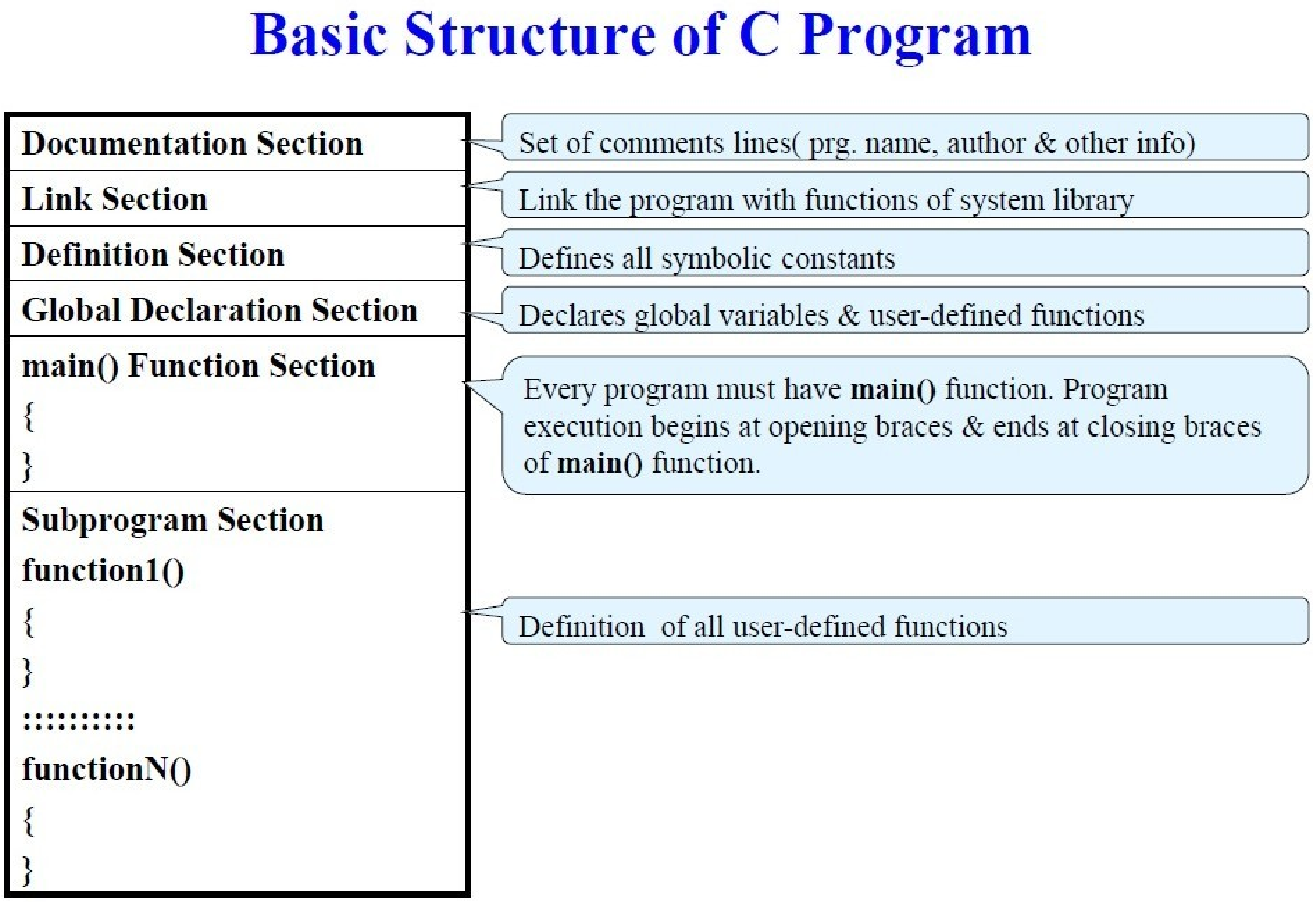
Function Name

Start of Program

Program Statements

End of Program

Basic Structure of C Programs


Basic Structure of C Program
Documentation Section
Link Section
Definition Section
Global Declaration Section
main() Function Section
{
}
Subprogram Section
function1()
{
}
::::::::::
functionN()
{
}
Set of comments lines(prg. name, author & other info)
Link the program with functions of system library
Defines all symbolic constants
Declares global variables & user-defined functions
Every program must have main() function. Program execution begins at opening braces & ends at closing braces of main() function.
Definition of all user-defined functions

Basic Structure of C Programs (with Example)

/* Program for Print, Author: Mr. X */	Documentation Section
#include<stdio.h>	Link Section
#define PI 3.1416	Definition Section
void print_pi();	Global Declaration Section

int main()
{
    print_pi();
    return 0;
}
	main() Function Section
	{
	   ………………
	   ………………
	}

void print_pi()
{
    printf("Value of PI is: %.4f",PI);
}
	Subprogram Section
	Function1()
	{
	}

Programming Style

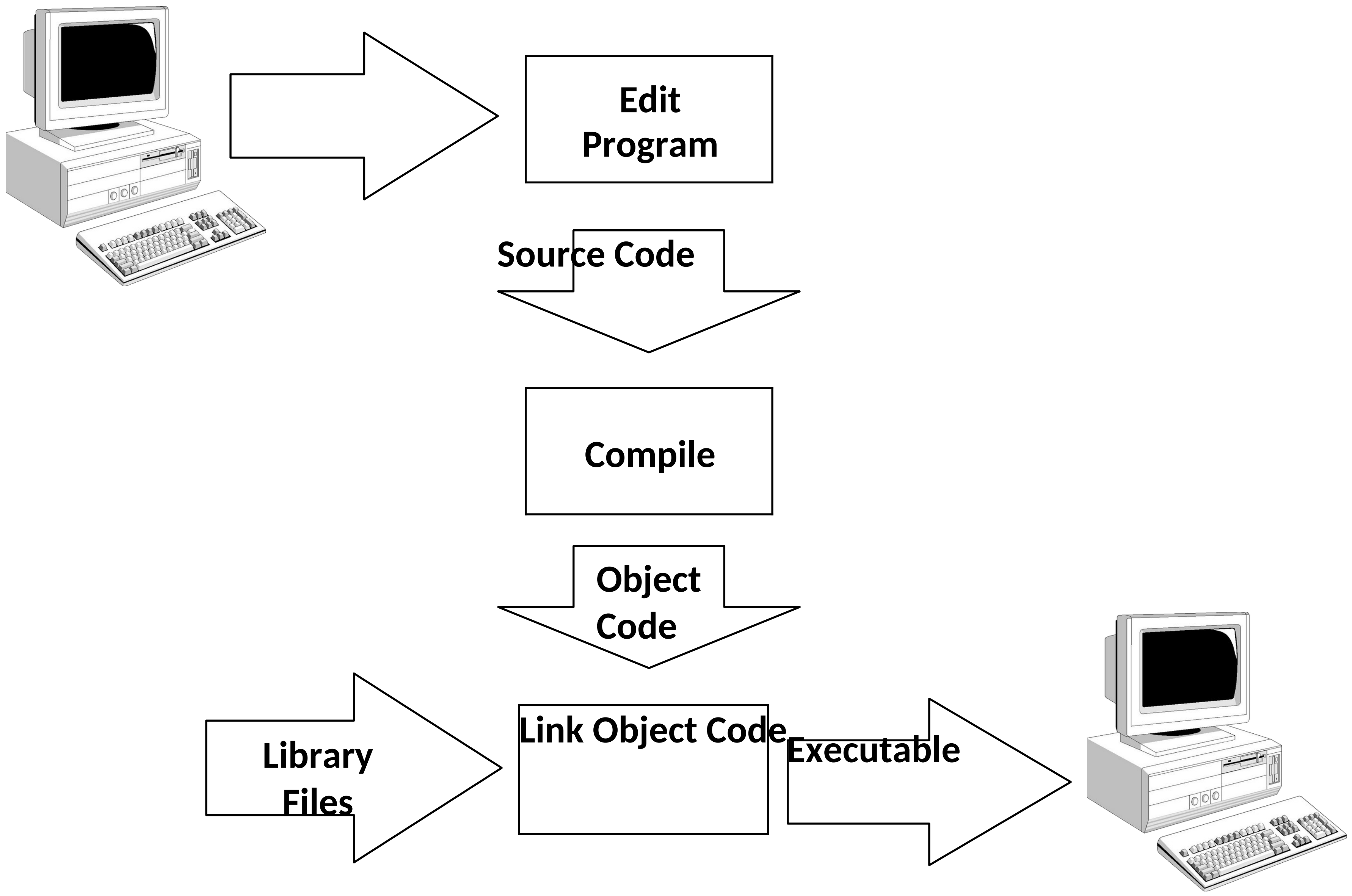
- You should follow one style for programming.
- We must develop the habit of writing programs in lowercase letters, because C programs statements are written in lowercase letters.
- Uppercase letters are used only for symbolic constants.
- Braces, {} indicates beginning and end of a functions.
- Need, braces to align for easy readability.
- Try to write one statement into one line, although C support multiple statement in a single line.

Executing A ‘C’ Program

- Executing a program written in C involves a series of steps. There are:

