



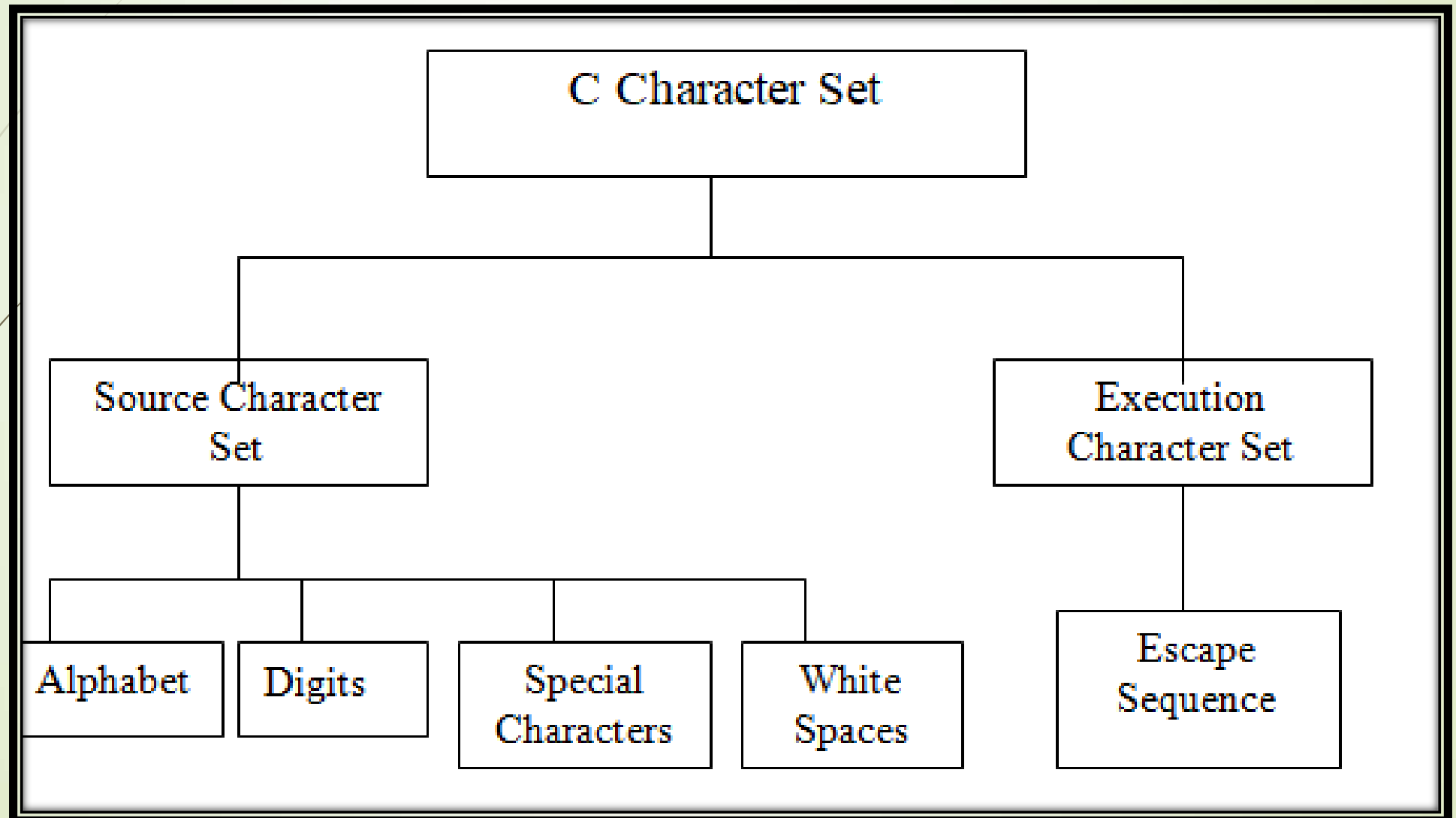
Tokens, Variables & Datatypes



TOPICS

- **CHARACTER SET**
- **CONSTANTS**
- **VARIABLES**
- **DATATYPES(PRIMITIVE)**
- **SAMPLE PROGRAMS & MCQ**

CHARACTER SET



SOURCE CHARACTER SET

- Used to construct the statements in the source program.
- A character denotes any alphabet, digit or special symbol used to represent information. The following Figure shows the valid alphabets, numbers and special symbols allowed in C.
- The alphabets, numbers and special symbols when properly combined form **tokens**.

Alphabets	A, B,, Y, Z a, b,, y, z
Digits	0, 1, 2, 3, 4, 5, 6, 7, 8, 9
Special symbols	~ ' ! @ # % ^ & * () _ - + = \ { } [] : ; " ' < > , . ? /

EXECUTION CHARACTER SET

- It is also called as “**Escape sequences**”.
- This set of characters are also called as non graphic characters because, **these characters are invisible and cannot be printed directly.**
- These characters will have effect only when the **program is executed.**

<u>CODE</u>	<u>MEANING</u>
\b	Backspace
\f	Form feed
\n	New line
\r	Carriage return
\t	Horizontal tab
\"	Double quote
\'	Single quote
\0	Null
\\	Backslash
\v	Vertical Tab
\a	Alert

