Programming in C

Dr. Rupali P. Patil

Department of Electronics and Telecommunications

Topics for today

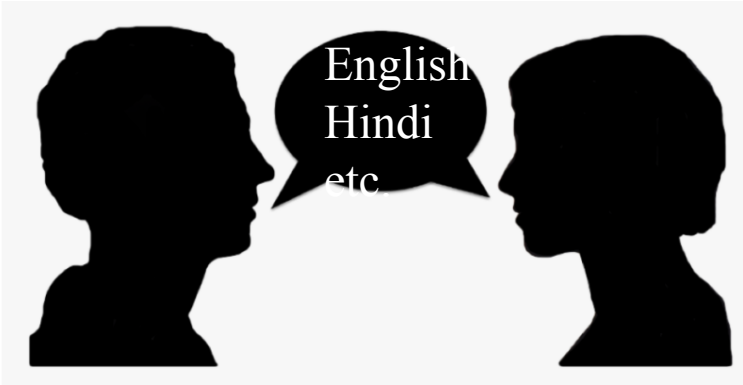
- Syllabus in brief
- Marking scheme
- Language/computer language
- Need of computer language
- Why to study C?
- Platform dependency in C

Introduction

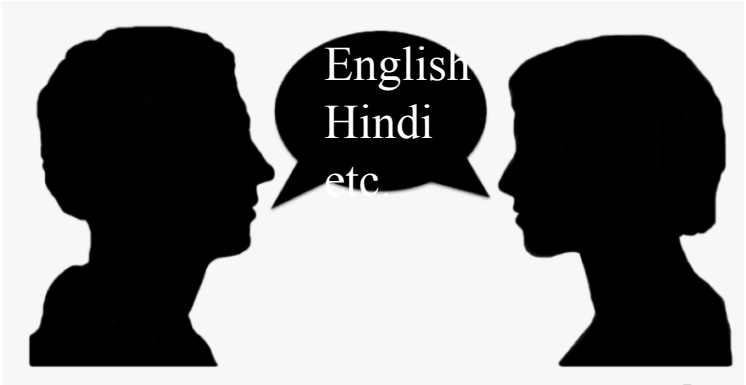
- **What is Language?**
- **What is Computer Language?**

- Follow set of instructions/ programs
- Application/ software

Introduction: Need of Computer Language

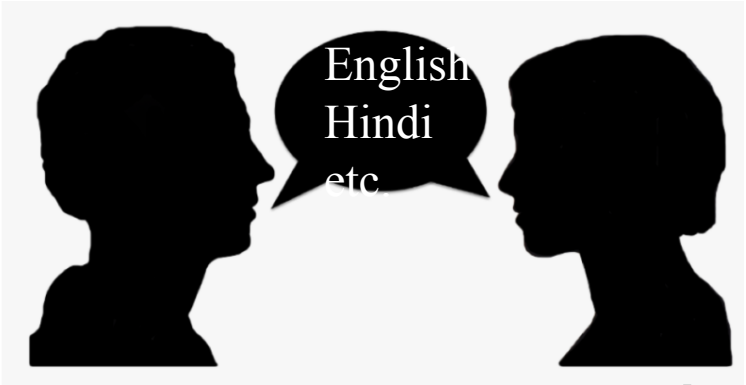


Introduction: Need of Computer Language



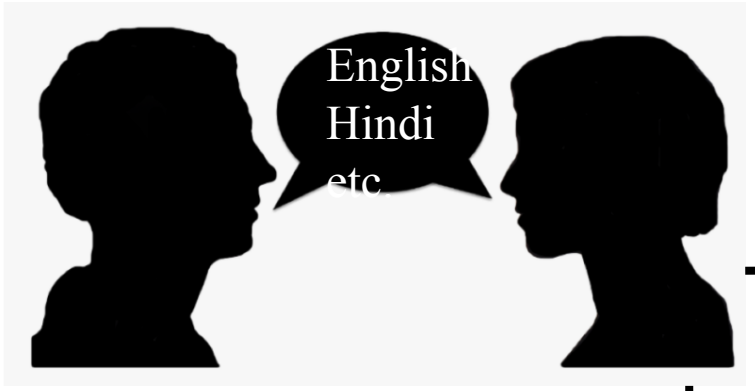
communication

Introduction: Need of Computer Language



communication X

Introduction: Need of Computer Language



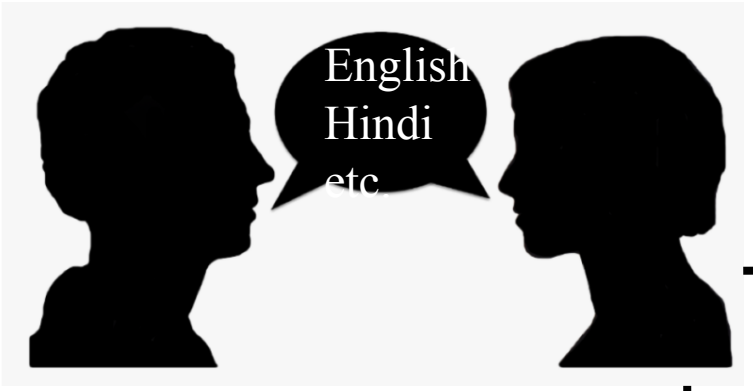
High level languages

C
C++
Java
Python, R
etc.

using

communication X

Introduction: Need of Computer Language



High level languages

C
C++
Java
Python, R
etc.

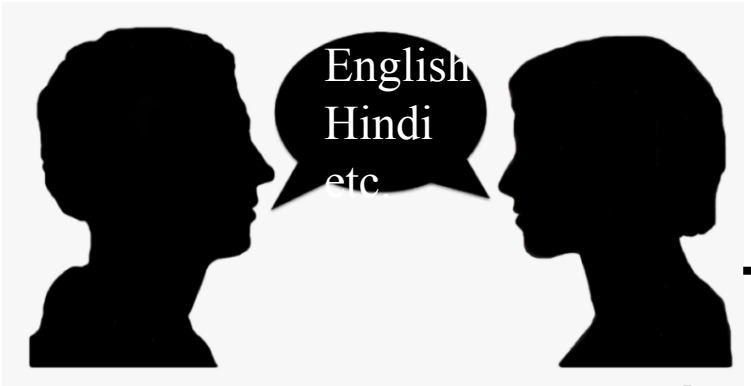
Writing programs

using

communication


```
a = 2;
b=6;
c=a + b;
Print c.
```

Introduction: Need of Computer Language



High level languages

C
C++
Java
Python, R
etc.

Writing programs

using

communication

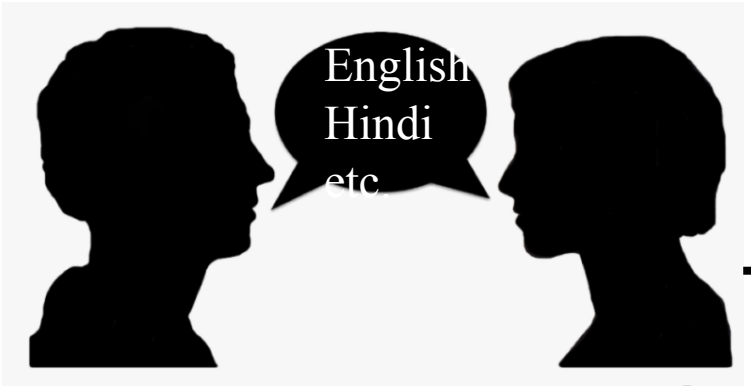
X

10110100
1000
10011
111101

Compiler

a = 2;
b=6;
C=a + b;
Print c.

Introduction: Need of Computer Language



High level languages

C
C++
Java
Python, R
etc.

Writing programs

using

communication

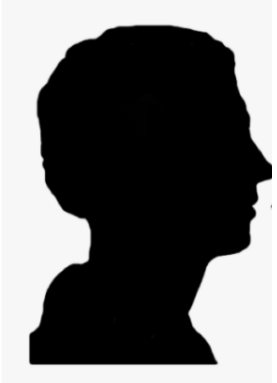
X

10110100
1000
10011
111101

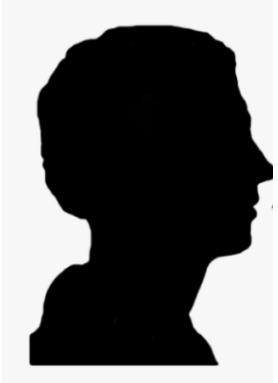
Compiler

a = 2;
b=6;
C=a + b;
Print c.

Introduction: Need of Computer Language

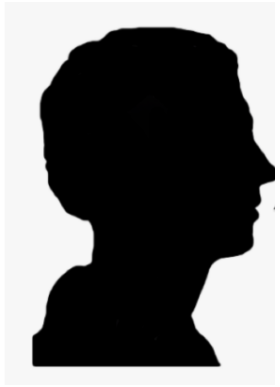


Programmer



End user

Introduction: Need of Computer Language



Programmer

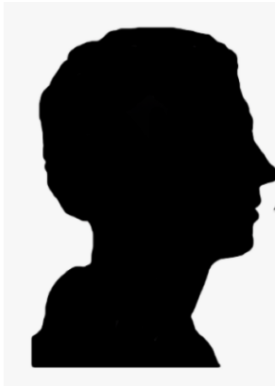
provides

Interface/Application

Choose the language

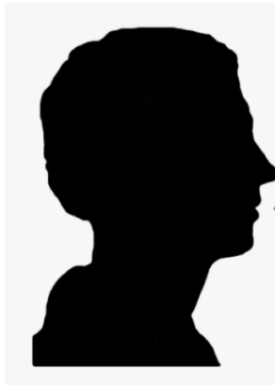
English
Hindi
etc.

Can communicate using



End user

Introduction: Need of Computer Language



Programmer

provides

Interface/Application

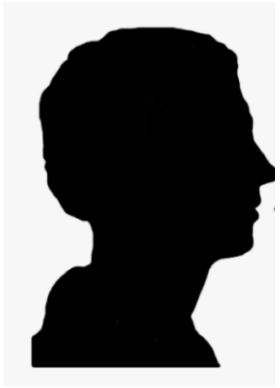
Choose the language

English
Hindi
etc.