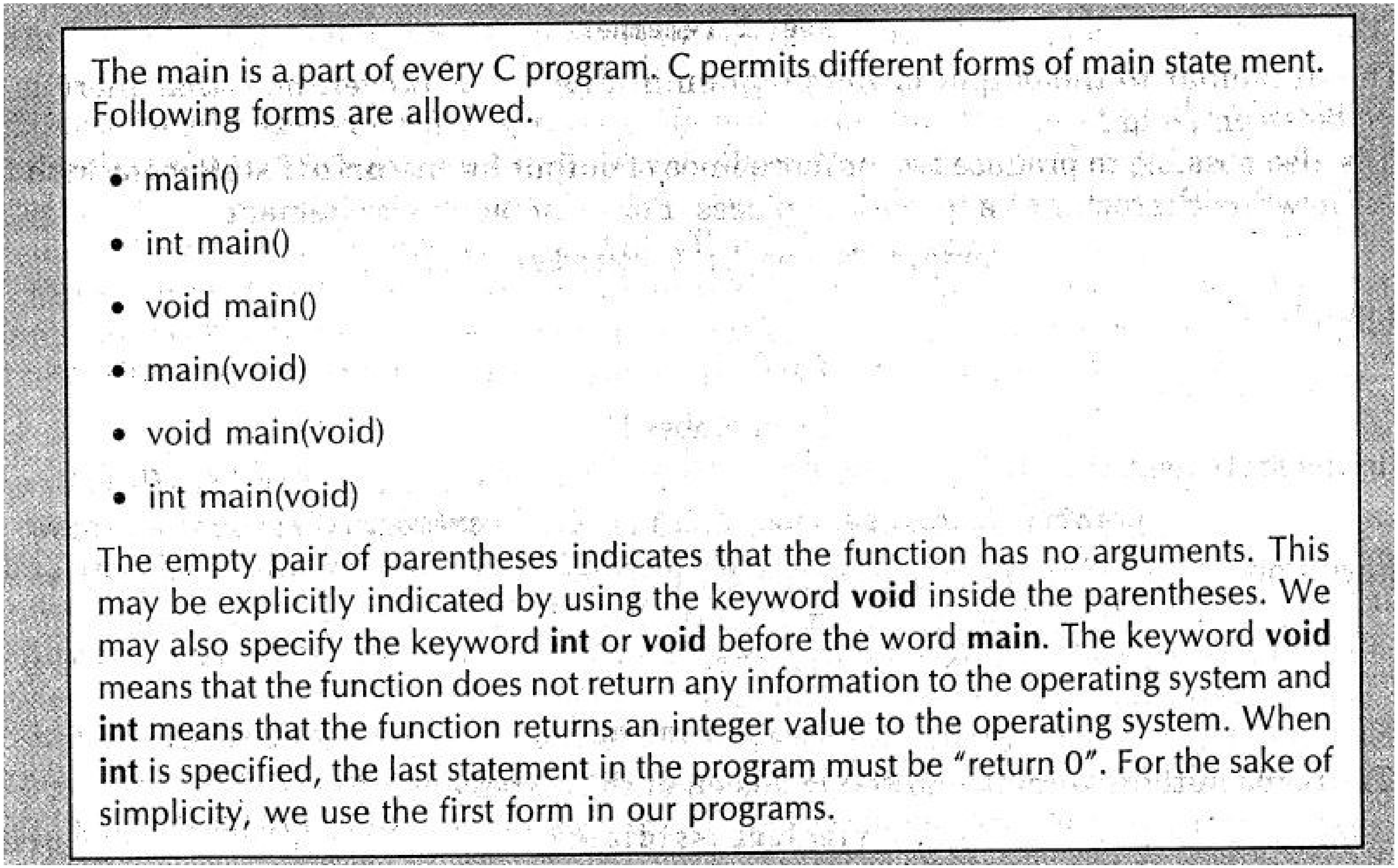
1. Creating the program;
2. Compiling the program;
3. Linking the program with functions that are needed from the C library;
4. Executing the program.

Process for Executing A ‘C’ Program



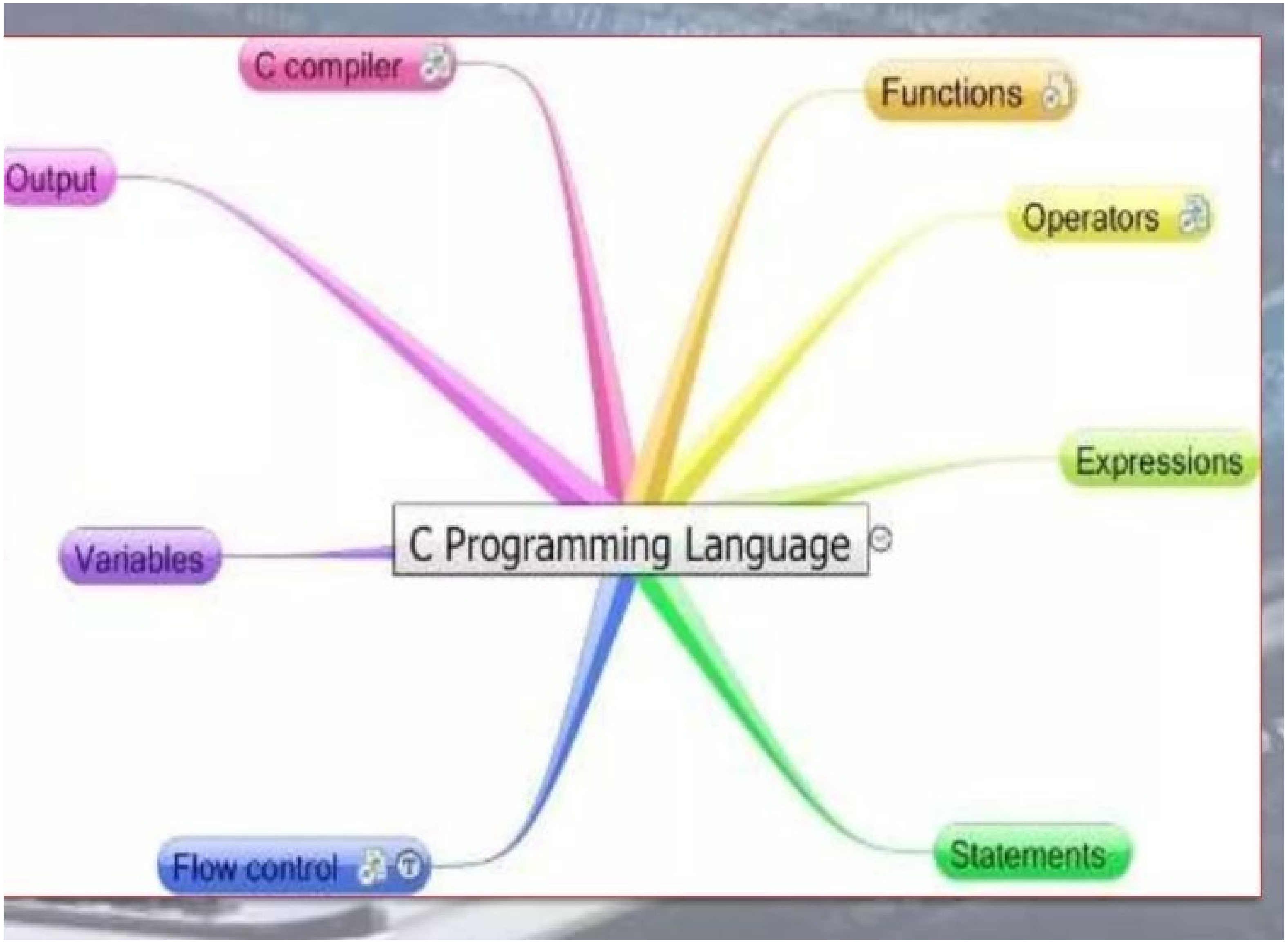
Rules to Remember:

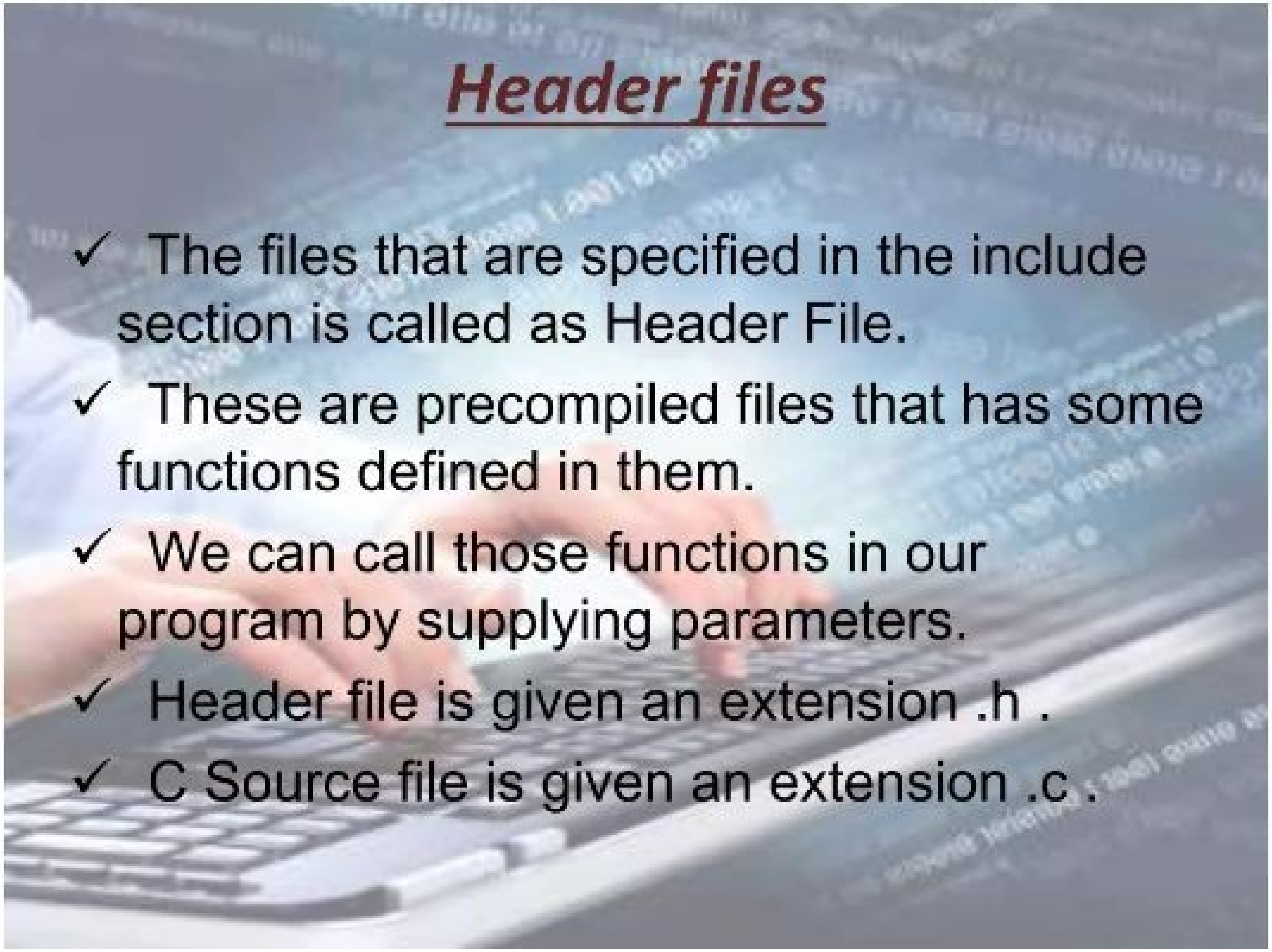
- Every C program requires a **main()** program (more than one main() illegal)
- Execution of a function begins at the **{ opening brace** and ends at the corresponding **} closing brace**.
- C programs are written in **lowercase letters**. Higher case used for **symbolic name** and **output strings**.
- Every program statement in a C language must end with a **semicolon**.
- **include** and **define** are special instruction s to compiler, they *do not end with a semicolon*.
- Using comment **/* Comment */** is a good programming habit.


The main is a part of every C program. C permits different forms of main state ment. Following forms are allowed.

