Chapter 2
Fundamentals of C

Overview of C

C is a computer programming language developed in 1972 by **Dennis M. Ritchie** at the Bell Telephone Laboratories to develop the UNIX Operating System. C is a simple and **structure oriented** programming language.

C is also called **mother Language** of all programming Language. It is the most widely use computer programming language, This language is used for develop system software and Operating System. All other programming languages were derived directly or indirectly from C programming concepts.

In the year 1988 'C' programming language standardized by ANSI (American national standard institute), that version is called **ANSI-C**. In the year of 2000 'C' programming Language standardized by 'ISO' that version is called C-99

C Language is mainly used for

Design Operating system

Design Language Compiler

Design Database

Language Interpreters

Utilities

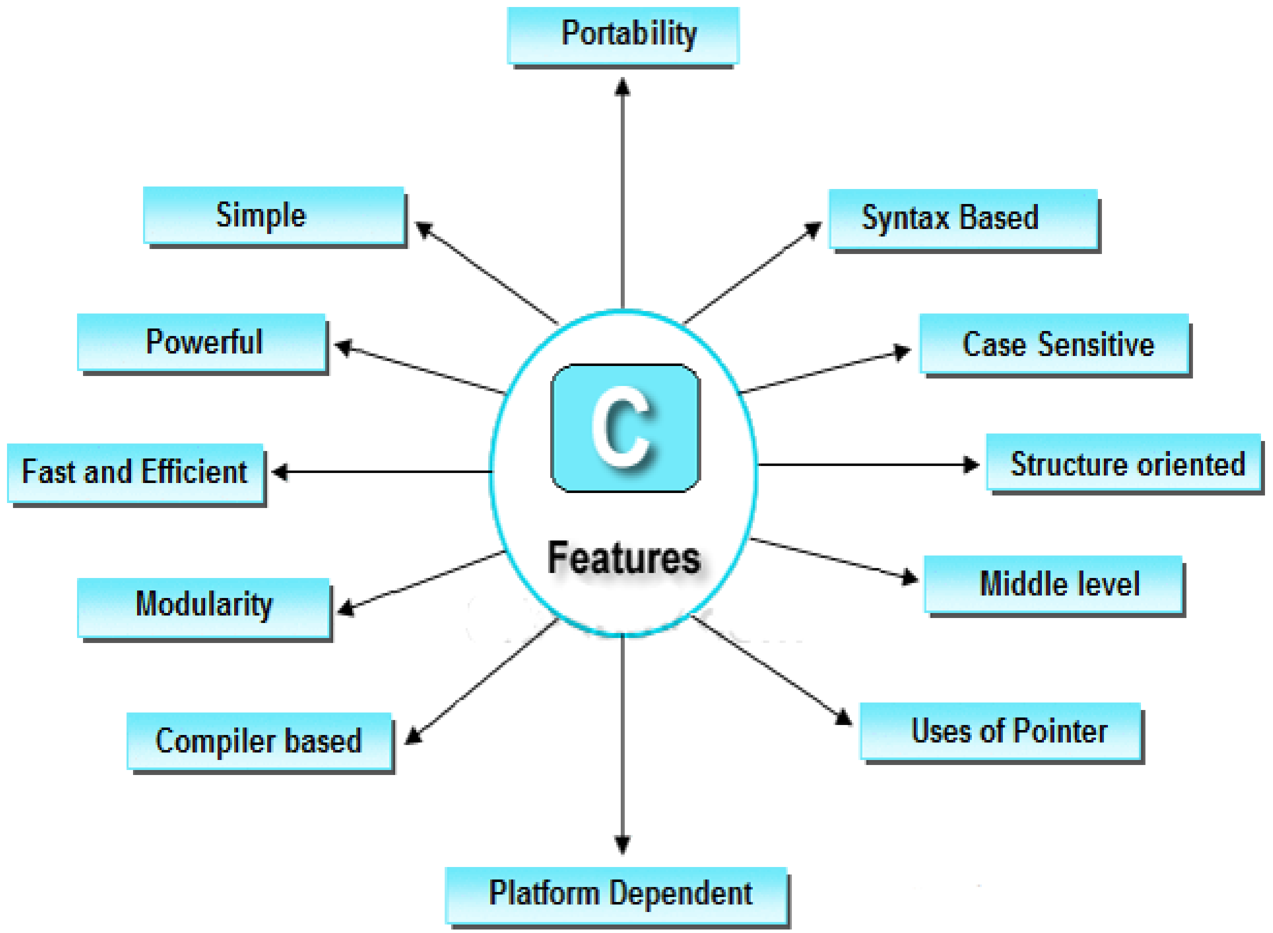
Network Drivers

Assemblers

Features of C Language

It is a very simple and easy language, C language is mainly used for develop desktop based application.

All other programming languages were derived directly or indirectly from C programming concepts.

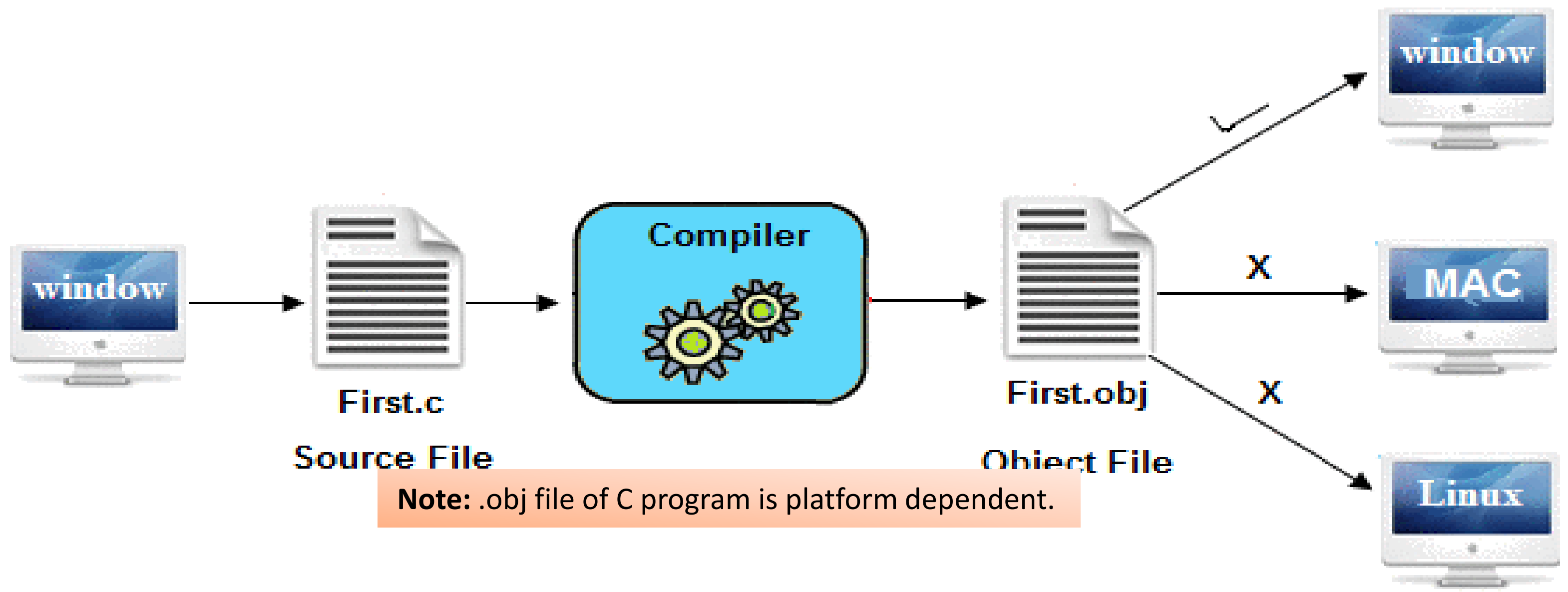


Simple

Every c program can be written in simple English language so that it is very easy to understand and developed by programmer.

Platform dependent

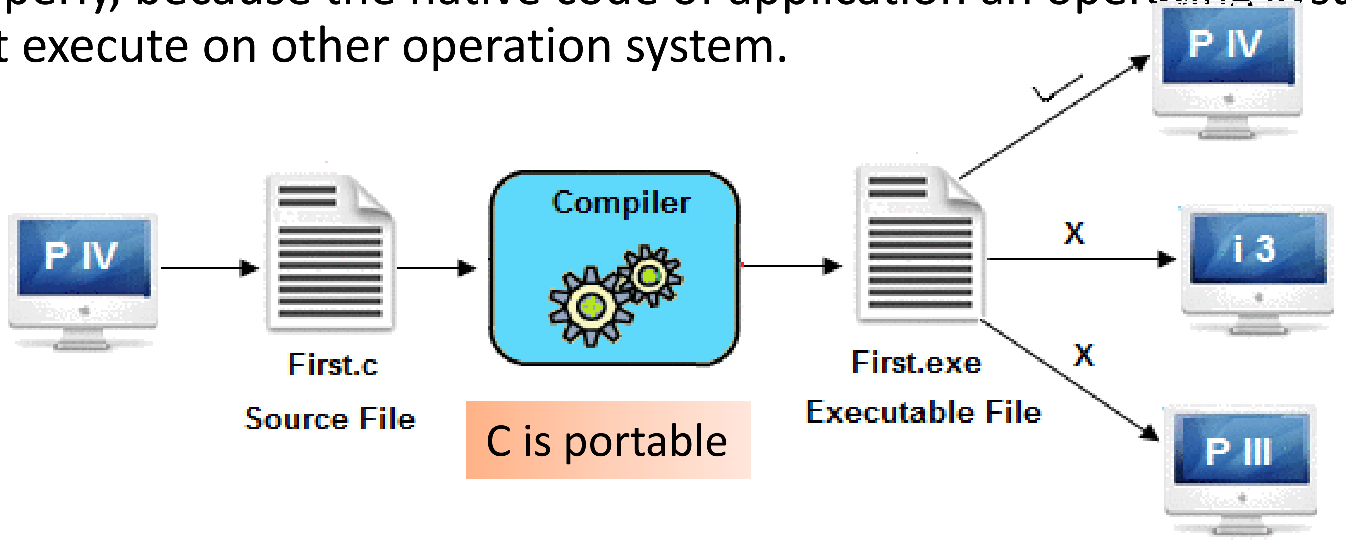
A language is said to be platform dependent whenever the program is execute in the same operating system where that was developed and compiled but not run and execute on other operating system. C is platform dependent programming language.



Note: .obj file of C program is platform dependent.

Portability

- It is the concept of carrying the instruction from one system to another system. In C Language **.c** file contain source code, we can edit also this code. **.exe** file contain application, only we can execute this file. When we write and compile any C program on window operating system that program easily run on other window based system.
- When we can copy .exe file to any other computer which contain window operating system then it works properly, because the native code of application an operating system is same. But this exe file is not execute on other operation system.



