IQBAL INSTITUTE OF TECHNOLOGY AND MANAGEMENT

Laloo, Sheshgaribagh, Hyderpora, Sgr

Phone No's: 2442570, 9419733670

C Language Basics

A Brief History of the C Language

Before we start any complex program in C, we must understand what really C is, how it came into existence and how it differs from other languages of that time.

C is a programming language which born at "**AT & T's Bell Laboratories" of USA in 1972**. It was written by **Dennis Ritchie**. This language was created for a specific purpose: to design the UNIX operating system (which is used on many computers). From the beginning, C was intended to be useful--to allow busy programmers to get things done.

Because C is such a powerful, dominant and supple language, its use quickly spread beyond Bell Labs. In the late 70's C began to replace widespread well-known languages of that time like PL/I, ALGOL etc. Programmers everywhere began using it to write all sorts of programs. Soon, however, different organizations began applying their own versions of C with a subtle difference. This posed a serious problem for system developers. To solve this problem, the **American National Standards Institute (ANSI)** formed a committee in **1983** to establish a standard definition of C. This committee approved a version of C in 1989 which is known as **ANSI C.** With few exceptions, every modern C compiler has the ability to adhere to this standard. ANSI C was then approved by the International Standards Organization **(ISO) in 1990**.

Now, what about the name? **Why it was named C**, why not something else. The C language is so named because its predecessor was called B. The B language was developed by **Ken Thompson of Bell Labs.**

Dennis Ritchie.

Why Use C?

In today's world of computer programming, there are many high-level languages to choose from, such as Pascal, BASIC, and Java. But C stands apart from all these languages. This is due to its many desirable qualities. It is a robust language whose rich set of built-in functions and operators can be used to write any complex logic program. The C language compiler combines the capabilities of a low level language with the features of a high level language. Therefore the language is suitable for writing both system software as well as business packages & other software.

• Program written in **c** are **very efficient and fast**. This is due to its variety of data types and powerful operators. It is many time faster than BASIC. This helps developers in saving their valuable time.

• **C** is a **powerful and flexible language** which helps system developers to deliver various complex tasks with ease. C is used for diverse projects as operating systems, word processors, graphics, spreadsheets, and even compilers for other languages.

• **C** is popular among professional programmers for programming, as a result, a wide variety of C compilers and helpful accessories are available.

• **C** is highly **portable language**. This means that a C program written for one computer system (an IBM PC, for example) can be run on another system (a DEC VAX system, perhaps) with little or no modification. Portability is enhanced by the ANSI standard for C, the set of rules for C compilers.

• **C**'s another striking feature is its ability **to extend itself**. A C program is basically a collection of various function supported by C library (also known as header files). We can also add our own functions to the C library. These functions can be reused in other applications or programs by passing pieces of information to the functions, you can create useful, reusable code.

• Writing **C** program with user-defined functions makes program **more simple and easy to understand**. Breaking a problem in terms of functions makes program **debugging, maintenance and testing easier.**

As these features shows that **C** is an **excellent c**hoice for your first programming language.

The C Character Set

A character denotes any **alphabet, digit or symbols to represent information.** The following are the valid alphabets, numbers and special symbols permitted in C

Numerals: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9

Alphabets: a, b,….z

A, B, ……..Z

Arithmetic Operations: +, -, *, /, %(Mod)

Special Characters:

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

CONSTANTS, VARIABLES AND KEYWORDS

A**’ constant’** is an entity that does not change, but a ‘**variable**’ as the name suggests may change. We do a number of calculations in a computer and the computed values are stored in some memory spaces. In order to retrieve and re-use those values from the computer’s memory locations they are given names. Since the value stored in each location may change, the names given to these locations are called as ‘**variable names**’.

