UNIT I

OVERVIEW OF C

- ☆ C is a general-purpose computer programming language developed between 1969 and 1973 by Dennis Ritchie at the Bell Telephone Laboratories for use with the UNIX operating system.
- ☆ Although C was designed for implementing system software, it is also widely used for developing portable application software.
- ☆ C is one of the most widely used programming languages of all time and there are very few computer architectures for which a C compiler does not exist.
- ☆ C has greatly influenced many other popular programming languages, most notably C++, which began as an extension to C.

History of C:

- ☆ C is one of the most popular computer languages.
- ☆ It is structured, high level, machine independent language.
- ☆ It is platform independent
- ☆ ALGOL is the first structured programming language used in Europe.
- ☆ In 1967, Martin Richards developed a language called BCPL (Basic Combined programming Language)
- ☆ In 1970, ken Thompson created a language using BCPL features called B.
- ☆ C was evolved from ALGOL, BCPL and B by Dennis Ritchie at Bell Laboratories in 1972.
- ☆ Unix Operating system, which was developed at Bell Laboratories, was coded almost entirely in C
- ☆ C is running under a variety of operating system and hardware platform.
- ☆ C proved to be an excellent programming language for writing system programs.
- ☆ “The C programming language” by Kernighan and Ritchie, 1977 is known as “K&RC”
- ☆ ANSI, in 1983 standard for C was introduced it is called an ANSI C.

Year	Language Development
1960	ALGOL (Algorithmic Language)
1970	BCPL (Basic Combined Programming Language)
1970	B
1971	Traditional C
1978	K&RC
1989	ANSI C
1990	ANSI/ISO C

Features of C:

C is attractive and popular because

- General-purpose language
- Structured Language
- Flexible and powerful language
- System programming Language
- Fast running and efficient Language
- Supports limited data types.
- Commands may be inserted anywhere in a program
- Programs are made up of functions
- More built-in functions
- Permits recursion.

Importance of C:

- It is a robust language
- It is well suited for writing both system software and business packages.
- Programs written in C are efficient and fast
- It is many times faster than BASIC
- Several standard functions are available
- It is highly portable
- C language is structured and it makes program debugging, testing and maintenance easier.
- It is a collection of functions that are supported by C library.
- We can add own functions
- The programming task becomes simple using functions.
-

Basic Structure of C program:

```
Documentation Section                         [File Inclusions]
Link Section                                          [Constant Definitions]
Definition Section                            [External Variables Definitions]
Global Declaration Section                    main()
Main()  Function Section       (or)      {
{                                                             Variable declarations
Declaration Section/part                              Program Statements;
Executable part                                       }
}                                                             [User-Defined Functions]
Subprogram Section
    Function 1
    Function 2
            .                            (user defined functions)
            .
    Function n
```

<u>Sample Program:</u>

main()

```
{
/* Printing Begins */
      Printf("Welcome to C Language");
/* ………Printing Ends …….*/
}
```

- main() is a special function which tells the computer where the program starts.
- The documentation section consists of a set of comment lines, it gives program name, author name and other details.
- The link section provides instruction to the compiler to link system library functions
- Definition section defines all symbolic constant.
- Global variables, which are used in other programs also declared in Global declaration section.
- C program must have one main() function.
- Declaration and executable part are two parts of main.
- All variables used in the executable part are declared in declaration section.
- At least one statement in the executable part.
- These two parts must appear in opening and closing braces.
- All statements in these two parts end with a semicolon.
- User-defined functions are generally placed immediately after the main function.
- Data consisting of number, characters and strings and to provide useful output known as information.
- Executing a sequence of precise instructions called a program.

<u>Character Set:</u>

- The characters in C are grouped letters, Digits, special characters, white spaces is known as its characters set.

<u>Character set in ANSI C</u>

Upper-case Alphabets :		A to Z
Lower-case Alphabets:		a to z
Decimal digits	:	0 to 9
Special Characters	:	+ - * / % = <> _ Blank : ; , . ‘ “ ? ! # \ $ ( ) [ ] { } & | ^ ~ White spaces.

<u>C TOKENS</u>:

- A Tokens may be a single character or a group of characters which has a specific meaning.
- In a passage of text, individual words and punctuation marks are called tokens.