EXECUTION CHARACTER SET-Example Program

```
#include <stdio.h>

int main()
{
    printf("\nHello World");
    return 0;
}
```

OUTPUT:

"Hello World"

```
#include <stdio.h>

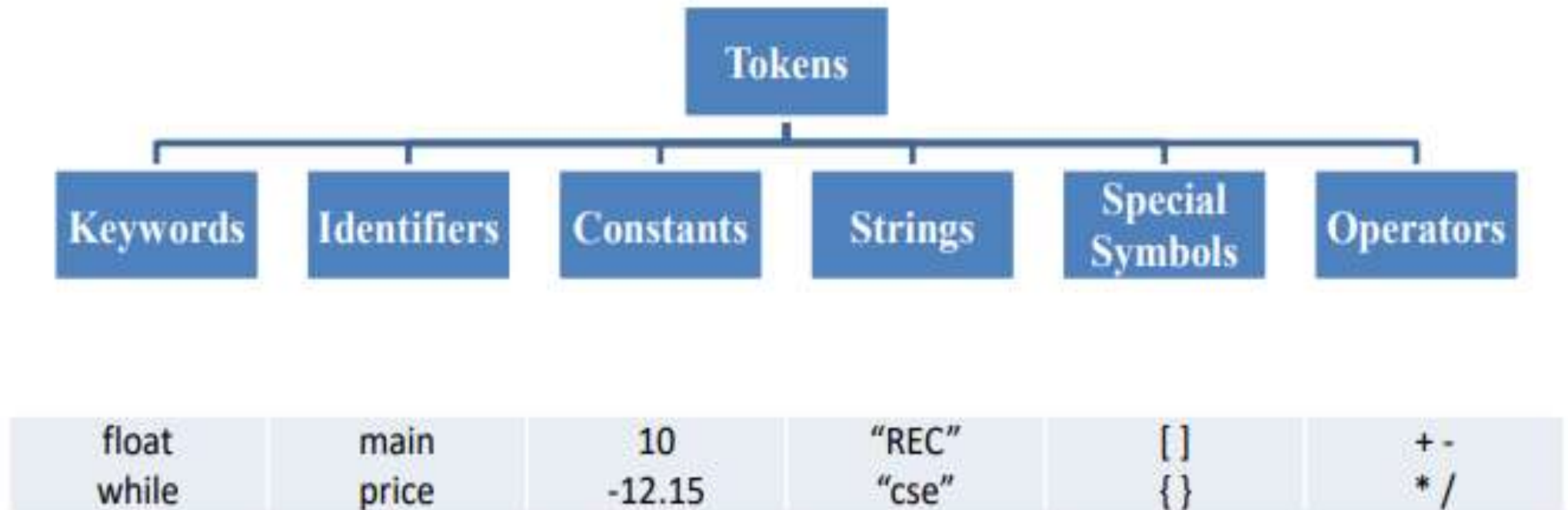
int main()
{
    printf("\nWel\come to \nREC");
    return 0;
}
```

OUTPUT:

'Wel come to
REC'

C TOKENS

- In C program, the smallest individual units are known as C tokens.
- **Types of Tokens:**



C KEYWORDS

- Keywords are the words whose meaning has already been explained to the C compiler.
- The keywords cannot be used as variable names The keywords are also called 'Reserved words'.
- There are only 32 keywords available in C. The following Figure gives a list of these keywords for your ready reference.

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

Identifiers in C

- Used for naming variables, functions, arrays, structures, etc.
- Identifiers in C are the user-defined words.
- **Rules** for constructing identifiers in C
 - The first character of an identifier should be either an **alphabet** or an **underscore**, and then it can be followed by any of the character, digit, or underscore.
 - It should **not** begin with any **numerical** digit.
 - In identifiers, both uppercase and lowercase letters are distinct. Therefore, we can say that identifiers are case sensitive.
 - **Commas or blank spaces cannot** be specified within an identifier.
 - **Keywords cannot** be represented as an identifier.
 - The length of the identifiers should not be more than **31** characters.

Strings in C

- Sequence of character.
- Strings in C are enclosed within **double quotes**, while characters are enclosed within **single characters**.

Example:

“Hello”

‘A’

Operators in C

➤ Symbol used to perform the operation

➤ Example:

