A decorative graphic on the left side of the slide, consisting of a black crosshair with colored squares (blue, red, yellow) at the ends of the lines.

IT 112: Introduction to Programming

Dr. Manish Khare

Dr. Bakul Gohel

A decorative graphic on the right side of the slide, consisting of a blue crosshair with a small circle at the intersection.

Lecture 4



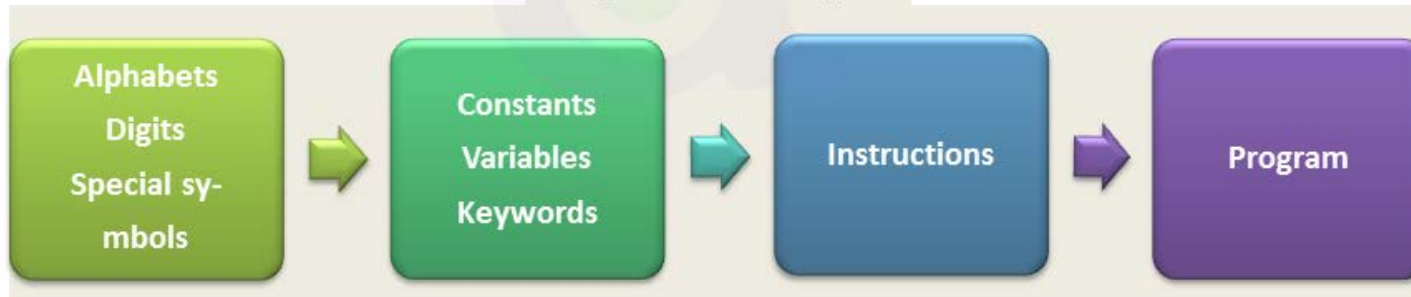
Programming in C

Language

Steps in learning English language



Steps in learning C:



C word vocabulary is limited

Grammatical mistake is not allowed. Computer have no I.Q. !

Let us do C programming

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>

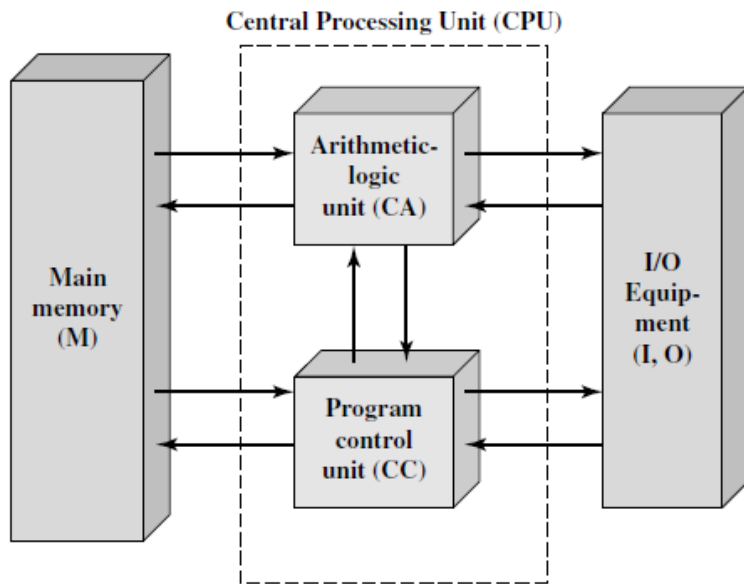
C programming

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 programming

C-program
File_name.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 l
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 Ćō
0000A0 C5D4 BF9E 5FDC E09E C8D4 BE9E 4ADC E09E Ūō:
0000B0 C8D4 BA9E 4ADC E09E 5269 6368 4BDC E09E Ćō:
0000C0 0000 0000 0000 0000 5045 0000 4C01 0300 ..
0000D0 7BE6 9D42 0000 0000 0000 0000 E000 0F0D {Ċ
0000E0 0801 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 408E 0000 A000 0000 ..
000150 00C0 0100 7414 0000 0000 0000 0000 0000 .Ĥ
000160 0092 4B00 0824 0000 0000 0000 0000 0000 .'!
000170 0071 0000 1C00 0000 0000 0000 0000 0000 n!
```

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

Machine
Independent code

Machine
Dependent code

Understand and
processed by computer

C programming

C-program
File_name.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

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

Machine Code
File_name.exe