Powerful

C is a very powerful programming language, it have a wide verity of data types, functions, control statements, decision making statements, etc.

Structure oriented

C is a Structure oriented programming language.Structure oriented programming language aimed on clarity of program, reduce the complexity of code, using this approach code is divided into sub-program/subroutines. These programming have rich control structure.

Modularity

It is concept of designing an application in subprogram that is procedure oriented approach. In c programming we can break our code in subprogram.

For example we can write a calculator programs in C language with divide our code in subprograms.

Case sensitive

It is a case sensitive programming language. In C programming 'break and BREAK' both are different.

If any language treats lower case latter separately and upper case latter separately than they can be called as **case sensitive** programming language [Example c, c++, java, .net are sensitive programming languages.] other wise it is called as **case insensitive** programming language [Example HTML, SQL is case insensitive programming languages].

Middle level language

C programming language can supports two level programming instructions with the combination of low level and high level language that's why it is called middle level programming language.

Compiler based

C is a compiler based programming language that means without compilation no C program can be executed. First we need compiler to compile our program and then execute.

Syntax based language

C is a strongly tight syntax based programming language. If any language follow rules and regulation very strictly known as strongly tight syntax based language. Example C, C++, Java, .net etc. If any language not follow rules and regulation very strictly known as loosely tight syntax based language.
Example HTML.

Fast & Efficient

Applications of C

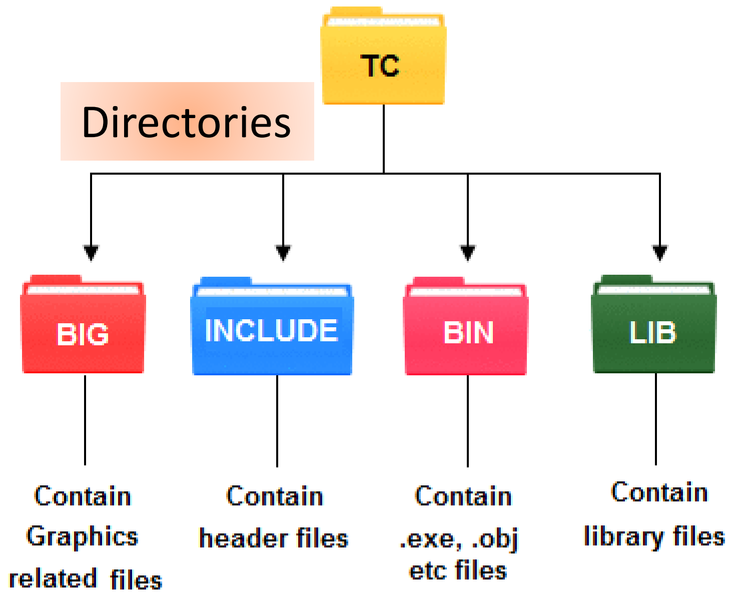
Mainly C Language is used for Develop Desktop application and system software. Some application of C language are given below.

- C programming language can be used to design the system software like operating system and Compiler.
- To develop application software like database and spread sheets.
- For Develop Graphical related application like computer and mobile games.
- To evaluate any kind of mathematical equation use c language.
- C programming language can be used to design the compilers.
- UNIX Kernal is completely developed in C Language.
- **For Creating Compilers** of different Languages which can take input from other language and convert it into lower level machine dependent language.
- C programming language can be used to design Operating System.
- C programming language can be used to design Network Devices.

Installation of TC

Installation of TC is very simple just download turbo C or C++ and run .exe files

When you install the Turbo C compiler on your system, then TC directory is created on the hard disk and various sub directories such as INCLUDE, and LIB etc. are created under TC.



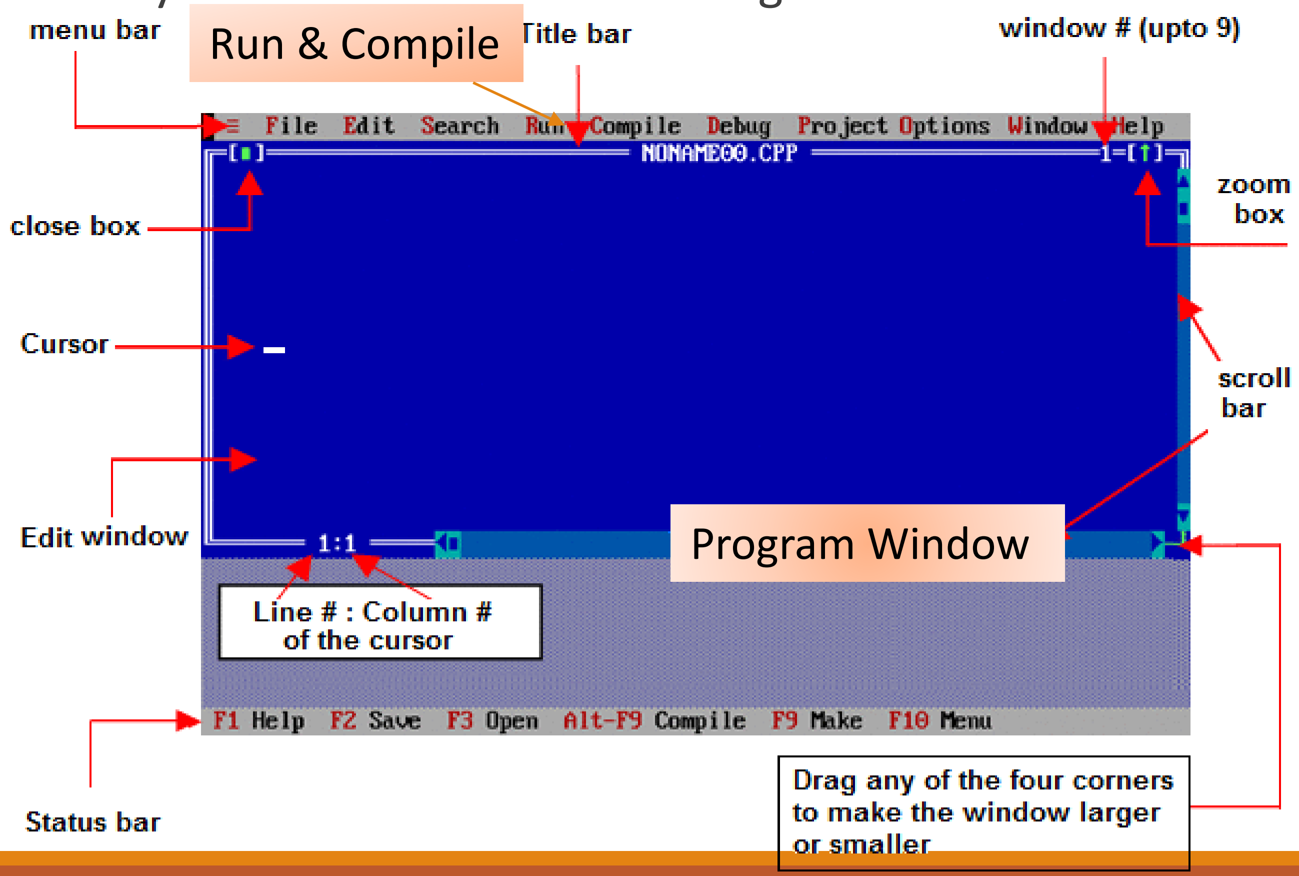
INCLUDE : Contain the header files of C.
LIB: Contain the library files of C.
BGI: Contain Graphics related files.
BIN: Contain .exe, .obj etc files.

TC Editor

TC Editor is very simple and easy to use; here i will give you all tips related to TC Editor and some shortcut keys related to TC Editor which is very useful at the time of coding. Turbo C is a most common C language compiler.

Clt + f9 : Run C Program.

Alt + f9 : Compile C Code.


menu bar
Run & Compile
Title bar
window # (upto 9)
≡ File Edit Search Run Compile Debug Project Options Window Help
[■]
NONAME00.CPP
1=[↑]
zoom box
close box
Cursor
scroll bar
Edit window
1:1
Program Window
Line # : Column # of the cursor
F1 Help F2 Save F3 Open Alt-F9 Compile F9 Make F10 Menu
Status bar
Drag any of the four corners to make the window larger or smaller

Source Code Vs Object Code

Source Code

Source code is in the form of Text form.

Source code is Human Readable Code.

Source code is Generated by Human or Programmer.

Source code is receive Compiler as a Input.

Object Code

Object Code is in the form of Binary Numbers.

Object Code is in Machine Readable formats.

Object Code is Generated by Compiler.

Object Code is Generated by Compiler as a Output.

Comments in C

Generally **Comments** are used to provide the description about the Logic written in program. Comments are not display on output screen.

When we are used the comments, then that specific part will be ignored by compiler.

In 'C' language two types of comments are possible

Single line comments

Multiple line comments

Single line comments

Single line comments can be provided by using / /...................

Multiple line comments

Multiple line comments can be provided by using /*.....................*/

Note: When we are working with the multiple line comments then nested comments are not possible.