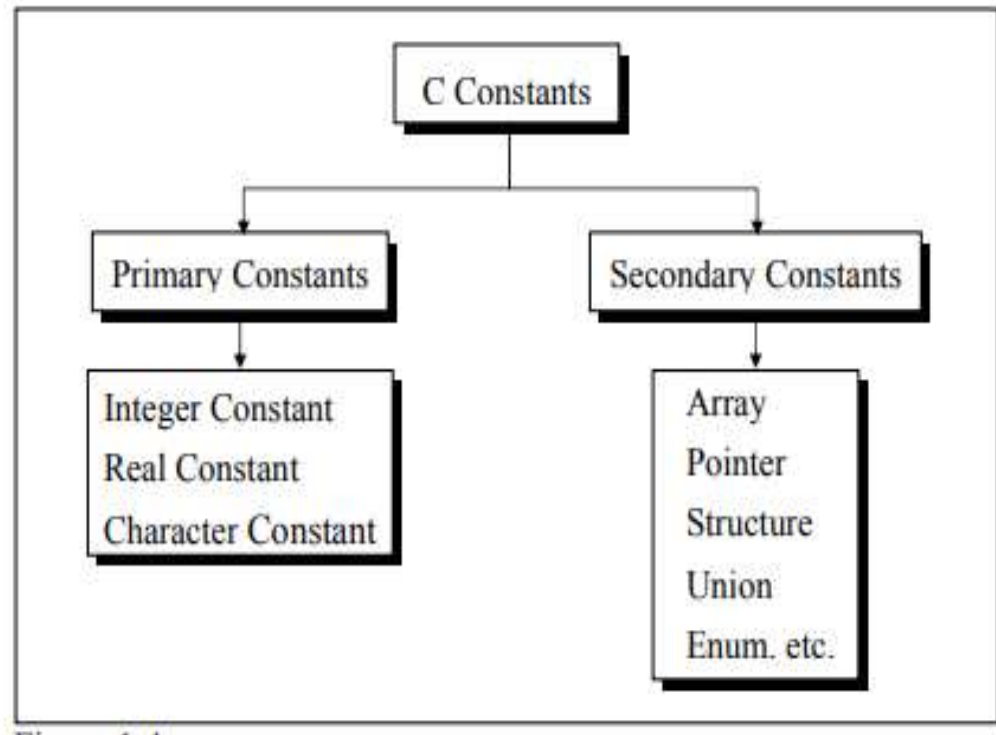
`+, -, /, *, %, ++, --, <<, >>, &&, ||, !`

Special Symbols

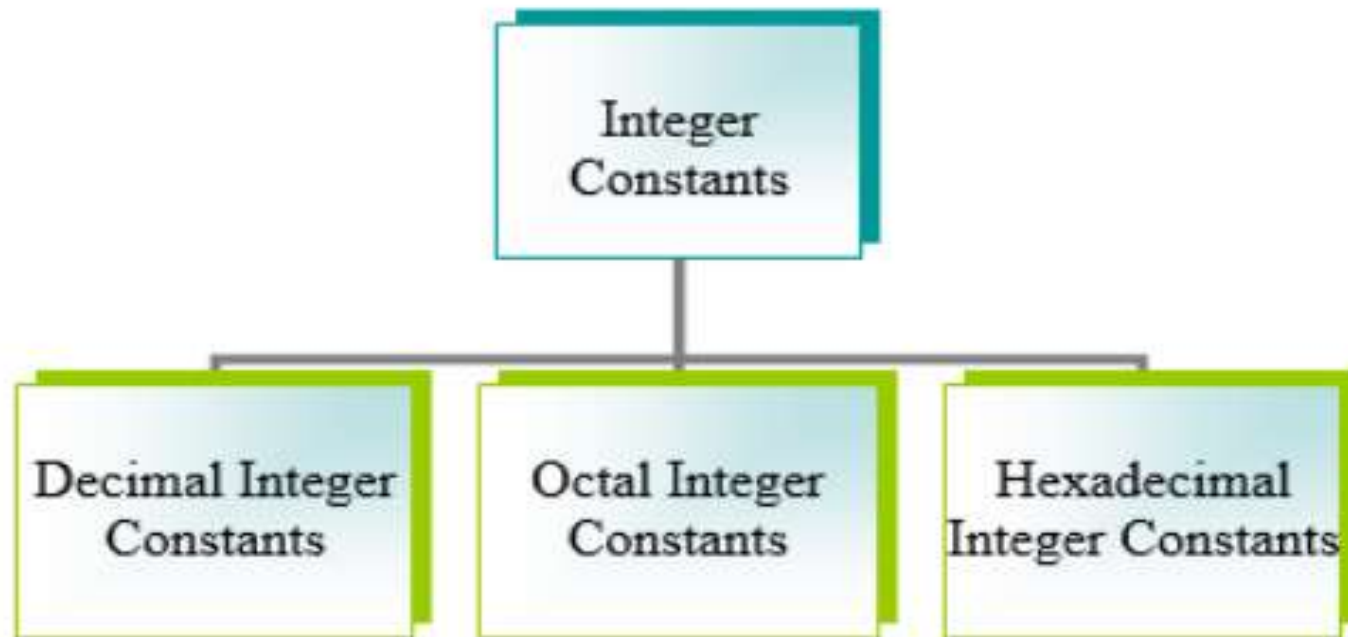
- **Square brackets []**: The opening and closing brackets represent the single and multidimensional subscripts.
- **Simple brackets ()**: It is used in function declaration and function calling. For example, printf() is a pre-defined function.
- **Curly braces { }**: It is used in the opening and closing of the code. It is used in the opening and closing of the loops.
- **Comma (,)**: It is used for separating for more than one statement and for example, separating function parameters in a function call, separating the variable when printing the value of more than one variable using a single printf statement.
- **Hash/pre-processor (#)**: It is used for pre-processor directive. It basically denotes that we are using the header file.
- **Asterisk (*)**: This symbol is used to represent pointers and also used as an operator for multiplication.
- **Tilde (~)**: It is used as a destructor to free memory.
- **Period (.)**: It is used to access a member of a structure or a union.

