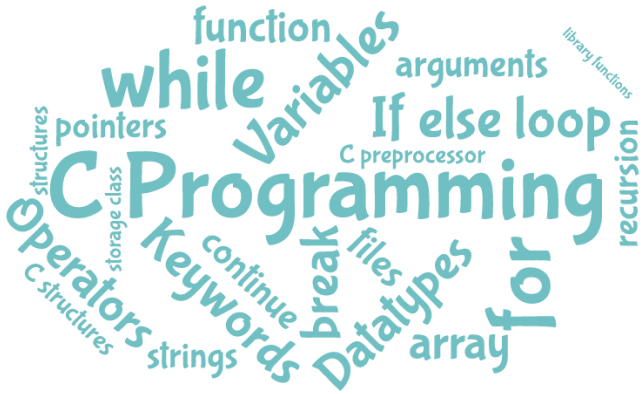
PROGRAMMING

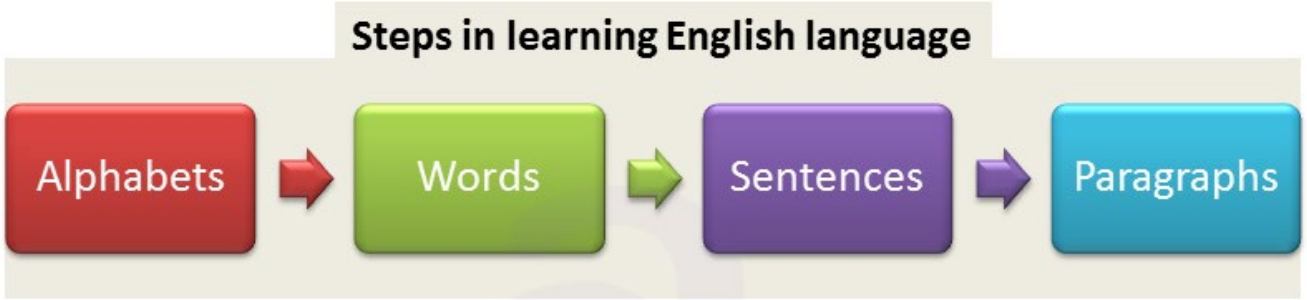
IT 112: Introduction to Programming

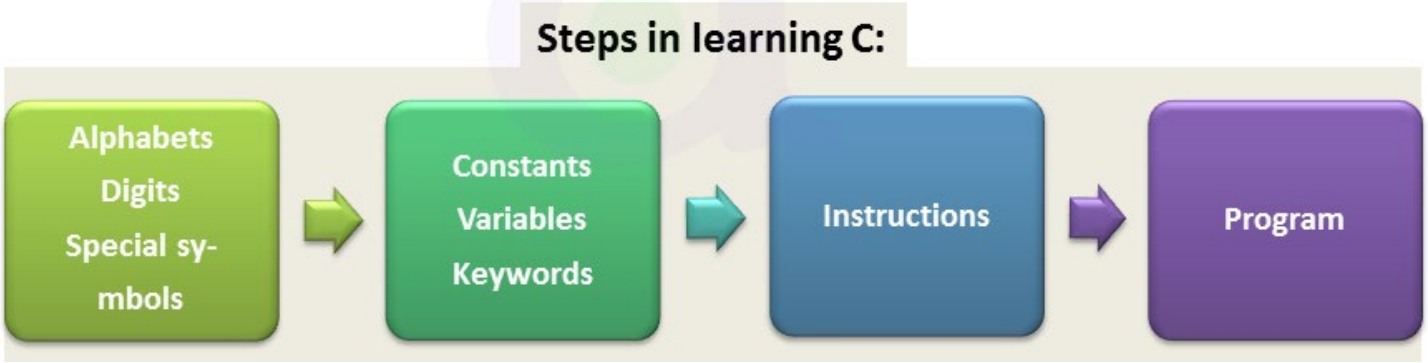
Dr. Manish Khare
Dr. Bakul Gohel



Programming in C

Language





C word vocabulary is limited
Grammatical mistake is not allowed. Computer have no I.Q. !

