C Programming Basics

The C Character Set

- A to Z
- a to z
- 0 to 9
- Special Character

+	-	*	/	=	%	&	#
!	?	^	“	‘	~	\	|
<	>	(	)	{	}	[	]
:	;	.	,	_	@	$	

blank space (Handled Specially)

- Combination of characters

\n \t \b etc used instead of controller characters

C Programming Basics

Identifiers: Name given to various programming element like variables, Functions etc.

Rules for Identifiers:

- Upper and lower case letters
- Digits (0 to 9)
- _(under score)
- case sensitive
- digit can not be the first character
- can start with a '_' (under score)

But generally not used.

An Identifier can be arbitrarily long **(meaning vs length)**

Some compiler recognize only first 8 some upto 31

Example: File_manager vs File_ management

C Programming Basics

Valid Identifiers:

X y12 sum_1 _temp

Names area Tax_Rate TABLE

InValid Identifiers:

4th “x” order-on order flag

Check for validity

$tax recORd1 name_and_address 123-45-67

Name and Adresss file-3 A123_45_678

C Programming Basics

Key Words: **Certain reserved words that have standard predefine meaning in C programming.**

can not be used as an Identifier (But uppercase names are allowed)

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

For some compilers

ada	far	near	asm	fortran	pascal	entry	huge

See the reference manuals for the Key Words

C Programming Basics

Constants:

Numeric

Integer constants: 45, -135, 0, 11.2

Floating point constants:

456.43, -4E35, -66.0e-98

12e-11.5

Characters

single character constant:

‘A’, ‘a’, ‘1’, ‘ ’

String constant:

“ Hello World”, “A”, “ ”

C Programming Basics

Data Types:

Basic Data Types

int	Used to store integer quantity
char	used to store a character
float	used to store real numbers/floting point numbers
double	used to store floating point numbers but of double precision

- Each data type has a predefined size of storage
- Basic Data types can be Augmented by the use of Qualifiers like

short long signed unsigned

C Programming Basics

Integers: **ANSI standard specifies 3 kinds of integer**

Short integers: *short int* or *short*

Integers: *int*

Long Integers: *long int* or *long*

Further each of these can be associated with *signed* or *unsigned*

ANSI Specified Sizes:

Short int >= 2bytes

Int >= 2bytes (Size of short int <= int)

Long int >= 4 bytes (Size of long int >= int)

The exact size of the various integer data type is implementation dependent: C-Compiler/ Hardware

C Programming Basics

Example: Integer size in a typical system (PC)

May allocate 2 bytes for short int

2 bytes for int

4 bytes for long int

2 Byte Integer (16 bits)

Maximum signed value = + 32767

Minimum signed value = - 32768

Minimum unsigned value = 0

Maximum unsigned value = 65535

Actual Size from Limit.h Or Sizeof

4 Byte Integer (32 bits)

Maximum signed value = + 2147483647 = + 2.14 X 10^9

Minimum signed value = - 2147483648 = - 2.14 X 10^9

Minimum unsigned value = 0

Maximum unsigned value = + 4294967295 = + 4.29 X 10^9

C Programming Basics

Integers Constants:

Decimal Integers: 678, -23

Octal Integers: 034,0112, 0678

Hexadecimal Integers: 0Xab234, 0x12fe

A suffix u or U indicates a unsigned integer

343U, 45u

A suffix l or L indicates a long integer

22uL, 345l

What is the data type of 1234???

if no suffix is there compiler tries to store it in int, if not possible compiler tries with a long int if it is non negative compiler tries with a unsigned long int. similar rules applied for octal and hexadecimal numbers

C Programming Basics

Rules as per K&R:

1. If it is unsuffixed and Decimal:

   int, long int, unsigned long int (if positive)

2. If it is unsuffixed Octal or Hexadecimal :

   int, unsigned int, long int, unsigned long int

3. If it is suffixed by u or U:

   unsigned int, unsigned long int

4. If it is suffixed by l or L:

   long int, unsigned long int

C Programming Basics

Program For Simple Input and Output of Integers

```
main()
{
   int x;
   printf( "\nPlease Enter a Integer:")
   scanf("%d",&x);
   /* %d is used for int*/
   printf("\nYou have entered %d",x);
   return 0;
}
```