CONSTANTS

- A constant is an entity that doesn't change its value.
- C constants can be divided into two major categories:
 - Primary Constants
 - Secondary Constants



INTEGER CONSTANTS



INTEGER CONSTANTS

- C accepts integer constants in three numbering systems - decimal, octal and hexadecimal.
- **Decimal Integer Constant**-Decimal integer constants can consist any combinations of digits from 0 to 9, preceded by an optional + or – sign.(Eg: 98, +47, -87)
- **Octal Integer Constant**-Octal integer constants can consist of any combinations of digits from 0 to 7. C specifies that the **octal integer** must be **preceded** by a **0**.(Eg:010, 07,0240)
- **Hexadecimal Integer Constant**-Hexadecimal integer constants can consist of any combination of digits from 0 to 9 and letters from A to F (either uppercase or lowercase represents 10 to15)
- A hexadecimal integer constant must begin with either **0x** or **0X**(Eg:0x2,0X2A,0Xab)

INTEGER CONSTANTS

➤ Rules for Constructing Integer Constants

- An integer constant must have at least one digit.
- It must not have a decimal point.
- It can be either positive or negative.
- If no sign precedes an integer constant it is assumed to be positive.
- No commas or blanks are allowed within an integer constant.
- The allowable range for integer constants depends upon the compiler
- Ex.: 426
- +782
- -8000
- -7605



Examples

- Valid Examples:

0	+47	179
-240	22099	

- Invalid Examples:

35 750	10,000	\$5000	12.55
--------	--------	--------	-------



Examples

- Valid Examples:

0

047

0240

- Invalid Examples:

01 70

010,000

\$05000

012.55

0786



Examples

- Valid Examples:

0x

0X2

0x7A

0xbcd

- Invalid Examples:

0x35 75A

0x10,000

\$0x5000

0x12.55

0x7AG

INTEGER CONSTANTS-Example

➤ Example: 1

➤ `printf (“%d %d %d %d”, 45, 045, 0x45, 0X45);`

➤ What it will print?

➤ The decimal equivalent of all the constants will be printed.

➤ **45 33 65 65**

➤ Example: 2

➤ `printf (“ %o %x”, 45, 45);`

➤ What it will print?

➤ The octal and hexadecimal equivalent of 45 will be printed.

➤ **55 2D**

INTEGER CONSTANTS-MCQ

➤ **An integer constant in C must have:**

- (1) At least one digit
- (2) At-least one decimal point
- (3) A comma along with digits
- (4) Digits separated by commas

➤ **ANSWER:1**

REAL CONSTANTS (FLOATING POINT CONSTANTS)

- Real constants are often called Floating Point constants. The real constants could be written in two forms—Fractional form and Exponential form.

- **Real constants expressed in fractional form**

- A real constant must have at least one digit.
 - It must have a decimal point.
 - It could be either positive or negative.
 - Default sign is positive.

No commas or blanks are allowed within a real constant.

- Ex.: +325.34
 - 426.0
 - -32.76
 - -48.5792



Examples

- Valid Examples:

0.0002

-0.96

179.47

+31.79

+.2

-.47

179.

.99

- Invalid Examples:

12,000.50

31

12.30.45

\$1000.75

15 750.25

REAL CONSTANTS (FLOATING POINT CONSTANTS)

➡ Example: 1

➡ `printf("%f %f",+456.67,-67.89);`

➡ What it will print?

➡ 456.670000 -67.890000

REAL CONSTANTS (FLOATING POINT CONSTANTS)

Real constants expressed in exponential form

- The mantissa part and the exponential part should be separated by a letter e.
- The mantissa part may have a positive or negative sign.
- Default sign of mantissa part is positive.
- The exponent must have at least one digit, which must be a positive or negative integer. Default sign is positive.
- Range of real constants expressed in exponential form is -3.4×10^{38} to 3.4×10^{38} .
- Ex.: $+3.2 \times 10^{-5}$
- 4.1×10^8
- -0.2×10^3
- -3.2×10^{-5}

Examples

- Valid Examples:

0.31e2

14e-3

3.1e+5

3.14E4

-1.2E-2

- Invalid Examples:

12,000e2

3.1e 2

3.1E+2.4

REAL CONSTANTS (FLOATING POINT CONSTANTS)

Example: 1

```
printf ("%f %f",4.1e8,-3.2e-5);
```

What it will print?

410000000.000000 -.000032

CHARACTER CONSTANTS

Rules for Constructing Character Constants

- A character constant is a single alphabet, a single digit or a single special symbol enclosed within single inverted commas.
- The maximum length of a character constant can be 1 character.
- Ex.: 'A'
- 'I'
- '5'
- '═'

CHARACTER CONSTANTS-Example

Example: 1

```
printf ("%c %c",'a','5');
```

```
printf ("%d %d",'a','5');
```

What it will print?

a 5 (%c will print the actual character as such)

97 53 (%d will print the ASCII value of the character)

CHARACTER CONSTANTS-MCQ

Example 2:

Which of the following is NOT a character constant

- (1) 'Thank You'
- (2) 'Enter values of P, N, R'
- (3) '23.56E-03'
- (4) All the above

ANSWER: 4

VARIABLES

- Variable is a name used for storing a data value.
- Its value may be changed during the program execution.
- Variable names are names given to locations in memory.
- These locations can contain integer, real or character constants.
- **Rules** for defining variables is same as Identifier.