Comments

Rules for Writing Comments

1. Program contains any number of comments at any place.

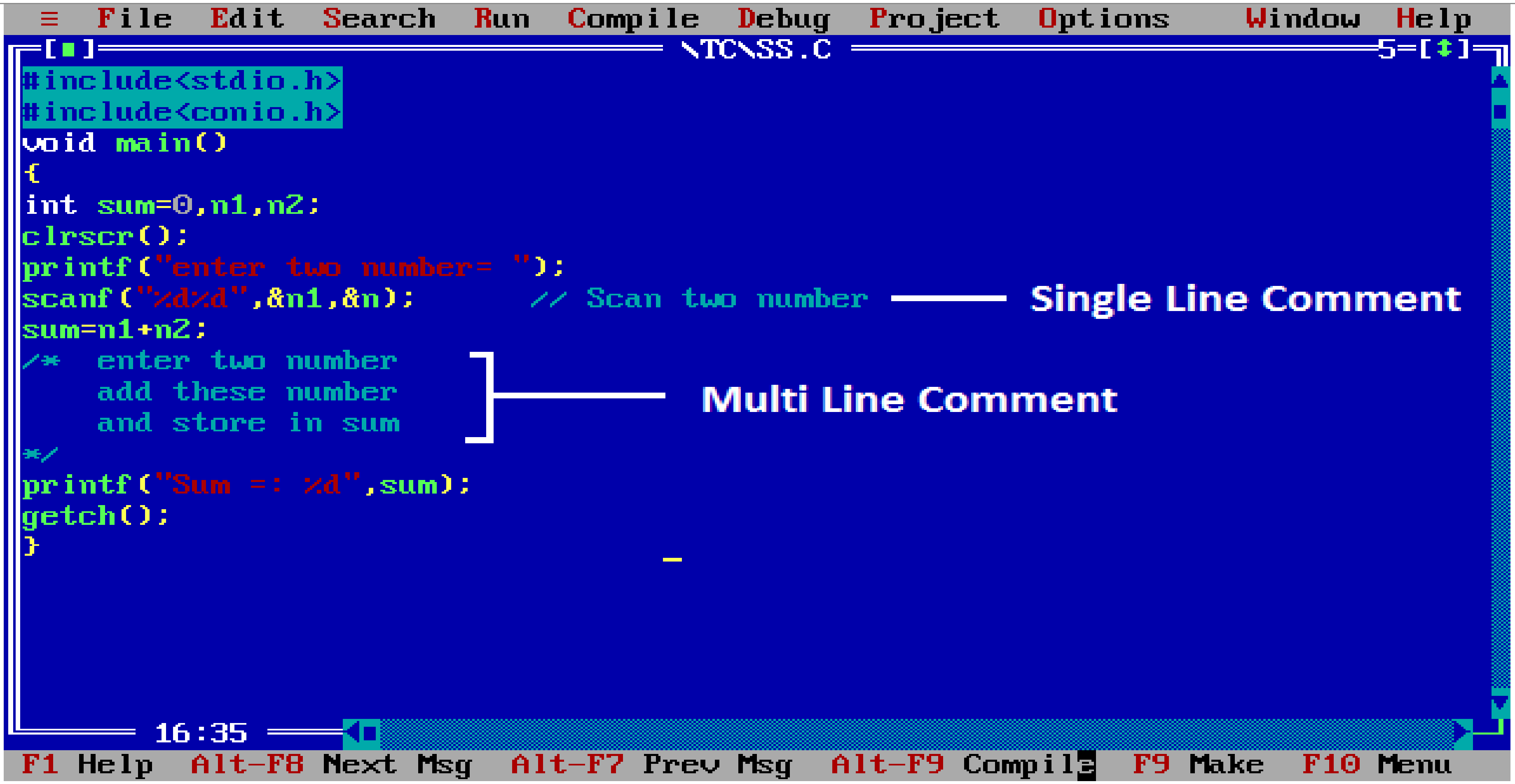
2. Nested Comments are not possible, that means comments within comments.

3. Comments can be splits over more than one line.

4. Comments are not case sensitive.

5. Single line comments start with "//"

Example


≡ File Edit Search Run Compile Debug Project Options Window Help
\TC\SS.C
Single Line Comment
Multi Line Comment
16:35
F1 Help Alt-F8 Next Msg Alt-F7 Prev Msg Alt-F9 Compile F9 Make F10 Menu

```
#include<stdio.h>
#include<conio.h>
void main()
{
int sum=0,n1,n2;
clrscr();
printf("enter two number= ");
scanf("%d%d",&n1,&n);        // Scan two number
sum=n1+n2;
/*   enter two number
     add these number
     and store in sum
*/
printf("Sum =: %d",sum);
getch();
}
```

Character Set

- ❑A character indicates any alphabet, digit or special symbol used to represent the data.
- ❑The characters are used to construct keywords, variables, identifiers, constants and expressions.
  - ❑Alphabets: Uppercase letters A-Z, Lowercase letters a-z
  - ❑Digits: 0-9
  - ❑Special Symbols: ~,!,@,#,$,%,^,&,*,(,),-,_,+,=,\,|,{,},[,],;,:,’,”,<,>,?,?,.

Tokens

- C tokens are defined as the smallest individual units in a C program.
- Tokens in C program can be
  - Keyword
  - Identifier
  - Constant
  - String
  - Symbol
  - Operators

Keywords in C

Keyword is a predefined or reserved word in C library with a fixed meaning and used to perform an internal operation. C Language supports 32 keywords.

Every **Keyword** exists in lower case latter like auto, break, case, const, continue, int etc.

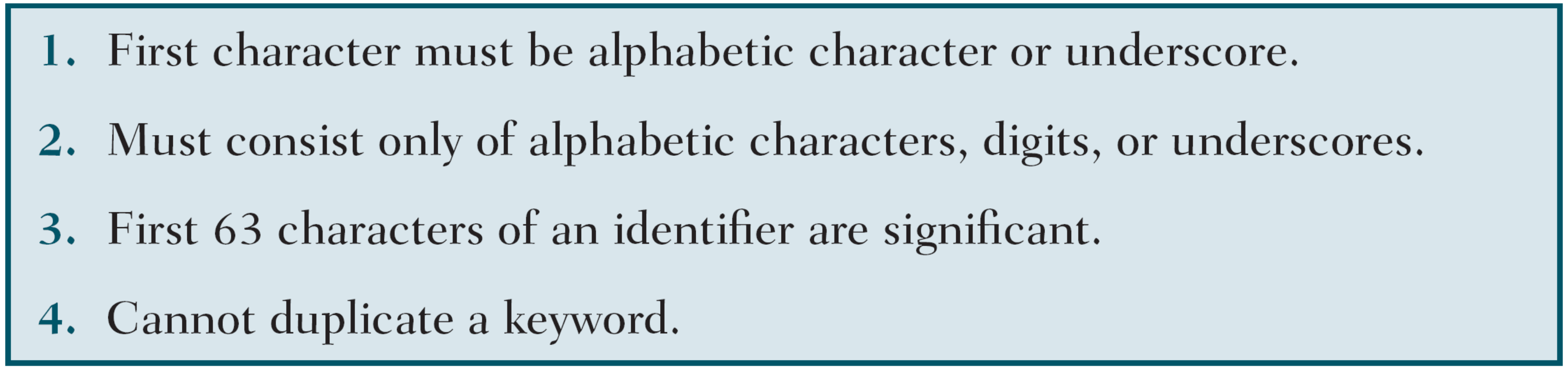
32 Keywords in C Language

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

Identifiers in C

One feature present in all computer languages is the identifier. Identifiers allow us to name data and other objects in the program. Each identified object in the computer is stored at a unique address.

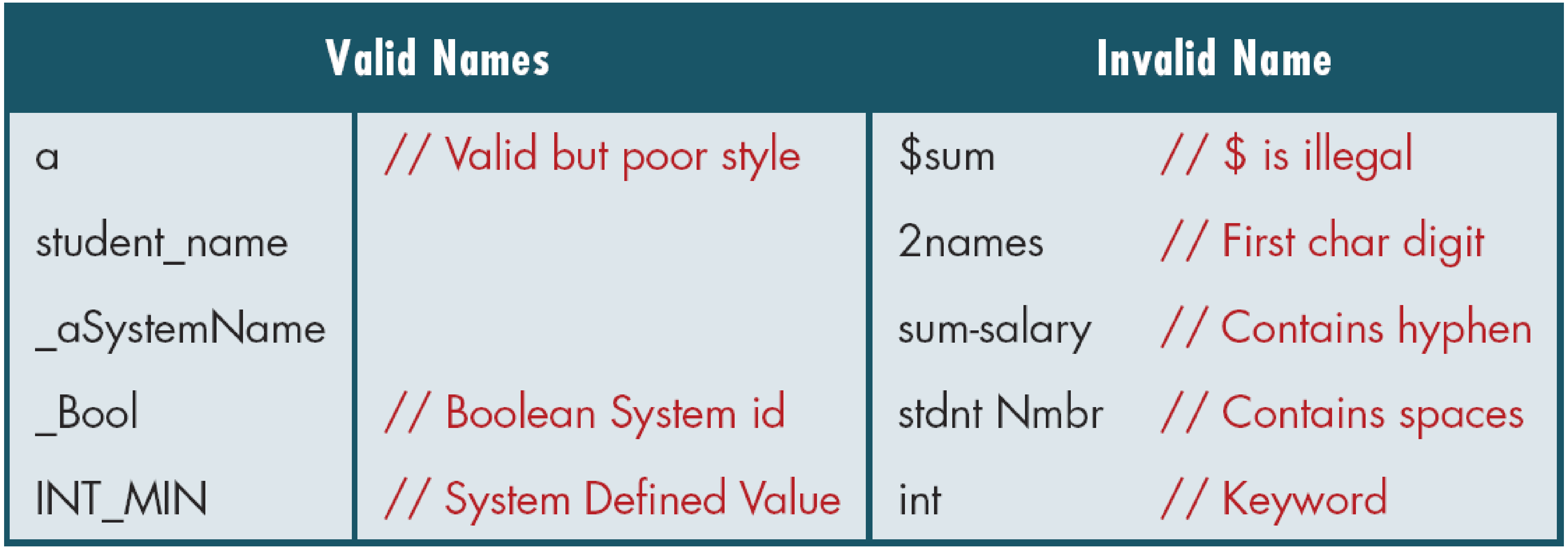
Rules for identifiers:



1. First character must be alphabetic character or underscore.
2. Must consist only of alphabetic characters, digits, or underscores.
3. First 63 characters of an identifier are significant.
4. Cannot duplicate a keyword.

Note: identifiers are case sensitive.

Examples of Valid and Invalid Names

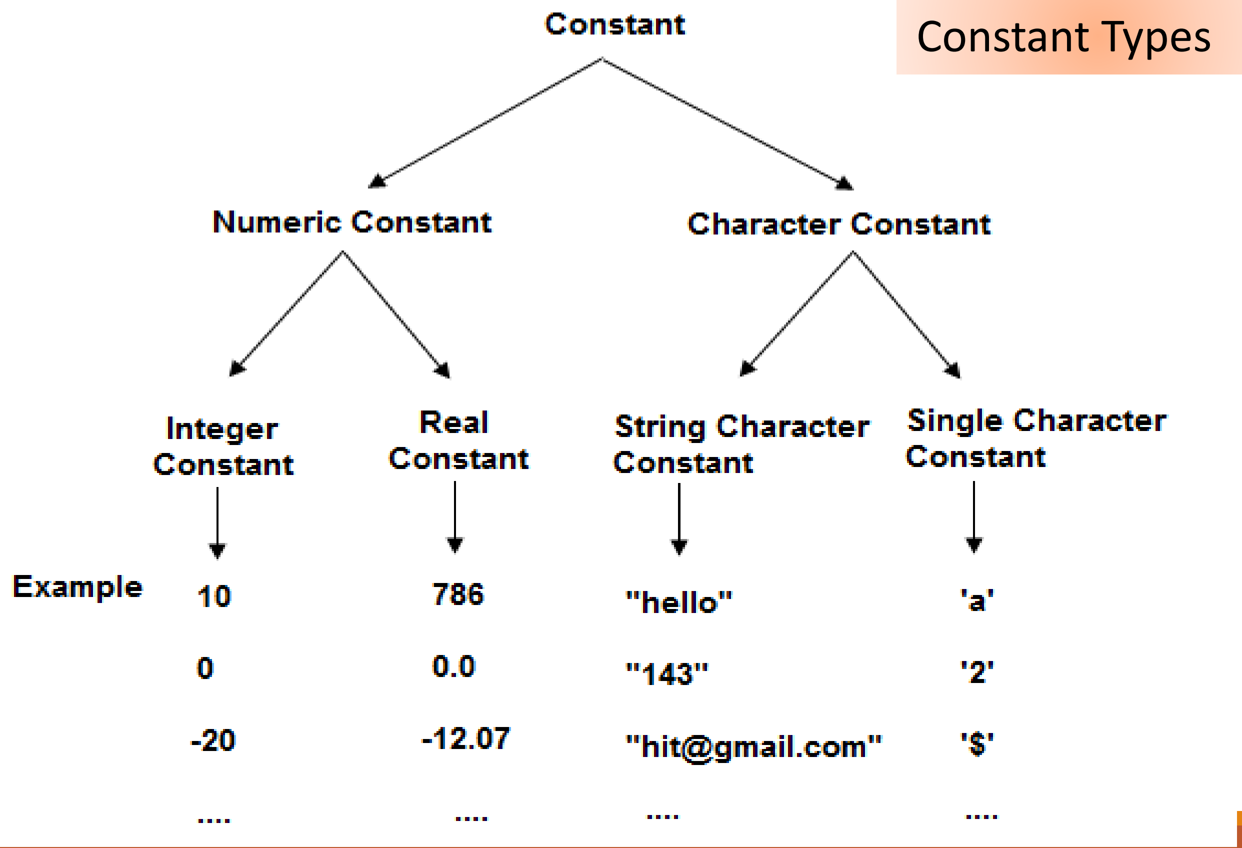


Valid Names		Invalid Name	
a	// Valid but poor style	$sum	// $ is illegal
student_name		2names	// First char digit
_aSystemName		sum-salary	// Contains hyphen
_Bool	// Boolean System id	stdnt Nmbr	// Contains spaces
INT_MIN	// System Defined Value	int	// Keyword