1) Identifier
2) Keywords
3) Constants
4) String Constants
5) Operators and
6) Separators

<u>Keywords and Identifiers:</u>

- All Keywords have fixed meanings and these meanings cannot be changed.
- Keywords must be written in lowercase.

<u>ANSI C Keywords</u>

Auto	double	int	struct
Break	else	long	switch
Case	enum	register	typedef
Char	extern	return	union

Const
Continue
Default
Do
float
for
goto
if
short
signed
sizeof
static
unsigned
void
volatile
while

Identifiers:

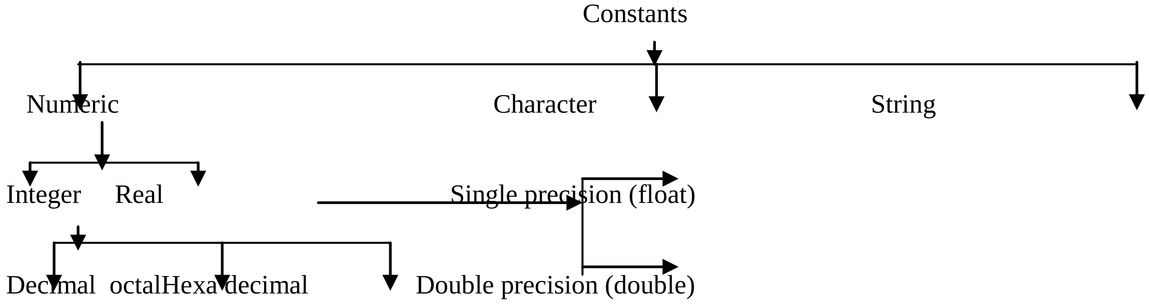
- Identifiers refer to the names of data types, constants, variables, functions and arrays etc in a program.
- Identifiers is a sequence of characters.
- Identifiers are formed by using alphabets, digits and _ an underscore.

Rules of Identifiers:

- First character must be an alphabet or underscore.
- Must consists of only letters, digits or underscore.
- Only first 31 characters are significant
- Cannot use a keyword.
- Must not contain white space.

CONSTANTS:

- Constants are the fixed values that do not change during the execution of a program.



Integer Constants:

- It refers sequence of digits without a decimal point.

Rules:

- Comma and blank spaces cannot be included within integer constants.
- Decimal integers consists of a set of digits 0 through 9, preceded by an optional – or + sign.

Valid	**Invalid**
0 1 +1 -1	345,123
-4236	34 51 23
94838639L	75.
600000u	34-51-23

L or l representing long integer.

U or u representing unsigned integer

UL or ul representing unsigned long integer.

OCTAL Integer constant:

- Octal Integer constant is formed form the octal number system 0 through 7, with a leading 0 (Zero)

Valid	**Invalid**
0001	627
0627	0628

-07757	06.27

Hexadecimal Integer Constant:

- It is formed from the hexadecimal number 0 through 9 and A through F (either upper or lowercase), leading with 0X, or 0x

Valid	**Invalid**
0x0	626
0x9A	0626
-0X9Aff	0Xapq

Floating Point Constant:

- Any number with a decimal point is called a floating point or real or single precision constant.
- The real number may also be expressed in exponential or scientific notation.

General Form:

Mantissa e exponent

e or E is used to represent exponent followed by a positive or negative integer.

Rules :-

1) Both the interger and fractional parts consists of a sequence of digits
2) Special character's except +,- and . are not allowed.
3) Floating constants may end with f or F

Example:

The value 215.65 may be written as 2.1565e2 in exponential notation e2 means multiply by 10^{2}.

Double precision floating constants:

- A double precision floating constant is similar to a single precision floating constant, but with higher range of values and greater precision.
- L or l is used for extend double precision numbers.

Valid	**Invalid**
698354L	1.5E +2.5
+5.0E3	-4.5e-2
3.5e-5	$255

Single Character Constants:

A character written within single quotes is called a character constant.

Valid	**Invalid**
'x' 'z' '8'	"y'
'?' '!' ';' '+'	"k"

For example,

The statement

Printf("%d",'a');

Would print the number 97, the ASCII value of the letter a.

Printf("%c",'97');

Would output the letter 'a'

String Constant:

- A string constant is a sequence of characters enclosed in double quotes.
- The characters may be letters, numbers, special characters and blank spaces.