VARIABLES

DECLARING A VARIABLE:

- The declaration of variables must be done before they are used in the program.
- It tells the compiler what the variable name is and it specifies what type of data the variable will hold.
- A variable declaration consists of a data type name followed by a list of one or more variables of that type separated by commas.

Syntax:

`datatype var1, var2, . . . varN;` // here var1, var2..varN are names of the variable

Eg: `int sum;`

- The variables will contain some garbage value when they are declared.

VARIABLES

ASSIGNING VALUES TO VARIABLE:

Values can be assigned to variables using the assignment operator =.

Syntax:

```
variablename = value;
```

```
int x;
```

```
x=100;
```

VARIABLES

INITIALIZING VALUE TO VARIABLE:

- The process of giving an initial value to variables is called initialization.
- C permits the initialization of more than one variable in one statement using multiple assignment operators.

Example:

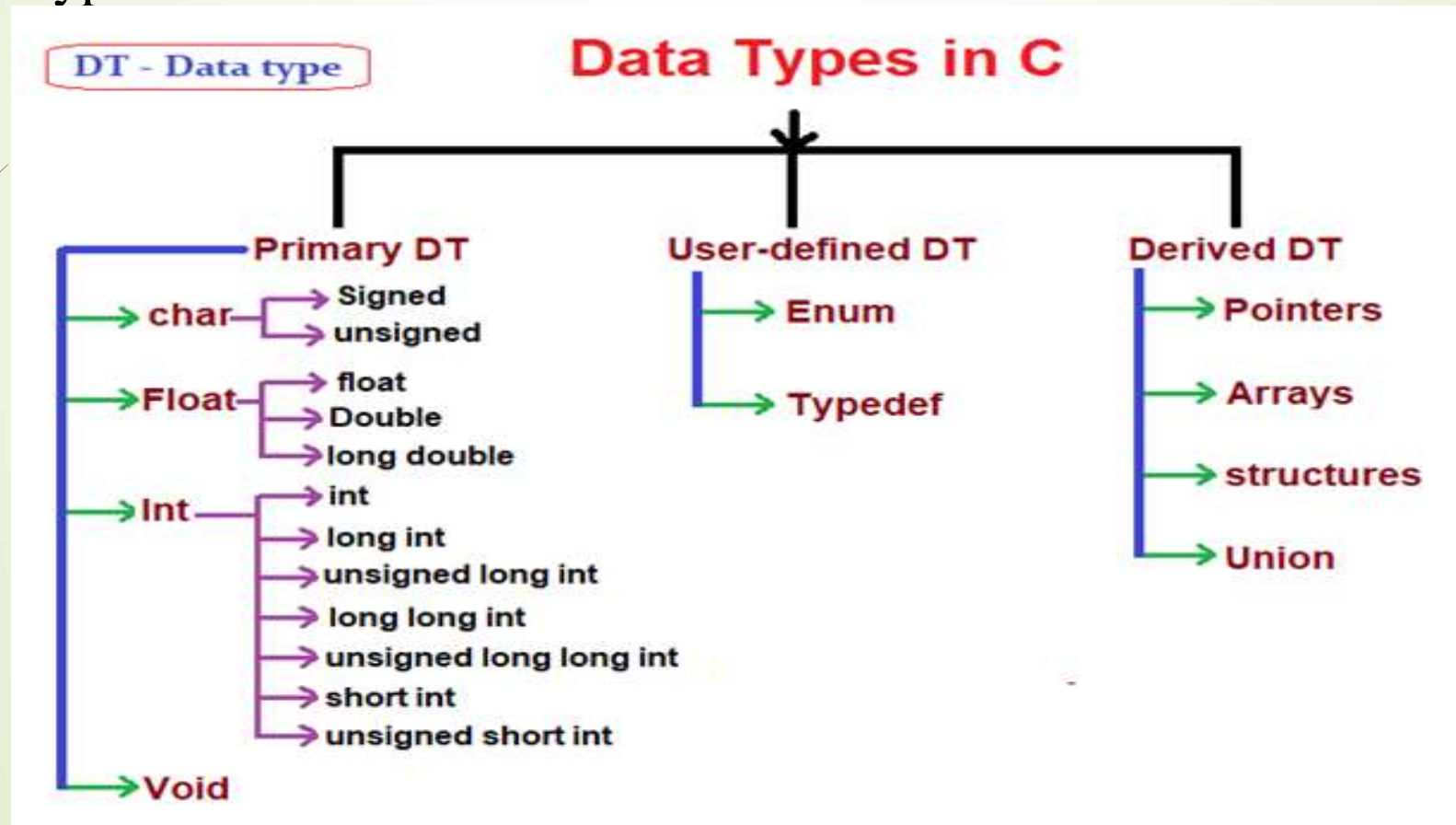
```
int a = 0;
```

```
int x,y,z;
```

```
x = y = z = 1;
```

DATA TYPES IN C

- A data type specifies the type of data that a variable can store such as integer, floating, character, etc.. ANSI C provides three types of data types:



PRIMARY (BUILT-IN) DATA TYPES:

char:

- The most basic data type in C. It stores a single character and requires a single byte of memory in almost all compilers.