Server

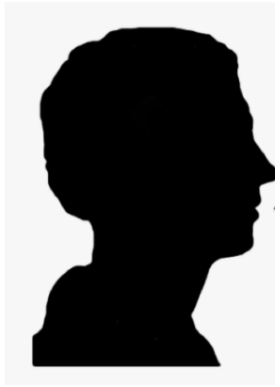
Database

Can communicate using



End user

Introduction: Need of Computer Language



Programmer

provides

Interface/Application

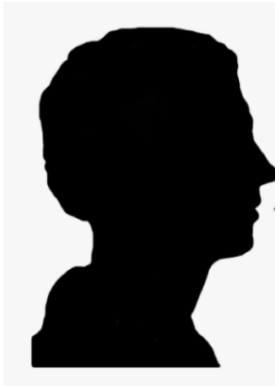
Choose the language

English
Hindi
etc.

Server

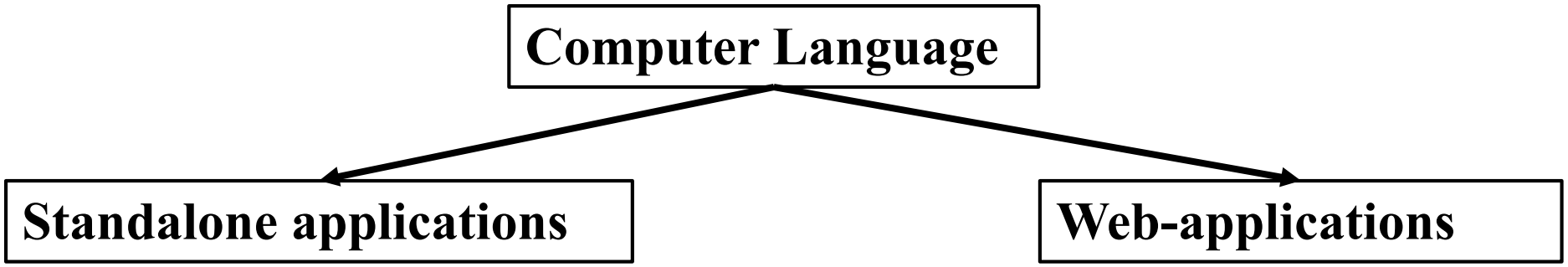
Database

Can communicate using

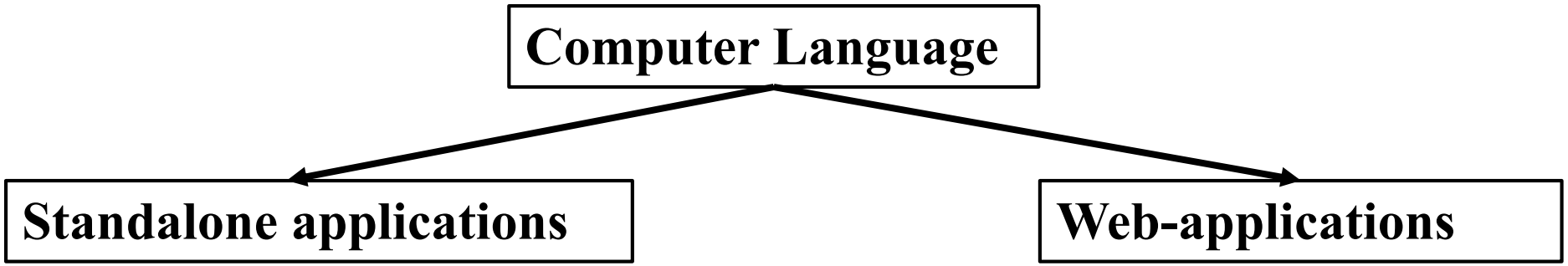


End user

Introduction: Computer Language

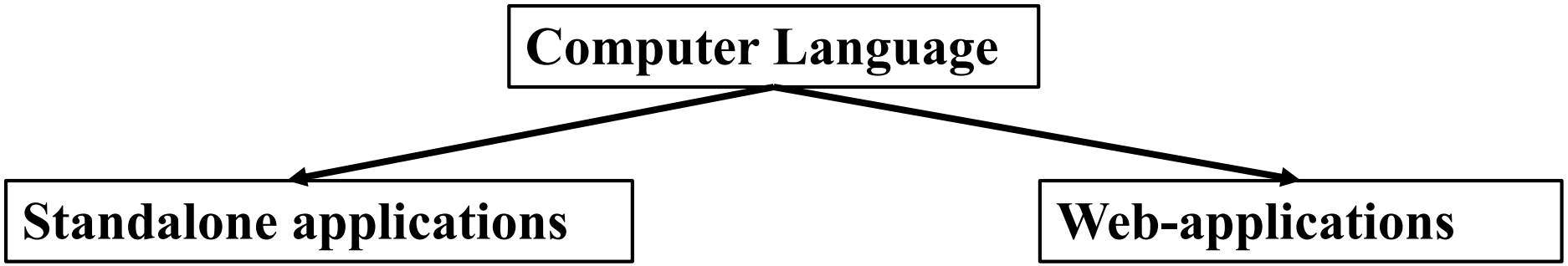


Introduction: Computer Language



- Must be installed
- Compatible to single OS

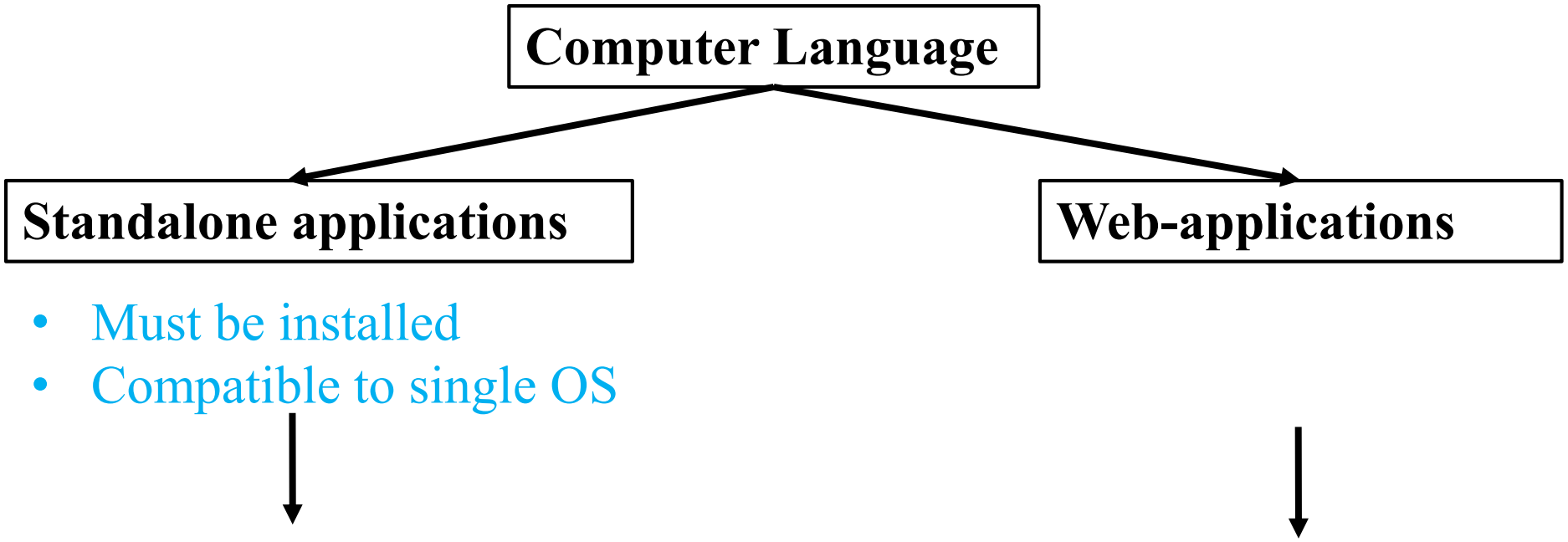
Introduction: Computer Language



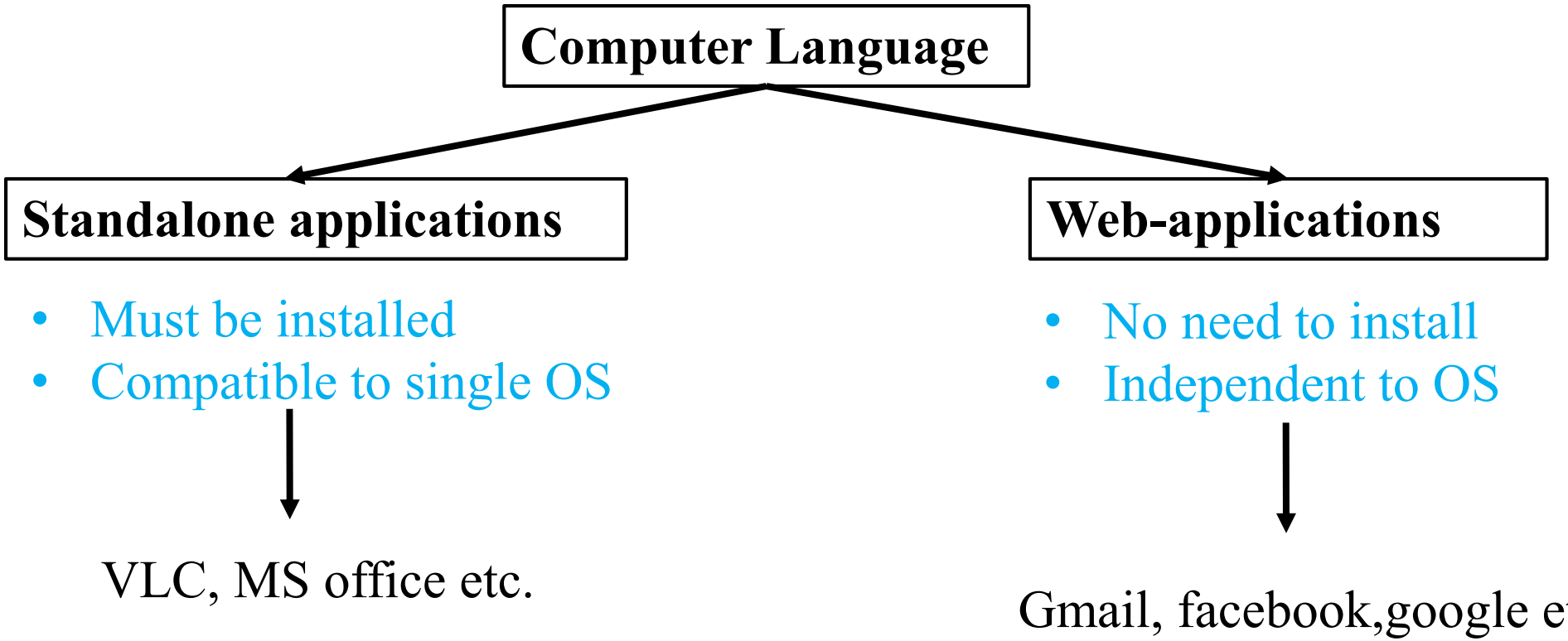
- Must be installed
- Compatible to single OS

VLC, MS office etc.