C Programming Basics

Floating Point:

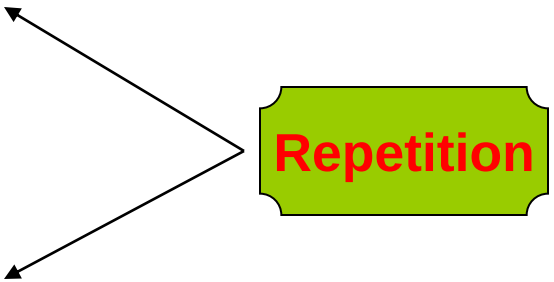
We have already discussed about representation of floating point numbers using Exponent and Mantissa.

Finite Precision

While storing a real number we need to understand we can't have infinite precision because in the world of computers we have a finite storage.

For example to store 0.2 with 5 binary digits

0.2 X 2 = 0	+	0.4
0.4 X 2 = 0	+	0.8
0.8 X 2 = 1	+	0.6
0.6 X 2 = 1	+	0.2
0.2 X 2 = 0	+	0.4
0.4 X 2 = 0	+	0.8



(0.2) = (0.00110011001100110011…….)

C Programming Basics

(0.2) = (0.00110011001100110011…….)

= .1875 with 5 bit precision

= (0.00110011001100110011…….)

= 0.1992 with 8 bit precision

- **This error is known as Round-off error which decreases with the increase in precision.**
- **Machine Accuracy of a 8 bit binary number**

  $\varepsilon = 2^{-7} = 7.8125 \times 10^{-3}$

- **How many digits in decimal??**

  $= \log_{10}(\varepsilon) = \log_{10}(7.8125 \times 10^{-3}) = 2.1072 = 2$ digits

- **So an accuracy upto 2 digits in decimal (In average)**

C Programming Basics

There are three floating point data types

1. **float**
2. **double**
3. **long double**

We can not use unsigned modifier for floating point number.

Condition:

Precision of long Double >= Precision of Double
>= Precision of Float

IEEE 754 standard is used: 32 bit, 64 bit

SIZE of Floating point numbers can be found in FLOAT.H

C Programming Basics

Program For Simple Input and Output of Floating point numbers

```
main()
   {
      float x;
      printf( "\nPlease Enter a Integer:")
      scanf("%f",&x);
      /* %f /%g /%e is used for Float*/
      /* %lf /%lg /%le is used for Double*/
      /* %Lf /%Lg /%Le is used for Long Double*/
      printf("\nYou have entered %f",x);
      return 0;
   }
```

C Programming Basics

Floating Point Constants

Valid values

123.45, 34.4e-76, 5E78 and 1. are valid constants.

- Default data type is Double
- A suffix ‘f’ or ‘F’ specify float value
- A suffix ‘l’ or ‘L’ specify a Long Double

Example

4.6L specifies a long double value

4.6f specifies a float value

C Programming Basics

Character Data Type

- We have discussed about ASCII / extended ASCII
- Character in C is stored as one byte integer
- So a character can be interpreted as a number also.
- Can be signed or unsigned
- Don’t have to bother: to use character between (0 - 127)
- Characters are represented in single quotes
- ‘0’ and 0 are different (48 and null=> ’\0’)

C Programming Basics

Escape Sequence

‘\n’	New line
‘\t’	Horizontal Tab
‘\v’	Vertical Tab
‘\b’	Back Space
‘\r’	Carriage Return
‘\f’	Form Feed
‘\a’	alert (bell)
‘\\’	\ (back slash)
‘\?’	?(Question mark)
‘\”’	‘ (Single Quote)
‘\”’’	“ (Double Quote)
\000	000 (Octal Number)
\xhh	hh (Hexadecimal Number)

