BASIC CONSTRUCTS OF C

char DATATYPE

- The *char* datatype is used when we want to store alphabets, numeric characters and special characters or symbols.
- Each character is assigned a unique integer value also known as the ASCII value.
- *char* variables are also declared in the same manner as any other variable.
- Syntax:

char variable_name = value (can be character/integer);

DATATYPE CONVERSIONS

- Binary conversion
  - In a binary arithmetic operation, the outcome finally depends on the higher data type.
- Type-casting/Coercion
  - We forcefully convert one data type into another.
  - Remember, the conversion is temporary and not permanent
- Assignment conversion
  - The value finally stored depends upon the datatype of the variable on the L.H.S of the assignment operator

QUICK EXERCISE

- Find whether correct or incorrect:
  - char ‘x’; X
  - char x = ‘x’; √
  - char x = 66; √
  - int x = ‘C’+1; √
- Program to print the ASCII value of a given character.

  Hint:

  printf(“The ASCII Value of character % c is %d”, ch, (int) ch);
- What will be the final value of variable z?

```
int x = 10, w = 8,z;
float y = 3.5;
z = x / (float)w * (int) y + 2.98;
```

  Output: 6

LITERAL CONSTANTS IN C

- **Integer literal**
  - In C programming we can specify an integer constant in three ways.
    - Decimal number (base 10).
    - Octal number (base 8).
    - Hexadecimal number (base 16).
  - Integer literal is followed by an optional size or sign qualifier.
    l or L is a size qualifier. It specifies size of an integer type as long.
    u or U is a sign qualifier. It specifies an integer type as unsigned
  - Decimal constant is defined using digits between 0-9. For example: 123.
  - Octal constants is prefixed with 0 followed by digits between 0-7. For example: 0123 (in octal) equivalent to 83 in decimal. Any number begins with 0 must only contain digits between 0-7 otherwise compiler will report an error on compilation.
  - Hexadecimal constant is prefixed with 0x or 0X following hexadecimal characters (i.e. digits from 0-9 and characters from a-f or A-F). For example: 0X1A or 0x1a or 0x1A (in hexadecimal) equivalent to 26 (in decimal).

LITERAL CONSTANTS IN C

In Decimal	In Octal	In Hexadecimal	Description
2016u	03740u	0x7E0u	Unsigned integer
2016U	03740U	0x7E0U	
2147483697l	020000000061l	0x80000031l	Long integer
2147483697L	020000000061L	0x80000031L	
2147483697ul	020000000061ul	0x80000031ul	Unsigned long integer
2147483697uL	020000000061uL	0x80000031uL	
2147483697Ul	020000000061Ul	0x80000031Ul	
2147483697UL	020000000061UL	0x80000031UL	

LITERAL CONSTANTS IN C

- **Float or real literal**
  - In C programming a float or real constant is specified either using decimal or using exponential notation.
  - **Decimal notation**
  - In decimal notation real constant is represented as whole number, followed by an optional decimal point and fractional part.
  - It may be preceded with + or - symbol representing negative or positive number.
  - Valid examples of float constants in decimal notation.
    +1, 1.2, -0.4, 0.1, .3 , -.3

LITERAL CONSTANTS IN C

- **Exponential notation**
- Exponential notations are helpful in representing any number with very big or small magnitude.
- Numbers such as, 7850000000000 in exponential notation expressed as 7.85e12,
- 0.000000000785 expressed as 7.85e-11.
- In exponential notation real constants are expressed in scientific format specifying mantissa and exponent.
- Valid syntax of writing real constant in scientific format.

  [+/-] <Mantissa> <e/E> [+/-] <Exponent>
- Valid examples of real constants in exponential notation 0.1e1, 0e-5, 5e25, +5E+10, -5.2e0

LITERAL CONSTANTS IN C

- **Character literal**
  - Character literal is a single character constant enclosed within single quotes.
  - In C programming character constant is a single byte.
  - There are many ways to represent a character constant in C programming.
    - Using character inside single quote. For example ‘a’ ,‘0’, ‘/’, ‘.’ etc.
    - Using escape sequence characters. For example,

      ‘\n’ ,‘\a’, ‘\b’ etc.
    - Using an integer ASCII representing of a character. For example, 65 in ASCII format is used to represent ‘A’.