Introduction: Computer Language

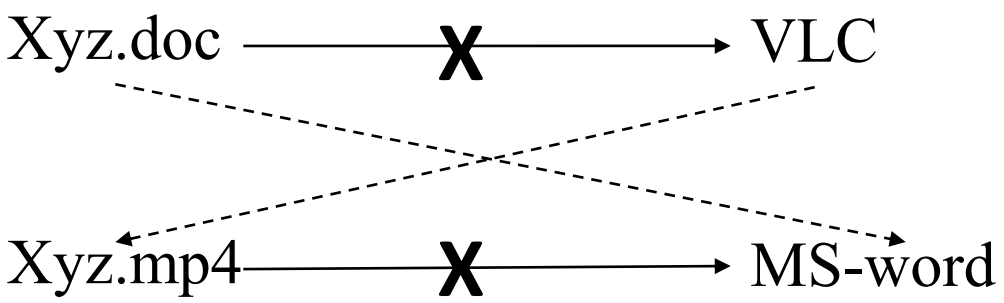


Introduction: Computer Language



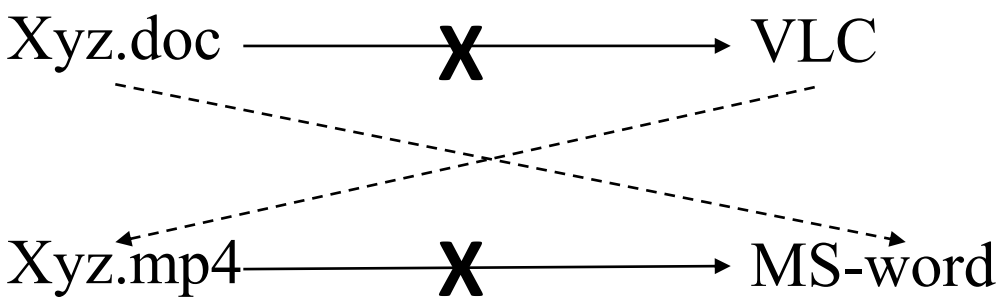
Introduction: Why OS understand only particular software?

File extensions



Introduction: Why OS understand only particular software?

File extensions

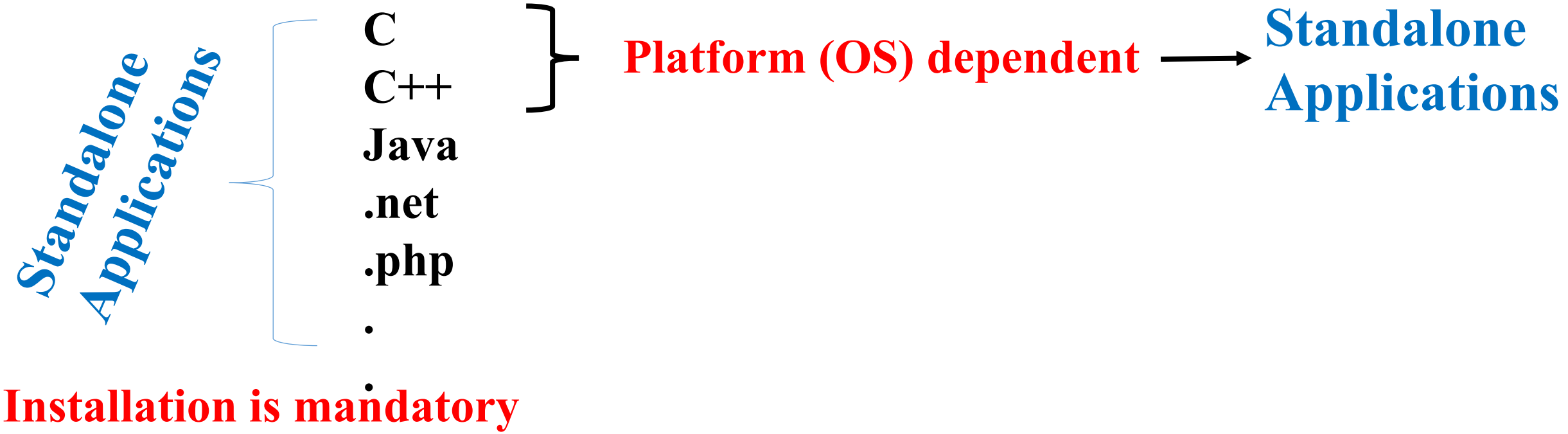


OS extensions

Windows---.exe
Mac---.dmg
Linux---.rpm,.tar

Introduction: Is programming language standalone or web-application?

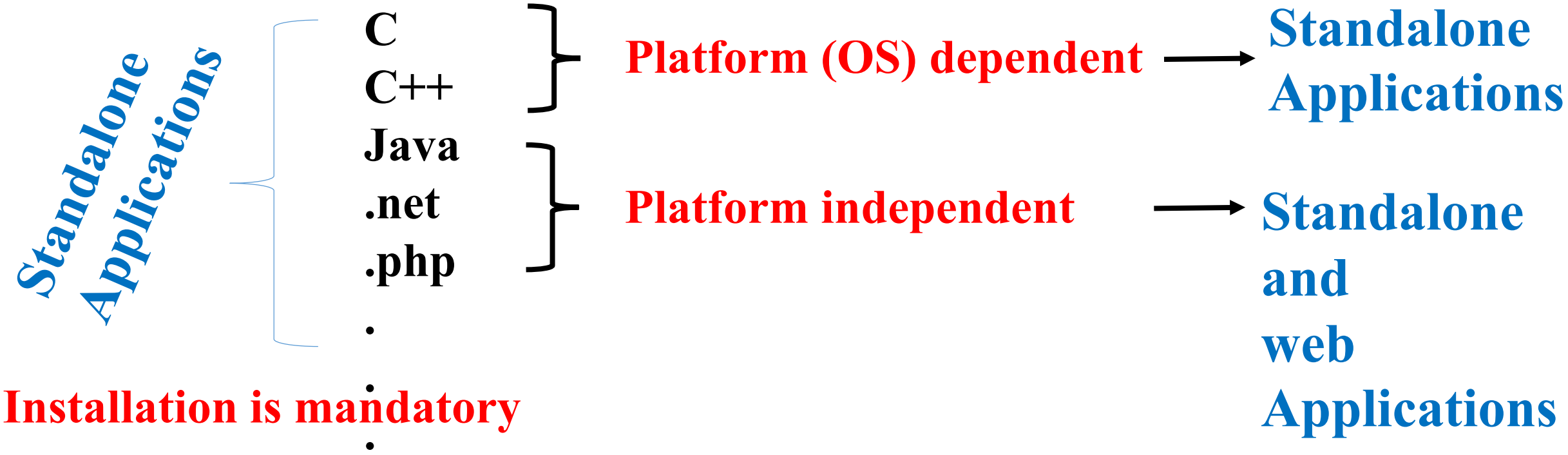
Programming languages



Installation is mandatory

Introduction: Is programming language standalone or web-application?

Programming languages



Installation is mandatory

Introduction: Why to study C language ?

C---

C++---

Java, .net, PHP---

Introduction: Why to study C language ?

C---embedded system programming

ie

Software used by electronic devices like washing machine, Refrigerator etc

C++---

Java, .net, PHP---

Introduction: Why to study C language ?

C---embedded system programming

ie

Software used by electronic devices like washing machine, Refrigerator etc

C++---Gaming library

Java, .net, PHP---

Introduction: Why to study C language ?

C---embedded system programming

ie

Software used by electronic devices like washing machine, Refrigerator etc

C++---Gaming library

Java, .net, PHP---Enterprise related applications or web applications

Introduction: Platform dependency in C ?

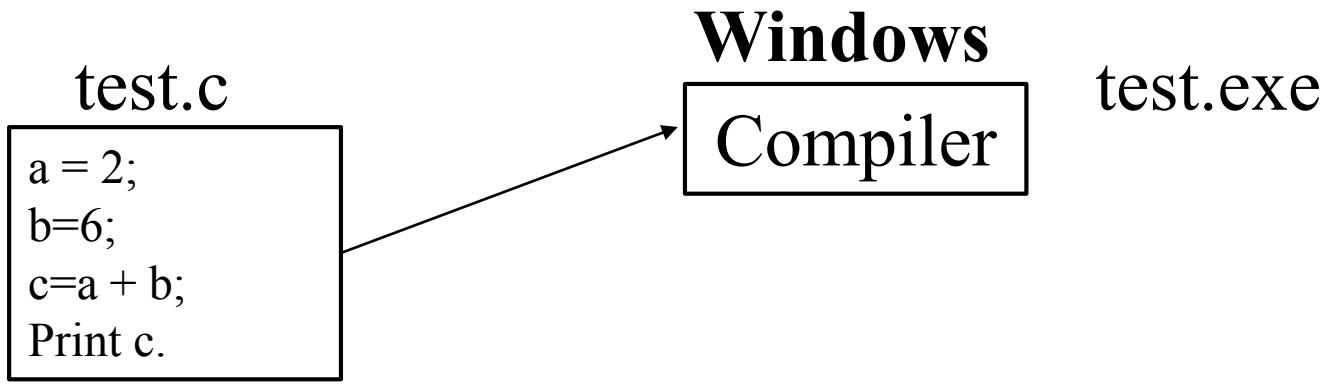
Windows--------C.exe----------Compiler + Library

Mac-----------C.dmg---------Compiler + Library

Introduction: Platform dependency in C ?