Example

\xb : \ 013 Vertical Tab

\x7 : \007 Bell character

C Programming Basics

Program For Simple Input and Output of Character

```
main()
   {
         char mychar;
         printf( "\nPlease Enter a character:")
         scanf("%c",&mychar);
         /* %c is used for character*/
         printf("\nYou have entered %c",mychar);
         printf("\nYou have entered %d",mychar);
         return 0;
   }
```

C Programming Basics

Trigraph Sequence of Characters

In IBM key board some of the character were not there

Trigraph	??=	??/	??’	??(	??)	??!	??<	??>	??-
Char	#	\	^	[	]	|	{	}	~

Wide Character

To have Multilanguage support a new data type wchar_t is Included

Wide character constants are preceded with L

Example: L‘x’

C Programming Basics

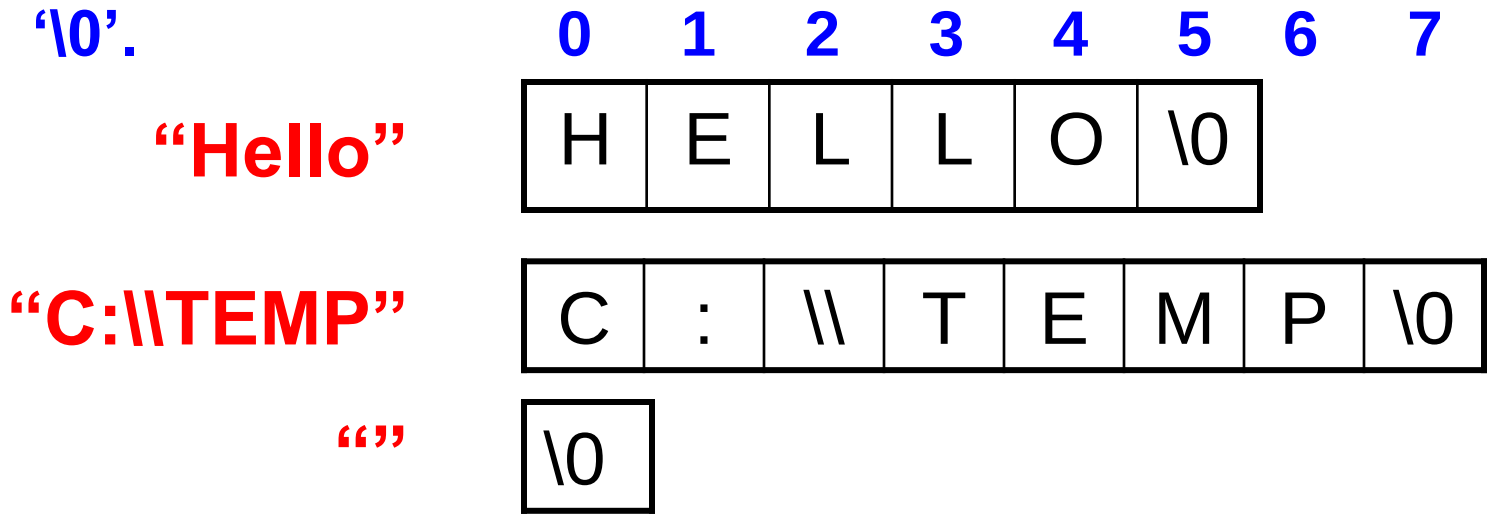
Strings in C

- A set of character within double quotes

  “Hello World” “C Language” “123”

  “\nHello” “\t\b\aFAT” “c:\\temp”