Let us do C programing

Lets ask computer to compute a area of circle

Give instruction to computer

```
#include<stdio.h>
#define PI 3.14

int main() {
   float radius, area;

   radius = 10 ; // mm

   area = PI * radius * radius;

   printf("\nArea of Circle : %f mm", area);

   return (0);
}
```

Computer output

```
Area of Circle : 314.000000 mm
```

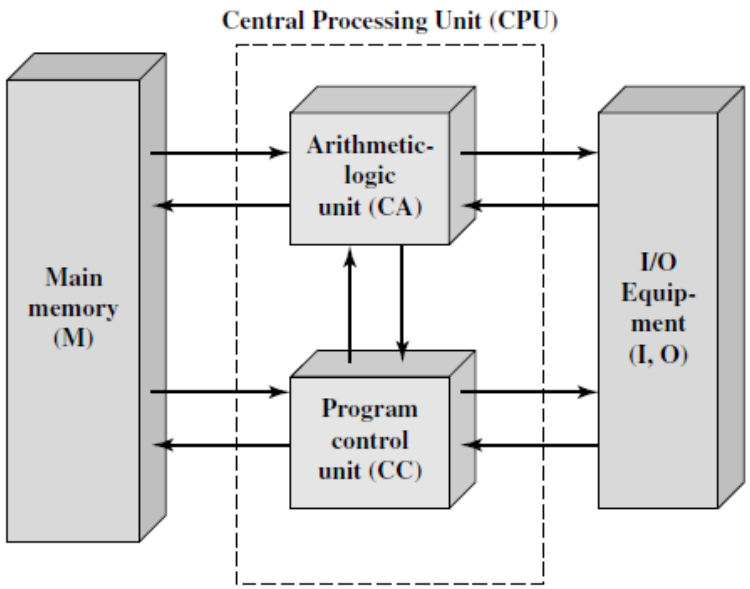
Does computer understand the C language ?

https://ideone.com/OAfGmq

C programing

Does computer (CPU) understand the C language ?
Answer: No ..!

what language computer (CPU) can understand ?
Answer: Machine Code / Binary Code



So we need a translator / converter……

C programing

C-program File_name.C

```c
#include<stdio.h>

#define PI 3.14
int main() {
  float radius, area;

  radius = 10 ; // mm

  area = PI * radius * radius;

  printf("\nArea of Circle : %f mm", area);

  return (0);
}
```

Assembly code File_name.s

```
LC2:
        .string "\nArea of Circle : %f mm"
main:
        pushq   %rbp
        movq    %rsp, %rbp
        subq    $16, %rsp
        movss   .LC0(%rip), %xmm0
        movss   %xmm0, -4(%rbp)
        cvtss2sd        -4(%rbp), %xmm1
        movsd   .LC1(%rip), %xmm0
        mulsd   %xmm0, %xmm1
        cvtss2sd        -4(%rbp), %xmm0
        mulsd   %xmm1, %xmm0
        cvtsd2ss        %xmm0, %xmm0
        movss   %xmm0, -8(%rbp)
        cvtss2sd        -8(%rbp), %xmm2
        movq    %xmm2, %rax
        movq    %rax, %xmm0
        movl    $.LC2, %edi
        movl    $1, %eax
        call    printf
        movl    $0, %eax
        leave
        ret
.LC0:
        .long   1092616192
.LC1:
        .long   1374389535
        .long   1074339512
```

Machine Code File_name.exe

```
000030 0000 0000 0000 0000 0000 0000 C800 0000  ..
000040 0E1F BA0E 00B4 09CD 21B8 014C CD21 5468  ..
000050 6973 2070 726F 6772 616D 2063 616E 6E6F  is
000060 7420 6265 2072 756E 2069 6E20 444F 5320  t
000070 6D6F 6465 2E0D 0D0A 2400 0000 0000 0000  mo
000080 0FBD 8ECD 4BDC E09E 4BDC E09E 4BDC E09E  .
000090 C8D4 BD9E 44DC E09E 4BDC E19E 20DC E09E
0000A0 C5D4 BF9E 5FDC E09E C8D4 BE9E 4ADC E09E
0000B0 C8D4 BA9E 4ADC E09E 5269 6368 4BDC E09E
0000C0 0000 0000 0000 0000 5045 0000 4C01 0300  ..
0000D0 7BE6 9D42 0000 0000 0000 0000 E000 0F0D
0000E0 0B01 070A 007A 0000 0018 0000 0000 0000  ..
0000F0 7259 0000 0020 0000 00A0 0000 0000 0001  rY
000100 0020 0000 0002 0000 0500 0200 0500 0200  .
000110 0400 0000 0000 0000 00E0 0100 0004 0000  ..
000120 3992 4C00 0200 0084 0000 0400 0020 0000
000130 0000 1000 0010 0000 0000 0000 1000 0000  ..
000140 0000 0000 0000 0000 408E 0000 A000 0000  ..
000150 00C0 0100 7414 0000 0000 0000 0000 0000
000160 0092 4B00 0824 0000 0000 0000 0000 0000
000170 D021 0000 1C00 0000 0000 0000 0000 0000
```

(In Hexadecimal format)
That is read by the operating system in binary format while executing it

Machine Independent code

Machine Dependent code

Understand and processed by computer

C programing

C-program
File_name.C

```c
#include<stdio.h>

#define PI 3.14
int main() {
  float radius, area;

  radius = 10 ; // mm

  area = PI * radius * radius;

  printf("\nArea of Circle : %f mm", area);

  return (0);
}
```