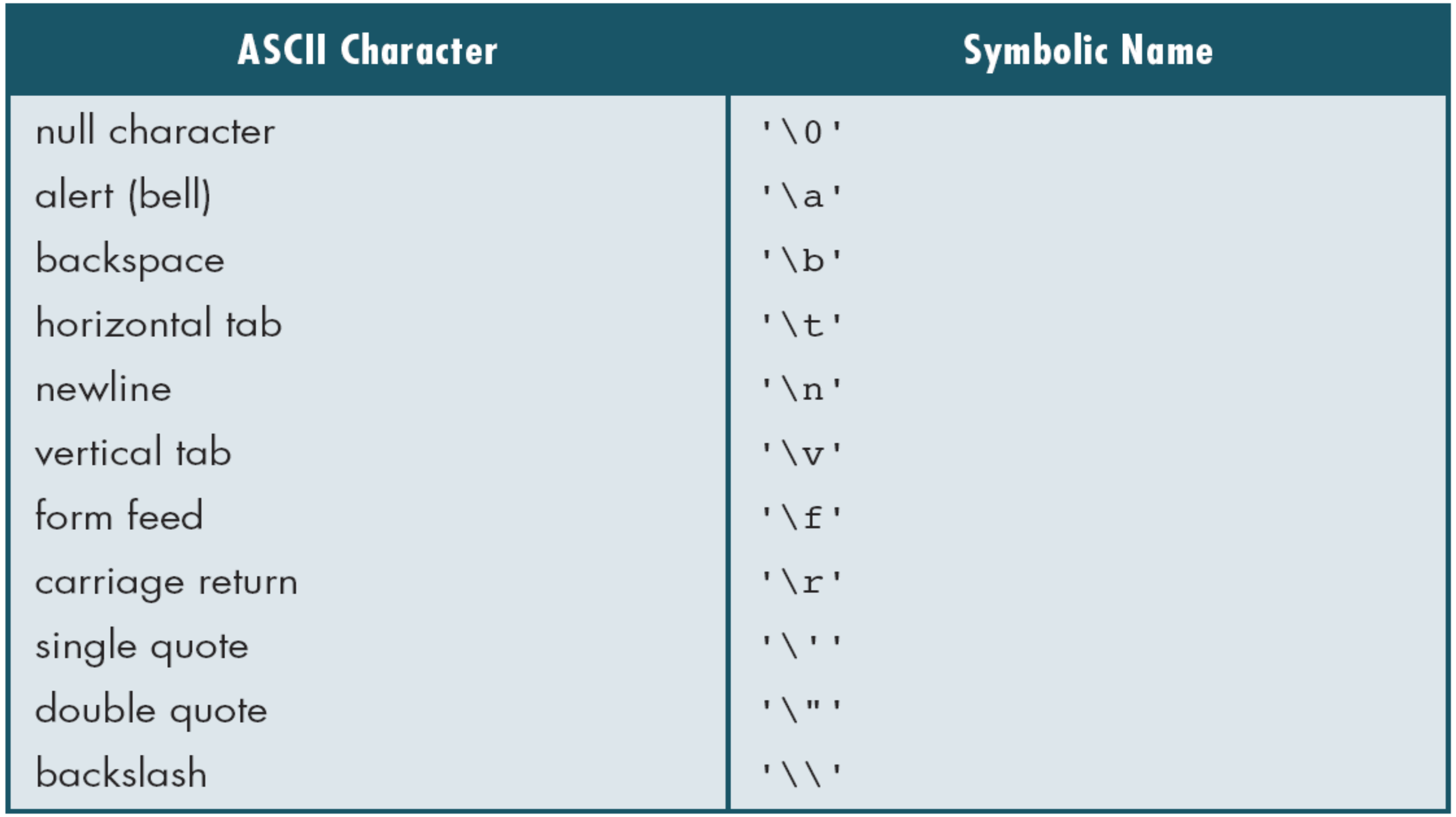
Types Constant in C

It is an identifier whose value can not be changed at the execution time of program. In general **constant** can be used to represent as fixed values in a C program. Constants are classified into following types.

Constant Types



Escape sequence and their meaning

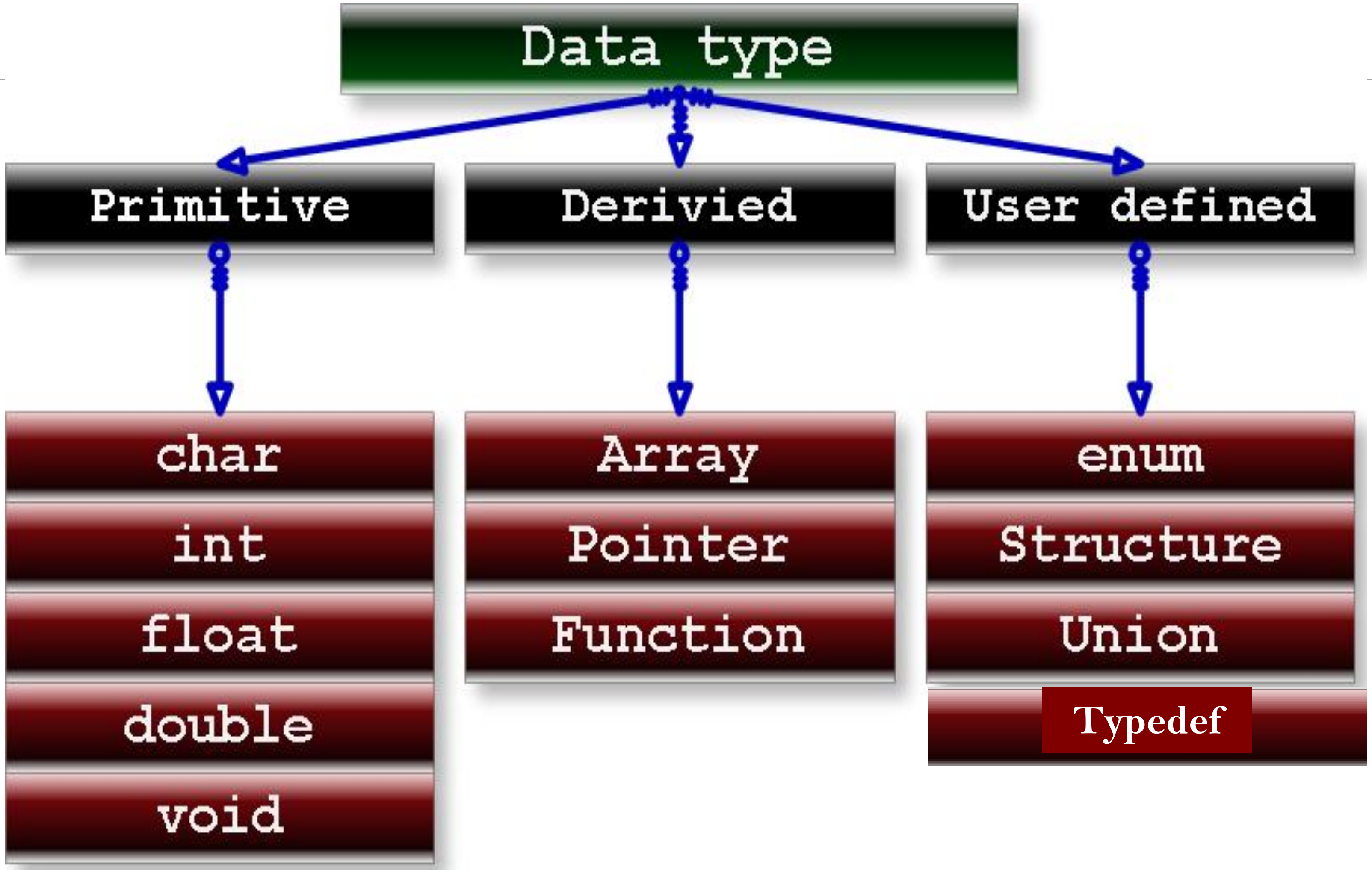


ASCII Character	Symbolic Name
null character	`'\0'`
alert (bell)	`'\a'`
backspace	`'\b'`
horizontal tab	`'\t'`
newline	`'\n'`
vertical tab	`'\v'`
form feed	`'\f'`
carriage return	`'\r'`
single quote	`'\''`
double quote	`'\"'`
backslash	`'\\'`

DataType in C

Data type is a keyword used to identify type of data. Data types are used for storing the input of the program into the main memory (RAM) of the computer by allocating sufficient amount of memory space in the main memory of the computer.

In other words data types are used for representing the input of the user in the main memory (RAM) of the computer.


Data type
Primitive
Derivied
User defined
char
int
float
double
void
Array
Pointer
Function
enum
Structure
Union
Typedef

Integer Type : Used to store whole numbers.

Type	Size (bytes)	Range
int	2	-32,768 to 32,767
unsigned int	2	0 to 65,535
short	2 bytes	-32,768 to 32,767
unsigned short	2 bytes	0 to 65,535
long	4 bytes	-2,147,483,648 to 2,147,483,647
unsigned long	4 bytes	0 to 4,294,967,295

Floating Type : Used to store real numbers

Type	Size (bytes)	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
long double	10 byte	3.4E-4932 to 1.1E+4932	19 decimal places

Character Type : Used to store real character values.

Type	Size (bytes)	Range
char	1 byte	-128 to 127
unsigned char	1 byte	0 to 255
signed char	1 byte	-128 to 127

Void type:

Void type means no value.