- main()
- int main()
- void main()
- main(void)
- void main(void)
- int main(void)

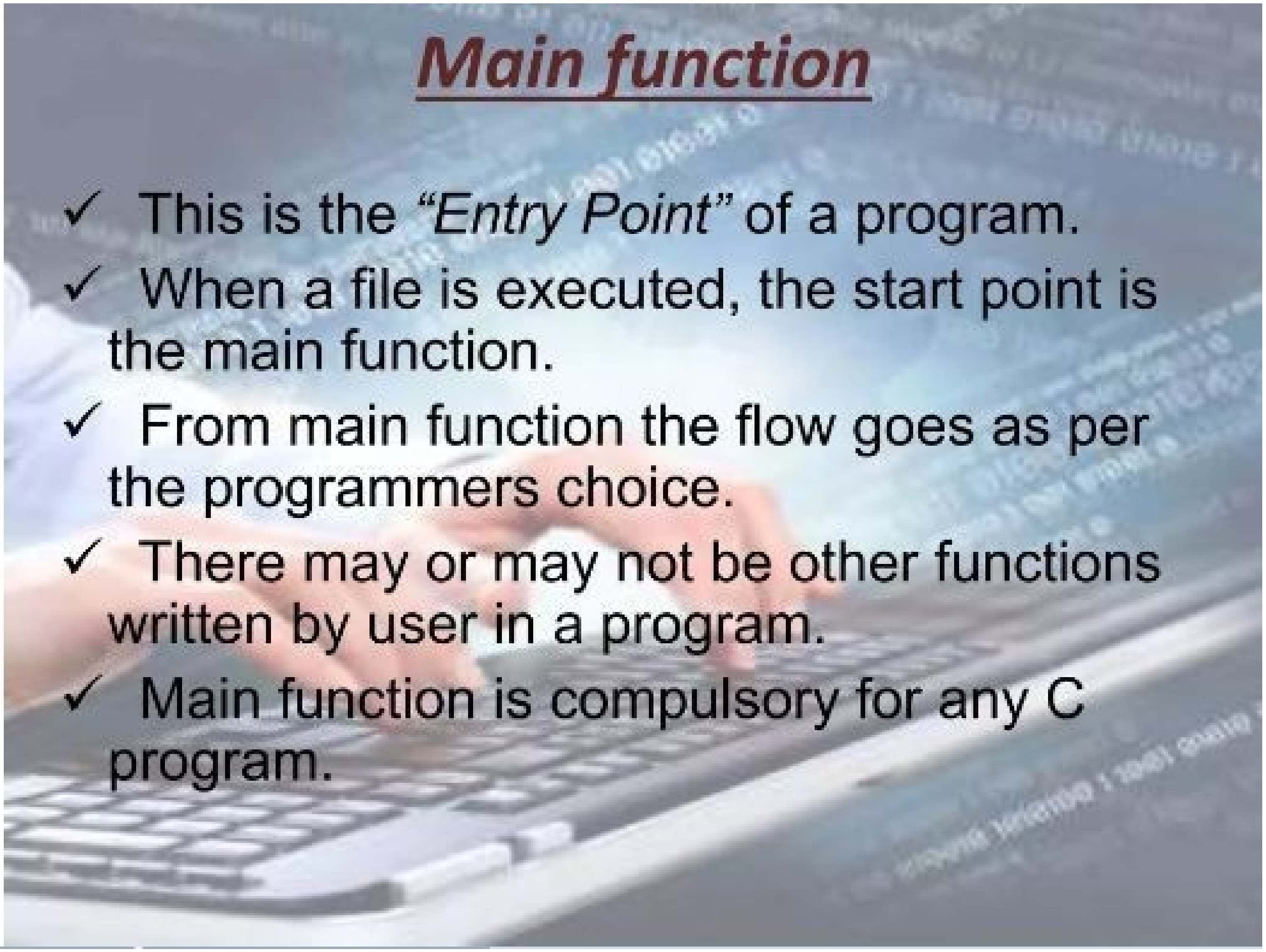
The empty pair of parentheses indicates that the function has no arguments. This may be explicitly indicated by using the keyword **void** inside the parentheses. We may also specify the keyword **int** or **void** before the word **main**. The keyword **void** means that the function does not return any information to the operating system and **int** means that the function returns an integer value to the operating system. When **int** is specified, the last statement in the program must be "return 0". For the sake of simplicity, we use the first form in our programs.


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



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.

“C” language TOKENS

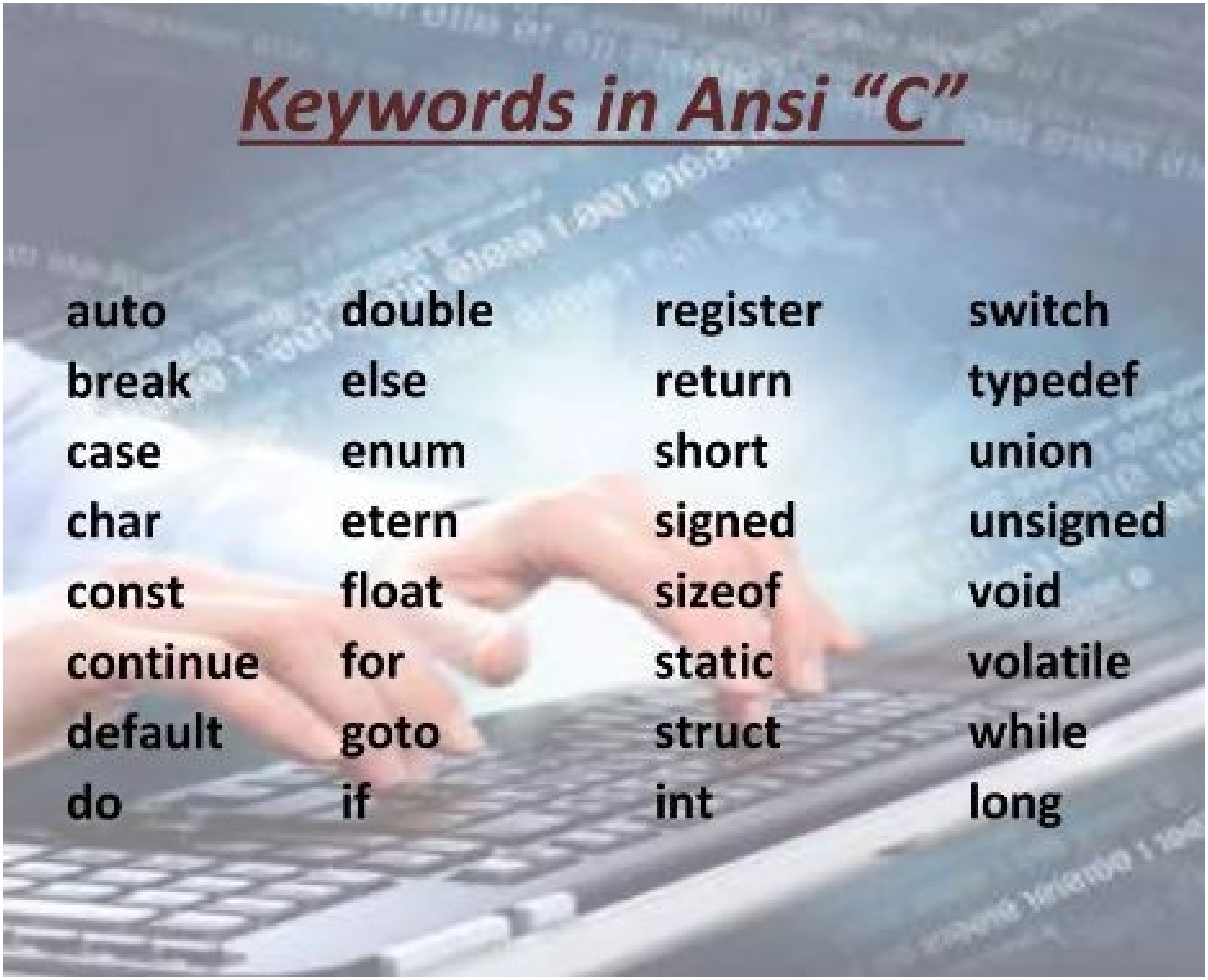
✓ The smallest individual units in a C program are known as tokens. In a C source program, the basic element recognized by the compiler is the "token." A token is source-program text that the compiler does not break down into component elements.

✓ 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. +, -, *]

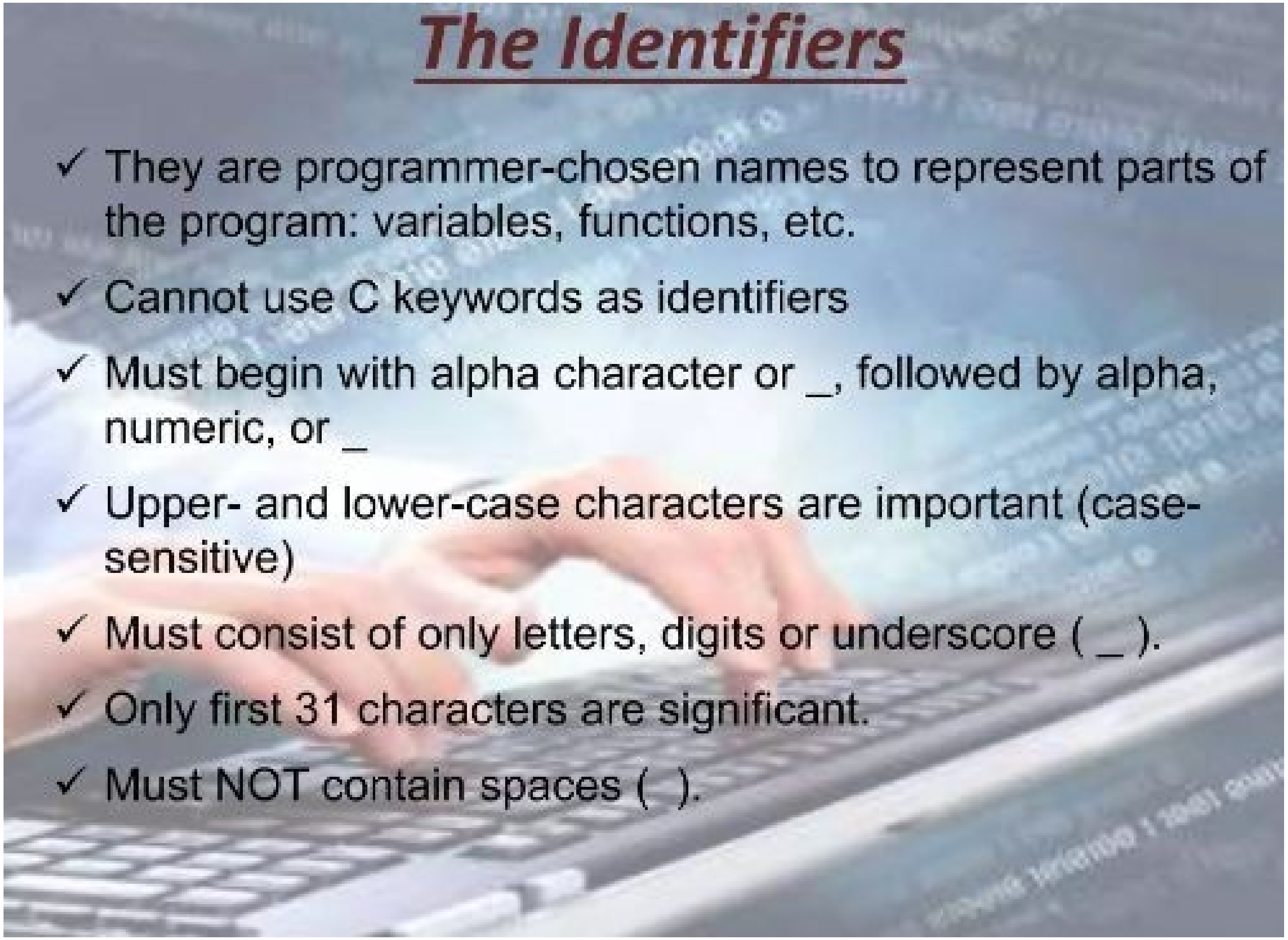
Key wor ds
Constants
Special Symbols
C tokens
Operators
Identifiers
Strin gs