Machine
Independent code

Assembly code
File_name.s

```asm
LC2:
        .string "\nArea of Circle : %f mm"
main:
        pushq   %rbp
        movq    %rsp, %rbp
        subq    $16, %rsp
        movss   .LC0(%rip), %xmm0
        movss   %xmm0, -4(%rbp)
        cvtss2sd        -4(%rbp), %xmm1
        movsd   .LC1(%rip), %xmm0
        mulsd   %xmm0, %xmm1
        cvtss2sd        -4(%rbp), %xmm0
        mulsd   %xmm1, %xmm0
        cvtsd2ss        %xmm0, %xmm0
        movss   %xmm0, -8(%rbp)
        cvtss2sd        -8(%rbp), %xmm2
        movq    %xmm2, %rax
        movq    %rax, %xmm0
        movl    $.LC2, %edi
        movl    $1, %eax
        call    printf
        movl    $0, %eax
        leave
        ret
.LC0:
        .long   1092616192
.LC1:
        .long   1374389535
        .long   1074339512
```

Machine
Dependent code

Machine Code
File_name.exe

```
000030 0000 0000 0000 0000 0000 0000 C800 0000  ..
000040 0E1F BA0E 00B4 09CD 21B8 014C CD21 5468  ..
000050 6973 2070 726F 6772 616D 2063 616E 6E6F  is
000060 7420 6265 2072 756E 2069 6E20 444F 5320  t b
000070 6D6F 6465 2E0D 0D0A 2400 0000 0000 0000  mo
000080 0FBD 8ECD 48DC E09E 48DC E09E 48DC E09E  .?
000090 C8D4 B09E 44DC E09E 48DC E19E 20DC E09E  ÈÔ
0000A0 C5D4 BF9E 5FDC E09E C8D4 BE9E 4ADC E09E  ÅÔ
0000B0 C8D4 BA9E 4ADC E09E 5269 6368 48DC E09E  ÈÔ
0000C0 0000 0000 0000 0000 5045 0000 4C01 0300  ..
0000D0 7816 9D42 0000 0000 0000 0000 E000 0F0D  {È
0000E0 0B01 070A 007A 0000 0018 0000 0000 0000  ..
0000F0 7259 0000 0020 0000 00A0 0000 0000 0001  rY
000100 0020 0000 0002 0000 0500 0200 0500 0200  .
000110 0400 0000 0000 0000 00E0 0100 0004 0000  ..
000120 3992 4C00 0200 0084 0000 0400 0020 0000  9'
000130 0000 1000 0010 0000 0000 0000 1000 0000  ..
000140 0000 0000 0000 0000 488E 0000 A000 0000  ..
000150 00C0 0100 7414 0000 0000 0000 0000 0000  .Ã
000160 0092 4800 0024 0000 0000 0000 0000 0000  .'
000170 D821 0000 1C00 0000 0000 0000 0000 0000  Ø!
```

In Hexadecimal format
That is read by the operating system
and convert it to binary format while
executing it

Understand and
processed by computer

C
Compiler

Assembler

High/Medium
Level Language

Medium/Low
Level Language

Low / Instruction
Level Language

IDE

Integrated Development Environment (**IDE**) is the software that allow editing, managing of the codes (programmes) and compilation of the codes(programmes)

IDE → Editor + Compiler

IDE provides all comprehensive functionalities required for software development

- *Visual Studio Code.*
- *Eclipse*
- *NetBeans*
- *Sublime Text*
- *Atom*
- *Code::Blocks*
- *CodeLite*

And many more…

IDE

- Example, NetBean IDE 8.1

FirstTrialProgram - NetBeans IDE 8.1

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help

Projects
- CppApplication_1
- FirstTrialProgram
  - Header Files
  - Resource Files
  - Source Files
    - main.c
  - Test Files
  - Important Files
- myFirstProgram
- Practice
- Practice1
- Welcome_3

main() - Navigator
- main()

Start Page | main.c

Source | History

```
/*
 * File:   main.c
 * Author: Vishu
 *
 * Created on December 11, 2015, 11:27 PM
 */

#include <stdio.h>
#include <stdlib.h>

/*
 *
 */
int main(void) {
    printf ("Hello, world!");
    fflush(stdout);
    return (0);
}
```

Output

FirstTrialProgram (Clean, Compile File) | FirstTrialProgram (Build, Run) | FirstTrialProgram (Run)

FirstTrialProgram (Run) 26:1 INS

The C Character Set

- A character denotes any alphabet, digit or special symbol used to represent information.

Types	Character Set
Uppercase Alphabets	A, B, C, ... Y, Z
Lowercase Alphabets	a, b, c, ... y, z
Digits	0, 1, 2, 3, ... 9
Special Symbols	~ ‘ ! @ # % ^ & * ( ) _ - + = | \ { } [ ] : ; " ' < > , . ? /
White spaces	Single space, tab, new line.