Windows--------C.exe----------Compiler + Library

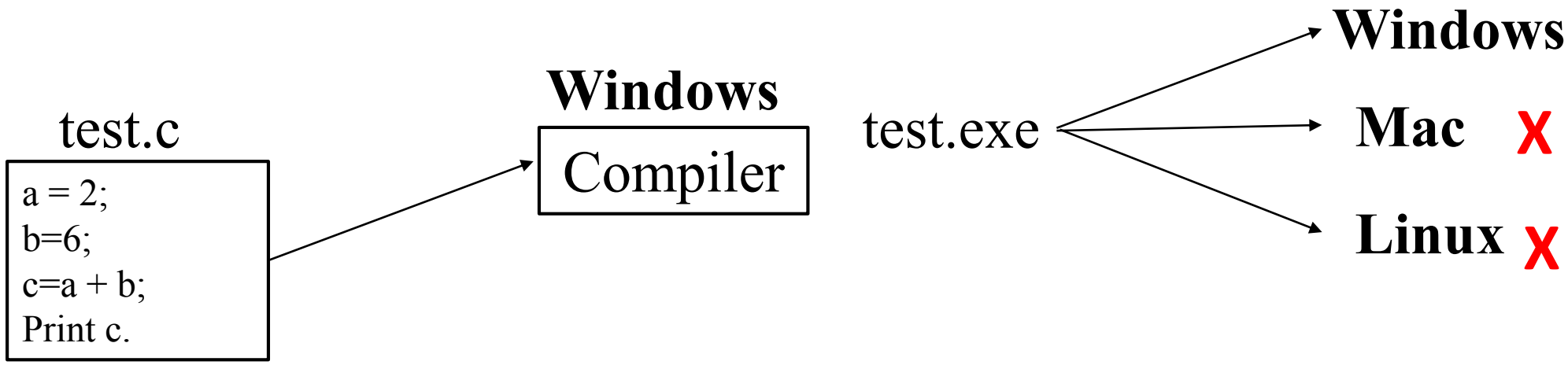
Mac-------------C.dmg---------Compiler + Library



Introduction: Platform dependency in C ?

Windows--------C.exe----------Compiler + Library

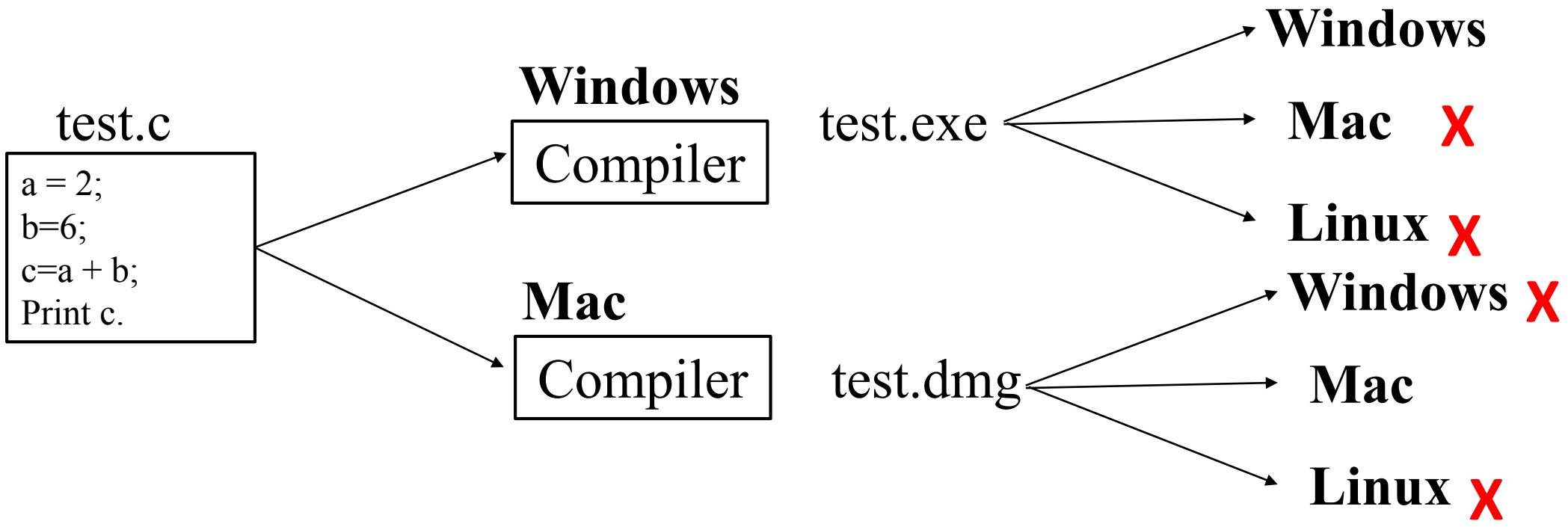
Mac-------------C.dmg---------Compiler + Library



Introduction: Platform dependency in C ?

Windows--------C.exe----------Compiler + Library

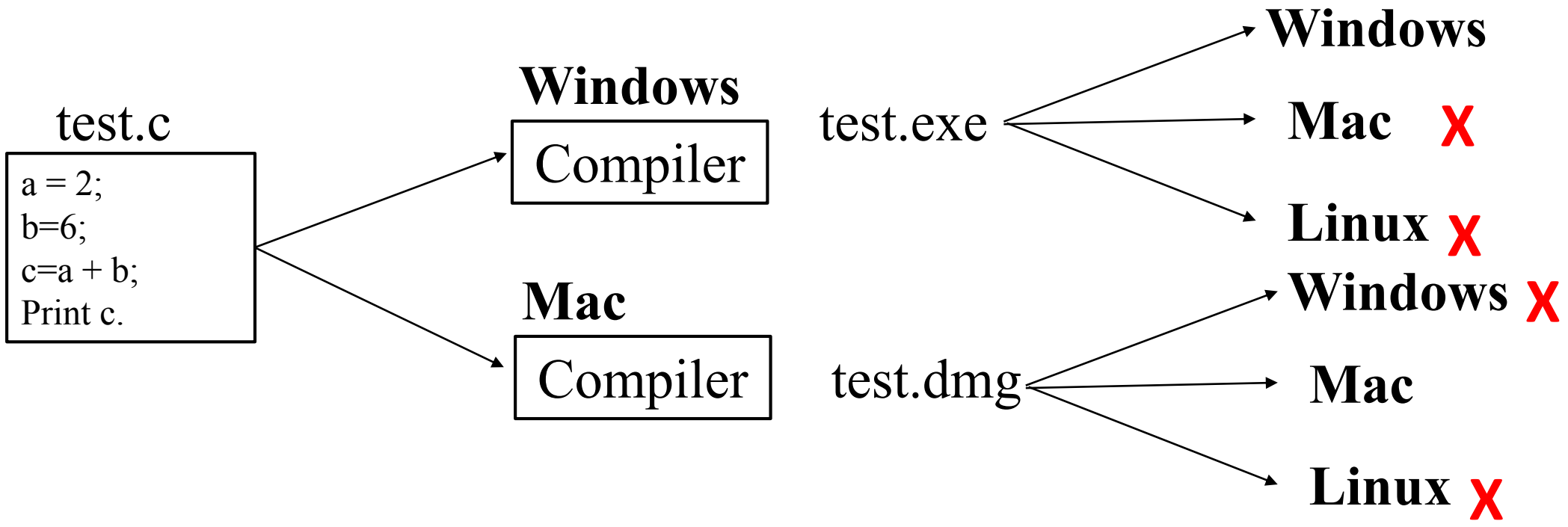
Mac-----------C.dmg---------Compiler + Library



Introduction: Platform dependency in C ?

Windows--------C.exe----------Compiler + Library

Mac-----------C.dmg---------Compiler + Library



Problem solving skill development: Logical skills, Algorithms and flowcharts

- **Understand the Problem**
- **Improve Logical skills**

Problem solving skill development: Logical skills, Algorithms and flowcharts

- A step by step way/method to solve any problem is algorithm.
- Algorithm contains:

  General statements
  Data processing
  Reasoning
  calculations
  In short everything that is required to solve any given problem

- Algorithm can be represented using any natural language, flow chart or pseudo code.

