UNIT I

OVERVIEW OF C – INTRODUCTION

C is a general-purpose high level language that was originally developed by Dennis Ritchie for the Unix operating system. It was first implemented on the Digital Eqquipment Corporation PDP-11 computer in 1972.

The Unix operating system and virtually all Unix applications are written in the C language. C has now become a widely used professional language for various reasons.

- Easy to learn
- Structured language
- It produces efficient programs.
- It can handle low-level activities.
- It can be compiled on a variety of computers.

Facts about C

- C was invented to write an operating system called UNIX.
- C is a successor of B language which was introduced around 1970
- The language was formalized in 1988 by the American National Standard Institute (ANSI).
- By 1973 UNIX OS almost totally written in C.
- Today C is the most widely used System Programming Language.
- Most of the state of the art software have been implemented using C

Why to use C?

C was initially used for system development work, in particular the programs that make-up the operating system. C was adoped as a system development language because it produces code that runs nearly as fast as code written in assembly language. Some examples of the use of C might be:

- Operating Systems
- Language Compilers
- Assemblers
- Text Editors
- Print Spoolers
- Network Drivers
- Modern Programs
- Data Bases
- Language Interpreters
- Utilities

C Program File

All the C programs are writen into text files with extension ".c" for example hello.c. You can use "vi" editor to write your C program into a file.

This tutorial assumes that you know how to edit a text file and how to write programming insturctions inside a program file.

C Compilers

When you write any program in C language then to run that program you need to compile that program using a C Compiler which converts your program into a language understandable by a computer. This is called machine language (ie. binary format). So before proceeding, make sure you have C Compiler available at your computer. It comes alongwith all flavors of Unix and Linux.

If you are working over Unix or Linux then you can type gcc -v or cc -v and check the result. You can ask your system administrator or you can take help from anyone to identify an available C Compiler at your computer.

If you don't have C compiler installed at your computer then you can use below given link to download a GNU C Compiler and use it.

CHARACTER SET

When you write a program, you express C source files as text lines (page ***) containing characters from the source character set. When a program executes in the target environment, it uses characters from the target character set. These character sets are related, but need not have the same encoding or all the same members.
Every character set contains a distinct code value for each character in the basic C character set. A character set can also contain additional characters with other code values. For example:
• The character constant 'x' becomes the value of the code for the character corresponding to x in the target character set.
• The string literal "xyz" becomes a sequence of character constants stored in successive bytes of memory, followed by a byte containing the value zero: {'x', 'y', 'z', '\0'}
A string literal is one way to specify a null-terminated string, an array of zero or more bytes followed by a byte containing the value zero.

C TOKENS
The smallest individual units in C are called tokens. C has 6 types of tokens as shown below

C Tokens

Keywords Identifiers ConstantsStrings Special symbols Operators

KEYWORDS

Keywords have a constant and fixed meaning and the meaning cannot be changed. The keywords serve as building blocks for program statements. All keywords must be written in lowercase.
C –keywords are:
autodoubleintstruct
breakelselongswitch
caseenumregistertypedef
charexternreturnunion
constfloatshortunsigned
continueforsignedvoid
defaultgotosizeofvolatile
doifstaticwhile

IDENTIFIERS

Identifiers refer to the name of variables, functions and arrays. They
are user-defined names consisting of a sequence of letters, digits with the letter as the first character. Underscore can be used to link two words.
Example: a, stu_no, marks_1, names etc.,
CONSTANTS
Constants refer fixed values that do not change during the execution of program. Constants consist of the following categories.

Constants

Numeric constants Character constants

Integer Constants Real constants Single character constants String constants

Decimal Octal Hexa decimal

Integer constants:
It refers to a sequence of digits.
Types of integer constants
1. Decimal integer constants (set of digits from 0 to 9 with an optional + or – sign)
2. Octal integer constants (consists of digits from 0 to 7 with a leading 0).
3. Hexa decimal integer constants (consists of the digits 0 to 9 and the letters a to f preceded by 0X or 0x)

Decimal integer constants examples:

Valid examples	Invalid examples
1. 123	1. 153 43
2. –76565	2. 20,000
3. 0	3. $667
4. +736	4. 9-898

Octal integer constants examples:

Valid examples	Invalid examples
1. 034	1.0X2
2. 07	2.0283
3. 07655	3.0987

Hexa decimal integer constants examples:

	Valid examples	Invalid examples
1.	0x2	1.0xnmjk
2.	0x87665	2.0x1286m
3.	0xabc	3.08972
4.	0x53647	4.0khjdi

Real constants:
The numbers with fractional part is called real or floating-point constants.