```
000020 0000 0000 0000 0000 0000 0000 0000 0000 ..
000040 0E1F B40E 0084 09CD 21B0 014C CD21 546B ..
000060 6973 2870 726F 6772 6160 2063 616E 666F is
000080 7420 6265 2072 756E 2069 6E20 444F 5120 t i
0000A0 6D6F 6465 2180 008A 2400 0000 0000 0000 mo
0000C0 0F8D E8CD 480C E89E 480C E89E 480C E89E -"
0000E0 C8D4 B09F 440C E89E 480C E18E 200C E89E C
000100 C8D4 B09F 5FDC E89E C8D4 B09E 440C E89E C
000120 C8D4 B09F 440C E89E 5269 636B 480C E89E C
000140 0000 0000 0000 0000 0000 5045 0000 4C01 ..
000160 756E 504C 0000 0000 0000 0000 0000 8F0D C
000180 0001 070A 007A 0000 0018 0000 0000 0000 ..
0001A0 7259 0000 0020 0000 0040 0000 0000 0001 rY
0001C0 0020 0000 0002 0000 0500 0200 0500 0200 .
0001E0 0400 0000 0000 0000 0000 0100 0004 0000 ..
000200 3992 4C00 0200 0084 0000 0400 0020 0000 9
000220 0000 1000 0010 0000 0000 0000 1000 0000 ..
000240 0000 0000 0000 0000 400E 0000 0000 ..
000260 00C8 0100 7414 0000 0000 0000 0000 ..
000280 0052 4300 0024 0000 0000 0000 0000 'i
0002A0 7471 0000 1700 0000 0000 0000 0000 R1
```

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

Machine
Independent code

Machine
Dependent code

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 and managing of the codes (programmes) and compilation of the codes(programmes)

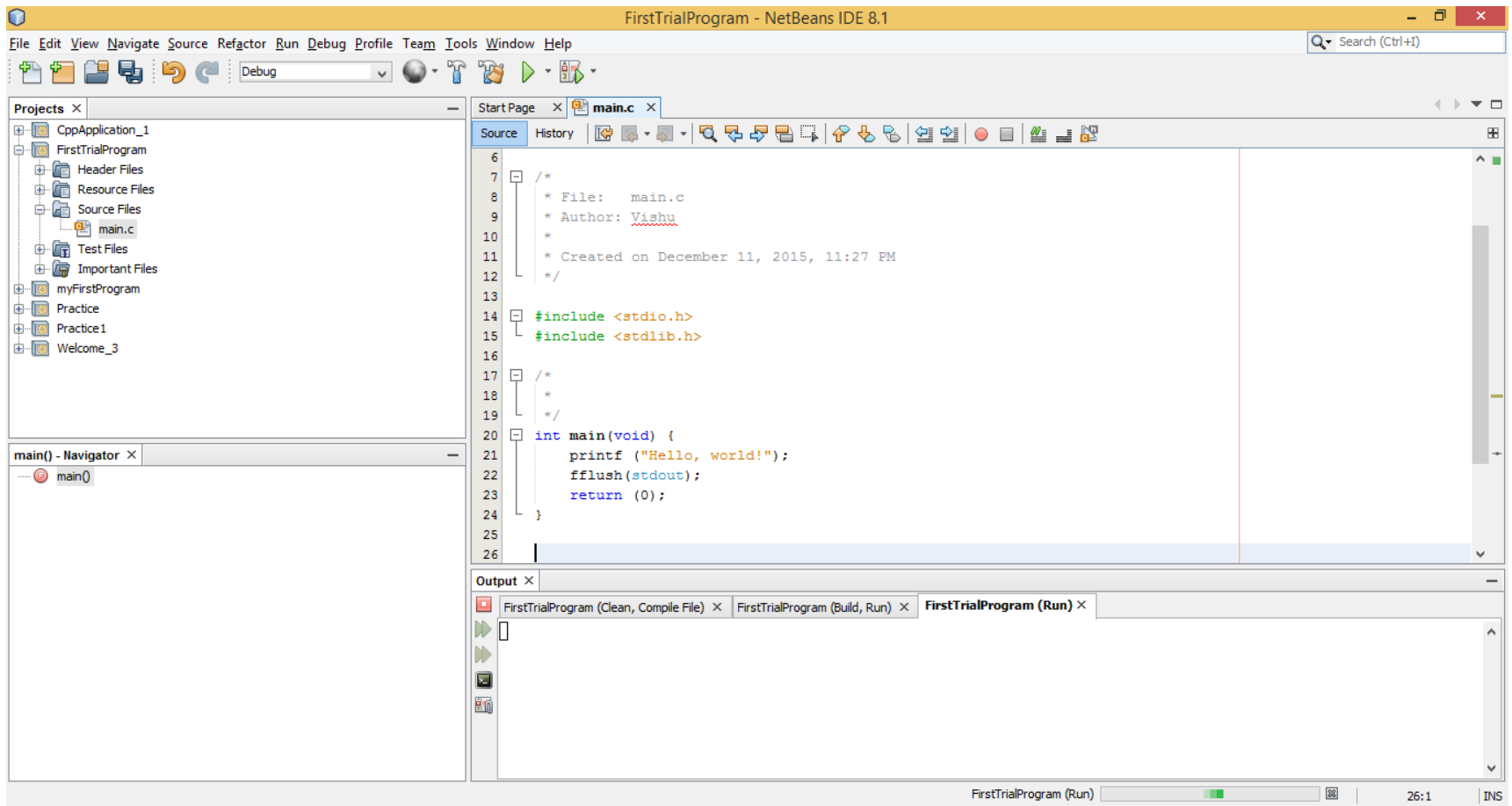
IDE → Editor + Compiler

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, NetBeans IDE 8.1



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.
 - Each compound statement is enclosed within a pair of braces: ‘{ ‘ and ‘}’

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

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

Structure of a C program..

- Each compound statement is enclosed within a pair of braces: ‘{ ‘ and ‘}’
 - The braces may contain combinations of elementary statements and other compound statements.
- 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
- _delay
- __speed

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;

Memory Map