Valid	**Invalid**
“Hello!”	Program
“987”	‘Area’
“Good”	Maths”
“5+3”	

‘K’ gives the integer value

“K” It is a string constant that contains the character K.

<u>Backslash Character Constants:</u>

- It is used in output functions.
- It is also called as escape sequence characters.

‘\a’	audible alert
‘\b’	back space
‘\f’	form feed
‘\n’	new line
‘\r’	carriage return
‘\t’	horizontal tab
‘\v’	vertical tab
‘\’’	single quote
‘\”’	double quote
‘\?’	question mark
‘\\’	backslash
‘\0’	null

<u>Variables:</u>

- ☆ A variable is a data name that may be used to store a data value.
- ☆ Variable may take different values at different times during execution.
- ☆ Variable names may consist of letters, digits and underscore (_) character.

<u>Rules:</u>

- They must begin with a letter or underscore.
- ANSI standard recognizes a length of 31 characters but normally should not be exceed more than eight characters.
- It should not be a keyword.
- White space is not allowed.

Valid	**Invalid**
John	123
X1	(area)
Mark	25th
T-raise	Char
int-type	group one

<u>Data Types:</u>

- Data type is a term that refers to the kind of data used in a program.

Two major categories of data types are

(i) Scalar data types or Fundamental data types.

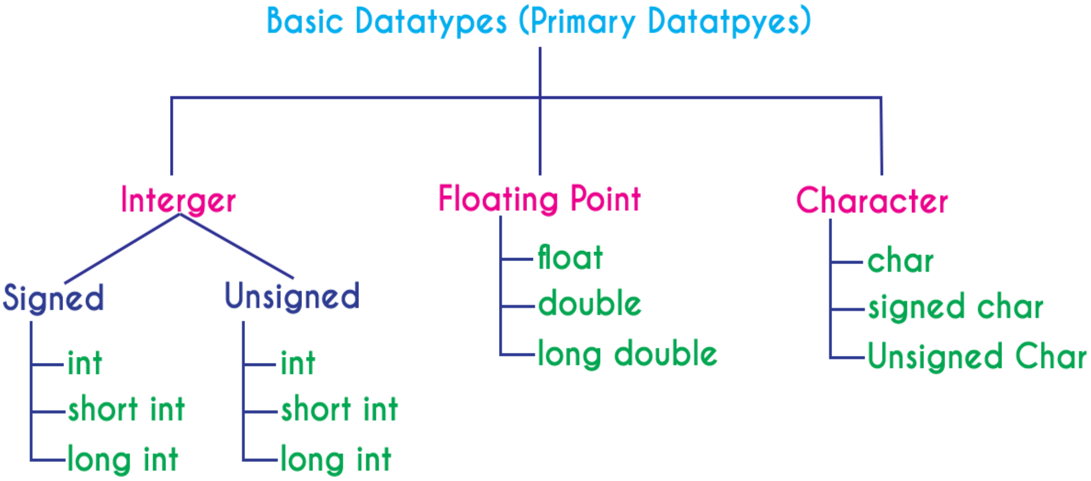
(ii) Derived data types or Structured data types.

- The data items of scalar data-types are numbers and hence they are known as Arithmetic data types.
- Arithmetic data-types are classified as integral and floating types.
- The enum, int and char types are known as integral types.
- Float and double are known as floating types.

Four scalar data types are

Data types	**Range of Values**
Int	-128 to 127
Char	-32,768 to 32,767
Float	3.4e-38 to 3.4e+e38
Double	1.7e-308 to 1.7e+308

- ☆ Also known as primitive or primary or fundamental or basic data types.
- ☆ Derived data types are derived from the collections of scalar data types. They are also known as structured data types.
  - ❖ Arrays
  - ❖ Functions
  - ❖ Pointers
  - ❖ Structures
  - ❖ Union



<u>Void types:</u>

- The void types has no values.
- This is usually used to specify the type of functions.

<u>Declarations of Variables:</u>

A variable can be used to store a value of any data type.

<u>Syntax:</u>

Data-type v1,v2,…..,vn;

- V1,v2,…,vn are the names of variables.
- Variables are separated by commas.