Example: 1. 54.98
2. 25.75
Notations used to represent real constants are:

1. Decimal notation
2. Scientific notation

Decimal notation:
1. Whole number followed by a decimal point and fractional part.
2. It is possible to omit the digits before the decimal point.
3. It is possible to omit the digits after the decimal point.

Example: 1) 215. 2) .125

Scientific (exponential) notation:
1. The mantissa is either a real number in decimal notation or an integer.
2. The exponent is an integer with an optional plus or minus sign.
3. The letter e separating mantissa and exponent part.
4. The exponent can be written in either lower or upper case.

The General form for exponential notation:

mantissa e exponent

Valid examplesInvalid examples
i. .65e4.2 3e 4(white space not allowed)
ii. 12e-212e*2(special characters not allowed)
iii. 1.5e+3120$($ symbol not allowed)

Examples of numeric constants:

Constant	Valid	Remarks
1. 89283L	yes	represents long integer
2. 26,000	no	Comma is not allowed
3. 5.3 e 2	no	No white space is allowed
4. 0x7	yes	represents hexa decimal integer
5. 077	yes	Represents octal integer

Single character constants:

A character enclosed by single quote marks is called single character constants.
Examples:1) ‘a’
2) ’#’
3) ’2’
4)’ ’
Points to remember:
1. The character constant ‘2’ is not same as the number 2.
2. Blank space enclosed by single quote is also called as character constant.
3. Character constants have integer values known as ASCII values.

String Constants:
A sequence of characters enclosed by double quotes. The character may be any letter, digits, special characters and blank space.
Example: 1. “Welcome to c programming”
2. “2001”
3. “Well done”
4. “34+23”
5. “d”

Backslash character constants:
Backslash character constants are used in output functions. Each backslash character constants contains two characters to represent a single character. These combinations are called escape sequences.
Examples:
1) ‘\n’- new line
2) ‘\t’-horizontal tab
3) ‘\v’-vertical tab
4) ‘\0’-null

VARIABLES
A variable is a data name that may be used to hold a data value. A variable may take different values at different times during the program execution.

Rules for forming a variable:
1. The starting character should be a letter. The first character may be followed by a sequence of letters or digits.
2. The maximum number of characters in a variable may be 8 characters. The number of characters differs from compiler to compiler.
3. Upper and lower case are significant. Example: TOTAL is not same as total or Total.
4. The variable should not be a keyword.
5. White space is not allowed.
6. Special characters except _(underscore) symbol are not allowed.

Examples:
Valid Invalid
1. john1868 --The first character should be a letter)
2. value (area)--Special characters are not allowed)
3. tot_amt27th--The first character should be a letter)

4. a1%--Special characters are not allowed)

DECLARATION OF VARIABLES
1. It tells the compiler what the variable name is.
2. It specifies what type of data the variable will hold.

Data-type v1,v2,v3….vn;
Syntax:
v1,v2,v3….vn- variables
Example:
1. int count;
2. int a,b;
3. float area,volume;
4. char array;
5. double a1;

DATA TYPES
C language data types can be broadly classified as
1. Primary data type
2. Derived data type
3. User-defined data type
1. Primary data type
All C Compilers accept the following fundamental data types
1.

Integer

int

2.

Character

char

3.

Floating Point

float

4.

Double precision floating point

double

5.

Void

void

The size and range of each data type is given in the table below

DATA TYPE

RANGE OF VALUES

char

-128 to 127

Int

-32768 to +32767

float

3.4 e-38 to 3.4 e+38

double

1.7 e-308 to 1.7 e+308

Integer Type:
Integers are whole numbers with a machine dependent range of values. A good programming language as to support the programmer by giving a control on a range of numbers and storage space. C has 3 classes of integer storage namely short int, int and long int. All of these data types have signed and unsigned forms. A short int requires half the space than normal integer values. Unsigned numbers are always positive and consume all the bits for the magnitude of the number. The long and unsigned integers are used to declare a longer range of values.
Floating Point Types:
Floating point number represents a real number with 6 digits precision. Floating point numbers are denoted by the keyword float. When the accuracy of the floating point number is insufficient, we can use the double to define the number. The double is same as float but with longer precision. To extend the precision further we can use long double which consumes 80 bits of memory space.
Void Type:
Using void data type, we can specify the type of a function. It is a good practice to avoid functions that does not return any values to the calling function.
Character Type:

A single character can be defined as a defined as a character type of data. Characters are usually stored in 8 bits of internal storage. The qualifier signed or unsigned can be explicitly applied to char. While unsigned characters have values between 0 and 255, signed characters have values from –128 to 127.