Structure of a C program

- Every C program consists of one or more functions.
  - One of the functions must be called *main*.
  - The program will always begin by executing the main function.

```
int main()
{
  statement 1;
  Statement 2;
        .
        .
  Statement n;

  return 0;              //end of the programme ; handover control to OS
}
```

Structure of a C program

- all statement is enclosed within a pair of braces: ‘{‘ and ‘}’
- Each statement is end or terminate by semicolon e.g.”;”
- Comments may appear anywhere in a program, enclosed within delimiters ‘/*’ and ‘*/’.
- “//” can be used for single line comment

C Keywords

- As every language has words to construct statements, C programming also has words with a specific meaning which are used to construct c program instructions.
- In the C programming language, keywords are special words with predefined meaning.
- Keywords are also known as reserved words in C programming language.
- In C programming language, there are **32 keywords**. All the 32 keywords have their own meaning which is already known to the compiler.

32 Keywords in C

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

No need to remember at present ; you naturally learn these keywords as course and lab progresses…

Identifier

- Identifier refer to names given to identify various programme elements such as **variables**, **functions**, structures, constants etc.
- It is user defined
- It must be different from keywords
- May consist of letters, digits and underscore ('_') character with no space between
- Case sensitive e.g. 'area', 'AREA', 'Area' are all different

Identifiers

Valid identifiers

- abc
- simple_interest
- Aa_123
- X
- LIST
- Stud_Name
- Empl_1
- avg_empl_sal
- average_employee_salary

Invalid Identifiers

- 10abc
- Simple interest
- “Aa_123”
- (X)
- %LIST

Variables

- It is a data name that can be used to store a data value.
- Variable may take different values in memory during execution.
- Variable names follow the naming convention for identifiers.
  - Examples :: temp, speed, name_2, Current
- We need to define the data type of the variable(s) during their initialization

Declaration of Variables

- we need to declare variable before using it
  - It tells the compiler what the variable name is.
  - It specifies what type of data the variable will hold.
- General syntax:
  - **data-type** variable-list;
- Examples:
  - **int** velocity, distance;
  - **int** X;
  - **float** radius, area;
  - **char** flag, option;

Data Types in C

Data Types in C

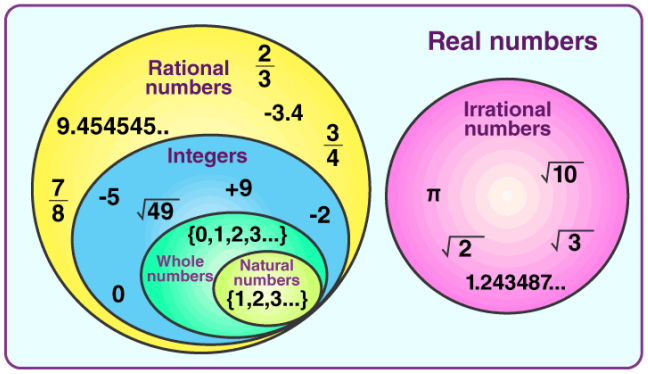
- Primary Data type
  - Integer
  - Float
  - Character
- Derived Data type
  - Array
  - Pointer
  - Structure
  - Union
  - Enum

|----user defined-------|

integer: integer number
e.g. 0, -125, 9999, -5555 etc.

float: point with decimal number
e.g. 0.11, 0, -99.25, 123.3333 etc.

char: single character
e.g. 'A', '2', '%', '\n' etc.



Binary to Decimal Conversion

101011 ➔ $1x2^5 + 0x2^4 + 1x2^3 + 0x2^2 + 1x2^1 + 1x2^0$

$= 43$

$(101011)_2 = (43)_{10}$

.0101 ➔ $0x2^{-1} + 1x2^{-2} + 0x2^{-3} + 1x2^{-4}$

$= .3125$

$(.0101)_2 = (.3125)_{10}$

101.11 ➔ $1x2^2 + 0x2^1 + 1x2^0 + 1x2^{-1} + 1x2^{-2}$

5.75

$(101.11)_2 = (5.75)_{10}$

Decimal to Binary Conversion

- Integer to Binary

2	239	
2	119	--- 1
2	59	--- 1
2	29	--- 1
2	14	--- 1
2	7	--- 0
2	3	--- 1
2	1	--- 1
2	0	--- 1

$(239)_{10} = (11101111)_2$

2	64	
2	32	--- 0
2	16	--- 0
2	8	--- 0
2	4	--- 0
2	2	--- 0
2	1	--- 0
2	0	--- 1

$(64)_{10} = (1000000)_2$

Decimal to Binary Conversion

- Fraction to Binary

.634 x 2 = 1.268
.268 x 2 = 0.536
.536 x 2 = 1.072
.072 x 2 = 0.144
.144 x 2 = 0.288
:
:

$(.634)_{10} = (.10100\ldots\ldots)_2$

.125 x 2 = 0.250
.250 x 2 = 0.500
.500 x 2 = 1.000
.000 x 2 = 0.000
.000 x 2 = 0.000
:
:

$(.125)_{10} = (.001000\ldots\ldots)_2$

Binary – Decimal Representation

- An n-bit binary number

  $B = b_{n-1}b_{n-2} \ldots b_2b_1b_0$

  - 2^n distinct combinations are possible, 0 to 2^n-1.
- For example, for n = 3, there are 8 distinct combinations.
  - 000, 001, 010, 011, 100, 101, 110, 111
- Range of numbers (**SYMBOLS**) that can be represented

  n=8 ➔ 0 to 2^8-1 (255)

  n=16 ➔ 0 to 2^{16}-1 (65535)

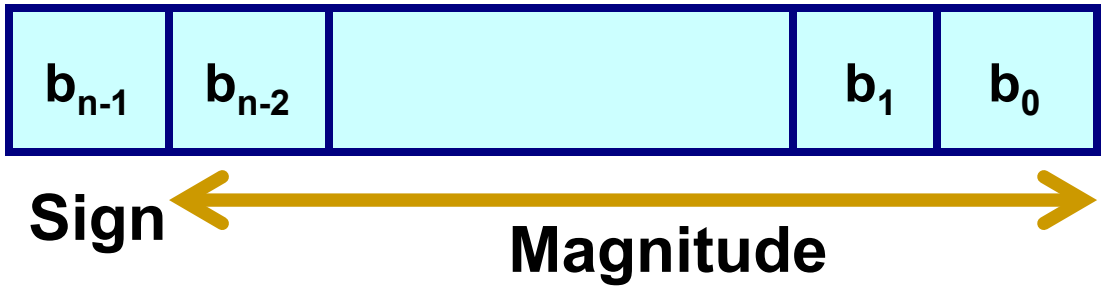
  n=32 ➔ 0 to 2^{32}-1 (4294967295)

Sign-magnitude Representation

- For an n-bit number representation
  - The most significant bit (MSB) indicates sign

    0 → positive

    1 → negative
  - The remaining n-1 bits represent magnitude.



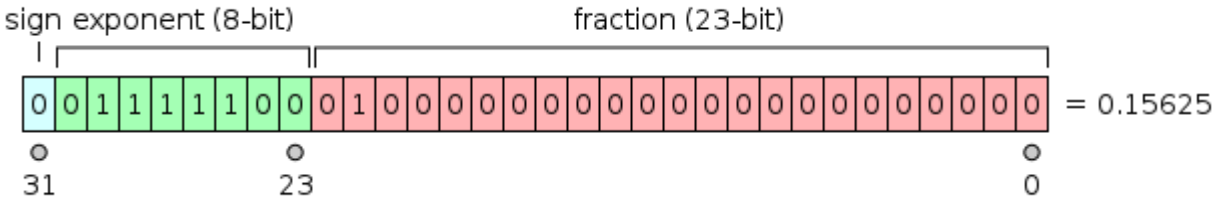
*Two's Complement Representation of integer (**self-study**)*

Floating point number: IEEE Standard 754

- Storage Layout

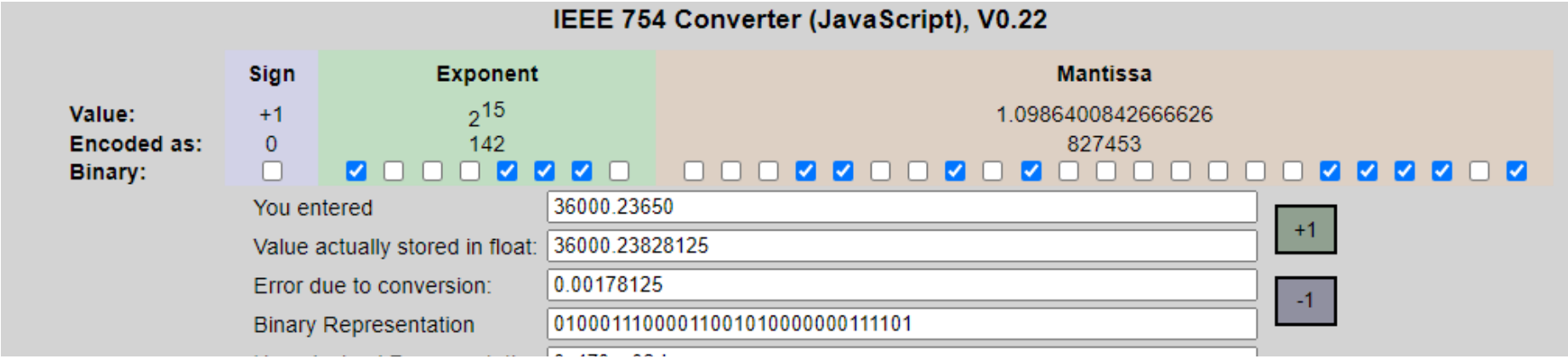
	Sign	Exponent	Fraction / Mantissa
Single Precision	1 [31]	8 [30–23]	23 [22–00]
Double Precision	1 [63]	11 [62–52]	52 [51–00]