➤ TURBO C COMPILER

Type	Length	Range
unsigned char	8 bits	0 to 255
char (or) signed char	8 bits	-128 to 127

➤ GCC COMPILER

Type	Length	Range
unsigned char	8 bits	0 to 255
char (or) signed char	8 bits	-128 to 127

PRIMARY (BUILT-IN) DATA TYPES:

int:

- int datatype is used to store whole numbers.
- TURBO C COMPILER

unsigned int	16 bits	0 to 65,535
short (or) short int (or) signed short int	16 bits	-32,768 to 32,767
int (or) signed int	16 bits	-32,768 to 32,767
unsigned long (or) unsigned long int	32 bits	0 to 4,294,967,295
long (or) long int (or) signed long int	32 bits	-2,147,483,648 to 2,147,483,647

PRIMARY (BUILT-IN) DATA TYPES:

int:

- int datatype is used to store whole numbers.
- GCC COMPILER

unsigned int	32 bits	0 to 4,294,967,295
short (or) short int (or) signed short int	16 bits	-32,768 to 32,767
int (or) signed int	32 bits	-2,147,483,648 to 2,147,483,647
unsigned long (or) unsigned long int	32 bits	0 to 4,294,967,295
long (or) long int (or) signed long int	32 bits	-2,147,483,648 to 2,147,483,647

PRIMARY (BUILT-IN) DATA TYPES:

float and double:

Floating types are used to store real numbers.

float: A single-precision floating point value.

Double: A double-precision floating point value.

➔ TURBO C COMPILER

float	32 bits	$3.4 * (10^{*-38})$ to $3.4 * (10^{**+38})$ 3.4E-38 to 3.4E+38
double	64 bits	$1.7 * (10^{*-308})$ to $1.7 * (10^{**+308})$ 1.7E-308 to 1.7E+308
long double	80 bits	$3.4 * (10^{*-4932})$ to $1.1 * (10^{**+4932})$ 3.4E-4932 to 1.1E+4932

PRIMARY (BUILT-IN) DATA TYPES:

float and double:

Floating types are used to store real numbers.

float: A single-precision floating point value.

Double: A double-precision floating point value.

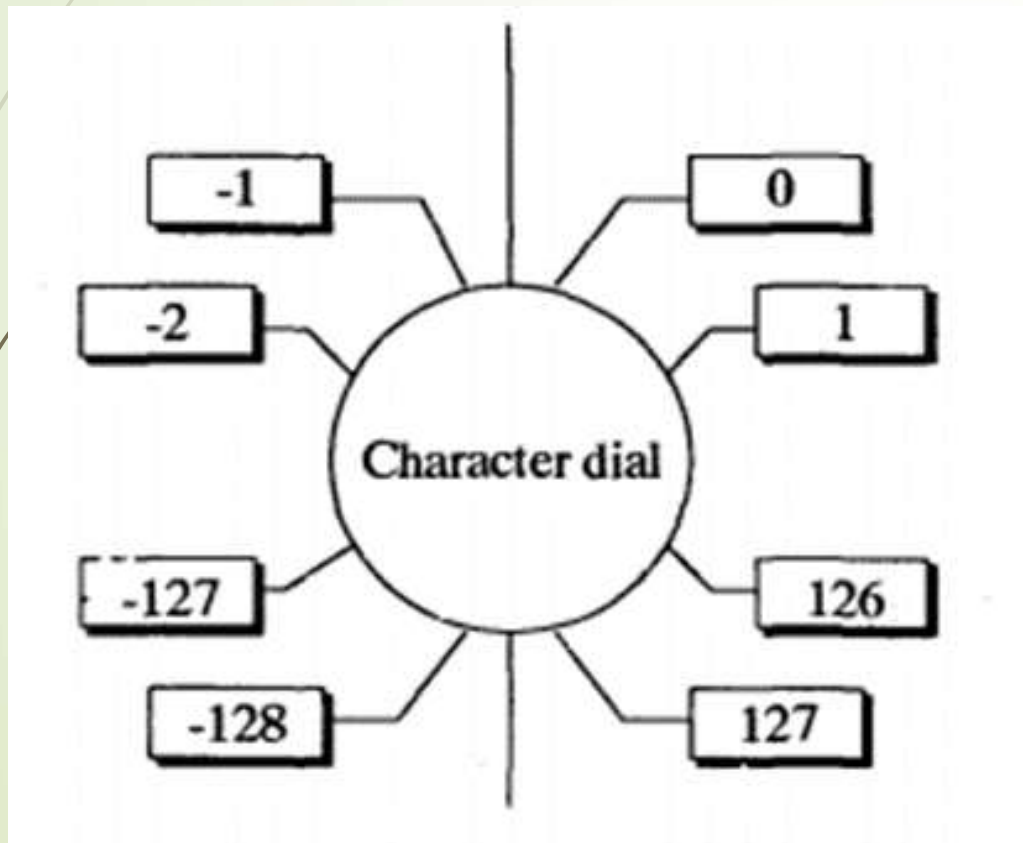
➡ GCC COMPILER

float	32 bits	$3.4 * (10^{*-38})$ to $3.4 * (10^{**+38})$ 3.4E-38 to 3.4E+38
double	64 bits	$1.7 * (10^{*-308})$ to $1.7 * (10^{**+308})$ 1.7E-308 to 1.7E+308
long double	96 bits	$3.36210 * (10^{*-4932})$ to $1.18973 * (10^{**+4932})$ 3.36210E-4932 to 1.18973E+4932

