

CS -01

PROBLEM SOLVING
METHODOLOGIES AND
PROGRAMMING IN C

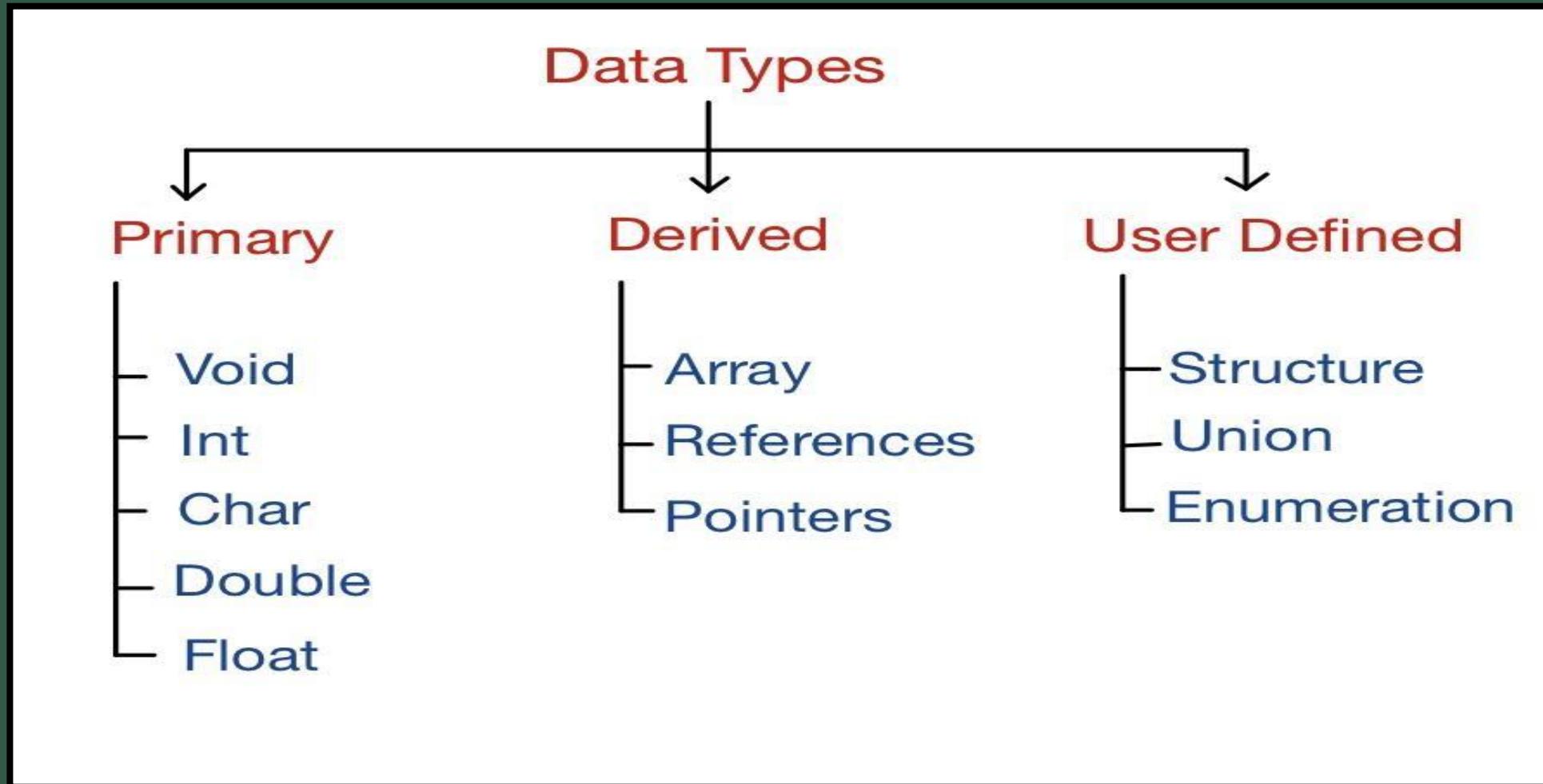
By Rachel

UNIT 1 - PART 4

TOPIC 8

DATA TYPES IN C

Data is differentiated into various types in C language.