Problem solving skill development: Logical skills, Algorithms and flowcharts

Problem: Print 1 to 20 numbers

Algorithm:

Step 1: Initialize variable “a” = 0;

Step 2: Increment “a”

Step 3: Print “a”

Step 4: Check “a” is less than 20, goto Step 2

Problem solving skill development: Logical skills, Algorithms and flowcharts

- A graphical or diagrammatical representation of an algorithm is Flowchart

Problem: Print 1 to 20 numbers

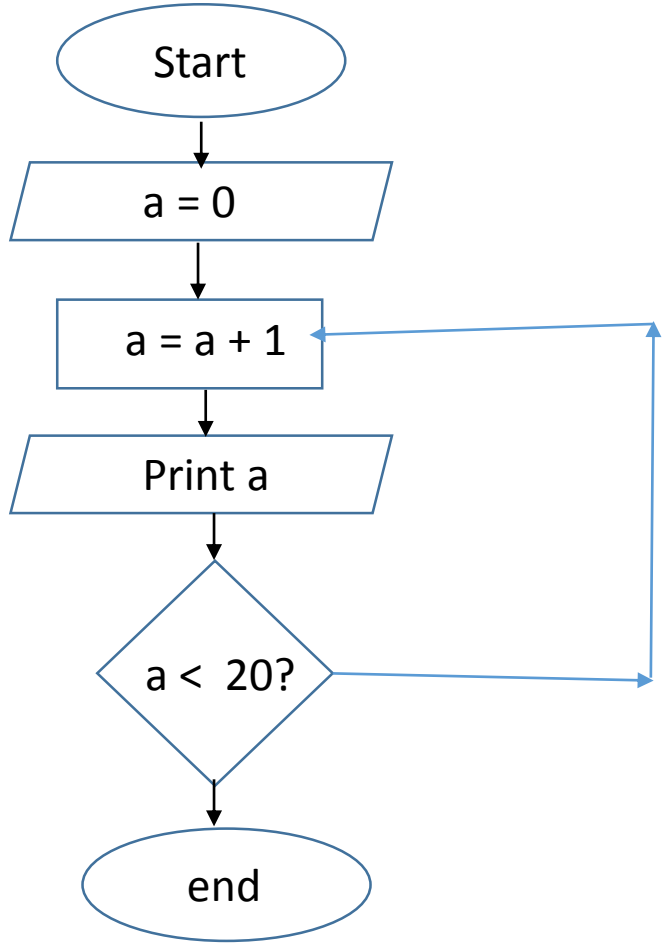
Algorithm:

Step 1: Initialize variable “a” = 0;

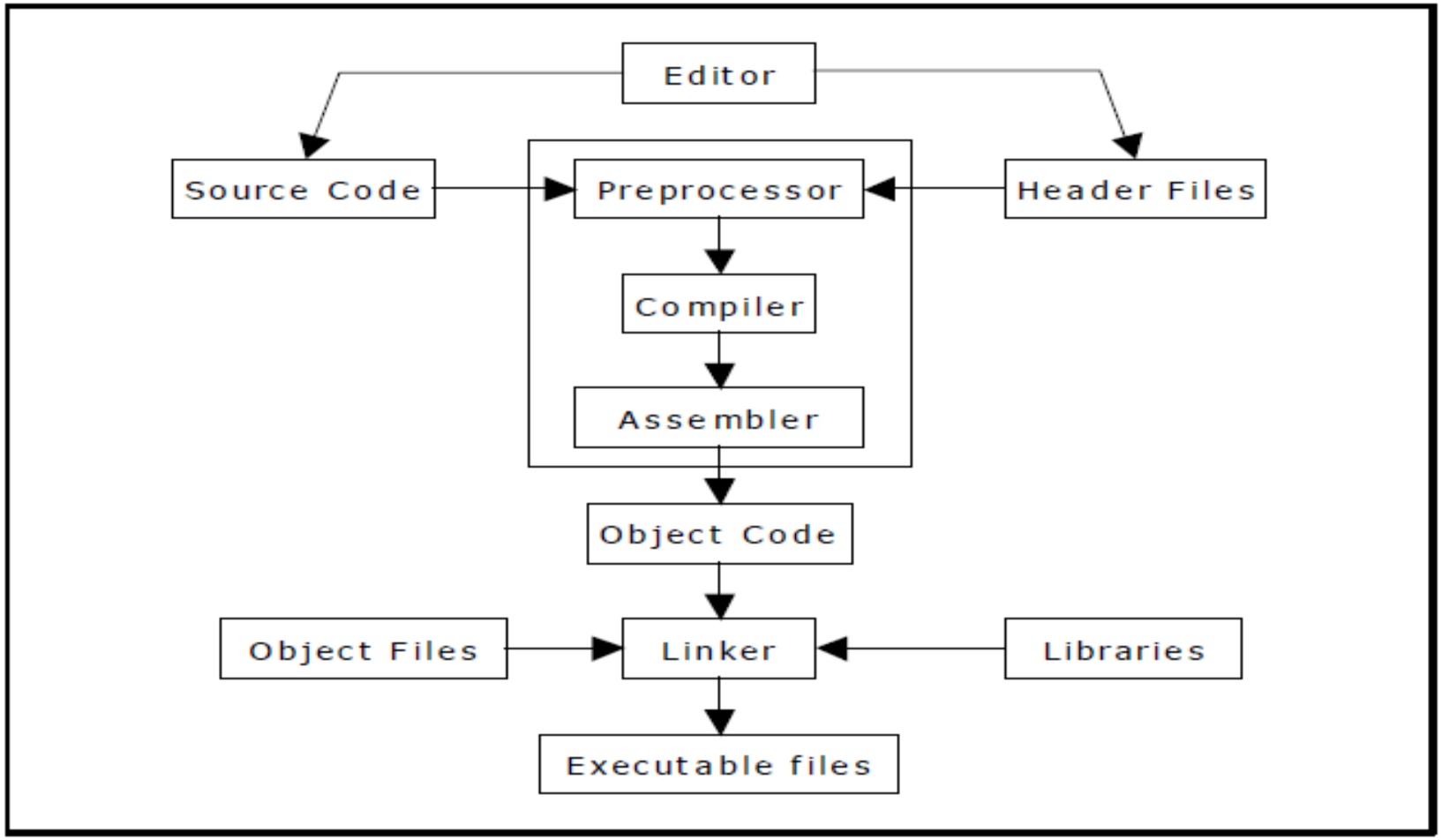
Step 2: Increment “a”

Step 3: Print “a”

Step 4: Check “a” is less than 20, goto Step 2

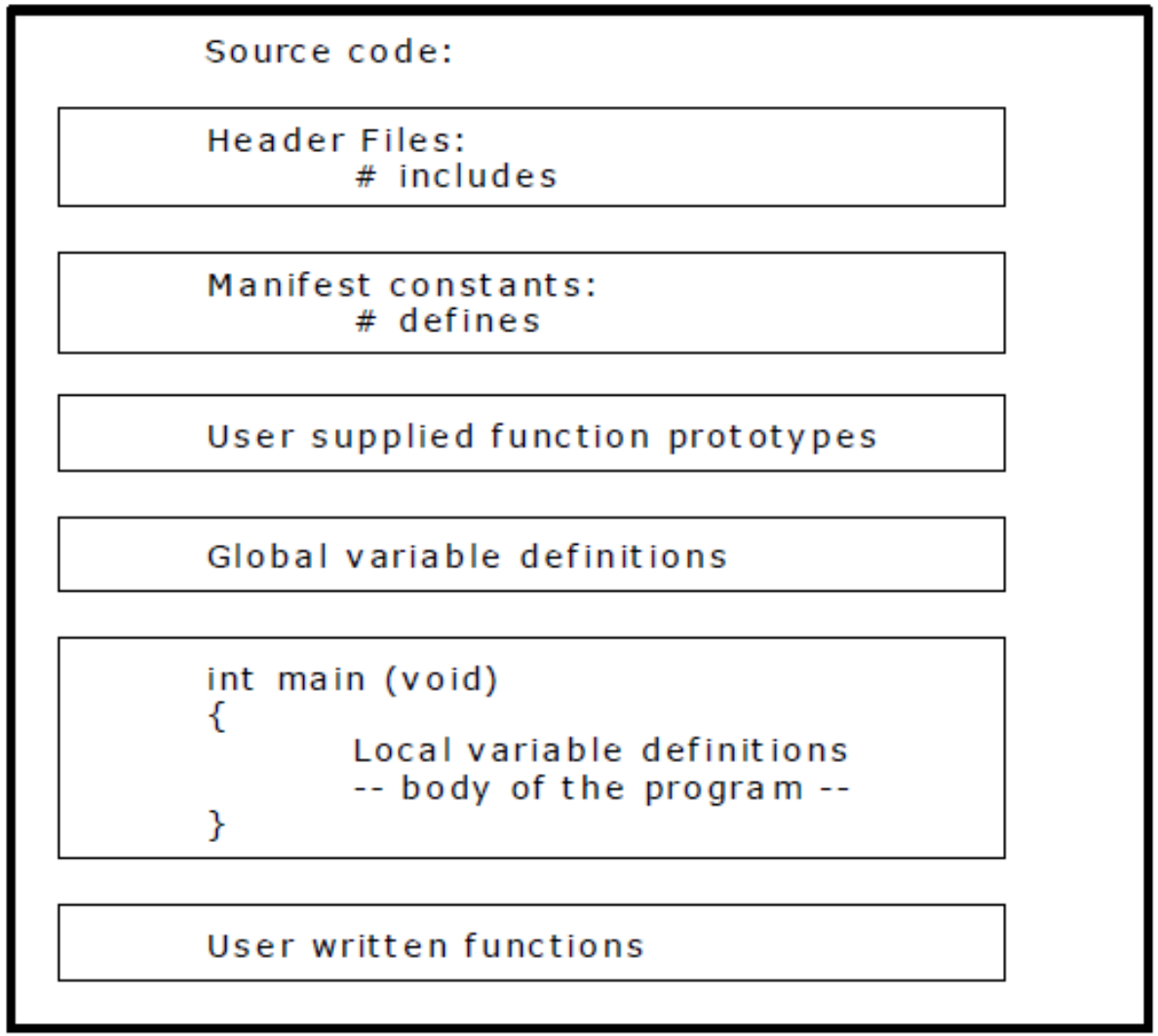


Compiling & Executing C Program



Stages of Compilation and Execution

Simple C program structure:



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

Elements of C program

- Token in C

A C program consists of various tokens and a token is either a keyword, an identifier, a constant, a string literal, or a symbol. For example, the following C statement consists of five tokens:

```
printf("Hello, World! \n");
```