This is specially used to specify the type of function.

Derived data types

These data type are derived from fundamental data type. Variables of derived data type allow us to store multiple values of same type in one variable but never allows to store multiple values of different types. These are the data type whose variable can hold more than one value of similar type.

In C language it can be achieve by array.

```
int a[] = {10,20,30}; // valid
 int b[] = {100, 'A', "ABC"}; // invalid
```

User defined data types

User defined data types related variables allows us to store multiple values either of same type or different type or both. This is a data type whose variable can hold more than one value of dissimilar type, in C language it is achieved by structure.

Example:

```
struct emp
{
int id;
char ename[10];
float sal;
};
```

Typedef

- It represents the type definition that allows user to define an identifier that would represent the existing data type.
- This user defined data type identifier can later be used to declare the variables.

Syntax:

```
typedef type identifier;
```

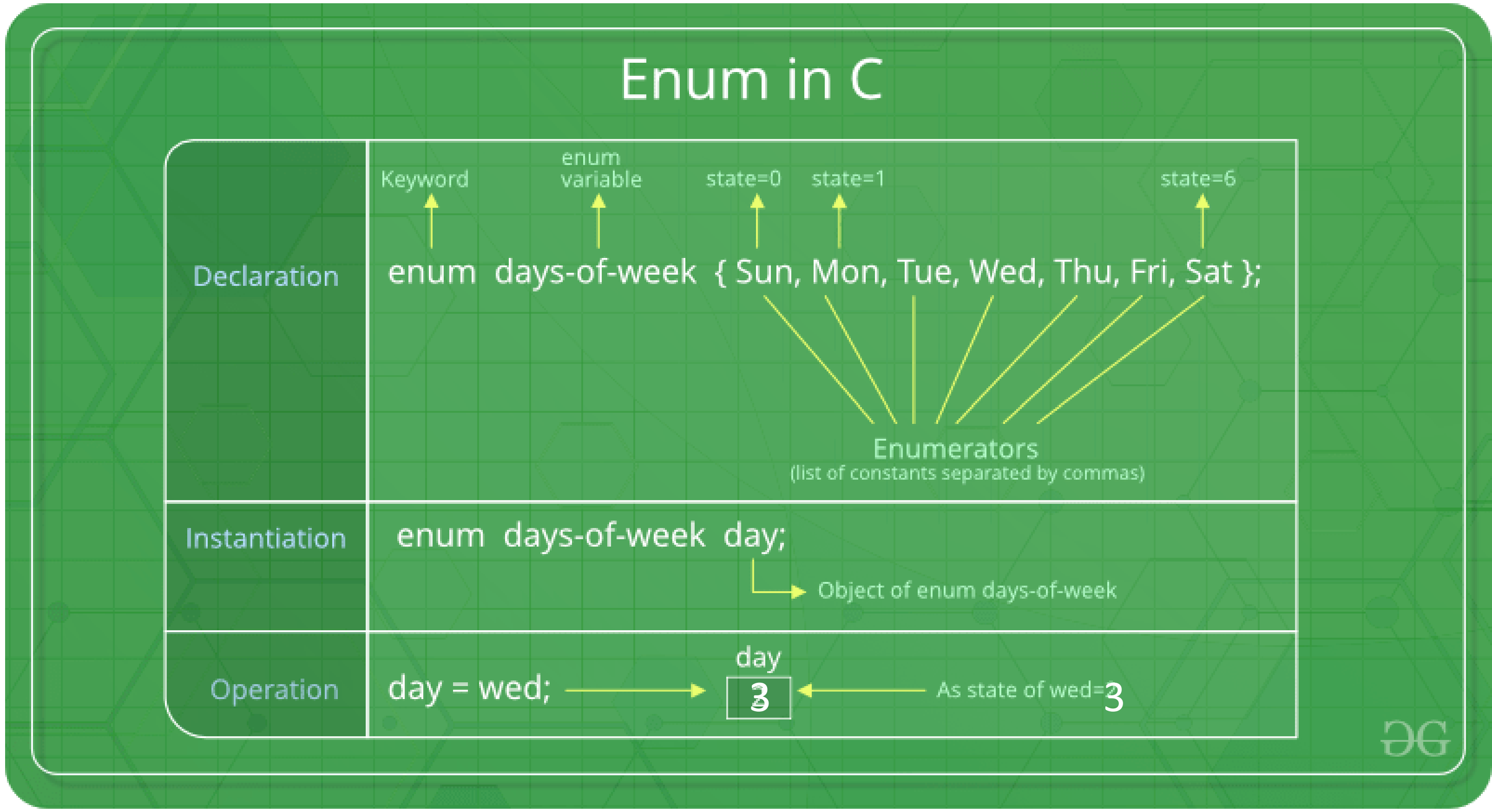
Example:

```
typedef float price;

price silver, gold;
```

enum

- It is mainly used to assign names to integral constants, the names make a program easy to read and maintain
- An enumeration is a list of constant values.
- It is used to declare variables that can have one of the values enclosed within the braces.
- These values are also called as enumeration constants.



enum

Example

```
void main()
{
enum day{Mon,Tues,Wed,Thu,Fri,Sat,Sun};
printf("%d",Fri);
getch();
}
```

a) 5 b) Error c) 4 d) Fri

Variable in C

Variable is an identifier which holds data or another one variable. It is an identifier whose value can be changed at the execution time of program. It is used to identify input data in a program.

Syntax:

```
Variable_name = value;
```

Rules to declare a Variable

To Declare any variable in C language you need to follow rules and regulation of C Language, which is given below;

- Every variable name should start with alphabets or underscore (_).
- No spaces are allowed in variable declaration.
- Except underscore (_) no other special symbol are allowed in the middle of the variable declaration (not allowed -> roll-no, allowed -> roll_no).
- Maximum length of variable is 8 characters depend on compiler and operation system.
- Every variable name always should exist in the left hand side of assignment operator (invalid -> 10=a; valid -> a=10;).
- No keyword should access variable name (int for <- invalid because for is keyword).
- **Note:** In a c program variable name always can be used to identify the input or output data.

If

Variable declarations

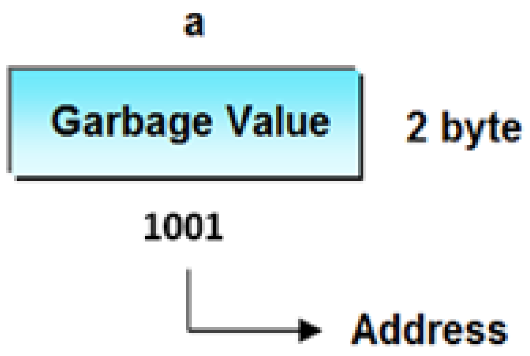
This is the process of allocating sufficient memory space for the data in term of variable.

Syntax

Datatype variable_name;

Example:

int a;



memory

If no input values are assigned by the user than system will gives a default value called garbage value.

Garbage value

Garbage value can be any value given by system and that is no way related to correct programs. This is a disadvantage of C programming language and in C programming it can overcome using variable initialization.

Variable initialization

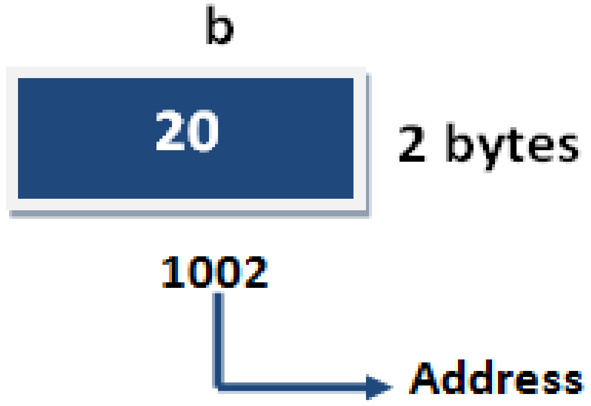
It is the process of allocating sufficient memory space with user defined values.

Syntax:

Datatype nariable_name=value;

Example:

int b = 20;



Scope of variable

Difference between Local variable and Global variable

In C language, a variable can be either of global or local scope.

Global variable

Global variables are defined outside of all the functions, generally on top of the program. The global variables will hold their value throughout the life-time of your program.

Local variable

A local variable is declared within the body of a function or a block. Local variable only use within the function or block where it is declare.

Example

```
#include<stdio.h>

#include<conio.h>

int a;   // global variable

void main()

{

int b;    // local variable

a=10, b=20;

printf("Value of a : %d",a);

printf("Value of b : %d",b);

getch();

}
```

```
Output:
Value of a: 10
Value of b: 20
```

Structure of C program

C program must have at least one function called as main() function has control over the other part of the program.

Execution of the program start from the main() function.

Structure of C program:

Documentation
Header files
Constant and global variables

Main()
{
Statement(s)
}

User defined functions and procedures with their body

C Program example

```
/* write a program to print
 “hello world”*/
#include<stdio.h>
void main()
{
printf(“hello world!”);
}
```

Output:

```
Hello world!
```

Operators

Operator is a special symbol that tells the compiler to perform specific mathematical or logical Operation.