S E M



Floating point number: IEEE Standard 754

Floating point Representation (Self-Study)



https://www.h-schmidt.net/FloatConverter/IEEE754.html

(just play with it)

ASCII Code

- Each individual character is numerically encoded into a unique 7-bit binary code.
  - A total of 2^7 or 128 different characters.
  - A character is normally encoded in a byte (8 bits), with the MSB not been used.

- The binary encoding of the characters follow a regular ordering.
  - Digits are ordered consecutively in their proper numerical sequence (0 to 9).
  - Letters (uppercase and lowercase) are arranged consecutively in their proper alphabetic order.

Some Common ASCII Codes

‘A’ :: 41 (H) 65 (D)

‘B’ :: 42 (H) 66 (D)

……

‘Z’ :: 5A (H) 90 (D)

‘a’ :: 61 (H) 97 (D)

‘b’ :: 62 (H) 98 (D)

……

‘z’ :: 7A (H) 122 (D)

‘0’ :: 30 (H) 48 (D)

‘1’ :: 31 (H) 49 (D)

……

‘9’ :: 39 (H) 57 (D)

‘(‘ :: 28 (H) 40 (D)

‘+’ :: 2B (H) 43 (D)

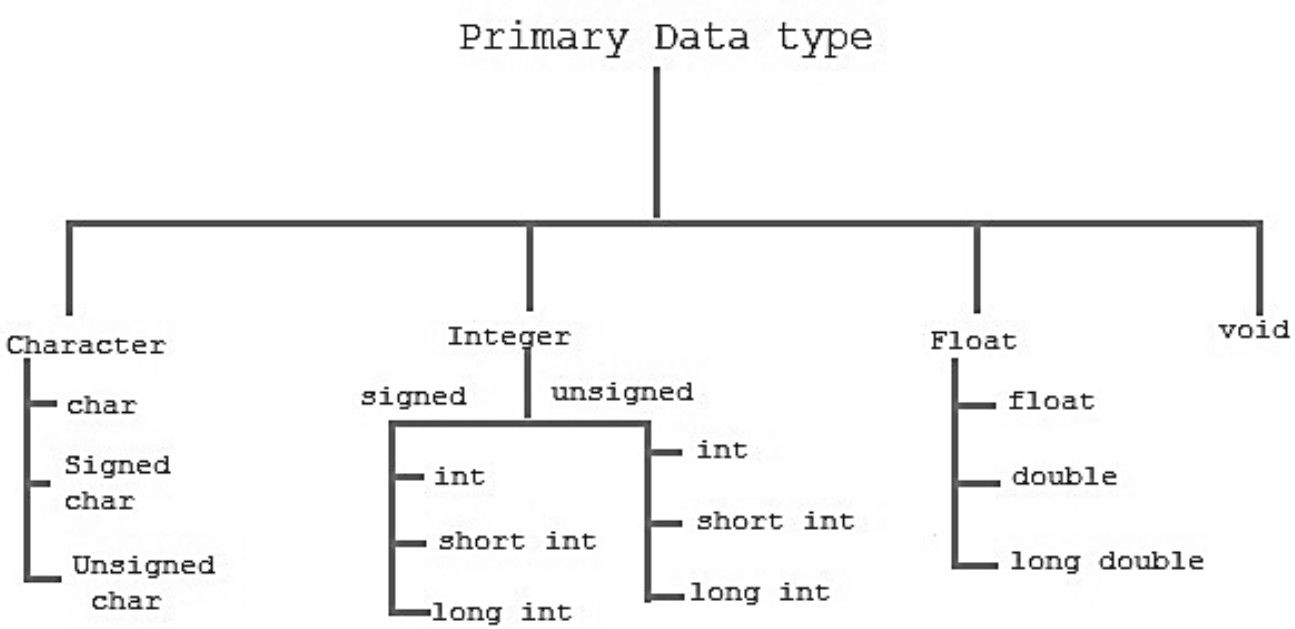
‘?’ :: 3F (H) 63 (D)

‘\n’ :: 0A (H) 10 (D)

‘\0’ :: 00 (H) 00 (D)