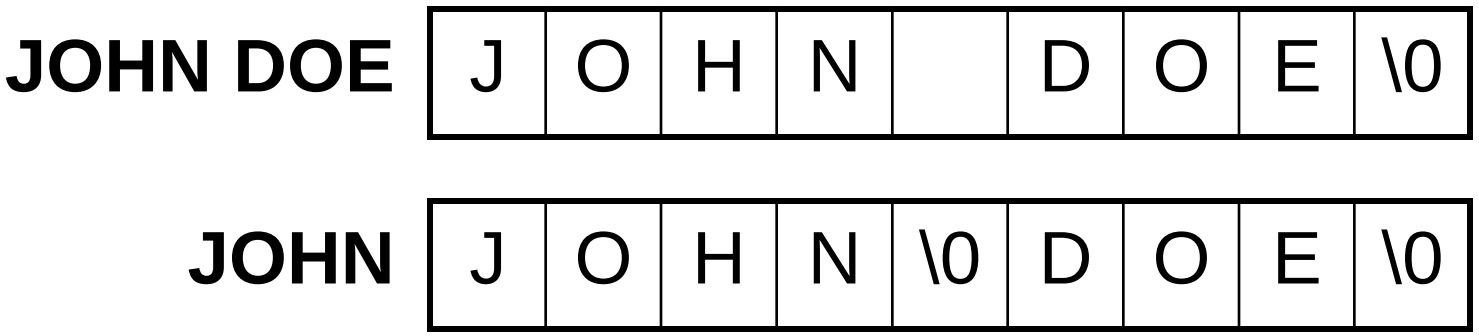
- In C language string is a null terminated array of characters. One extra space is required to store the ‘\0’.



C Programming Basics

Declaring Character Array

- char string_name[n_chars] = string constant;
- char greet[6] =“Hello”;
  - 6 chars stored, 6 bytes declared
- char myname[25] =“John Doe”;
  - 9 chars stored, 25 bytes declared
- This string terminates at the first null character encountered in the character set



C Programming Basics

Simple Input/Output String

```
Int Main()
{
        char mystr[100];
        printf("\nEnter a String: ");
        scanf("%s", mystr);
        printf("\n The string you have entered is \n
        %s"mystr);
        return 0;
}
```

C Programming Basics

Variable and Its Declaration

- A variable is an Identifier for a memory location
  In which the data can be stored and retrieve
- Variable are Associated with a Data Type
- Once Assigned, Data type can not be changed. But value in the variable can be changed.
- Name of the variable should explain its feature and its use.

  Example:

  Bad naming style: f,m,a for Force, Mass and Acceleration

  Good naming style: Force, Mass, Acceleration

  Example: Number of students

  Bad Naming: No_Student

  Good Naming: Num_Student;

Different Naming Standards!!!

C Programming Basics

Variable Declaration

```
DataType VarableList;
```

Examples:

```
int a, test, check;
float takeFirst, takeSecond;
```

Can also be written as:

```
int a;
int test;
int check;
float takeFirst;
float takeSecond;
```

C Programming Basics

Example:

```
main()
{
        int x, z, k;
        k = x + z; /* value of x and Z are not defined */
        printf(" Value of K is %d",k);
}
```

Need to assign value before using a variable

Better to assign the values at the time of declaration

C Programming Basics

Variable Initialization

Assigning values to the variables at the time of declaration is called Initialization

Example:

```
int x=23,y=2;
float k;
k = x+y;
```

Not Compulsory but Recommended

C Programming Basics

Symbolic Constants

```
# define SYMBOLIC_CONSTANT Contsant_Value
```

Constants can of any data type

Merits:

- **Maintainability**
- **Readability**

Example:

 # define VATRATE 4.5
 # define TRUE 1
 # define FRIEND "SUSAN"

Example: Printf("%s is my FRIEND", FRIEND);

C Programming Basics

Expressions

Represent a single entity like a constant, variable or logical conditions (True or False) etc

x

X=3

a + b

x = y

c = a + b

x <= y

x == y

In C each expression has a value

C Programming Basics

Statements

A statement causes a computer to carried out some action

3 different types of statements

Expression Statement: Expression followed by a semicolon

```
a = 3;
c = a + b;
++a ;
printf("enter a vale");
;            /* A NULL Statement */
```

Control Statements: Special Statements used for control action, logical tests, loops and branches like For/ while loops, If Else Statements

Compound Statement: Consists of several statements within braces { }. Individual statements may be expression statements or control statements or compound statements