LITERAL CONSTANTS IN C

- String literals
  - String literal is a sequence of characters enclosed within double quotes.
  - Size of a string constant is (total count of characters + 1) byte.
  - The additional 1 byte is used for \0 (NULL) character. A NULL value is added explicitly at the end of the string to specify termination of the string.
  - You can also concatenate string literals using + operator.
  - Examples of some valid string constants:
  - "I love programming!”
  - “I love Coding.\n" + "I love India”.
  - Important note: 'A' and "A" both looks similar but in real both are different. 'A' is a character constant occupying 1 byte space in memory. Whereas "A" is a string constant and occupies 2 byte space in memory.

IN-BUILT FUNCTIONS IN C

No.	Name	Description
1	stdio.h	Input/Output Functions
2	conio.h	console input/output
3	assert.h	Diagnostics Functions
4	ctype.h	Character Handling Functions
5	cocale.h	Localization Functions
6	math.h	Mathematics Functions
7	setjmp.h	Nonlocal Jump Functions
8	signal.h	Signal Handling Functions
9	stdarg.h	Variable Argument List Functions
10	stdlib.h	General Utility Functions
11	string.h	String Functions
12	time.h	Date and Time Functions
13	complex.h	A set of function for manipulating complex numbers
14	stdalign.h	For querying and specifying the alignment of objects
15	errno.h	For testing error codes
16	locale.h	Defines localization functions
17	stdatomic.h	For atomic operations on data shared between threads
18	stdnoreturn.h	For specifying non-returning functions
19	uchar.h	Types and functions for manipulating Unicode characters
20	fenv.h	A set of functions for controlling the floating-point environment
21	wchar.h	Defines wide string handling functions
22	tgmath.h	Type-generic mathematical functions
23	stdarg.h	Accessing a varying number of arguments passed to functions
24	stdbool.h	Defines a boolean data type

Function	Description
floor (x)	This function returns the nearest integer which is less than or equal to the argument passed to this function.
round (x)	This function returns the nearest integer value of the float/double/long double argument passed to this function. If decimal value is from ".1 to .5", it returns integer value less than the argument. If decimal value is from ".6 to .9", it returns the integer value greater than the argument.
ceil (x)	This function returns nearest integer value which is greater than or equal to the argument passed to this function.
sin (x)	This function is used to calculate sine value.
cos (x)	This function is used to calculate cosine.
cosh (x)	This function is used to calculate hyperbolic cosine.
exp (x)	This function is used to calculate the exponential "e" to the x^{th} power.
tan (x)	This function is used to calculate tangent.
tanh (x)	This function is used to calculate hyperbolic tangent.
sinh (x)	This function is used to calculate hyperbolic sine.
log (x)	This function is used to calculates natural logarithm.
log10 (x)	This function is used to calculates base 10 logarithm.
sqrt (x)	This function is used to find square root of the argument passed to this function.
pow (a,b)	This is used to find the power of the given number.
trunc()	This function truncates the decimal value from floating point value and returns integer value.

QUICK EXERCISE

The program asks user to enter value of principal (p), rate (r) and time (t) and finally calculates the compound interest by following formula.

- Total Amount = Principal * (1 + Rate / 100) ^{time}
- Compound interest = Total Amount – Principal.

QUICK EXERCISE

The program asks user to enter value of principal (p), rate (r) and time (t) and finally calculates the compound interest by following formula.

- Total Amount = Principal * (1 + Rate / 100) ^{time}
- Compound interest = Total Amount – Principal.

Solution:

```
#include <stdio.h>
#include<math.h>
int main()
{
float principal, roi,  interest, total;
int t;
printf("Enter the principal, rate of interest and time");
scanf("%f%f%d", &principal, &roi, &t);
total = principal * pow((1 + roi / 100),t);
printf("The interest is %f", total-principal);
return 0;
}
```