Data Types in C

- Some of the basic data types can be augmented by using certain data type qualifiers:
  - short
  - long
  - signed
  - unsigned



Data Types in C

- Based on system (CPU 8 vs16 vs 32 vs 64) configuration, number of bits or bytes for each data may varies
- Data type representation size for 32-bit and 64-bit system as follow

Type Name	32–bit Size	64–bit Size
char	1 byte	1 byte
short	2 bytes	2 bytes
int	4 bytes	4 bytes
long	4 bytes	8 bytes
long long	8 bytes	8 bytes

Type Name	32–bit Size	64–bit Size
float	4 bytes	4 bytes
double	8 bytes	8 bytes
long double	16 bytes	16 bytes

Data Types in C

Type	Storage size	Value range
char	1 byte	-128 to 127 or 0 to 255
unsigned char	1 byte	0 to 255
signed char	1 byte	-128 to 127
int	2 or 4 bytes	-32,768 to 32,767 or -2,147,483,648 to 2,147,483,647
unsigned int	2 or 4 bytes	0 to 65,535 or 0 to 4,294,967,295
short	2 bytes	-32,768 to 32,767
unsigned short	2 bytes	0 to 65,535
long	8 bytes or (4bytes for 32 bit OS)	-9223372036854775808 to 9223372036854775807
unsigned long	8 bytes	0 to 18446744073709551615

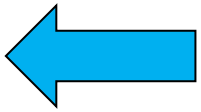
Type	Storage size	Value range	Precision
float	4 byte	1.2E-38 to 3.4E+38	6 decimal places
double	8 byte	2.3E-308 to 1.7E+308	15 decimal places

Literals

- Value that is expressed as itself ; used to assign value to variable or represent in a statement or expression

- In some literatures it is also called *constant*

```
int a = 85;
int aa = -420;
float b = 0.00314;
float bb = 3.14e-2;
char x = 'Y';
float c;
c = 1.33+a + b*5;
```



literals

Integer Literals

- An integer literals must have at least one digit.
- It must not have a decimal point.
- It can be any of zero, positive or negative.
- If no sign precedes an integer literal, it is assumed to be positive.
- No commas or blanks are allowed within an integer literal.

- Example
  - 426
  - +782
  - -8000
  - -7605

Integer Literals

- Different integer data type literals
- It is important at different application and need to take into consideration

```
85        /* decimal */
0213      /* octal */
0x4b      /* hexadecimal */
30        /* int */
30u       /* unsigned int */
30l       /* long */
30ul      /* unsigned long */
```

We will see more about it whenever need it during the course…

Real or float Literals

- Real literals could be written in two forms
  - Fractional form
  - Exponential form

Real or float Literals

- Fractional form
  - A real literal 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.
  - Example
    - +325.34
    - 426.0
    - -32.76
    - -48.5792

Real or float Literals

- Exponential form
  - The exponential form is usually used if the value is either too large or too small.
  - In exponential form of representation, the real value is represented in two parts. The part appearing before ‘e’ is called **mantissa**, whereas the part following ‘e’ is called **exponent**
  - For ex, 0.000342 can be written in exponential form of 3.42e-4 (which is equivalent to 3.42×10^{-4})

Real or float Literals

- Exponential form
  - The mantissa part and the exponential part should be separated by a letter e or 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.

  - Example
    - +3.2e-5
    - 4.1e8
    - -0.2e+3
    - -3.2e-5

Character literals

- A character literal is a single alphabet, a single digit or a single special symbol enclosed between single quotes.

- The maximum length of a character literals can be 1 character.

  - ’A’
  - ’I’
  - ‘5’
  - ‘=’
  - ‘$’

C - Input / Output

- Output operation
  - an instruction that displays information stored in memory e.g. printf
- Input operation
  - an instruction that copies data from an input device into memory e.g. scanf
- Input/output function
  - A C function that performs an input or output operation e. g scanf, printf

- These functions is available in “stdio.h” standard library file
- Therefore, we need to include it as **“#include<stdio.h>”** when ever we use printf or scanf in the programme