1) Arithmetic Operators

2) Relational Operators

3) Logical Operators

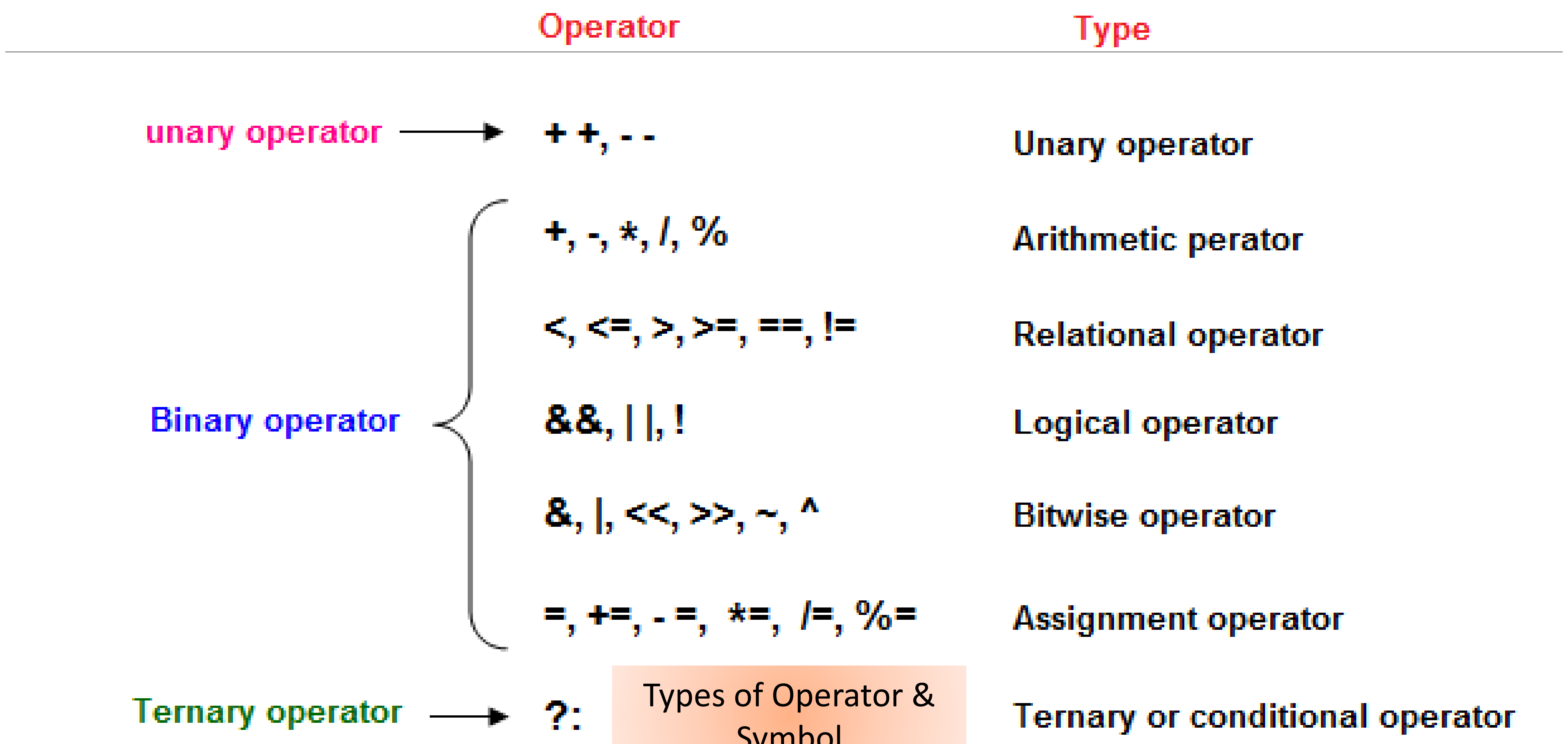
4) Bitwise Operators

5) Assignment Operators

6) Ternary or Conditional Operators

7) Increment & Decrement Operator (Unary Operator)

8) Size of operator



	Operator	Type
unary operator	++, - -	Unary operator
Binary operator	+, -, *, /, %	Arithmetic perator
	<, <=, >, >=, ==, !=	Relational operator
	&&, ||, !	Logical operator
	&, |, <<, >>, ~, ^	Bitwise operator
	=, +=, - =, *=, /=, %=	Assignment operator
Ternary operator	?:	Ternary or conditional operator

Types of Operator & Symbol

Arithmetic Operators

Given table shows all the Arithmetic operator supported by C Language. Lets suppose variable **A**hold 8 and **B** hold 3.

Operator	Example (int A=8, B=3)	Result
+	A+B	11
-	A-B	5
*	A*B	24
/	A/B	2
%	A%4	0

Relational Operators

Which can be used to check the Condition, it always return true or false. Lets suppose variable **A**hold 8 and **B** hold 3.

Operators	Example (int A=8, B=3)	Result
<	A<B	False
<=	A<=10	True
>	A>B	True
>=	A<=B	False
==	A== B	False
!=	A!=(-4)	True

Logical Operator

Which can be used to combine more than one Condition?. Suppose you want to combined two conditions **A<B** and **B>C**, then you need to use **Logical Operator** like (A<B) && (B>C). Here **&&**is Logical Operator.

Operator	Example (int A=8, B=3, C=-10)	Result
&&	(A<B) && (B>C)	False
||	(B!=-C) || (A==B)	True
!	!(B<=-A)	True

Truth table of Logical Operator

C1	C2	C1 && C2	C1 || C2	!C1	!C2
T	T	T	T	F	F
T	F	F	T	F	T
F	T	F	T	T	F
F	F	F	F	T	T

T- True
F- False

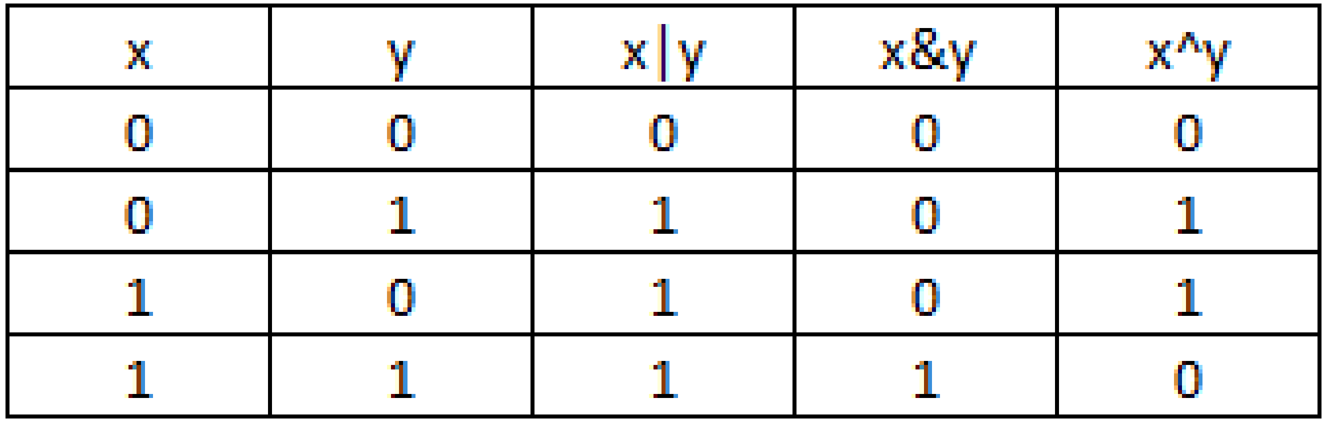
Bitwise operator

These operators are used to perform bit operations. Decimal values are converted into binary values which are the sequence of bits and bit wise operators work on these bits.

Bit wise operators in C language are

& (bitwise AND), | (bitwise OR), ~ (bitwise NOT), ^ (XOR), << (left shift) and >> (right shift).

TRUTH TABLE FOR BIT WISE OPERATION & BIT WISE OPERATORS:



x	y	x|y	x&y	x^y
0	0	0	0	0
0	1	1	0	1
1	0	1	0	1
1	1	1	1	0

& (bitwise AND) Takes two numbers as operands and does AND on every bit of two numbers. The result of AND is 1 only if both bits are 1.

| (bitwise OR) Takes two numbers as operands and does OR on every bit of two numbers. The result of OR is 1 any of the two bits is 1.

^ (bitwise XOR) Takes two numbers as operands and does XOR on every bit of two numbers. The result of XOR is 1 if the two bits are different.

<< (left shift) Takes two numbers, left shifts the bits of the first operand, the second operand decides the number of places to shift.

>> (right shift) Takes two numbers, right shifts the bits of the first operand, the second operand decides the number of places to shift.

~ (bitwise NOT) Takes one number and inverts all bits of it

Bitwise AND operator &

The output of bitwise AND is 1 if the corresponding bits of two operands is 1. If either bit of an operand is 0, the result of corresponding bit is evaluated to 0.

Let us suppose the bitwise AND operation of two integers 12 and 25.

12 = 00001100 (In Binary)

25 = 00011001 (In Binary)

Bit Operation of 12 and 25

```
  00001100
& 00011001
  ________
  00001000  = 8 (In decimal)
```

Bitwise OR operator |

The output of bitwise OR is 1 if at least one corresponding bit of two operands is 1. In C Programming, bitwise OR operator is denoted by |.

12 = 00001100 (In Binary)

25 = 00011001 (In Binary)

Bitwise OR Operation of 12 and 25

```
  00001100
| 00011001
  ________
  00011101  = 29 (In decimal)
```

Bitwise XOR (exclusive OR) operator ^

The result of bitwise XOR operator is 1 if the corresponding bits of two operands are opposite. It is denoted by ^.

12 = 00001100 (In Binary)

25 = 00011001 (In Binary)

Bitwise XOR Operation of 12 and 25

```
  00001100
^ 00011001
  ________
  00010101  = 21 (In decimal)
```

Bitwise complement operator ~

Bitwise compliment operator is an unary operator (works on only one operand). It changes 1 to 0 and 0 to 1. It is denoted by ~.

35 = 00100011 (In Binary)

Bitwise complement Operation of 35

```
~ 00100011
  ________
  11011100  = 220 (In decimal)
```

Left Shift Operator

Left shift operator shifts all bits towards left by certain number of specified bits. It is denoted by <<.

15 = 0000 1111 (In binary)

15<<1 = _____________ (In binary) [Left shift by one bit] =_____________(decimal)

15<<0 =____________ (Shift by 0) =_____________(decimal)

15<<4 = _____________ (In binary) = ___________ (In decimal)

Right Shift Operator

Right shift operator shifts all bits towards right by certain number of specified bits. It is denoted by >>.

15 = _____________ (In binary)

15>>2 = ___________(In binary) [Right shift by two bits]=_____________(decimal)

15>>7 = _____________(In binary) =_____________(decimal)

15>>8 = ____________ =_____________(decimal)

15>>0 = _______________(No Shift) =_____________(decimal)

Assignment operators

Which can be used to assign a value to a variable. Lets suppose variable **A** hold 8 and **B** hold 3.

Operator	Example (int A=8, B=3)	Result
+=	A+=B or A=A+B	11
-=	A-=3 or A=A-3	5
*=	A*=7 or A=A*7	56
/=	A/=B or A=A/B	2
%=	A%=5 or A=A%5	3
=a=b	Value of b will be assigned to a	

Ternary Operator (conditional operator)

If any operator is used on three operands or variable is known as **Ternary Operator**. It can be represented with **? :** . It is also called as **conditional operator**

Advantage of Ternary Operator

Using **?:** reduce the number of line codes and improve the performance of application.