✓ 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

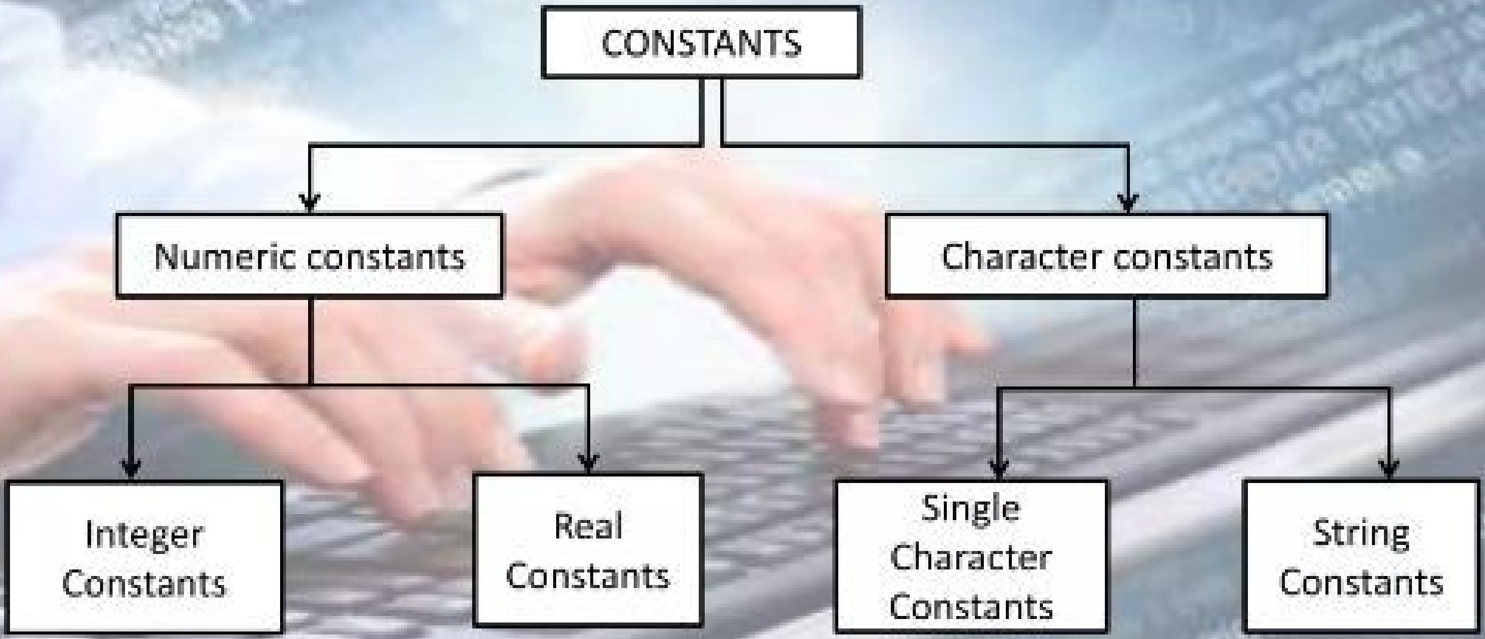


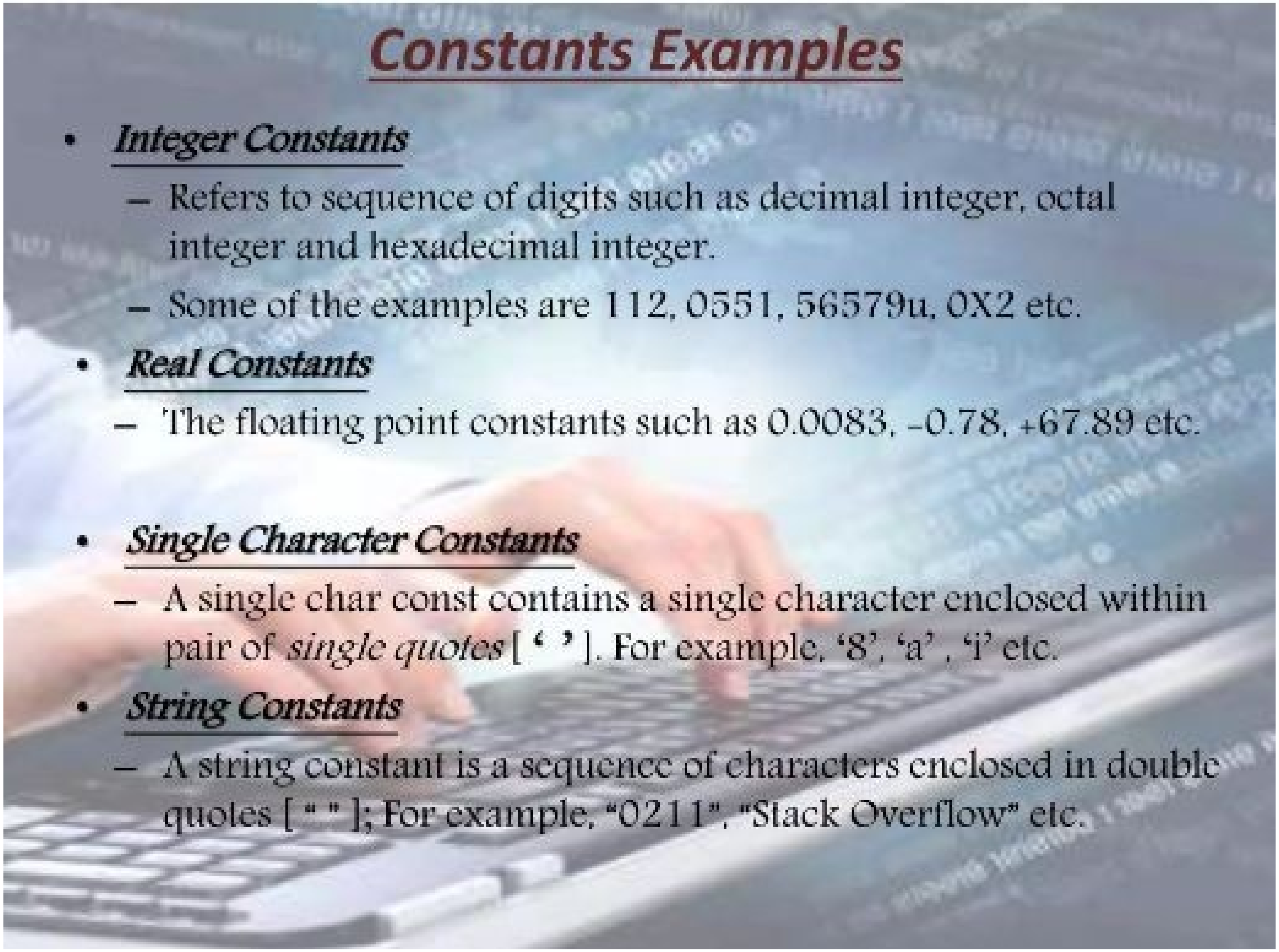
The Identifiers

- ✓ 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

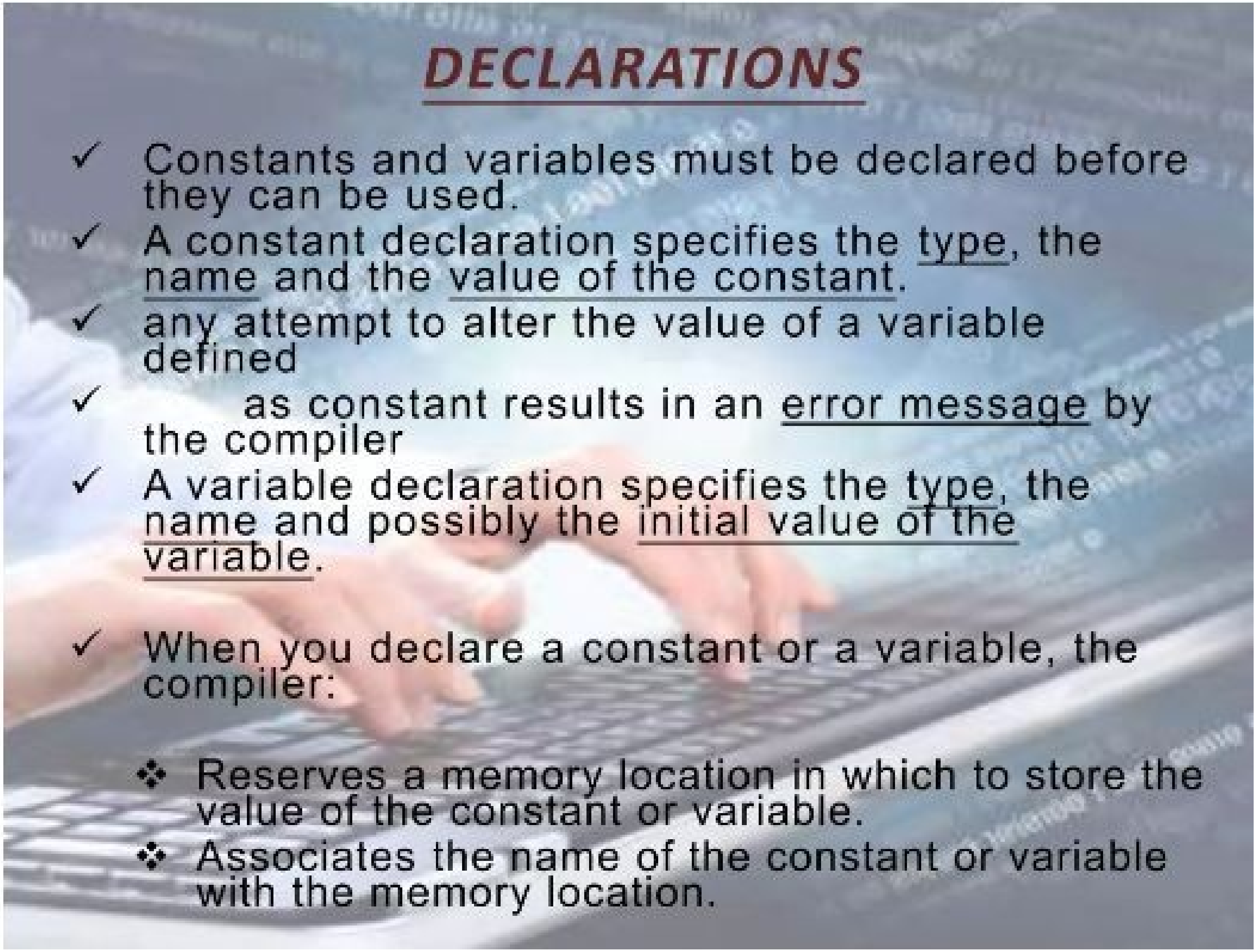
➢ **Constants** in C are the fixed values that do not change during the execution of a program.