Size and Range of Data Types on 16 bit machine.

TYPE

SIZE (Bits)

Range

Char or Signed Char

8

-128 to 127

Unsigned Char

8

0 to 255

Int or Signed int

16

-32768 to 32767

Unsigned int

16

0 to 65535

Short int or Signed short int

8

-128 to 127

Unsigned short int

8

0 to 255

Long int or signed long int

32

-2147483648 to 2147483647

Unsigned long int

32

0 to 4294967295

Float

32

3.4 e-38 to 3.4 e+38

Double

64

1.7e-308 to 1.7e+308

Long Double

80

3.4 e-4932 to 3.4 e+4932

DECLARATION OF VARIABLES
Every variable used in the program should be declared to the compiler. The declaration does two things.
1. Tells the compiler the variables name.
2. Specifies what type of data the variable will hold.
The general format of any declaration

datatype v1, v2, v3, …….. vn;

Where v1, v2, v3 are variable names. Variables are separated by commas. A declaration statement must end with a semicolon.

Example:
Int sum;
Int number, salary;
Double average, mean;

Datatype	Keyword Equivalent
Character	char
Unsigned Character	unsigned char
Signed Character	signed char
Signed Integer	signed int (or) int
Signed Short Integer	signed short int (or) short int (or) short
Signed Long Integer	signed long int (or) long int (or) long
UnSigned Integer	unsigned int (or) unsigned
UnSigned Short Integer	unsigned short int (or) unsigned short
UnSigned Long Integer	unsigned long int (or) unsigned long
Floating Point	float
Double Precision Floating Point	double
Extended Double Precision Floating Point	

long double

User defined type declaration

In C language a user can define an identifier that represents an existing data type. The user defined datatype identifier can later be used to declare variables. The general syntax is

typedef type identifier;

here type represents existing data type and 'identifier' refers to the 'row' name given to the data type.

Example:
typedef int salary;
typedef float average;

Here salary symbolizes int and average symbolizes float. They can be later used to declare variables as follows:

Units dept1, dept2;
Average section1, section2;

Therefore dept1 and dept2 are indirectly declared as integer datatype and section1 and section2 are indirectly float data type.

The second type of user defined datatype is enumerated data type which is defined as follows.

Enum identifier {value1, value2 …. Value n};

The identifier is a user defined enumerated datatype which can be used to declare variables that have one of the values enclosed within the braces. After the definition we can declare variables to be of this 'new' type as below.
enum identifier V1, V2, V3, …… Vn

The enumerated variables V1, V2, ….. Vn can have only one of the values value1, value2 ….. value n

Example:

enum day {Monday, Tuesday, …. Sunday};
enum day week_st, week end;
week_st = Monday;
week_end = Friday;
if(wk_st == Tuesday)
week_en = Saturday;
Declaration of Storage Class
Variables in C have not only the data type but also storage class that provides information about their location and visibility. The storage class divides the portion of the program within which the variables are recognized.

auto: It is a local variable known only to the function in which it is declared. Auto is the default storage class.

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: Social variables which are stored in the register.

DEFINING SYMBOLIC CONSTANTS

A symbolic constant value can be defined as a preprocessor statement and used in the program as any other constant value. The general form of a symbolic constant is

define symbolic_name value of constant

Valid examples of constant definitions are :

define marks 100
define total 50
define pi 3.14159

These values may appear anywhere in the program, but must come before it is referenced in the program. It is a standard practice to place them at the beginning of the program.

Declaring Variable as Constant

The values of some variable may be required to remain constant through-out the program. We can do this by using the qualifier const at the time of initialization.

Example:
Const int class_size = 40;
The const data type qualifier tells the compiler that the value of the int variable class_size may not be modified in the program.

OPERATORS AND EXPRESSIONS

➢ C supports a rich set of operators.

➢ An operator is a symbol that tells the computer to perform certain mathematical or logical operations.

➢ They are used to manipulate data.

➢ C operates can be classified into number of categories :

They include:

- Arithmetic operators
- Relational operators
- Logical operators
- Assignment operators
- Increment and decrement operators
- Conditional operators

• Bitwise operators
• Special operators

ARITHMETIC OPERATORS:

Arithmetic operators are used to perform arithmetic operations like addition, subtraction, multiplication, division and to find the remainder in division. The corresponding symbols are +(addition or unary plus), - (subtraction or unary minus), *, /,%(modulo division). C does not have an operator for exponentiation.