Type	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 65,535
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	3.4 e -38 to 3.4 e +38
Double	64	1.7e -308 to 1.7 e +308
Long double	80	3.4 e -4932 to 1.1 e +4932

Program

```
/* C Program to Find the Size of int, float, double, and
char using sizeof operator */
#include <stdio.h>
int main()
{
    int a;
    char b;
    float c;
    double d;
    // Determine and Print the size of a
    printf("\nSize of int is: %d bytes", sizeof(a));
    // Determine and Print the size of b
    printf("\nSize of char is: %d bytes", sizeof(b));
    // Determine and Print the size of c
    printf("\nSize of float is: %d bytes", sizeof(c));
    // Determine and Print the size of d
    printf("\nSize of double is: %d bytes", sizeof(d));
    return 0;
}
```

Output

```
Size of int is: 4
Size of char is: 1
Size of float is: 4
Size of double is: 8
```

TOPIC 9

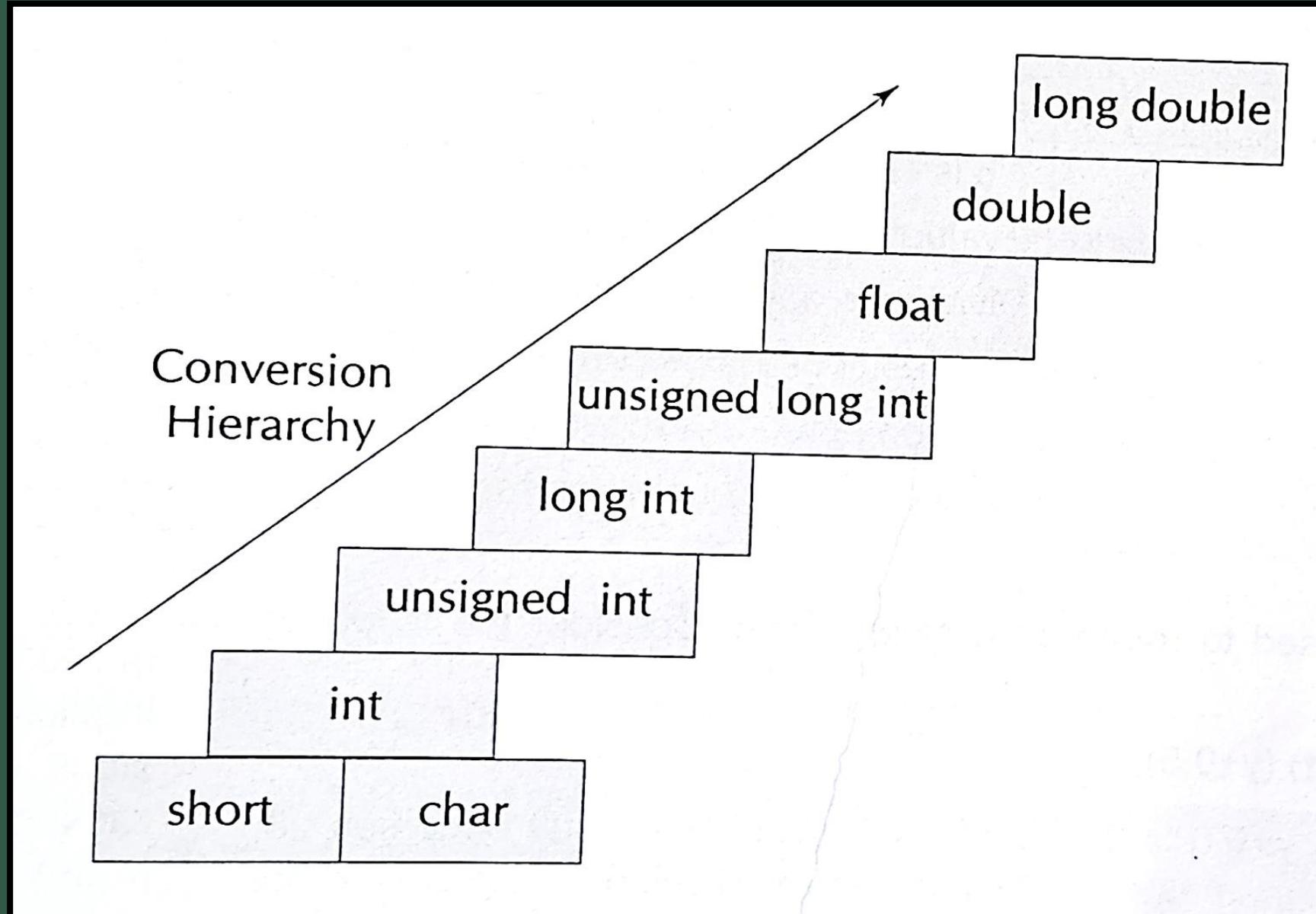
TYPE CASTING

C permits constants and variables of different types in an expression.
C facilitates type casting which allows conversion of data types.
Type conversion can be either **implicit** or **explicit**.

Rules of Implicit Type Conversion

1. if one of the operands is **long double**, the other will be converted to **long double** and the result will be **long double**;
2. else, if one of the operands is **double**, the other will be converted to **double** and the result will be **double**;
3. else, if one of the operands is **float**, the other will be converted to **float** and the result will be **float**;
4. else, if one of the operands is **unsigned long int**, the other will be converted to **unsigned long int** and the result will be **unsigned long int**;