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.

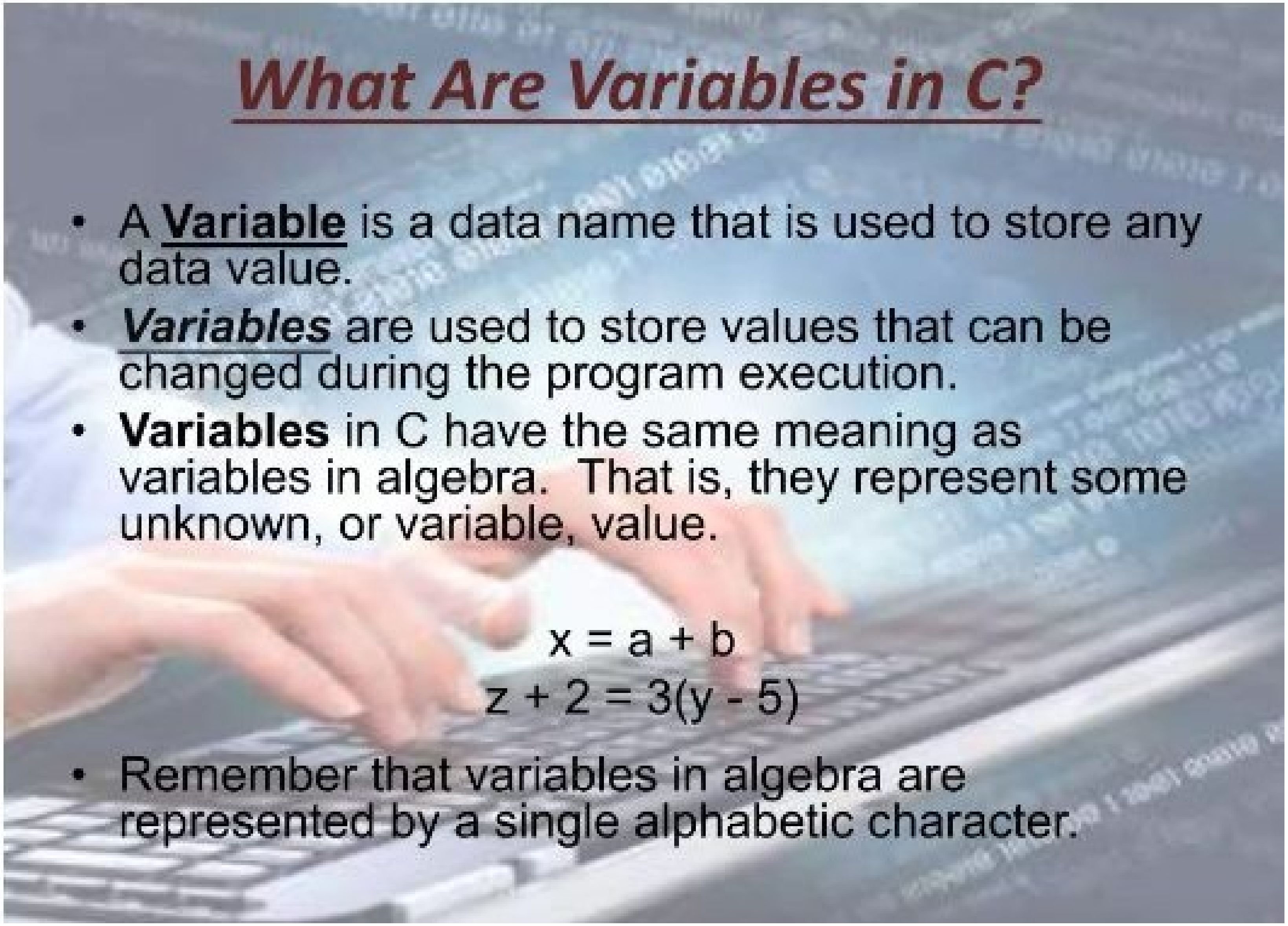


DECLARATIONS

- ✓ 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.

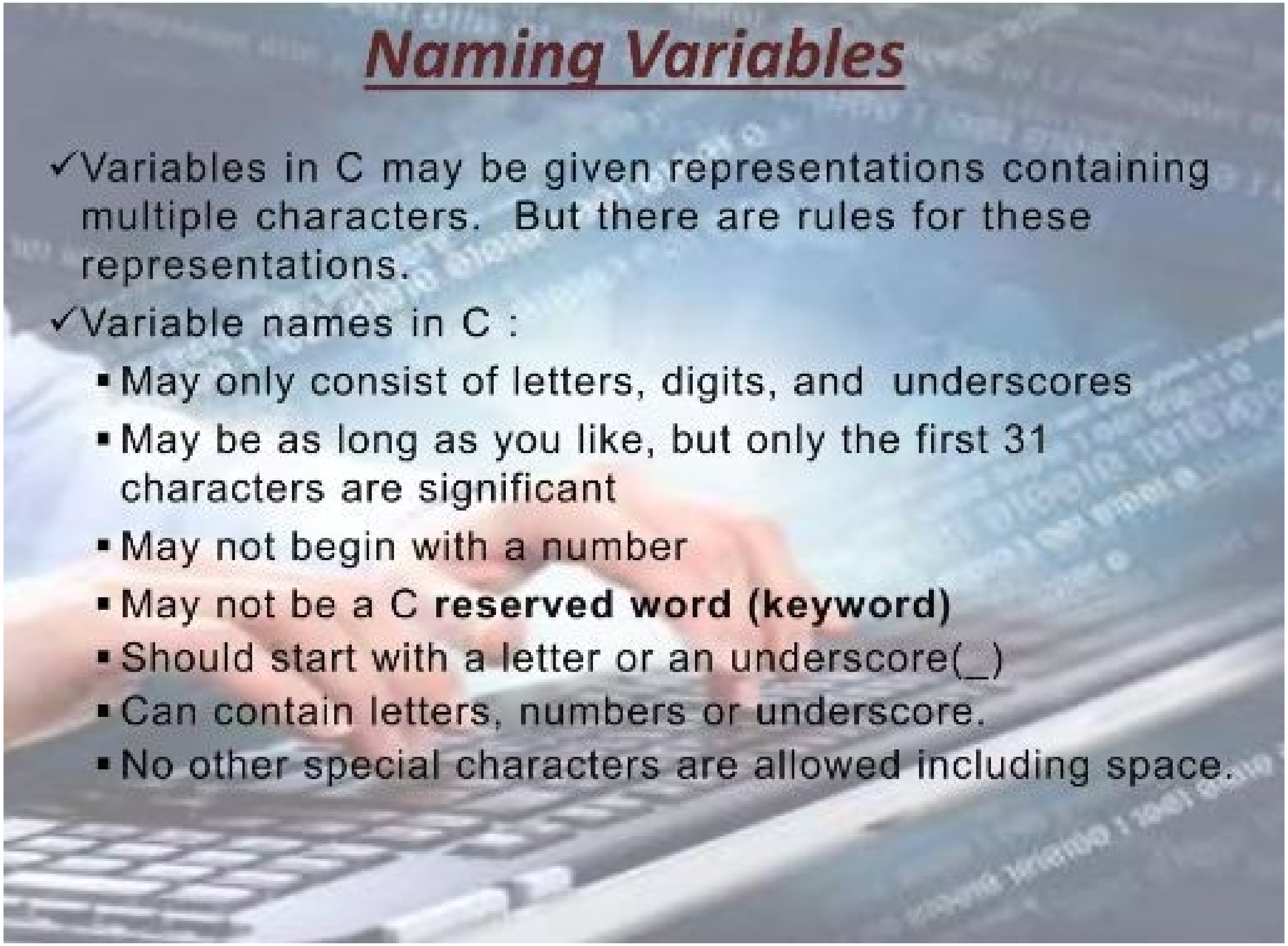


What Are Variables in C?

- 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.

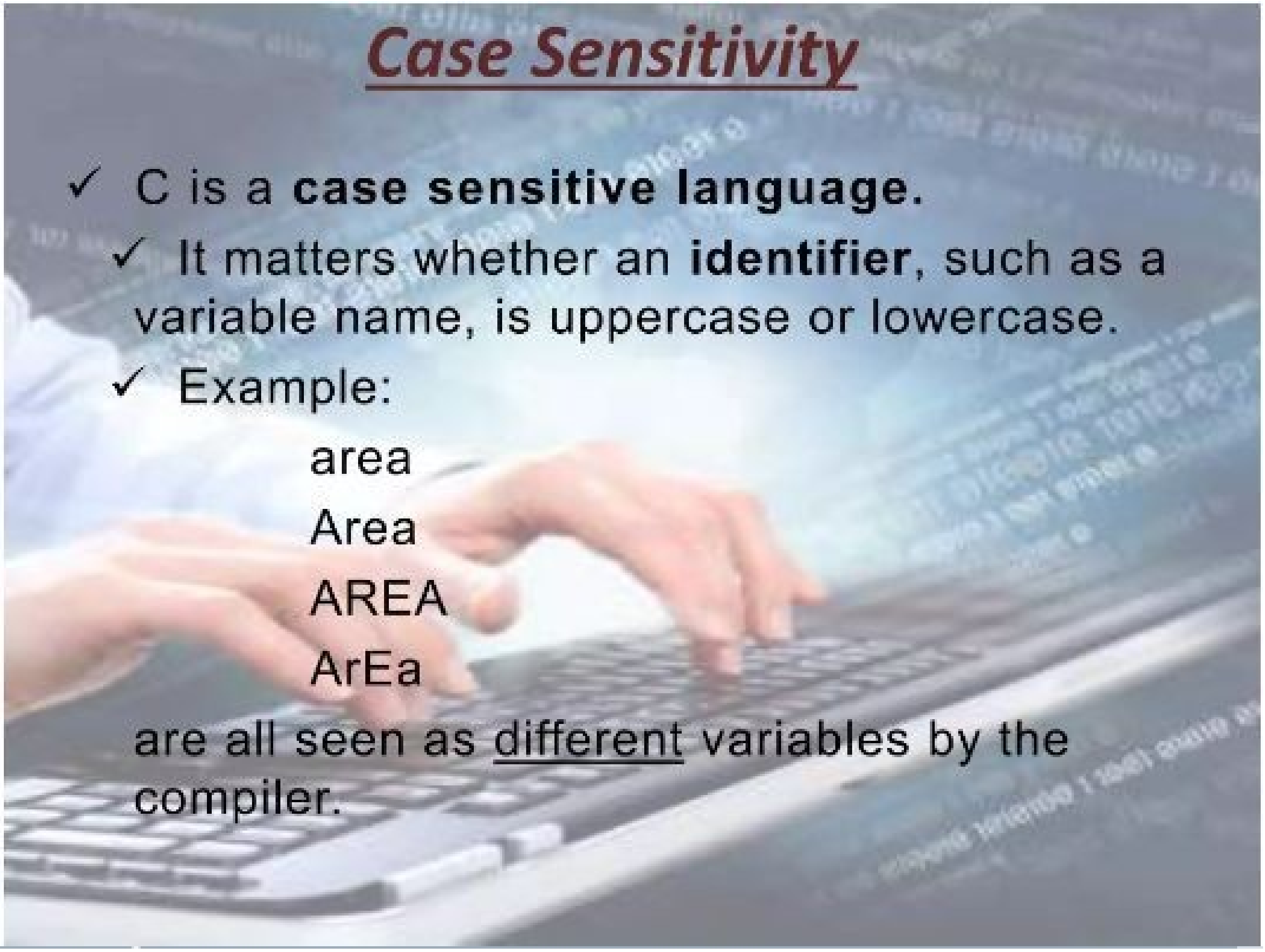


Naming Variables

✓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 different variables by the compiler.