Syntax:

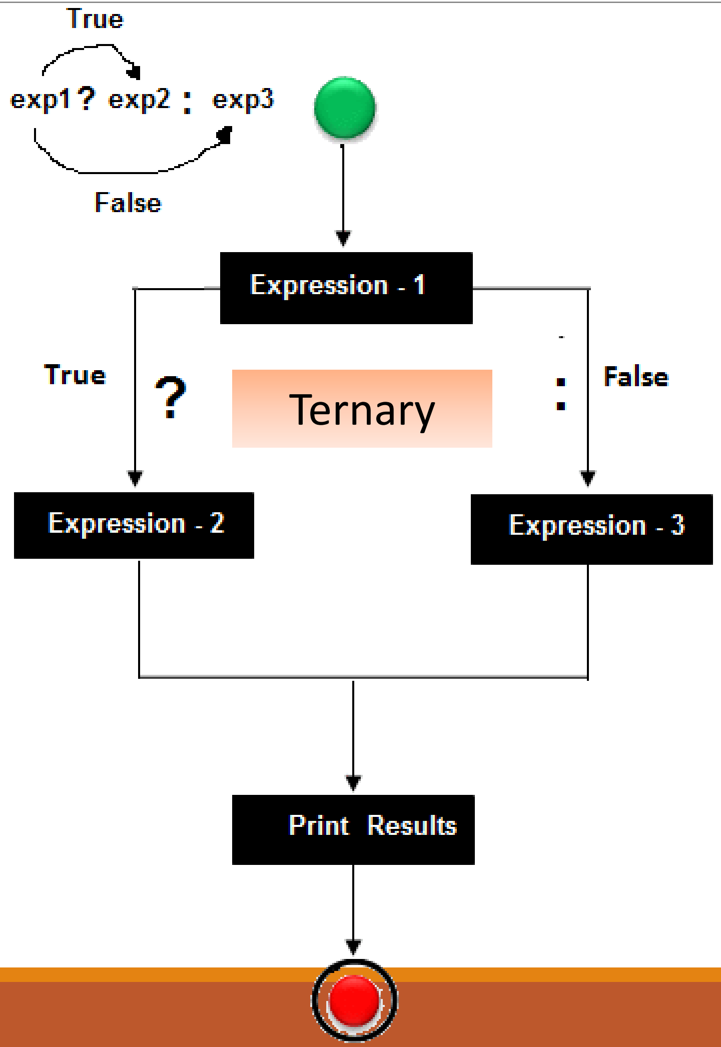
expression-1 ? expression-2 : expression-3

In the above symbol expression-1 is condition and expression-2 and expression-3 will be either value or variable or statement or any mathematical expression. If condition will be true expression-2 will be execute otherwise expression-3 will be executed.

Example:

```
a<b ? printf("a is less") : printf("a is greater");
```

Flow Diagram



Find largest number among 3 numbers using ternary operator

```
#include<stdio.h>

#include<conio.h>

void main()

{

int a, b, c, large;

clrscr();

printf("Enter any three number: ");

scanf("%d%d%d",&a,&b,&c);

large=a>b ? (a>c?a:c) : (b>c?b:c);

printf("Largest Number is: %d",large);

getch();

}
```

Increment and Decrement Operator in C

Increment Operators are used to increased the value of the variable by one and **Decrement Operators** are used to decrease the value of the variable by one in C programs.

Both increment and decrement operator are used on a single operand or variable, so it is called as a unary operator. Unary operators are having higher priority than the other operators it means unary operators are executed before other operators.

Syntax:

```
++ // increment operator
 -- // decrement operator
```

Example

Increment and decrement operators are can not apply on constant.

```
x= 4++; // gives error, because 4 is constant
```

Type of Increment Operator
pre-increment
post-increment

pre-increment (++ variable)

In pre-increment first increment the value of variable

and then used inside the expression

(initialize into another variable).

```
#include<stdio.h>
#include<conio.h>
void main()
 {
 int x,i;
 i=10;
 x=++i;
 printf("x: %d",x);
 printf("i: %d",i);
 getch();
}
```

Output:

```
x: 11
i: 11
```

post-increment (variable ++)

In post-increment first value of variable is used in the expression (initialize into another variable) and then increment the value of variable.

```
#include<stdio.h>
#include<conio.h>
 void main()
 {
 int x,i;
 i=10;
 x=i++;
 printf("x: %d",x);
 printf("i: %d",i);
getch();
 }
Output:
x: 10
i: 11
```

Pre-decrement (-- variable)

In pre-decrement first decrement the value of variable and then used inside the expression (initialize into another variable).

```
#include<stdio.h>
#include<conio.h>

void main()
{
int x,i;
i=10;
x=--i;
printf("x: %d",x);
printf("i: %d",i);
getch();
```

```
Output:
x: 9
 i: 9
```

post-decrement (variable --)

In Post-decrement first value of variable is used in the expression (initialize into another variable) and then decrement the value of variable.

```
#include<stdio.h>
#include<conio.h>

void main()
{
int x,i;
i=10;
x=i--;
printf("x: %d",x);
printf("i: %d",i);
getch();
}
```

Output:

```
x: 10
i: 9
```

Example of increment and decrement operator

```
#include<stdio.h>
#include<conio.h>
void main()
{
int x,a,b,c;
a = 2;
b = 4;
c = 5;
x = a-- + b++ - ++c;
printf("x: %d",x);
getch();
}
```

sizeof operator

To get the exact size of a type or a variable on a particular platform, you can use the **sizeof** operator. The expressions *sizeof(type)* yields the storage size of the object or type in bytes. Given below is an example to get the size of int type on any machine −

Syntax:

sizeof(variable);

Example:

```
#include<stdio.h>
#include<conio.h>
void main()
{
int a;
float b;
double c;
char d;
printf("Size of Integer: %d bytes\n",sizeof(a));
printf("Size of float: %d bytes\n",sizeof(b));
printf("Size of double: %d bytes\n",sizeof(c));
printf("Size of character: %d byte\n",sizeof(d));
getch();
}
```

Type casting (Type conversion)

- **Type casting** is a way to convert a variable from one data type to another data type. For example, if you want to store a long value into a simple integer then you can typecast long to int. You can convert values from one type to another explicitly using the cast operator. There are two types of **type casting** in c language that are Implicit conversions and Explicit Conversions.
- When compiler automatically convert data type of the data then it is called as a implicit type conversion
- When type casting forcefully converts the value of one type into another type then it is called as a explicit type conversion.

Syntax: (type) expression;

Example:

1) x=(int)10.45;

2) int a=34; z=(float)a;

Expressions Evaluation

In c language expression evaluation is mainly depends on priority and associativity.

Priority

This represents the evaluation of expression starts from "what" operator.

Associativity

It represents which operator should be evaluated first if an expression is containing more than one operator with same priority.

Operator	Priority (precedence)	Associativity
{}, (), []	1	Left to right
++, --, !	2	Right to left
*, /, %	3	Left to right
+, -	4	Left to right
<, <=, >, >=, ==, !=	5	Left to right
&&	6	Left to right
||	7	Left to right
?:	8	Right to left
=, +=, -=, *=, /=, %=	9	Right to left

I/O Functions (Input/ Output Functions)

We can classify I/O function into two categories:

1) Console I/O function 2) File I/O function

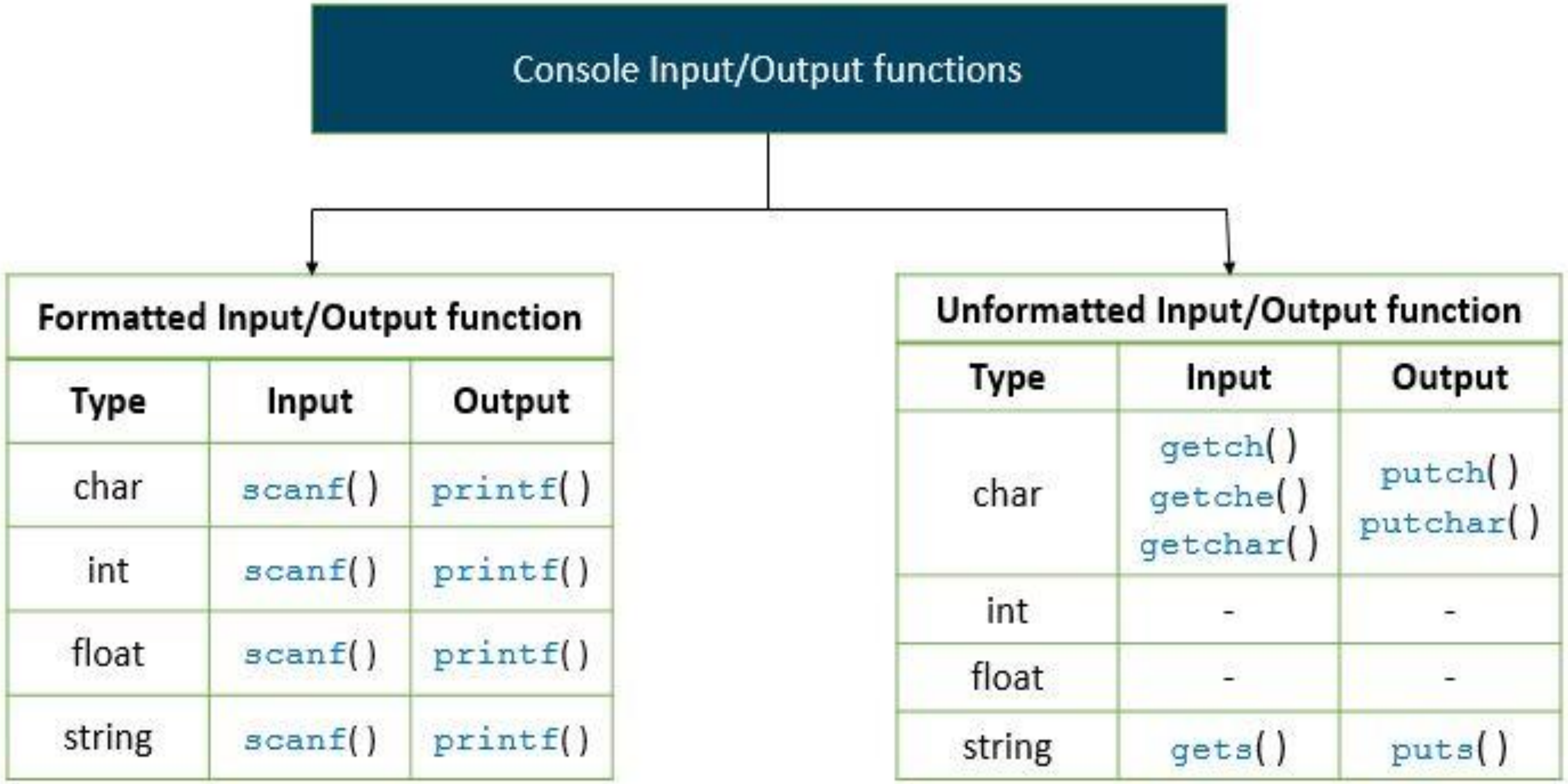
1) Console Input-Output functions

Console simply means screen and keyboard. There are two types of a console I/O functions:

(a) Formatted input-output function

(b) Unformatted input-output function

The major difference is that formatted function allows us to format the input from the keyboard and the output to be displayed on the screen.



Formatted Input/Output function

Type	Input	Output
char	`scanf()`	`printf()`
int	`scanf()`	`printf()`
float	`scanf()`	`printf()`
string	`scanf()`	`printf()`

Unformatted Input/Output function

Type	Input	Output
char	`getch()` `getche()` `getchar()`	`putch()` `putchar()`
int	-	-
float	-	-
string	`gets()`	`puts()`

Formatted i/o function

Printf()

Printf is a predefined function in "stdio.h" header file, by using this function, we can print the data or user defined message on console or monitor. While working with printf(), it can take any number of arguments but first argument must be within the double cotes (" ") and every argument should separated with comma (,) Within the double cotes, whatever we pass, it prints same, if any format specifies are there, then that copy the type of value. The scientific name of the monitor is called console.