printf and scanf

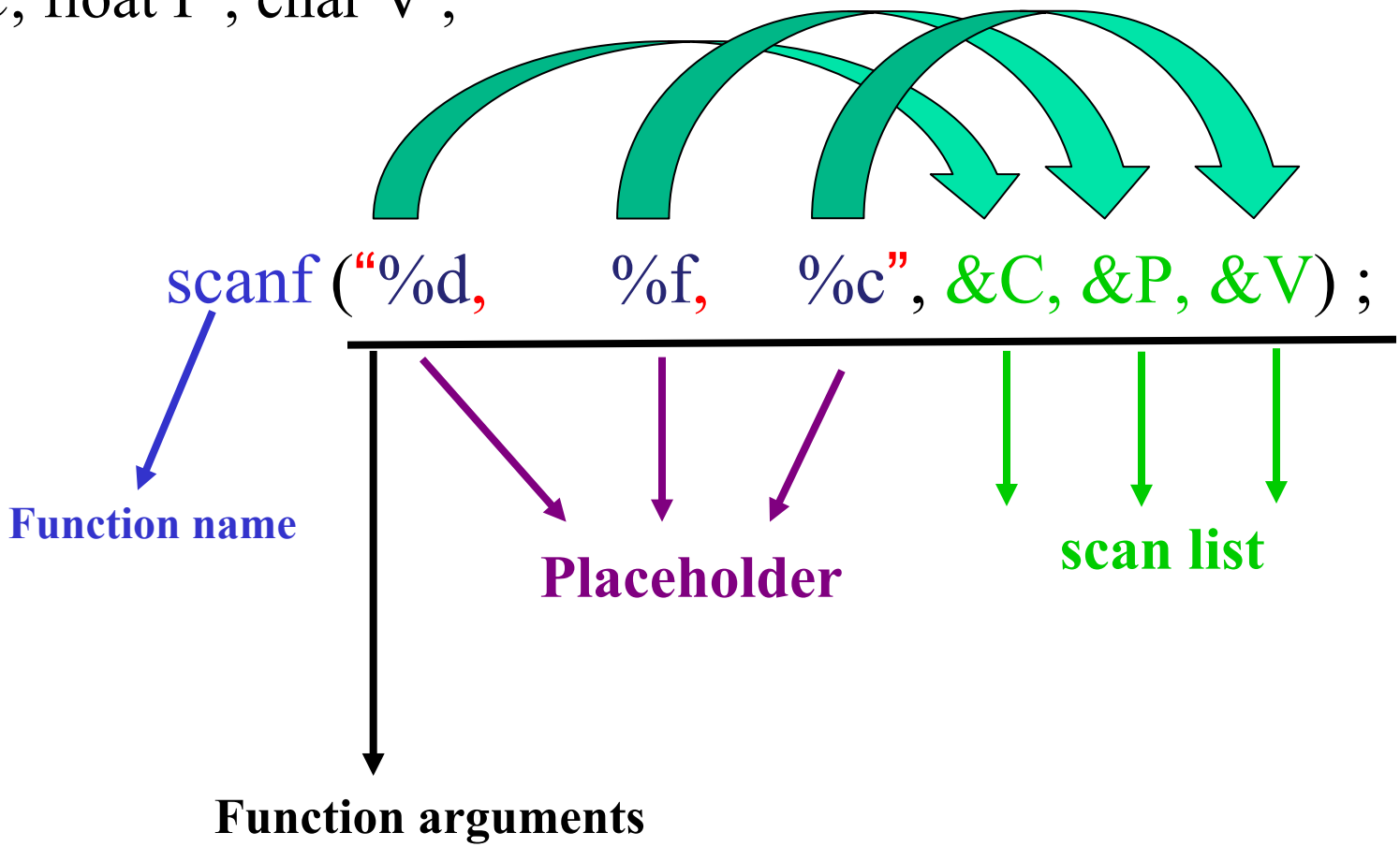
```
int main()
{
    int C ; float P; char V;

    scanf("%d %f %c",&C, &P, &V);
    printf("Cars: %d, Price: %f, Variant: %c. \n", C, P, V) ;

    return 0;
}
```

scanf

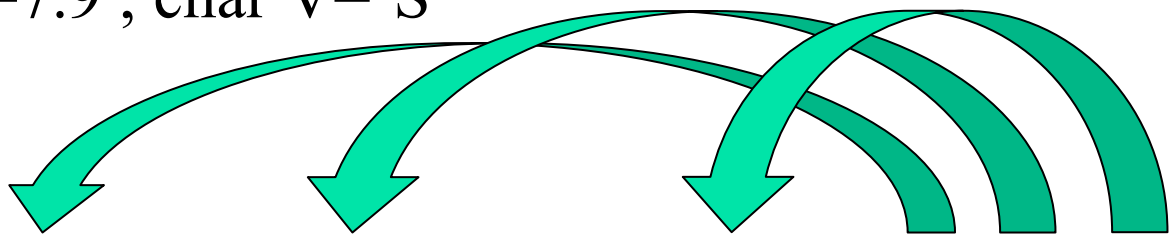
int C; float P ; char V ;



Input -> 10, 7.9, S

printf

int C=10 ; float P=7.9 ; char V='S'



printf ("Cars: %d, Price: %f, Variant: %c. \n", C, P, V) ;

Function name

Format string

Placeholder format specifier

Function arguments

Print list

Output -> Cars: 10, Price: 7.900000, Variant: S.

printf and scanf

Placeholder	Variable Type
%c	char
%d	int
%f	double
%lf	Long double