The individual tokens are:

```
printf
(
"Hello, World! \n"
)
;
```

Elements of C program

- Identifiers in C

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
```

Elements of C program

- Keywords in C

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.

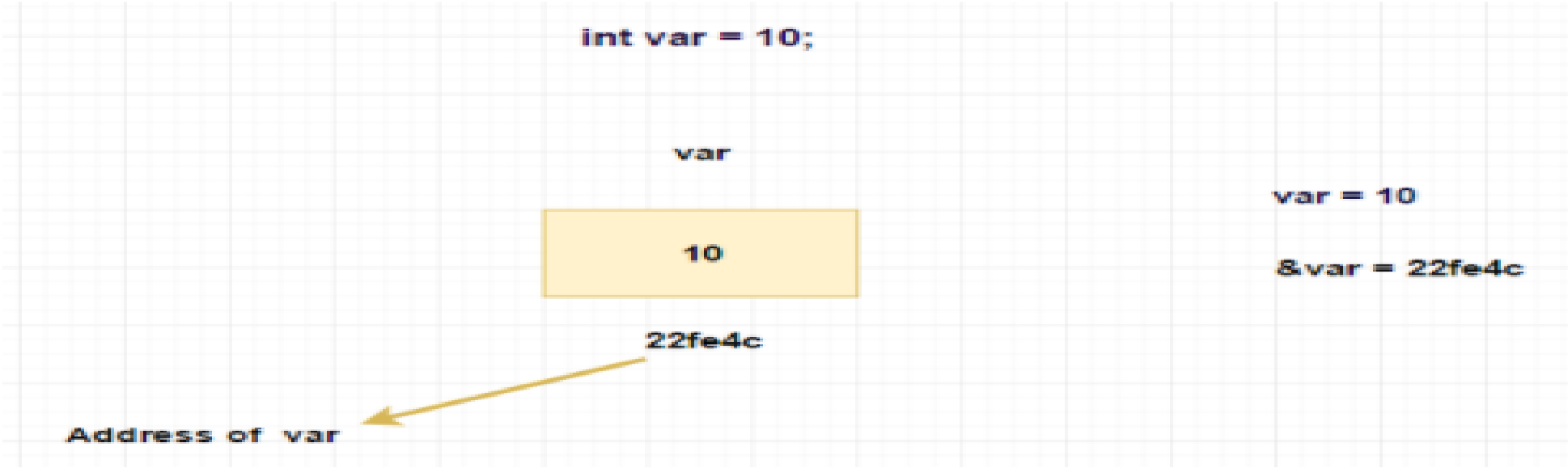
auto	else	Long	switch
break	enum	register	typedef
case	extern	return	union
char	float	short	unsigned
const	for	signed	void
continue	goto	sizeof	volatile
default	if	static	while
do	int	struct	_packed
double			

Compiler specific

Elements of C program

- Constants and variables in C

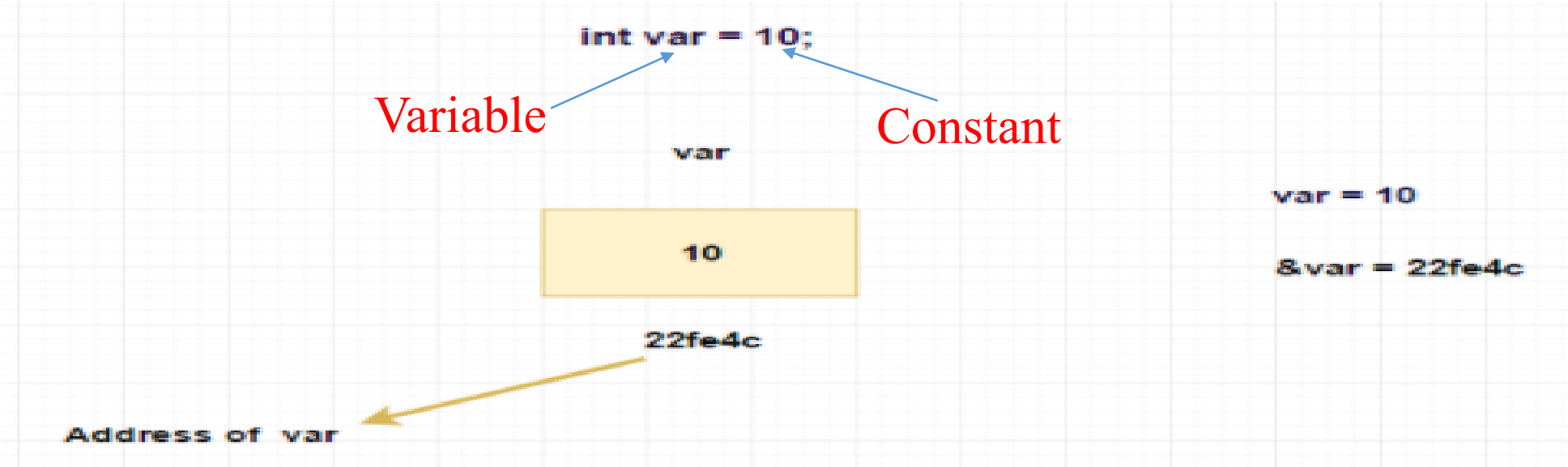
A constant is an entity that doesn't change whereas a variable is an entity that may change.



Elements of C program

- Constants and variables in C

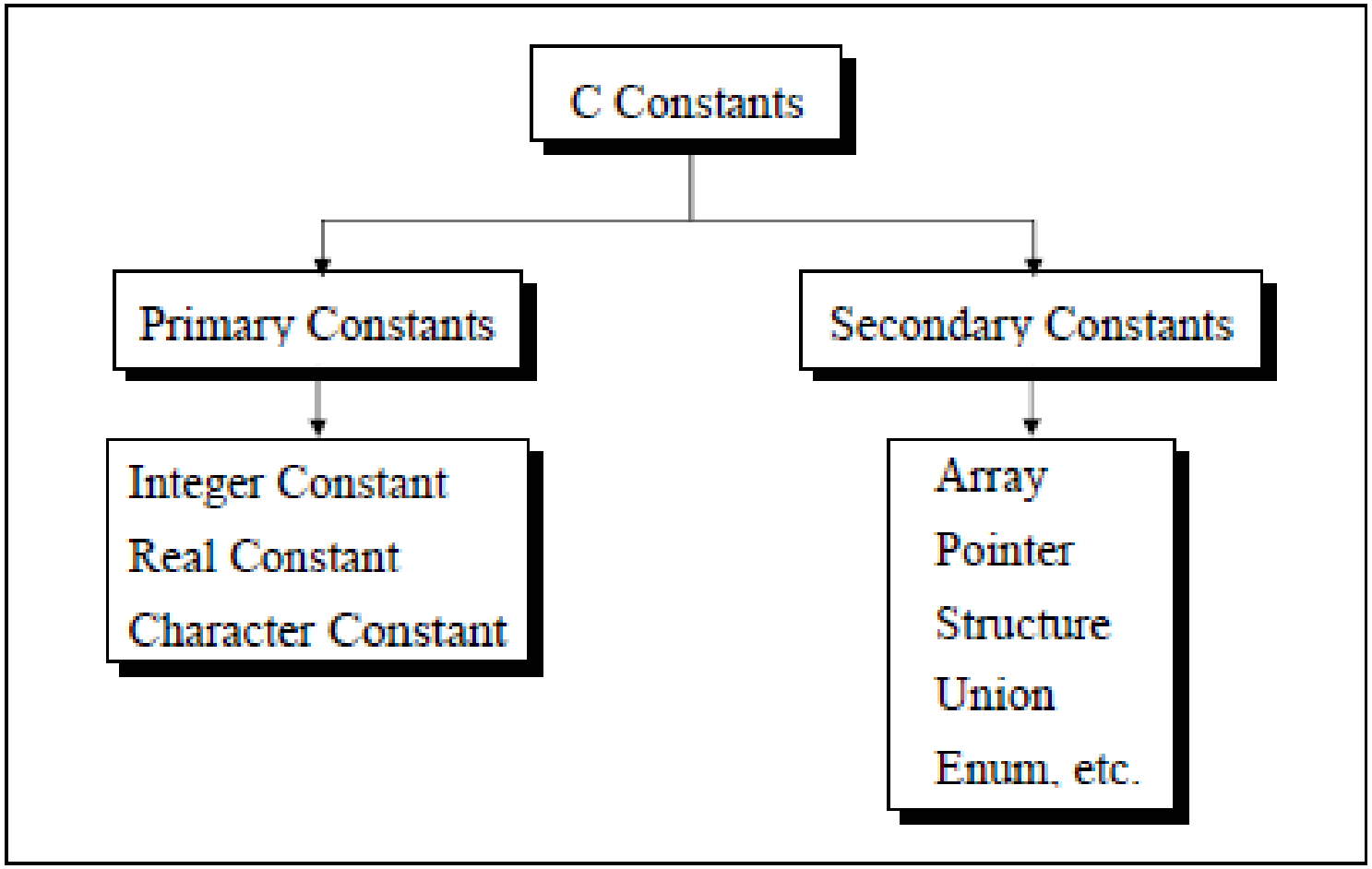
A constant is an entity that doesn't change whereas a variable is an entity that may change.



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.

Elements of C program

- Types of Constants in C



Elements of C program

- Rules for Constructing Integer Constants in C

(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.

Elements of C program

- Rules for Constructing Character Constants in C

(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'
'='

Elements of C program

- Rules for Constructing Variable Names in C

(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

Elements of C program

- Data types in C

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

Elements of C program

- Data types in C

To get the exact size of a type or a variable on a particular platform, you can use the **sizeof** operator. The expressions **sizeof(type)** yields the storage size of the object or type in bytes. Following is an example to get the size of **int** type on any machine:

```
#include <stdio.h>
#include <limits.h>

int main()
{
   printf("Storage size for int : %d \n", sizeof(int));

   return 0;
}
```

Program to take input of various datatypes in C

- Taking integer as input from user: input and display two numbers at a time
- Taking float as input from user
- Taking character as input from user

Program to take input of various datatypes in C

%d and %i, both are used to take numbers as input from the user.
%f is the format specifier to take float as input from the user
%c is the format specifier to take character as input from the user

Program to take input of various datatypes in C

Lets try this