Format Specifier	Type
%c	Character
%d	Signed integer
%e or %E	Scientific notation of floats
%f	Float values
%g or %G	Similar as %e or %E
%hi	Signed integer (short)
%hu	Unsigned Integer (short)
%i	Unsigned integer
%l or %ld or %li	Long
%lf	Double
%Lf	Long double
%lu	Unsigned int or unsigned long
%lli or %lld	Long long
%llu	Unsigned long long
%o	Octal representation
%p	Pointer
%s	String
%u	Unsigned int
%x or %X	Hexadecimal representation
%n	Prints nothing
%%	Prints % character

CHARACTER DIAL

- The range of the signed char datatype is -128 to 127. The range values is logically viewed in the form of a dial.



```
#include <stdio.h>
int main()
{
    char a = 128;
    printf("%d",a );
    return 0;
}
```

OUTPUT:-128

POINT OUT THE ERRORS, IF ANY, IN THE FOLLOWING C STATEMENTS:

➤ 1. `int = 314.562 * 150 ;`

➤ Error: variable name is missing/identifier is expected

➤ 2. `name = 'Ajay' ;`

➤ Error: datatype is missing/ name variable is undeclared

➤ 3. `varchar = '3';`

➤ Error: datatype is missing/ name variable is undeclared

POINT OUT THE ERRORS, IF ANY, IN THE FOLLOWING C STATEMENTS:

➤ 1. `int = 314.562 * 150 ;`

➤ Error: variable name is missing/identifier is expected

➤ 2. `name = 'Ajay' ;`

➤ Error: datatype is missing/ name variable is undeclared

➤ 3. `varchar = '3';`

➤ Error: datatype is missing/ name variable is undeclared

MULTIPLE CHOICE QUESTIONS

What is the output of this C code?

```
#include <stdio.h>

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

- A. Hello World! x;
- B. Hello World! followed by a junk value
- C. Compile time error
- D. Hello World!

➡ ANSWER:C

MULTIPLE CHOICE QUESTIONS

```
#include <stdio.h>
int main()
{
int main = 10;
printf("%d", main);
return 0;
}
```

- A. It will cause a compilation error
- B. It will cause a run-time error
- C. It will run without any error and prints 10
- D. It will experience infinite looping

➡ **ANSWER: C** (It will run without any error and prints 10 as 'main' is not a keyword and can be used as a variable)

MULTIPLE CHOICE QUESTIONS

What will be the output of the following C code? [Assume it's a 32-bit system]

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

```
int main()
```

```
{
```

```
int x = 70;
```

```
printf("%c", x);
```

```
return 0
```

```
}
```

A. 70

B. F

C. 70.0

D. Compilation error

ANSWER:B

MULTIPLE CHOICE QUESTIONS

The statement `char ch = 'Z'` would store what in `ch`

- i. The character Z
- ii. ASCII value of Z
- iii. Z along with the single inverted commas
- iv. Both (1) and (2)

ANSWER: B



MULTIPLE CHOICE QUESTIONS

A C variable cannot start with

- A. An alphabet
- B. A number
- C. A special symbol other than underscore
- D. Both (2) & (3) above

ANSWER:D

WHAT WILL BE THE OUTPUT?

```
#include <stdio.h>

int main()
{
    char a;
    float b;
    int c;
    printf("%d %d %d\n", sizeof(char), sizeof(int), sizeof(float));
    printf("%d %d %d\n", sizeof(a), sizeof(b), sizeof(c));
    printf("%d %d %d ", sizeof('7'), sizeof(7), sizeof(7.0));
    return 0;
}
```

OUTPUT:

1 4 4

1 4 4

4 4 8

WHAT WILL BE THE OUTPUT?

```
#include <stdio.h>

int main()
{
    char ch = 291;
    printf("%d %d %c", 2147483648, ch, ch);
    return 0;
}
```

OUTPUT:

-2147483648 35 #

WHAT WILL BE THE OUTPUT?

```
#include <stdio.h>
int main()
{
    unsigned char c = 257;
    char a='1270';
    char b='123a';
    printf("%c %d\n",a,a);
    printf("%c %d\n",b,b);
    printf("%d %d\n", c, c);
    return 0;
}
```

OUTPUT

0 48

a 97

1 1

PROBLEM 1:

If the mark obtained by a student is given for 3 subjects find out the aggregate mark and percentage obtained by the student. Assume that the maximum marks can be obtained by a student in each subject is 100.

LOGIC:

AGGREGATION: ADDING THE MARKS

PERCENTAGE: SUM DIVISION BY 3



```
/* Program to find the sum and percentage of 3 subject marks*/
```

```
/* Assume maximum mark is 100*/
```

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