Write a C program to display following variables.
a+ c, x + c,dx + x, ((int) dx) + ax, a + x, s + b, ax + b, s + c, ax + c, ax + ux

```
#include <stdio.h>
void main()
{
    int a = 125, b = 12345;
    long ax = 1234567890;
    short s = 4043;
    float x = 2.13459;
    double dx = 1.1415927;
    char c = 'W';
    unsigned long ux = 2541567890;

    printf("a + c = %d\n", a + c);
    printf("x + c = %f\n", x + c);
    printf("dx + x = %f\n", dx + x);
    printf("((int) dx) + ax = %ld\n", ((int) dx) + ax);
    printf("a + x = %f\n", a + x);
    printf("s + b = %d\n", s + b);
    printf("ax + b = %ld\n", ax + b);
    printf("s + c = %hd\n", s + c);
    printf("ax + c = %ld\n", ax + c);
    printf("ax + ux = %lu\n", ax + ux);

 }
```

Sample Output:

```
a + c =  212
x + c = 89.134590
dx + x = 3.276183
((int) dx) + ax =  1234567891
a + x = 127.134590
s + b =  16388
ax + b = 1234580235
s + c =  4130
ax + c = 1234567977
ax + ux = 3776135780
```

Table 2.1 *C 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 (or less than sign)
“ quotation mark	> closing angle bracket (or greater than sign)
! exclamation mark	(left parenthesis
| vertical bar	) right parenthesis
/ slash	[left bracket
\ backslash	] right bracket
~ tilde	{ left brace
_ under score	} right brace
$ dollar sign	# number sign
% percent sign	

Table 2.1 C Character Set

Letters	Digits
Uppercase A.....Z Lowercase a.....z	All decimal digits 0 9

Special Characters

, comma	& ampersand
. period	^ caret
; semicolon	* asterisk
: colon	– minus sign
? question mark	+ plus sign
' apostrophe	< opening angle bracket (or less than sign)
" quotation mark	> closing angle bracket (or greater than sign)
! exclamation mark	(left parenthesis
| vertical bar	) right parenthesis
/ slash	[left bracket
\ backslash	] right bracket
~ tilde	{ left brace
_ under score	} right brace
$ dollar sign	# number sign
% percent sign	

number sign

White Spaces

Blank space
Horizontal tab
Carriage return
New line
Form feed

Table 2.2 *ANSI C Trigraph Sequences*

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

C Quiz

1. Who is the father of C language?
a) Steve Jobs
b) James Gosling
c) Dennis Ritchie
d) Rasmus Lerdorf

Answer: C

Explanation: Dennis Ritchie is the father of C Programming Language. C programming language was developed in 1972 at American Telephone & Telegraph Bell Laboratories of USA.

C Quiz

1. Who is the father of C language?
a) Steve Jobs
b) James Gosling
c) Dennis Ritchie
d) Rasmus Lerdorf

C Quiz

2. Which of the following is true for variable names in C?
a) They can contain alphanumeric characters as well as special characters
b) It is not an error to declare a variable to be one of the keywords(like goto, static)
c) Variable names cannot start with a digit
d) Variable can be of any length

C Quiz

Answer: c

Explanation: According to the syntax for C variable name, it cannot start with a digit.

C Quiz

3. Which of the following cannot be a variable name in C?
a) volatile
b) true
c) friend
d) export

C Quiz

Answer: a

Explanation: volatile is C keyword.

C Quiz

4. Which of the following declaration is not supported by C language?

a) String str;
b) char *str;
c) float str = 3e2;
d) Both String str; & float str = 3e2

C Quiz

Answer: a

Explanation: It is legal in Java, but not in C language.

C Quiz

5. How is search done in #include and #include “somelibrary.h” according to C standard?

a) When former is used, current directory is searched and when latter is used, standard directory is searched
b) When former is used, standard directory is searched and when latter is used, current directory is searched
c) When former is used, search is done in implementation defined manner and when latter is used, current directory is searched
d) For both, search for ‘somelibrary’ is done
in implementation-defined places

C Quiz

Answer: d

C Quiz

6. What will be the output of the following C code?

```
#include <stdio.h>
int main()
{
    int y = 10000;
    int y = 34;
    printf("Hello World! %d\n", y);
    return 0;
}
```

a) Compile time error
b) Hello World! 34
c) Hello World! 1000
d) Hello World! followed by a junk value

C Quiz

Answer: a

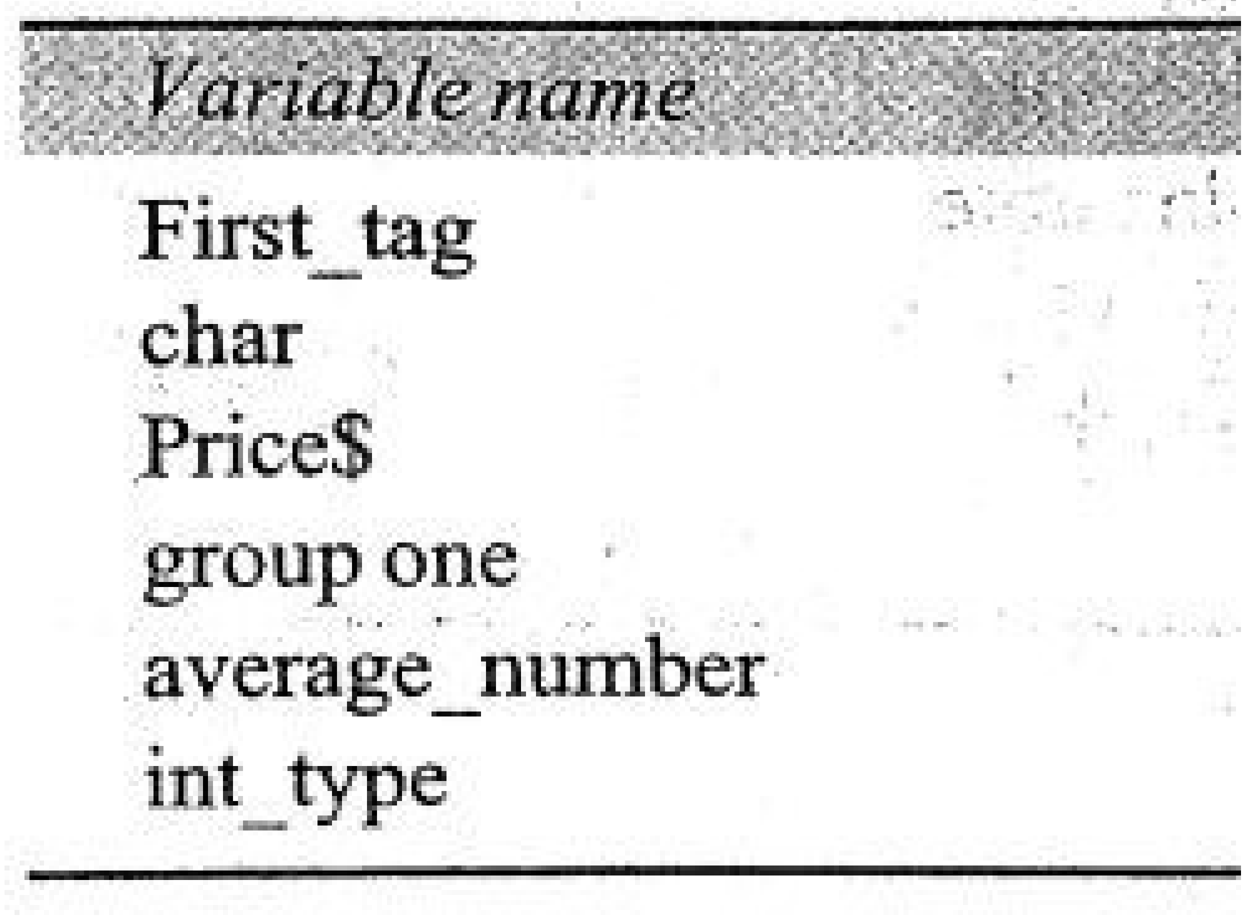
Explanation: Since y is already defined, redefining it results in an error.

Output:

```
$ cc pgm2.c
pgm2.c: In function ‘main’:
pgm2.c:5: error: redefinition of ‘y’
pgm2.c:4: note: previous definition of ‘y’ was here
```