Operator	Meaning	Example	Result
+	Addition	5+2	7
-	Subtraction	5-2	3
*	Multiplication	5*2	10
/	Division	5/2	2
%	modulo division	5%2	1

Relational operators:

It is used to relate any two or more quantities.

Operator	Meaning	example	result
<	less than	5<2	F
>	greater than	5>2	T
>=	greater than or equal to	5<=2	F
<=	lesser than or equal to	5>=2	T
==	equal to	5==2	F
!=	not equal to	5!=2	T

Unary + and – Operator:

When only one operand is used with + or – operator, the operation is called unary plus or unary minus, does not refer to addition or subtraction.
Ex: +28.-5.

LOGICAL OPERATORS:

It is used to combine two or more relational expressions. The value combined using logical expressions is always logical ie., either true or false.

Operator	Meaning	Example	Result
&&	logical and	(5<2)&&(5>3)	F
||	logical or	(5<2)||(5>3)	T
!	logical not	!(5<2)	T

ASSIGNMENT OPERATORS:

v op =exp;

= is used to assign value to a variable. In addition to this C has a set of shorthand assignment operators.
Syntax:

Statement with simple Assignment operator.	Statement with short Hand operator.
a=a+1	a+=1

a=a-1	a-=1
a=a*1	a*=1
a=a/1	a/=1
a=a%b	a%=b

Advantages:

1. No need to repeat the operator again in the right hand side.
2. The statement is more concise and clear.
3. The statement is more efficient.
4. It is easy to read.

INCREMENT AND DECREMENT OPERATORS:

To decrement the value of the variable by one or to increment the value of the variable by one.

Operators:

++ - adds 1 to the operand
-- - subtracts 1 unary operators

Example:

++m; (++ used as prefix operator)
m++; (++ as postfix operator)

Consider the following

m=5;
y=++m;

In the above case m and y would be 6.

m=5;
y=m++;

In the above case m would be 6 and y would be 5.

Similarly for the --operator.

CONDITIONAL OPERATOR OR (TERNARY OPERATOR):

A ternary operator pair "?:" is used to check a condition and select a value depending on the value of the condition.

exp1?exp2:exp3
Syntax:

Example :

x=(a>b)?a:b;

In the above the condition is evaluated first. If true then a will be assigned to x or else b will be assigned to x. The above syntax is equivalent to if …else statement.

BITWISE OPERATORS:

Manipulates the data at bit level.

Operator	Meaning
&	bitwise AND
|	bitwise OR

^	bitwise EXCLUSIVE OR
<<	shift left
>>	shift right
~	one's complement

SPECIAL OPERATORS:

sizeof() - to determine the size of the variable
, (comma) - to link the related expressions together
Example : value=(x=10,y=17,x+y)
. and -> - member selection operator
& and * - pointer operator

sizeof operator

➢ It is used to find the number of bytes occupied by a variable /data type in computer memory.
➢ Size of(float)---returns 4 bytes.
➢ Int m,x[50];
Size of(m)—2 bytes.
Size of(x)—returns 100
Comma operator

➢ It is used to link related expression to make the program more compact.
➢ Example:
Temp =x;
X=y;
Y=temp;
Can be written as
Temp=x,x=y,y=temp.

ARITHMETIC EXPRESSIONS
An expression is a combination of variables constants and operators written according to the syntax of C language. In C every expression evaluates to a value i.e., every expression results in some value of a certain type that can be assigned to a variable. Some examples of C expressions are shown in the table given below.

Algebraic Expression
C Expression

a x b – c

a * b – c

(m + n) (x + y)

(m + n) * (x + y)

(ab / c)

a * b / c

3x2 +2x + 1

3*x*x+2*x+1

(x / y) + c

x / y + c

EVALUATION OF EXPRESSIONS

Expressions are evaluated using an assignment statement of the form
Variable = expression;
Variable is any valid C variable name. When the statement is encountered, the expression is evaluated first and then replaces the previous value of the variable on the left hand side. All variables used in the expression must be assigned values before evaluation is attempted.

Example of evaluation statements are
x=a*b–c
y=b/c*a
z=a–b/c+d;

The following program illustrates the effect of presence of parenthesis in expressions.

```
.
main ()
{
float a, b, c x, y, z;
a = 9;
b = 12;
c = 3;
x = a – b / 3 + c * 2 – 1;
y = a – b / (3 + c) * (2 – 1);
z = a – ( b / (3 + c) * 2) – 1;
printf (“x = %fn”,x);
printf (“y = %fn”,y);
printf (“z = %fn”,z);
}
.
```

Output
x=10.00
y=7.00
z = 4.00