```
void main()
```

```
{
```

```
    int mark1,mark2,mark3;
```

```
    int sum=0;
```

```
    float percent;
```

```
    printf("Enter the 3 subjects marks");
```

```
    scanf("%d %d %d",&mark1,&mark2,&mark3);
```

```
    sum=mark1+mark2+mark3;
```

```
    percent=sum/3;
```

```
    printf("The sum is:%d",sum);
```

```
    printf("\nThe percentage is:%f",percent);
```

```
}
```

PROBLEM 2:


A cashier has currency notes of denominations 10, 50 and 100. If the amount to be withdrawn is input through the keyboard in tens, find the total number of currency notes of each denomination the cashier will have to give to the withdrawer.

LOGIC:

DIVIDE GIVEN VALUE BY 100

FIND REMAINDER BY DIVIDING GIVEN VALUE BY 100

DO THE SAME ABOVE TWO STEPS FOR THE REMAINDER WITH RESPECT TO 50 AS WELL AS 10



/* Program to find the total number of currency notes given to the withdrawer by the cashier. Give input value in multiple of 10s */

#include <stdio.h>

int main()

{

int denom1=10,denom2=50,denom3=100;

int w_input;

int temp=0;

printf("The input given by withdrawer is:");

scanf("%d",&w_input);

printf("currency note count for Rs 100 is:%d",w_input/denom3);

temp=w_input%denom3;

printf("\ncurrency note count for Rs 50 is:%d",temp/denom2);

temp=temp%denom2;

printf("\ncurrency note count for Rs 10 is:%d",temp/denom1);

return 0;

}

PROBLEM 3:

In a town, the percentage of men is 52. The percentage of total literacy is 48. If total percentage of literate men is 35 of the total population, write a program to find the total number of illiterate men and women if the population of the town is 80,000.

LOGIC:

FIND MEN, WOMEN COUNT IN 80,000 COUNT(52% MEN THEN 48% FOR WOMEN)

FIND LITERACY MEN THEN LITERACY WOMEN(WITH 35% MEN LITERATE AND OVERALL 48% LITERATE)

FIND ILLITERATE MEN AND WOMEN(SUBTRACT)

```
/* Program to find the total number of illiterate men and women fro the given population 80,000*/
#include <stdio.h>

void main()
{
    int t_pop=80000;
    int t_men=t_pop*52/100;    //total men percentage is 52
    int t_women=t_pop-t_men;
    int t_lit=t_pop *48/100;    //total literacy percentage is 48
    int l_men=t_pop*35/100;    //total literacy men percentage is 35
    int l_women=t_lit-l_men;
    int il_men=t_men-l_men;
    int il_women=t_women-l_women;
    printf("Total Illiteracy men is:%d",il_men);
    printf("\nTotal Illiteracy women is:%d",il_women);
}
```

PROBLEM 4:

Ram is an employee in a departmental store where his manager wants to test his English skills so he gives him a task to find the previous and next alphabet of given alphabet. Help him out by giving him a program to find the same.

IMPORTANT POINT:

In C programming, a character variable holds ASCII value (an integer number between 0 and 127) rather than that character itself. That value is known as it's ASCII value.

Eg:ASCII value of A is 65

If A is assigned to a character variable, the value 65 is only saved.

Hence we can perform all arithmetic operations over char.

If %c is used character will be printed if %d used then ASCII value will be printed



```
/* Program to find the previous and next alphabet of a given alphabet*/
```

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

```
void main()
```

```
{
```

```
    char alpha;
```

```
    char prev,next;
```

```
    printf("Enter the alphabet:");
```

```
    scanf("%c",&alpha);
```

```
    prev=alpha-1;
```

```
    next=alpha+1;
```

```
    printf("The previous alphabet is:%c",prev);
```

```
    printf("\nThe next alphabet is:%c",next);
```

```
}
```



MCQS

What will be the output of the following program?

```
#include<stdio.h>

int main()
{
    char ch='A';
    printf("%d", ch+1);
    return 0;
}
```

- A
- 66
- Compilation Error
- Run time Error
- **ANSWER: B**

MCQS

What will be the output of the following program?(When 21 and 31 are entered)

```
#include<stdio.h>
int main()
{
    int a;
    float b;
    printf("Enter the numbers");
    scanf("%d %f", &a, &b);
    printf("%d %f", a, b);
    return 0;
}
```

- Error
- 21 31.000000
- 21 31.0
- 21 31
- **ANSWER: B**

MCQS

```
int main()
{
    char a,b;
    printf("Enter the numbers\n");
    scanf("%c %c", &a, &b);
    printf("%d %d", a, b);
    return 0;
}
```

(if numbers 8 9 is given)

➤ Error

➤ 56 55

➤ 21 31

➤ 56 57

➤ **ANSWER: D**

MCQS

```
int main()
{
    char a,b;
    printf("Enter the numbers\n");
    scanf("%c %c", &a, &b);
    printf("%d", b-a);
    return 0;
}
```

(if numbers 8 9 is given)

➤ Error

➤ -1

➤ 1

➤ 56

➤ **ANSWER: C**

Write the program

PROGRAM 1:

Alfred buys an old scooter for Rs. X and spends Rs. Y on its repairs. If he sells the scooter for Rs. Z ($Z > X + Y$). Write a program to help Alfred to find his gain percent. Get all the above mentioned values through keyboard and find the gain percent.

Write the program

PROGRAM 2:

Ram was given the following task to understand how to work with constants and variables in C language. He should write a program to accomplish all the tasks.

Task 1: He was given a series of 3 numbers and was asked to find its octal and hexadecimal equivalent.

Task 2: He was asked to find the sum of the ASCII values of the Vowels both uppercase and lowercase