```
#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);
```

```
}
```



Address 0

Address 1

Address 2

Address 3

Address 4

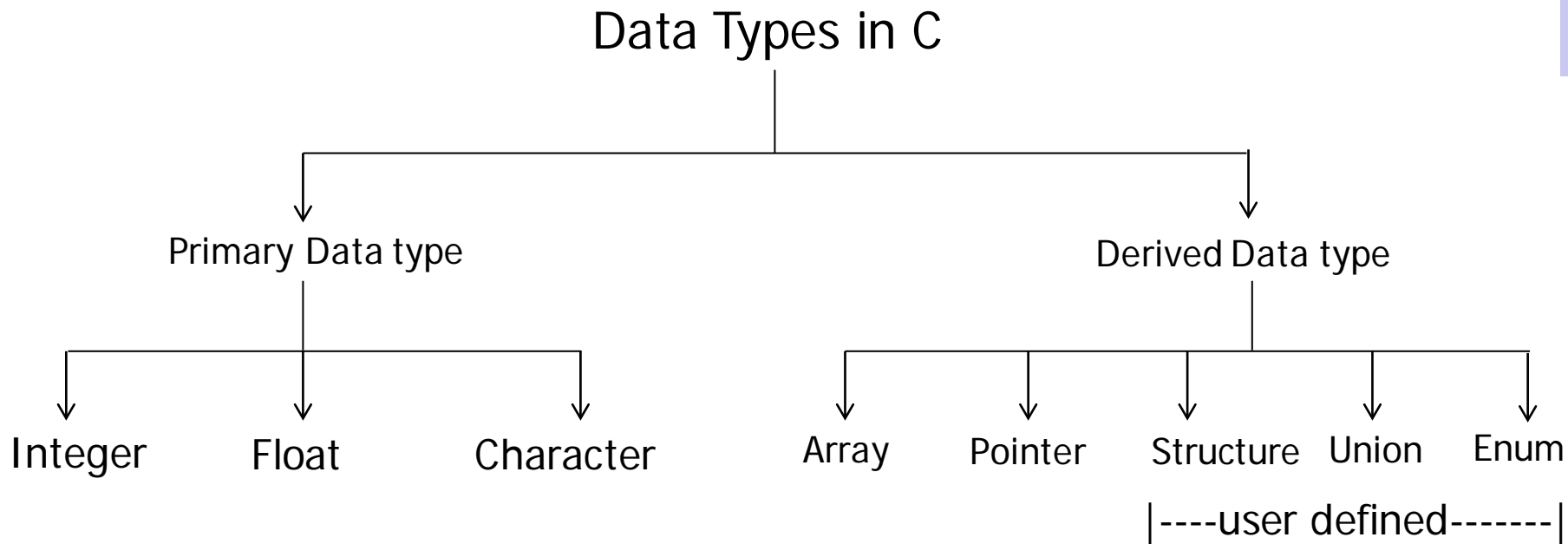
Address 5

Address 6

Address N-1

Every variable is mapped to a particular memory address

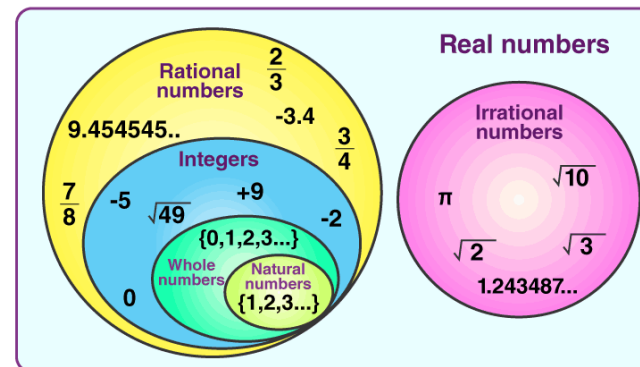
Data Types in C



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.

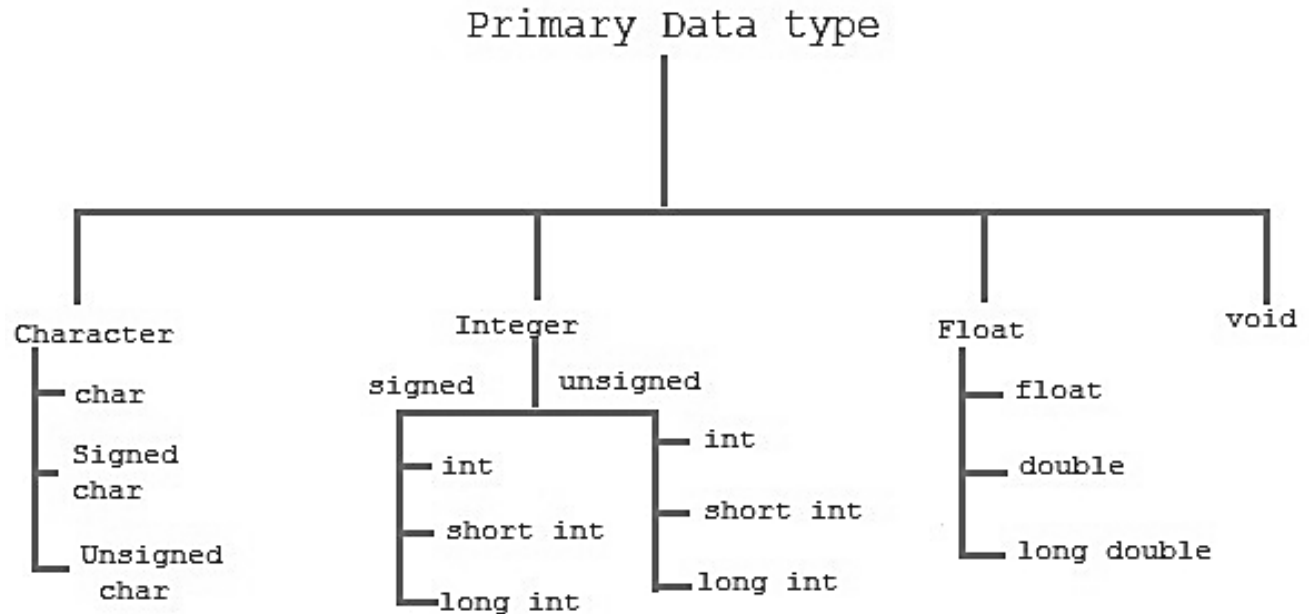
char: single character
e.g. 'A', '2', '%', '\n' etc.



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 vs 16 vs 32 vs 64) configuration, number of bits or bytes for each data may vary
- 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

Binary Representation

➤ Integer to Binary

Successive Division by 2

$$\begin{array}{r}
 2 \overline{) 29} \\
 \underline{2 14} \\
 2 \overline{) 7} \\
 \underline{2 3} \\
 2 \overline{) 3} \\
 \underline{2 1} \\
 2 \overline{) 1} \\
 \underline{2 0} \\
 0
 \end{array}$$

Remainders

1 LSB

0

1

1

1

1 MSB

Read the remainders
from the bottom up

29 decimal = 11101 binary

Binary to Decimal

1	1	0	1	1	0	1	1	
								$1 \times 2^0 = 1 \times 1 = 1$
								$1 \times 2^1 = 1 \times 2 = 2$
								$0 \times 2^2 = 0 \times 4 = 0$
								$1 \times 2^3 = 1 \times 8 = 8$
								$1 \times 2^4 = 1 \times 16 = 16$
								$0 \times 2^5 = 0 \times 32 = 0$
								$1 \times 2^6 = 1 \times 64 = 64$
								$1 \times 2^7 = 1 \times 128 = 128$

$1 + 2 + 8 + 16 + 64 + 128 = 219$

$(11011011)_2 = (219)_{10}$

© w3resource.com

(decimal ↔ binary conversion here is just for understanding... not the part of the syllabus)

➤ Integer to Binary

$$(99)_{10} = (00000000 \ 00000000 \ 00000000 \ 01100011)_2$$

➤ Float to Binary (*Self study...*)

$$(99.0)_{10} = (01000010 \ 11000110 \ 00000000 \ 00000000)_2$$

Binary Representation (ASCII)

➤ Character as integer value

ASCII TABLE

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	\$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[ENG OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]

No need to remember....