```
int a=25;
float b=5.67;
char ch=‘g’;
char s[]=“Hello”;
printf ( "\n%c %d %f", ch, ch, ch ) ;
printf ( "\n%s %d %f", s, s, s ) ;
printf ( "\n%c %d %f",a ,a, a ) ;
printf ( "\n%f %d\n", b, b ) ;
```

ASCII value table

Dec=ASCII value

Dec	Hex	Oct	Char	Dec	Hex	Oct	Char	Dec	Hex	Oct	Char	Dec	Hex	Oct	Char
0	0	0		32	20	40	[space]	64	40	100	@	96	60	140	`
1	1	1		33	21	41	!	65	41	101	A	97	61	141	a
2	2	2		34	22	42	"	66	42	102	B	98	62	142	b
3	3	3		35	23	43	#	67	43	103	C	99	63	143	c
4	4	4		36	24	44	$	68	44	104	D	100	64	144	d
5	5	5		37	25	45	%	69	45	105	E	101	65	145	e
6	6	6		38	26	46	&	70	46	106	F	102	66	146	f
7	7	7		39	27	47	'	71	47	107	G	103	67	147	g
8	8	10		40	28	50	(	72	48	110	H	104	68	150	h
9	9	11		41	29	51	)	73	49	111	I	105	69	151	i
10	A	12		42	2A	52	*	74	4A	112	J	106	6A	152	j
11	B	13		43	2B	53	+	75	4B	113	K	107	6B	153	k
12	C	14		44	2C	54	,	76	4C	114	L	108	6C	154	l
13	D	15		45	2D	55	-	77	4D	115	M	109	6D	155	m
14	E	16		46	2E	56	.	78	4E	116	N	110	6E	156	n
15	F	17		47	2F	57	/	79	4F	117	O	111	6F	157	o
16	10	20		48	30	60	0	80	50	120	P	112	70	160	p
17	11	21		49	31	61	1	81	51	121	Q	113	71	161	q
18	12	22		50	32	62	2	82	52	122	R	114	72	162	r
19	13	23		51	33	63	3	83	53	123	S	115	73	163	s
20	14	24		52	34	64	4	84	54	124	T	116	74	164	t
21	15	25		53	35	65	5	85	55	125	U	117	75	165	u
22	16	26		54	36	66	6	86	56	126	V	118	76	166	v
23	17	27		55	37	67	7	87	57	127	W	119	77	167	w
24	18	30		56	38	70	8	88	58	130	X	120	78	170	x
25	19	31		57	39	71	9	89	59	131	Y	121	79	171	y
26	1A	32		58	3A	72	:	90	5A	132	Z	122	7A	172	z
27	1B	33		59	3B	73	;	91	5B	133	[	123	7B	173	{
28	1C	34		60	3C	74	<	92	5C	134	\	124	7C	174	|
29	1D	35		61	3D	75	=	93	5D	135	]	125	7D	175	}
30	1E	36		62	3E	76	>	94	5E	136	^	126	7E	176	~
31	1F	37		63	3F	77	?	95	5F	137	_	127	7F	177	

Elements of C program

- Local and Global Variables in C

Local Variable:

The variables which are declared inside the function, compound statement (or block) are called Local variables.

```
void function_1()
{
   int a, b; // you can use a and b within braces only
}

void function_2()
{
   printf("%d\n", a); // ERROR, function_2() doesn't know any variable a
}
```

Elements of C program

- Local and Global Variables in C

Local Variable:

The variables which are declared inside the function, compound statement (or block) are called Local variables.

```
int main()
{
   int a = 100;

   {
      int a = 10;
      printf("Inner a = %d\n", a);
   }

   printf("Outer a = %d\n", a);

   return 0;
}
```

Elements of C program

- Local and Global Variables in C

Global Variable:

- The variables declared outside any function are called global variables.
- They are not limited to any function.
- Any function can access and modify global variables.
- Global variables are automatically initialized to 0 at the time of declaration.
- Global variables are generally written before main() function.

```
#include<stdio.h>
void func_1();
void func_2();
int a, b = 10;  // declaring and initializing global variables
int main()
{
   printf("Global a = %d\n", a);
   printf("Global b = %d\n\n", b);

   func_1();
   func_2();
   return 0;
}
void func_1()
{
   printf("From func_1() Global a = %d\n", a);
   printf("From func_1() Global b = %d\n\n", b);
}
void func_2()
{
   int a = 5;
printf("Inside func_2() a = %d\n", a);
}
```

Elements of C program

- Constants in C

```
#include <stdio.h>
#define num 25
#define pi 3.14
int main() {
   float p, r=2.5;
   printf("The value of pi is: %f", pi);
   p= 2* pi *r;
   printf("The value of perimeter is: %f", p);
   return 0;
}
```

Elements of C program

- Input/output functions

There are numerous library functions available for I/O. These can be classified into two broad categories:

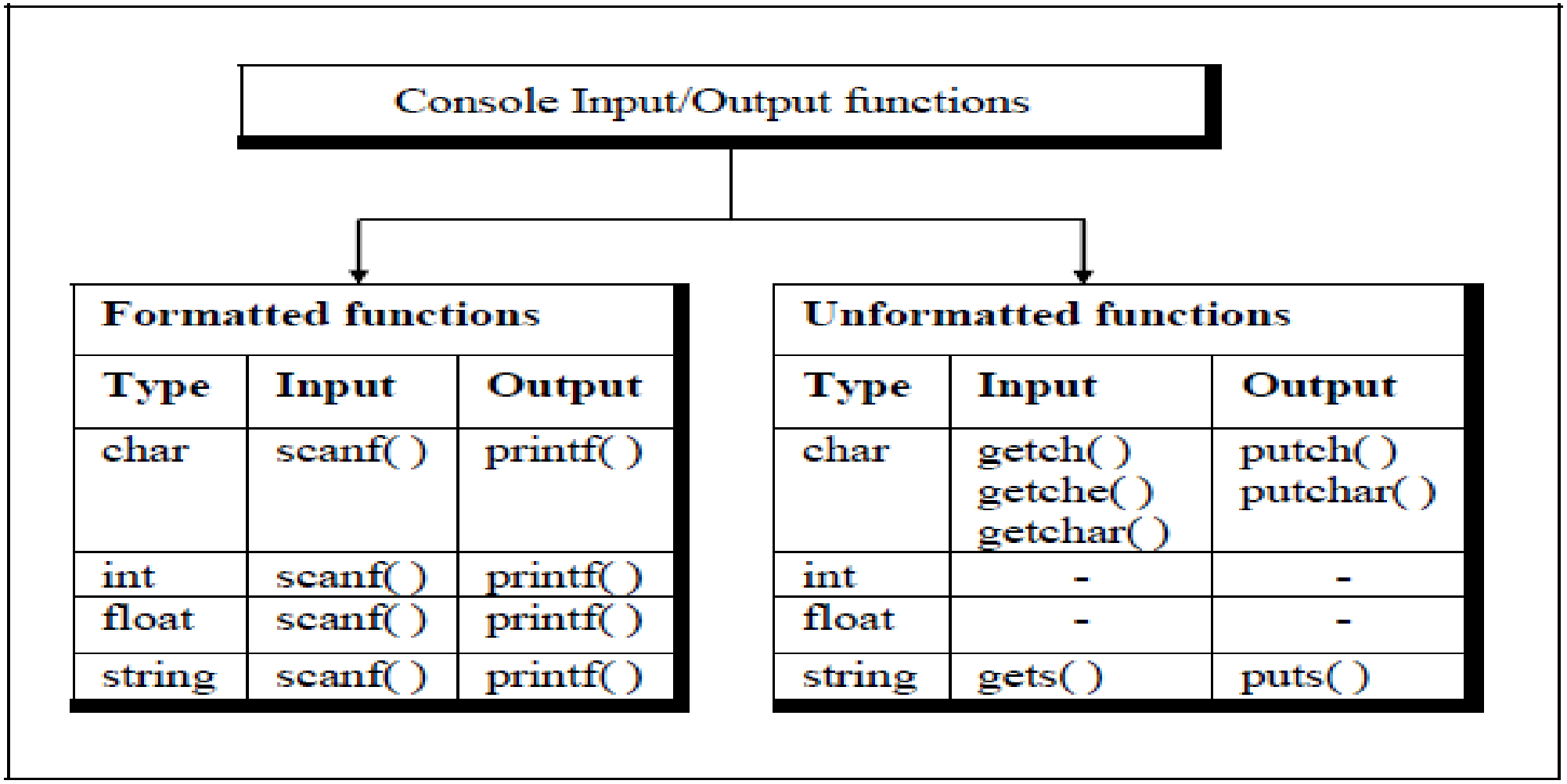
(a) Console I/O functions - Functions to receive input from keyboard and write output to VDU.

(b) File I/O functions - Functions to perform I/O operations on a floppy disk or hard disk.

In this chapter we would be discussing only Console I/O functions.

Elements of C program

- Console Input/output functions



Formatted functions

Type	Input	Output
char	scanf( )	printf( )
int	scanf( )	printf( )
float	scanf( )	printf( )
string	scanf( )	printf( )

Unformatted functions

Type	Input	Output
char	getch( ) getche( ) getchar( )	putch( ) putchar( )
int	-	-
float	-	-
string	gets( )	puts( )

Elements of C program

- Formatted console Input/output functions

The formatted functions allow the input read from the keyboard or the output displayed on the VDU to be formatted as per our requirements.

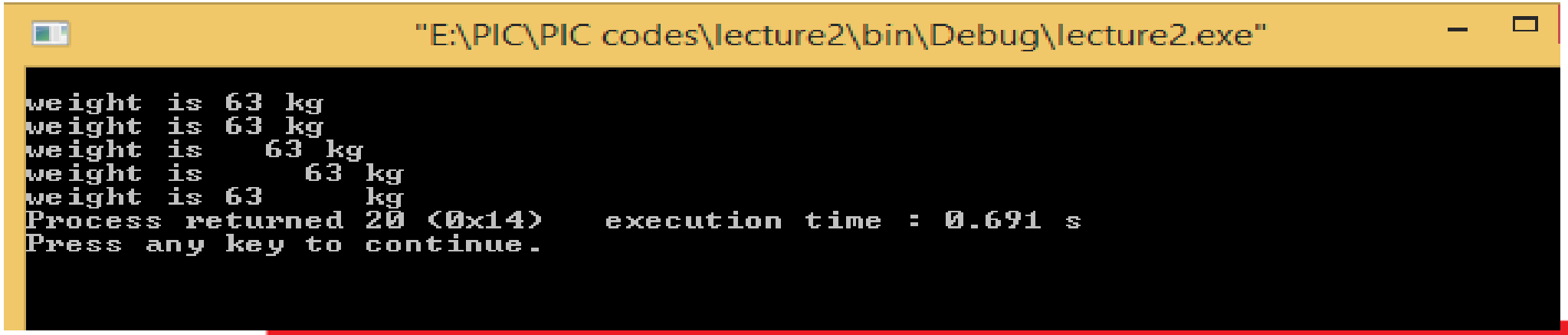
For example, if values of average marks and percentage marks are to be displayed on the screen, then the details like where this output would appear on the screen, how many spaces would be present between the two values, the number of places after the decimal points, etc. can be controlled using formatted functions.

Elements of C program

- Formatted console Input/output functions-Format Specifiers