5. else, if one of the operands is **long int** and the other is **unsigned int**, then
 - (a) if **unsigned int** can be converted to **long int**, the **unsigned int** operand will be converted as such and the result will be **long int**;
 - (b) else, both operands will be converted to **unsigned long int** and the result will be **unsigned long int**;
6. else, if one of the operands is **long int**, the other will be converted to **long int** and the result will be **long int**;
7. else, if one of the operands is **unsigned int**, the other will be converted to **unsigned int** and the result will be **unsigned int**.

Conversion Hierarchy



Explicit Type Conversion

The general form of explicit type conversion is
(type-name) expression

Examples:

Example	Action
<code>x = (int) 7.5</code>	7.5 is converted to integer by truncation.
<code>a = (int) 21.3/(int)4.5</code>	Evaluated as 21/4 and the result would be 5.
<code>b = (double)sum/n</code>	Division is done in floating point mode.
<code>y = (int) (a+b)</code>	The result of a+b is converted to integer.
<code>z = (int)a+b</code>	a is converted to integer and then added to b.
<code>p = cos((double)x)</code>	Converts x to double before using it.

Program for Implicit Type Conversion

```
//Program to illustrate type conversion
#include <stdio.h>

int main()
{
    int number = 34.78;
    printf("%d", number);
    return 0;
}
```

Output

34

Program for Explicit Type Conversion

```
//Program to illustrate type conversion
#include<stdio.h>

int main()
{
    // create an integer variable
    int number = 97;
    printf("Integer Value: %d\n", number);
    // (char) converts number to character
    char alphabet = (char) number;
    printf("Character Value: %c", alphabet);
    return 0;
}
```