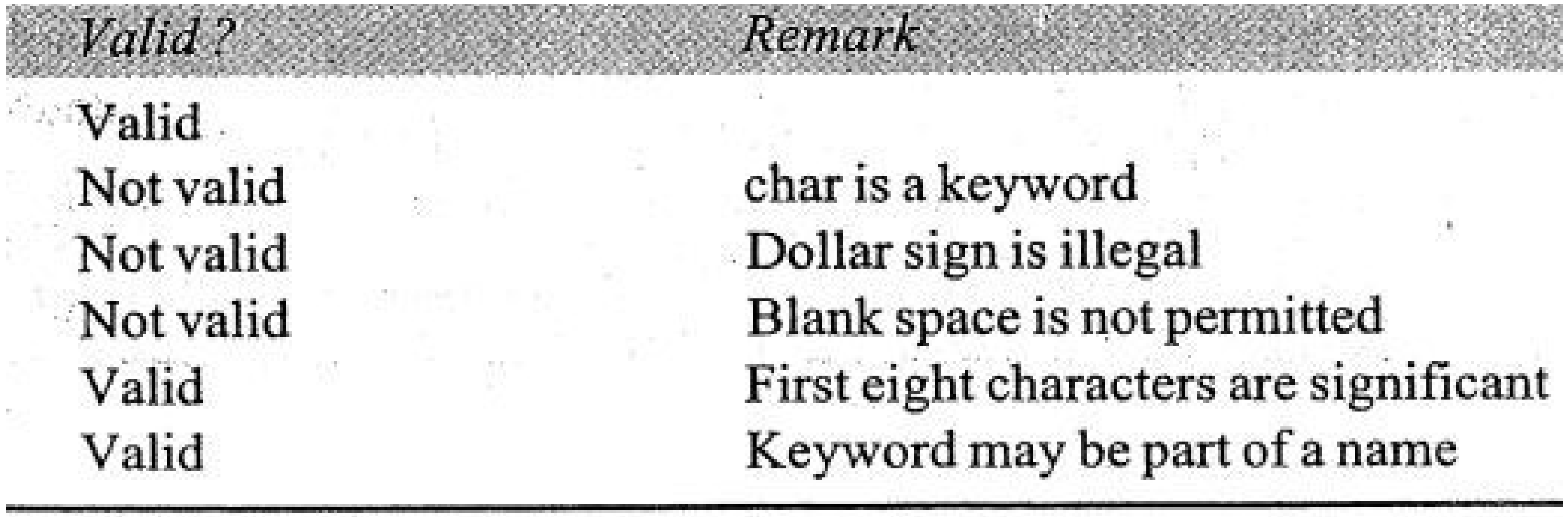
C Quiz

Q: Identify the Valid Variables and Invalid Variables



Variable name
First_tag
char
Price$
group one
average_number
int_type

C Quiz



Valid ?	*Remark*
Valid	
Not valid	char is a keyword
Not valid	Dollar sign is illegal
Not valid	Blank space is not permitted
Valid	First eight characters are significant
Valid	Keyword may be part of a name

C Quiz

Algorithm:
Algorithm of this program is to find ____________________________of two numbers.

```
START
   Step 1 → Define two variables - A, B
   Step 2 → Set loop from 1 to max of A, B
   Step 3 → Check if both are completely divided by same
loop number, if yes, store it
   Step 4 → Display the stored number is
________________________
STOP
```

C Quiz

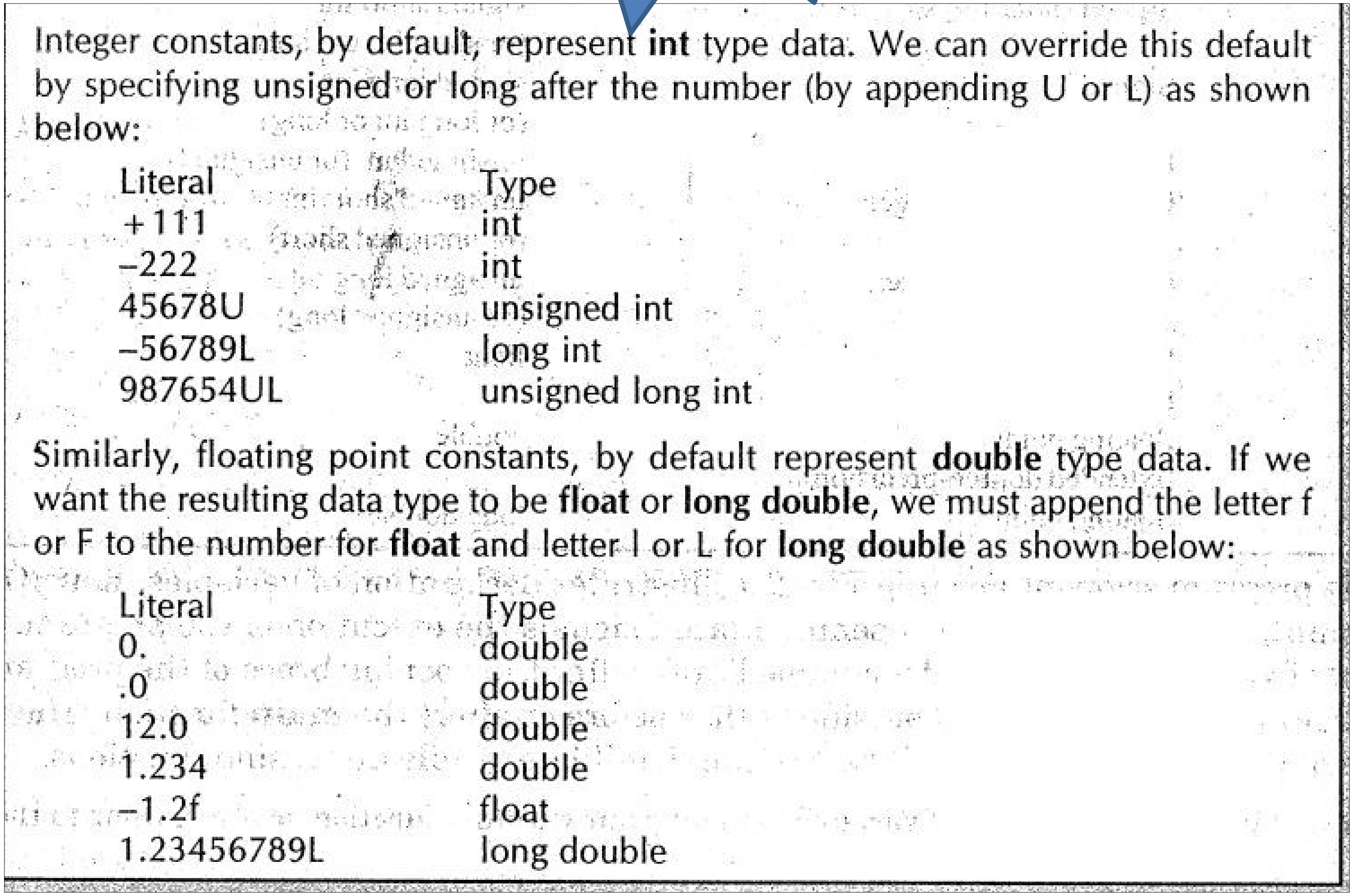
An H.C.F or Highest Common Factor, is the largest common factor of two or more values.

For example factors of 12 and 16 are −

12 → 1, 2, 3, 4, 6, 12
16 → 1, 2, 4, 8, 16

The common factors are 1, 2, 4 and the highest common factor is 4.

Food Of Thought

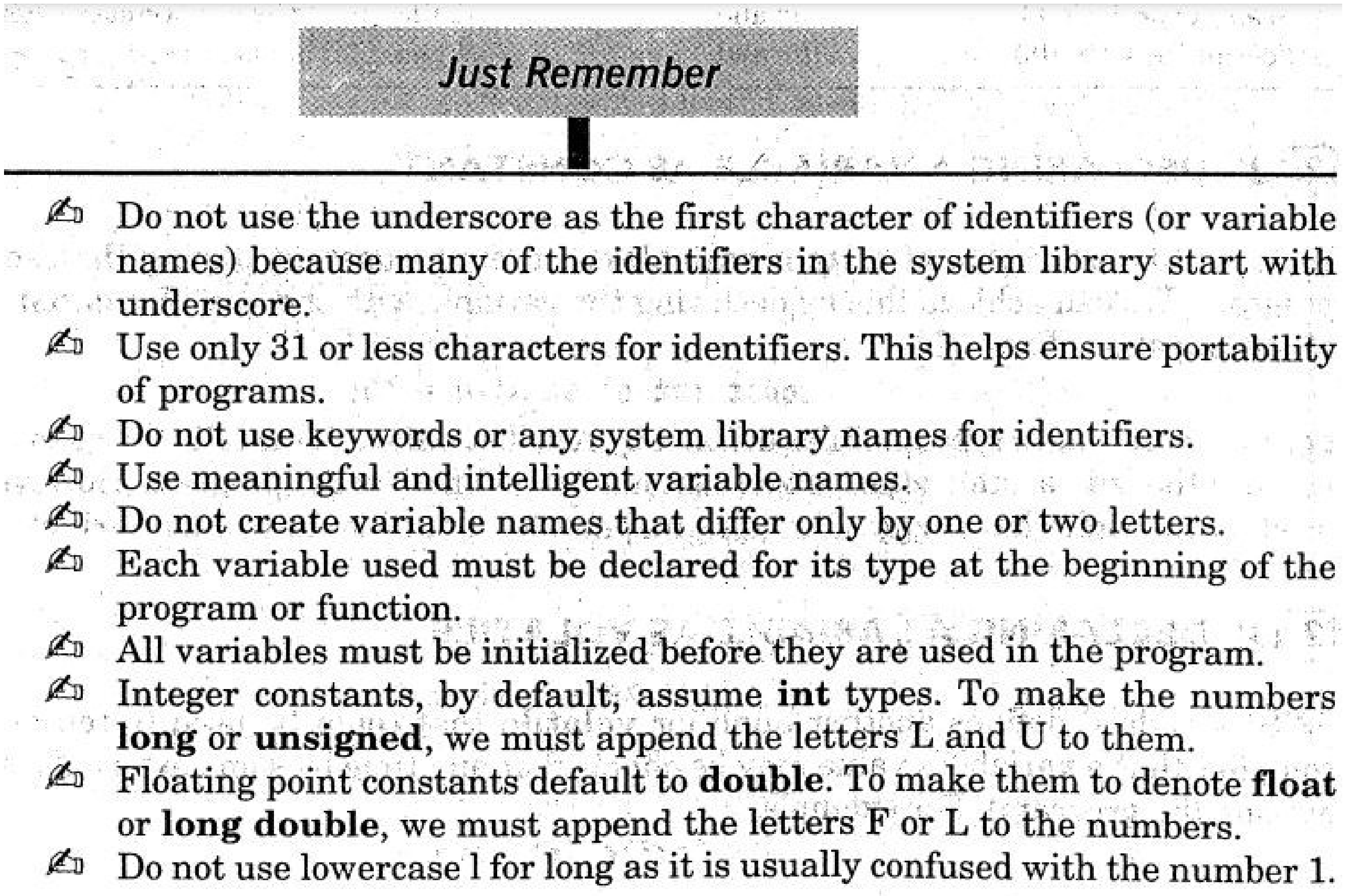


Integer constants, by default, represent **int** type data. We can override this default by specifying unsigned or long after the number (by appending U or L) as shown below:

Literal	Type
+111	int
–222	int
45678U	unsigned int
–56789L	long int
987654UL	unsigned long int

Similarly, floating point constants, by default represent **double** type data. If we want the resulting data type to be **float** or **long double**, we must append the letter f or F to the number for **float** and letter l or L for **long double** as shown below:

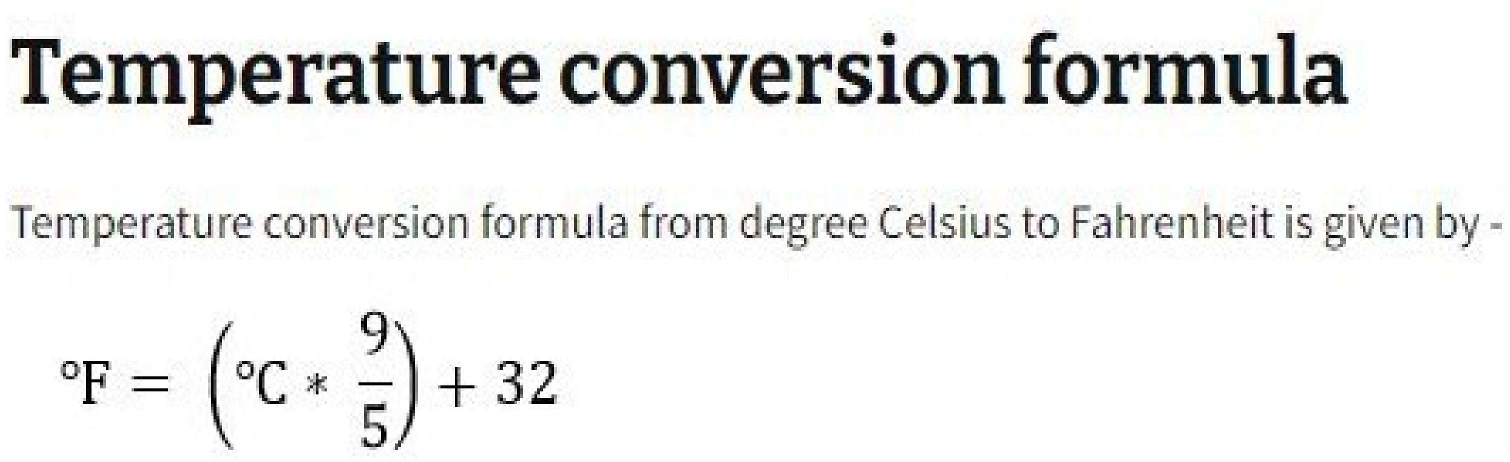
Literal	Type
0.	double
.0	double
12.0	double
1.234	double
–1.2f	float
1.23456789L	long double



Just Remember

- Do not use the underscore as the first character of identifiers (or variable names) because many of the identifiers in the system library start with underscore.
- Use only 31 or less characters for identifiers. This helps ensure portability of programs.
- Do not use keywords or any system library names for identifiers.
- Use meaningful and intelligent variable names.
- Do not create variable names that differ only by one or two letters.
- Each variable used must be declared for its type at the beginning of the program or function.
- All variables must be initialized before they are used in the program.
- Integer constants, by default, assume **int** types. To make the numbers **long** or **unsigned**, we must append the letters L and U to them.
- Floating point constants default to **double**. To make them to denote **float** or **long double**, we must append the letters F or L to the numbers.
- Do not use lowercase l for long as it is usually confused with the number 1.

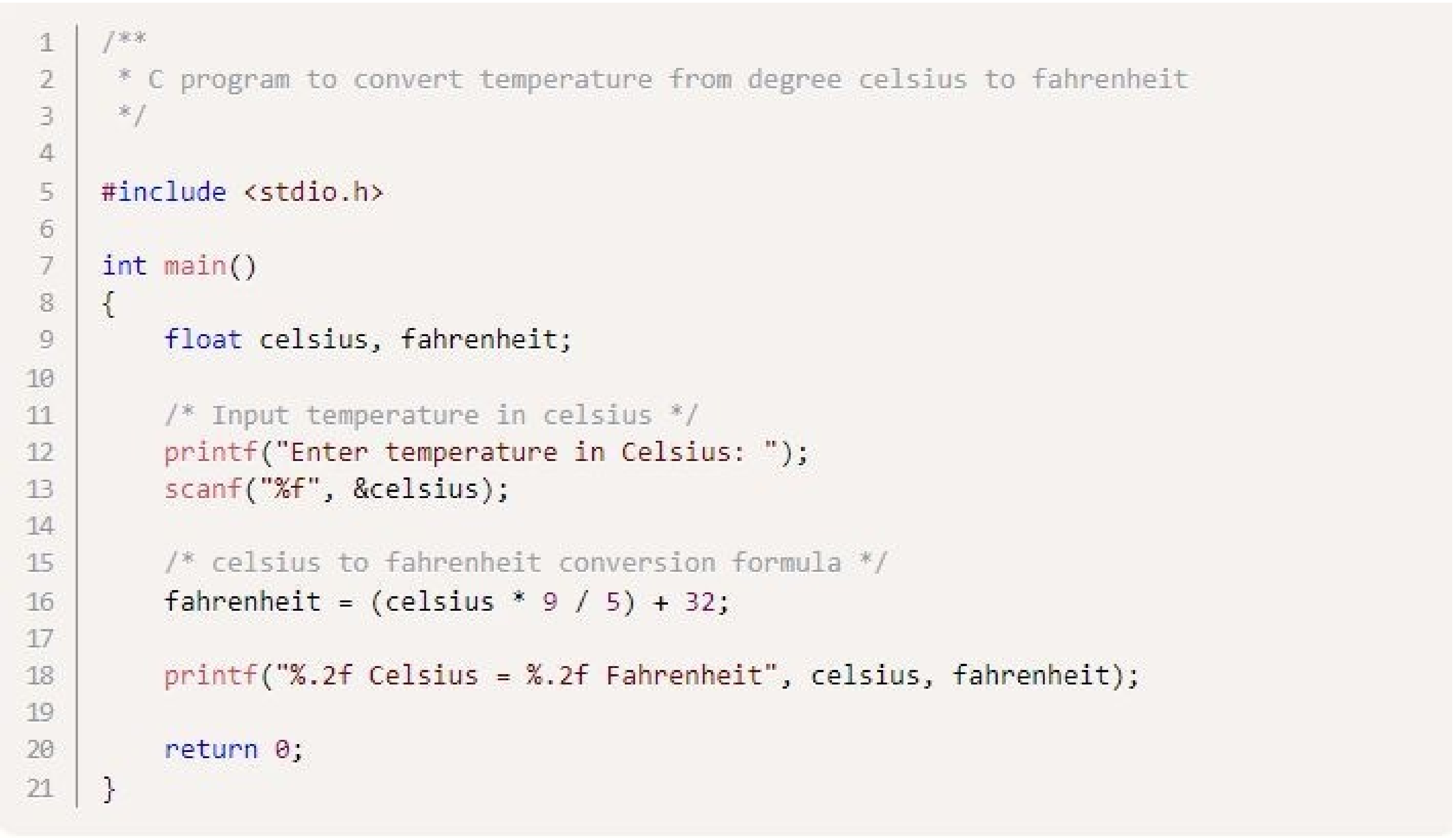
Program: Temperature Conversion Problem



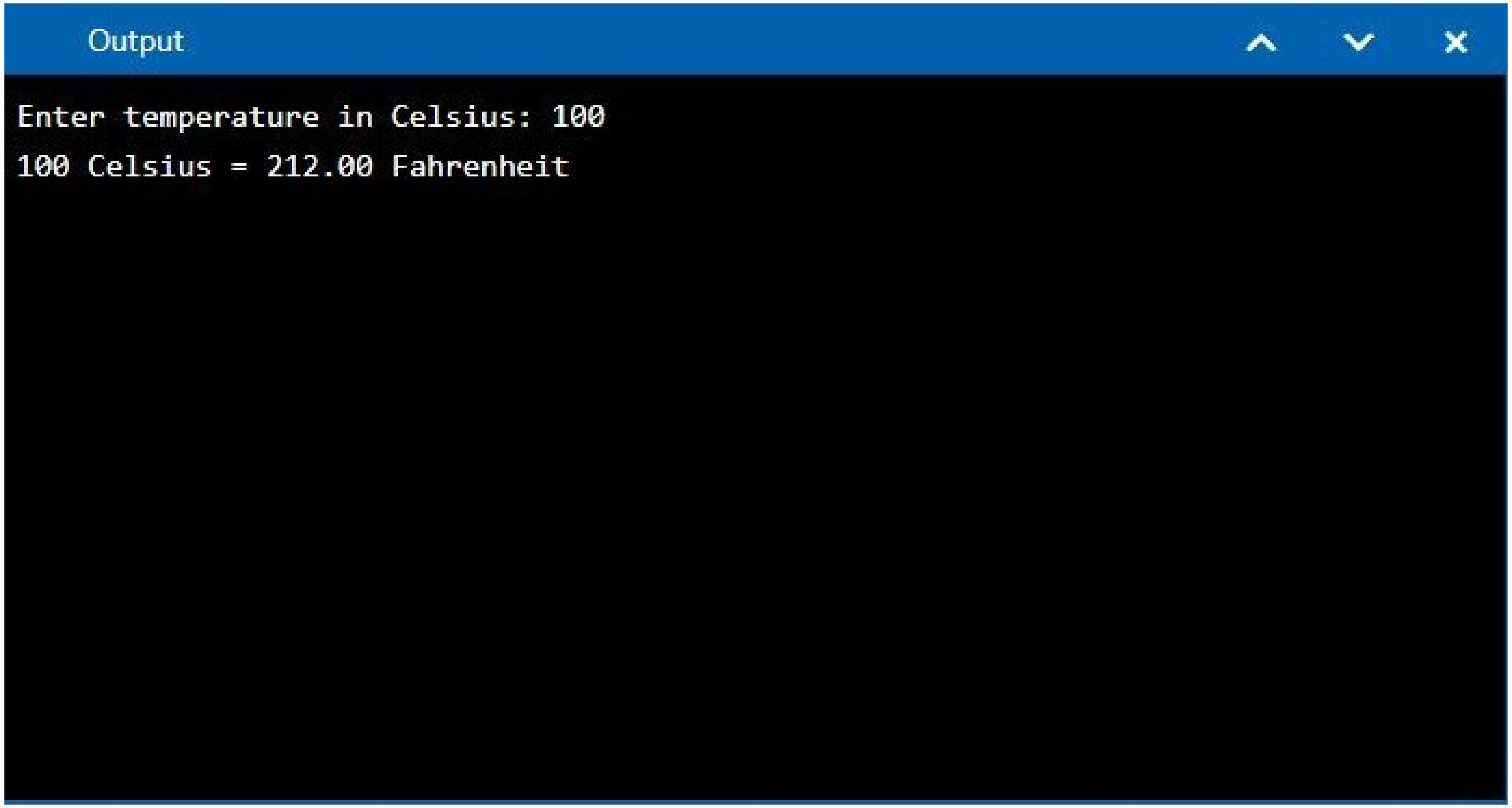
Temperature conversion formula

Temperature conversion formula from degree Celsius to Fahrenheit is given by -

$$°F = \left(°C * \frac{9}{5}\right) + 32$$



```c
/**
 * C program to convert temperature from degree celsius to fahrenheit
 */

#include <stdio.h>

int main()
{
    float celsius, fahrenheit;

    /* Input temperature in celsius */
    printf("Enter temperature in Celsius: ");
    scanf("%f", &celsius);

    /* celsius to fahrenheit conversion formula */
    fahrenheit = (celsius * 9 / 5) + 32;

    printf("%.2f Celsius = %.2f Fahrenheit", celsius, fahrenheit);

    return 0;
}
```



```
Enter temperature in Celsius: 100
100 Celsius = 212.00 Fahrenheit
```

Any Question?

Thank You All