```
main( )
{
int weight = 63 ;
printf ( "\nweight is %d kg", weight ) ;
printf ( "\nweight is %2d kg", weight ) ;
printf ( "\nweight is %4d kg", weight ) ;
printf ( "\nweight is %6d kg", weight ) ;
printf ( "\nweight is %-6d kg", weight ) ;
}
```

The output of the program would look like this ...



Elements of C program

- Formatted console Input/output functions-Format Specifiers

```
/* Formatting strings with printf( ) */
main( )
{
char firstname1[ ] = "Sandy" ;
char surname1[ ] = "Malya" ;
char firstname2[ ] = "AjayKumar" ;
char surname2[ ] = "Gurubaxani" ;
printf ( "\n%20s%20s", firstname1, surname1 ) ;
printf ( "\n%20s%20s", firstname2, surname2 ) ;
}
```

And here's the output...
Columns

```
01234567890123456789012345678901234567890123456789
                   Sandy               Malya
                AjayKumar           Gurubaxani
```

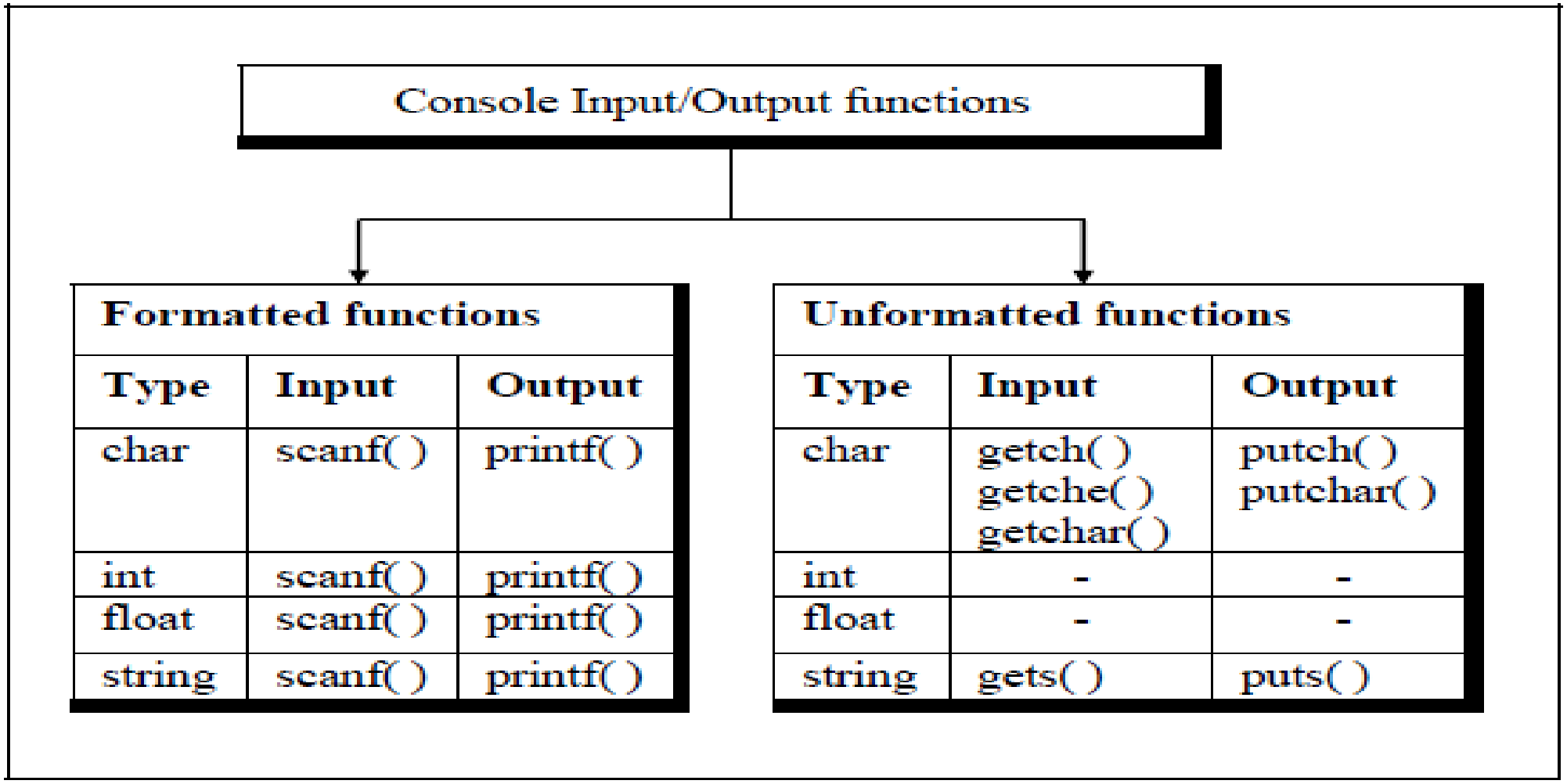
Elements of C program

- Formatted console Input/output functions- Escape sequences

Esc. Seq.	Purpose	Esc. Seq.	Purpose
\n	New line	\t	Tab
\b	Backspace	\r	Carriage return
\f	Form feed	\a	Alert
\'	Single quote	\"	Double quote
\\	Backslash		

Elements of C program

- UnFormatted console Input/output functions



Formatted functions

Type	Input	Output
char	scanf( )	printf( )
int	scanf( )	printf( )
float	scanf( )	printf( )
string	scanf( )	printf( )

Unformatted functions

Type	Input	Output
char	getch( ) getche( ) getchar( )	putch( ) putchar( )
int	-	-
float	-	-
string	gets( )	puts( )

Elements of C program

- UnFormatted console Input/output functions

```
main( )
{
char ch ;
printf ( "\nPress any key to continue" ) ;
getch( ) ; /* will not echo the character */

printf ( "\nType any character" ) ;
ch = getche( ) ; /* will echo the character typed */
printf ( "\nType any character" ) ;

getchar( ) ; /* will echo character, must be followed by enter key */
printf ( "\nContinue Y/N" ) ;
fgetchar( ) ; /* will echo character, must be followed by enter key */
}
```

```
main( )
{
char ch = 'A' ;
putch ( ch ) ;
putchar ( ch ) ;
fputchar ( ch ) ;
putch ( 'Z' ) ;
putchar ( 'Z' ) ;
fputchar ( 'Z' ) ;
}
```

Elements of C program

- UnFormatted console Input/output functions

The limitation of putch( ), putchar( ) and fputchar( ) is that they can output only one character at a time.

Solution:

gets( ) and puts( )

gets( ) receives a string from the keyboard. Why is it needed?

Because **scanf( )** function has some limitations:

```
main( )
{
char name[50] ;
printf ( "\nEnter name " ) ;
scanf ( "%s", name ) ;
printf ( "%s", name ) ;
}
```

Elements of C program

- UnFormatted console Input/output functions –gets() and puts()

The solution to this problem is to use gets( ) function.

- As said earlier, it gets a string from the keyboard.
- It is terminated when an Enter key is hit. Thus, spaces and tabs are perfectly acceptable as part of the input string.
- More exactly, gets( ) gets a newline (\n) terminated string of characters from the keyboard and replaces the \n with a \0.
- The puts( ) function works exactly opposite to gets( ) function. It outputs a string to the screen.

Here is a program which illustrate

Elements of C program

- UnFormatted console Input/output functions –gets() and puts()

```
main( )
{
char footballer[] ;
puts ( "Enter name" ) ;
gets ( footballer ) ; /* sends base address of array */
puts ( "Happy footballing!" ) ;
puts ( footballer ) ;
}
```

Elements of C program

- Completed first module from syllabus

Module No.	Unit No.	Details	Hrs. (Tutorial and Lab)	CO
1	**Introduction to C**			
	1.1	**Problem solving skill development:** Problem Definition, fundamentals of algorithms and flowcharts, Algorithms and flowchart development	04	CO1
	1.2	**Structure of C program and its Elements:** Character Set, C Tokens, Keywords and Identifiers, Literals , Variables, Data Types and its qualifiers, Declaration and Initialization of Variables, Local and Global Variables, Declaring Constants, Formatted Input/output functions and unformatted input/output functions	04	CO2

Program to display your complete name, roll no., department, college and 12th percentage entered from the user in C

Program to swap two numbers entered by user in C

Input: a= 67, b=32;

Output: a=32,b=67

The length & breadth of a rectangle and radius of a circle are input through the keyboard. Write a program to calculate the area & perimeter of the rectangle, and the area & circumference of the circle.

Program to swap two numbers entered by user in C

```
{

    int a,b,t;
    printf("Enter value of a:");
    scanf("%d",&a);
    printf("Enter value of b:");
    scanf("%d",&b);
    t=b;
    b=a;
    a=t;
    printf("a=%d,b=%d",a,b);
 }
```

Input: a= 67, b=32;

Output: a=32,b=67