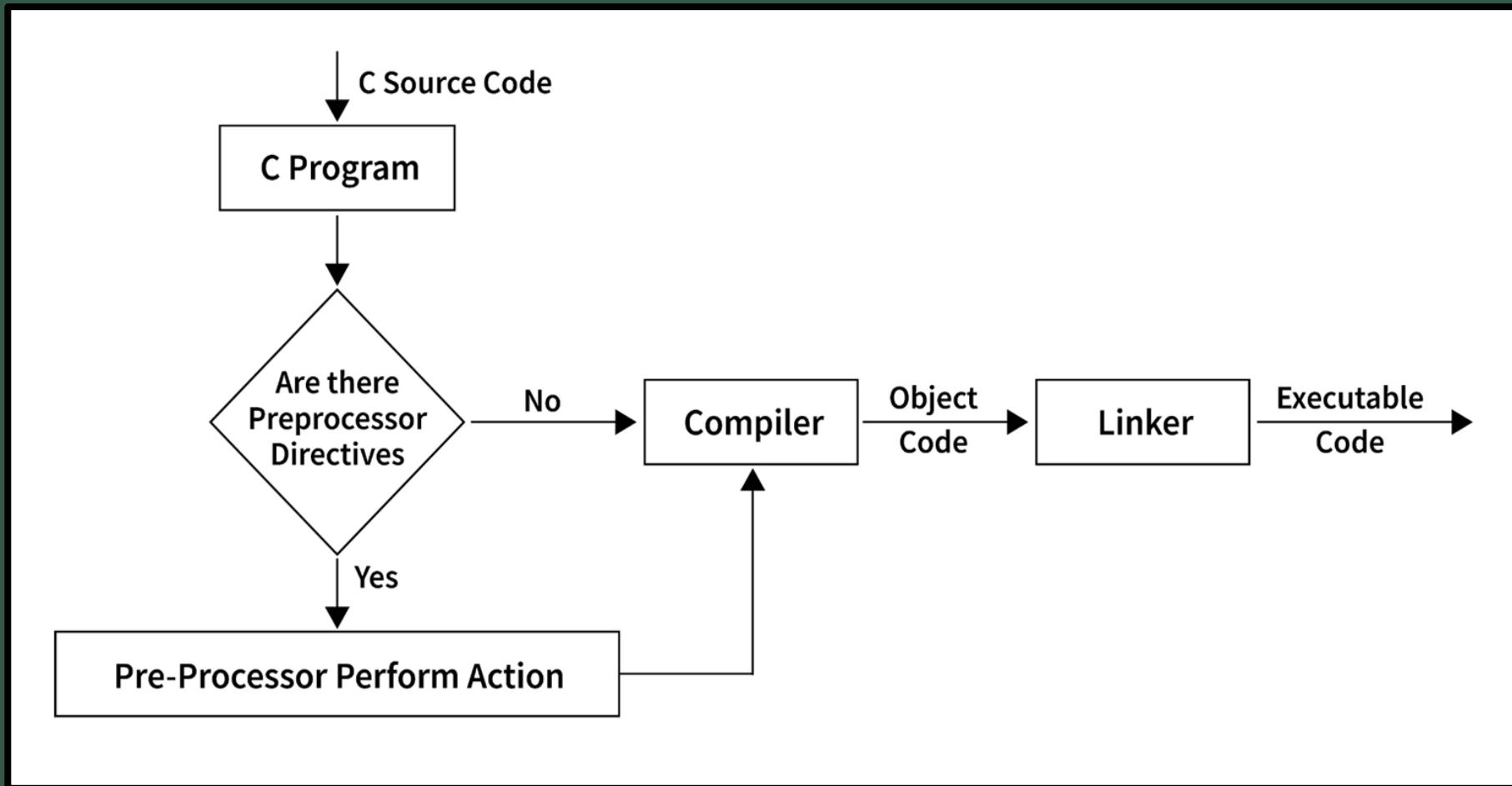
Output

```
Integer Value: 97
Character Value: a
```

TOPIC 10

PRE-PROCESSORS IN C

A pre-processor is a program that processes our source program before it is passed to the compiler.



Preprocessor Directives in C

The following table lists down all the important preprocessor directives –

Directive	Description
# define	Substitutes a preprocessor macro.
#include	Inserts a particular header from another file.
#undef	Undefines a preprocessor macro.
#ifdef	Returns true if this macro is defined.
#ifndef	Returns true if this macro is not defined.
#if	Tests if a compile time condition is true.
#else	The alternative for #if.
#elif	#else and #if in one statement.
#endif	Ends preprocessor conditional.
#error	Prints error message on stderr.
#pragma	Issues special commands to the compiler, using a standardized method.