<u>Example:</u>

int count;

intnumber,total;

double ratio;

<u>User-Defined Type Declaration:</u>

“Type Definition” that allows users to define an identifier that would represent an existing data type.

<u>Syntax:</u>

Typedef type identifier;

- Type refers to an existing data type and “identifier” refers to the “new” name given to the data type.
- Typedef cannot create a new type

typedefint units;

typedef float marks;

unit symbolizes in and marks symbolizes float.

Units	batch1,batch2;
Marks	name, class;

- Advantage of typedef is that we can create meaningful data type names for increasing the readability of the program.

<u>Enumerated Data type:</u>

- An enumeration is a special integer data type.
- This data type associated integer constants to identifiers.
- Enumerated data type is also user defined data type.

<u>Syntax:</u>

Enum identifier {value1, value2,…..,valuen);

It can be used to declare variables that can have one of the values enclosed within the braces (known as enumeration constants).

Enum identifier v1,v2,….,vn;

```
enum tag
{
        enumerator-1;
        enumerator-2;
        …
        …
        enumerator-n;
}
```

where

enum is a keyword indicating enumerated data type.

Tag is a name used to identify the specific enumeration. It is optional.

Enumerator-1,enumerator-2,…,enumerator-n are list of identifiers.

The list of identifiers enclosed within braces.

The enumerators may have

Default integer constant settings (or)

Explicit assignment given by programmer.

Syntax for declaring enumeration data type is

Enum tag var1, var2, …..,varn;

Example:

```
        Enum flowers { ROSE,JASMINE, LOTUS};
        Enum vehicles {BUS,TRAIN,SCOOTER,CAR};
        Enum currency {rupee=1,dollar=50,pound=60,euro=40};
```

- If there is no value the identifier represent the values 0,1,2,3 and so on, from left to right.
- During explicit assignment, the value may be in any order.

```
main()
{
        int i;
        enumswtch
        {   OFF,ON
        }
        enumswtch flag;
        i = setflag(flag);
        printf("Flag value : %d \n",i);
}
setflag(flag)
{
        printf("Enter the flag value \n");
        printf("0 for OFF and 1 for ON \n");
        scanf("%d",&flag);
        return flag;
}
```

Declaration of Storage class:

- Storage class that provides information about the variable location and visibility.
- The storage class decides the portion of the program within which the variables are recognized.

Storage class	**Meaning**
Auto	Local variable known only to the function in which it is declare. Default is auto.
Static	Local variable which exists and retains its value even after the control is transferred to the calling function.
Extern	global variable known to all functions in the file.
Register	Local variable which is stored in the register.

Storage class is another qualifier (like long or unsigned) that can be added to a variable declaration.

```
Auto int count;
Register char ch;
Static int x;
Extern long total;
```

- Static and extern variables are automatically initialized to zero.
- Auto variables contain undefined values known as 'garbage' unless they are initialized explicitly.

Example:

```
        int m;
        main()
        {
```

```
        int i;
        float balance;
        …
        …
        function1();
    }
    function1()
    {
        int i;
        float sum;
    …
    …
    }
```

- m variable is declared before main is called global variable. It can be used in all the functions.
- Global variable is also known as an external variable.
- I, balance, sum are called local variables.
- Local variables are visible and meaningful only inside the functions.
- Note the variable i has been declared in both the functions.
- Any change in the value of i in one function does not affect its value in the other.

Assigning values to variables:

- Variables are created for use in program statement.

```
Value = amount + inrate * amount;
While (year <= PERIOD)
{
  …
 year = year + 1;
}
```

- the numeric value stored in the variable inrate is multiplied by the value stored in amount and the product is added to amount. The result is stored in the variable value.
- The variable value is called the target variable.

Assignment Statement:

Syntax:

```
    Variable-name= constant/ variable/expression;
```

Ex:

```
    Initial-value = 0;
    Balance = 75.84;
    Final-value = balance * 1000;
```

C permits multiple assignments in one line.

Initial-value = 0; final-value = 100;

Assign a value to a variable at the time the variable is declared.

Ex:

```
int final-value = 100;

int yes = 1;

float balance = 756.56;

char yellow = ‘y’;
```

- The process of giving initial values to variables is called initialization.
- Initialization of more than one variable in one statement using multiple assignment operators.

```
p=q=s=0;

x=y=z=100;
```