Syntax:

printf(format-control-string, other-arguments);

Example:

```
float a ;  int b ;
scanf ( “%f%d”, &a, &b ) ;
printf ( “You entered %f and %d \n”, a, b ) ;
```

- Format control string in printf( ) function describes the output format which consists of conversion specifiers, precisions, flags, field widths and literal characters.
- Each conversion specifier starts with % sign and ends with a conversion specifier.

Format specifier:

Format Specifier

Format specifier	Type of value
%d	Integer
%f	Float
%lf	Double
%c	Single character
%s	String
%u	Unsigned int
%ld	Long int
%lf	Long double

Example:

1) printf("%2d",2019);

		2	0	1	9

2) a=25.551

printf("%5.2f,a);

2	5	.	5	5

scanf()

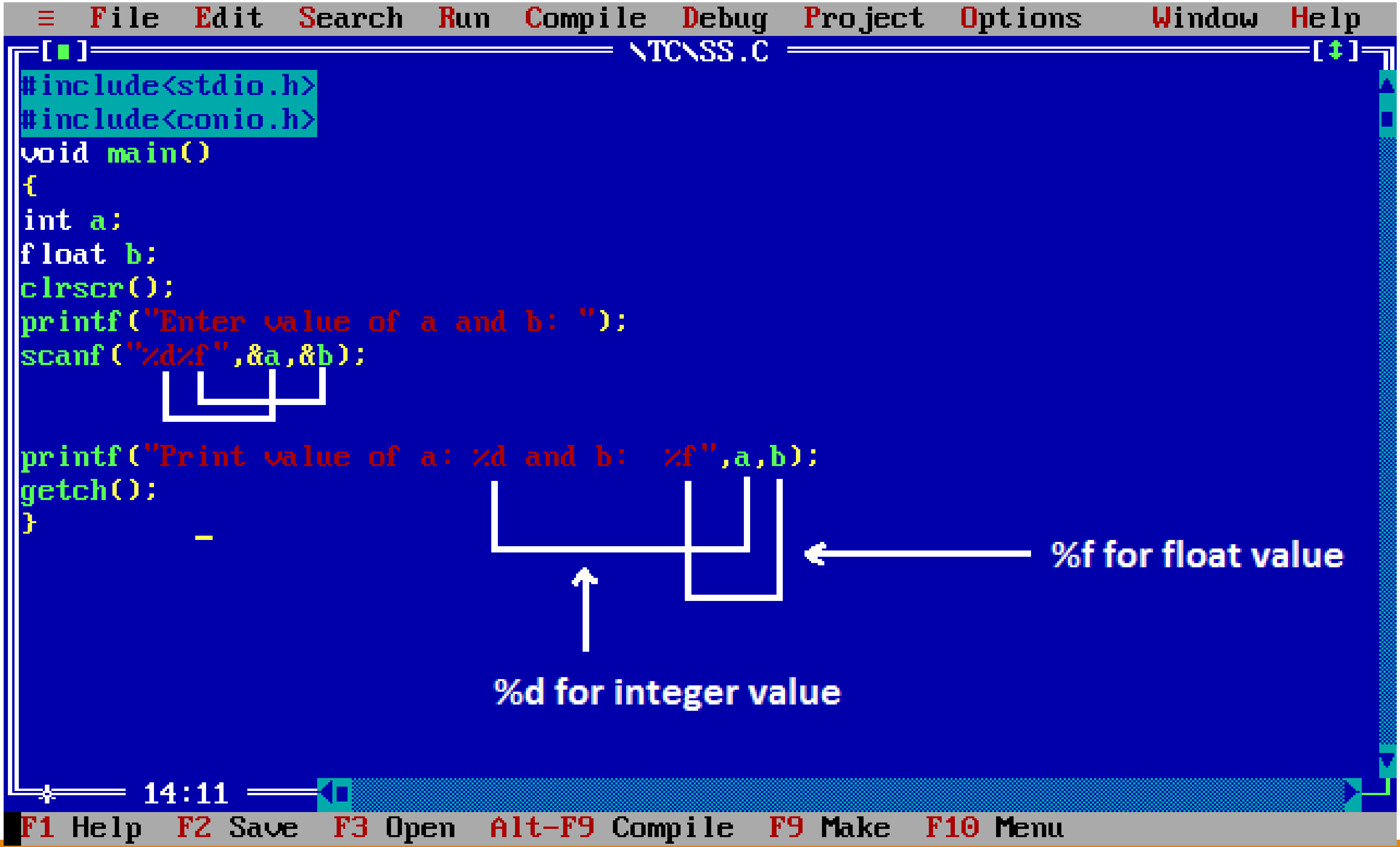
- scanf() is a predefined function in "stdio.h" header file. It can be used to read the input value from the keyword.
- scanf( ) is the standard library function that is used for precise input formatting.

Syntax:

scanf(format-control-string, other-arguments);

Example:

```
float a;  int b;
scanf (“%f%d”, &a, &b);
```


≡ File Edit Search Run Compile Debug Project Options Window Help
\TC\SS.C
14:11
F1 Help F2 Save F3 Open Alt-F9 Compile F9 Make F10 Menu

```
#include<stdio.h>
#include<conio.h>
void main()
{
int a;
float b;
clrscr();
printf("Enter value of a and b: ");
scanf("%d%f",&a,&b);

printf("Print value of a: %d and b:  %f",a,b);
getch();
}
```

%f for float value

%d for integer value

Unformatted i/o functions

getchar ( ) ;

This function provides for getting exactly one character from the keyboard.

Example:

```
char ch;
ch = getchar ( ) ;
```

putchar (char) ;

This function provides for printing exactly one character to the screen.

Example:

```
char ch;
ch = getchar ( ) ;  /* input a character from kbd*/
putchar (ch) ;      /* display it on the screen */
```

getch()

It is a predefined function in "conio.h" (console input output header file) will tell to the console wait for some time until a key is hit given after running of program.

By using this function we can read a character directly from the keyboard. Generally getch() are placing at end of the program after printing the output on screen.

putch()

putch displays any alphanumeric characters to the standard output device. It displays only one character at a time.

Syntax Declaration:

```
putch(variable_name);
```

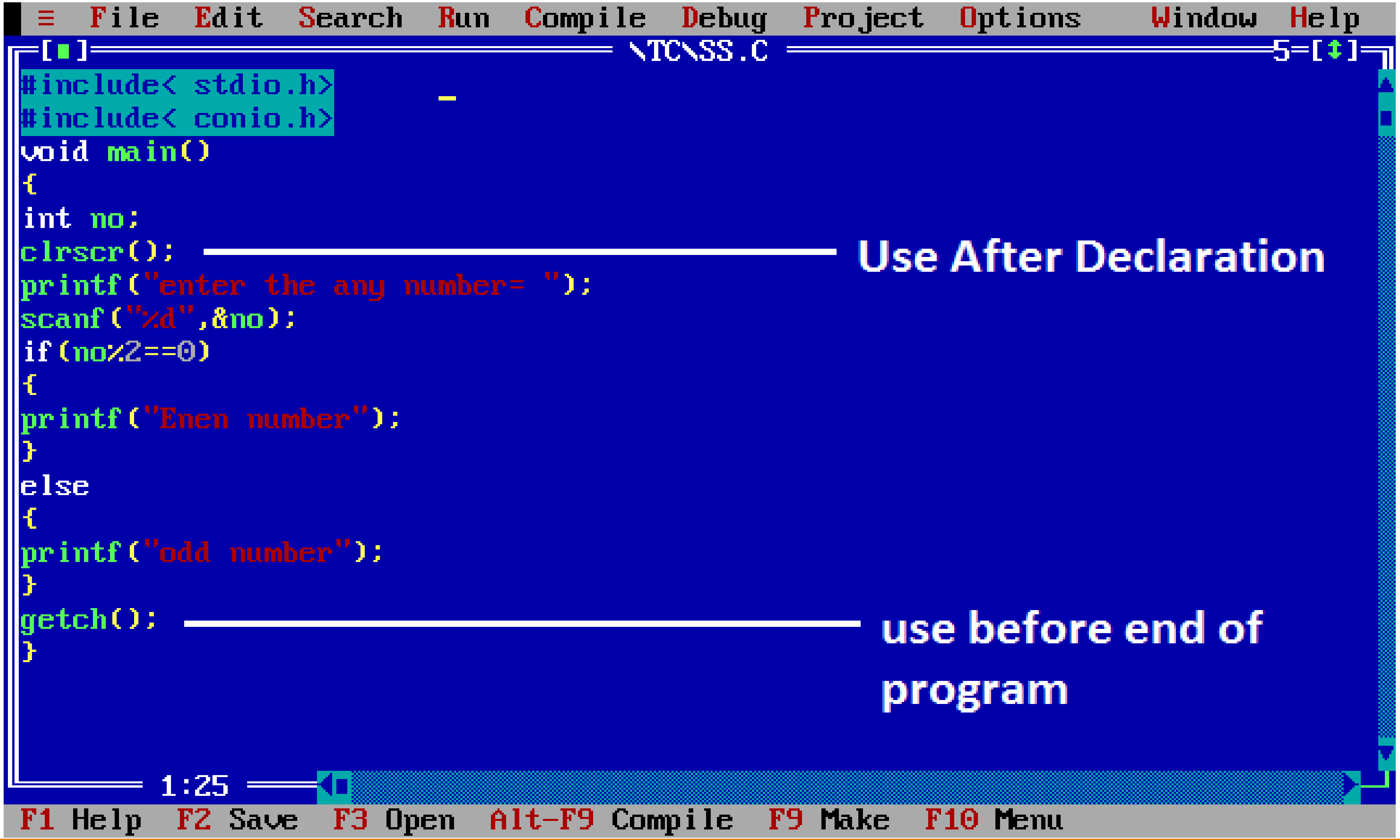
Example Declaration:

```
char ch = 'a';

putch(ch);
```

clrscr()

It is a predefined function in "conio.h" (console input output header file) used to clear the console screen. It is a predefined function, by using this function we can clear the data from console (Monitor). Using of clrscr() is always optional but it should be place after variable or function declaration only.


≡ File Edit Search Run Compile Debug Project Options Window Help
\TC\SS.C
5
1:25
F1 Help F2 Save F3 Open Alt-F9 Compile F9 Make F10 Menu

```
#include< stdio.h>
#include< conio.h>
void main()
{
int no;
clrscr();
printf("enter the any number= ");
scanf("%d",&no);
if(no%2==0)
{
printf("Enen number");
}
else
{
printf("odd number");
}
getch();
}
```

Use After Declaration

use before end of program

String i/o function

gets

This is a predefined function in C language which is available in stdio.h header file. This function is used to read a single string from keyboard.

Syntax:

gets(string);

puts

puts is same as gets function but this is used to display a string on screen or console. This function takes single arguments.

Syntax:

puts(string);