Constants:

There are mainly three types of constants namely: **integer, real and character constants.**

Integer Constants:

The integer constants are:

- Whole Numbers
- E.g. 25, 35, -25, -46

- Computer allocates only 2 bytes in memory.
- 16th bit is sign bit. (If 0 then +ve value, if 1 then –ve value)

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

2^{14} 2^{13} 2^{12} 2^{11} 2^{10} 2^{9} 2^{8} 2^{7} 2^{6} 2^{5} 2^{4} 2^{3} 2^{2} 2^{1} 2^{0}

= 1*1 + 4*1 + 8*1 + 16*1 + 32*1 + 64*1 + 128*1 + 256*1 + 512*1 + 1024*1 + 2048*1 + 4096*1 + 2*1 + 8192*1 + 16284*1

= 32767 (32767 Bits can be stored for integer constants)

- 32768 is negative
- -32767 is minimum

(i) Decimal Integer Constant:

- 0 to 9
- E.g.: 49, 58, -62 … (40000 cannot come bcoz it is > 32767)

(ii) Octal Integer Constant:

- 0 to 7
- Add “0” before the value.
- E.g.: 045, 056, 067

(iii) Hexadecimal Integer:

- 0 to 9 and A to F
- Add 0x before the value
- E.g.: 0x42, 0x56, 0x67

REAL CONSTANTS:

The real or floating point constants are in two forms namely **fractional form** and the **exponential form**.

A real constant in fractional form must:

- Have at least one digit.
- It must have a decimal point.
- Could have positive or negative sign (default sign is positive).
- Must not have commas or spaces within it.
- Allots 4 bytes in memory.

Ex: +867.9, -26.9876, 654.0

In exponential form, the real constant is represented as two parts. The part lying before the **'e'** is the **'mantissa'**, and the one following **'e**' is the **'exponent'**.

The real constant in exponential form must follow the following rules:

- The mantissa part and the exponential part should be separated by the letter 'e'.
- The mantissa may have a positive or negative sign (default sign is positive).
- The exponent must have at least one digit.
- The exponent must be a positive or negative integer (default sign is positive)
- The range of real constants in exponential form is 3.4 e-38 to 3.4 e+38.

Ex: +3.2e-4, 4.1e8, -0.2e+4, -3.2e-4

CHARACTER CONSTANTS

A character constant is an alphabet**, a single digit or a single special symbol** enclosed within inverted commas. The maximum length of a character constant can be 1 character. Allots 1 byte of memory.

Ex: 'B', 'l', '#'

Types of C Variables

Variable names are names given to locations in the memory. These locations can contain integer, real or character constants. **An integer variable can hold only an integer constant, a real variable can hold only a real constant and a character variable can hold only a character constant.**

Rules for Constructing Variable Names

A variable name is any combination of **1 to 31 alphabets, digits or underscores**. Some compilers allow variable names whose length could be up to 247 characters.

- The first character in the variable name must be an alphabet.
- No commas or blanks are allowed within a variable name.
- No special symbol other than an underscore (as in net_sal) can be used in a variable name.

Ex.: si_int

e_hra

pod_e_81

C compiler makes it compulsory for the user to declare the type of any variable name that he wishes to use in a program. This type declaration is done at the beginning of the program.

Following are the examples of type declaration statements:

Ex.: int si, e_hra ;

float bas_sal ;

char code ;

Since, the maximum allowable length of a variable name is **31 characters**; an enormous number of variable names can be constructed using the above-mentioned rules. It is a good practice to exploit this enormous choice in naming variables by using meaningful variable names.

Some Useful Tips:

Declaring Variables in C

Variables are what make your programs zoom. Programming just can't get done without them. E.g. Valerie Variable is a numeric variable. She loves to hold numbers — any number; it doesn't matter. Whenever she sees an equal sign, she takes to a value and holds it tight. But if there is another equal sign, and she takes on a new value. In that way, Valerie is a little flaky. You could say that Valerie's values vary, which is why she's a variable.

Now suppose Victor Variable is a string variable. He contains bits of text — everything from one character to several of them in a row. As long as it's a character, Victor doesn't mind. But which character? Victor doesn't care — because he's a variable, he can hold anything.

• Yes, there is a point here. There are two main types of variables in C: numeric variables that hold only numbers or values, and string variables that hold text, from one to several characters long.

• There are several different types of numeric variables, depending on the size and precision of the number.

• Before you use a variable, it must be declared.

This is — oh, just read the next section.

"Why must I declare a variable?"

You are required to announce your variables to the C compiler before you use them. You do this by providing a list of variables near the beginning of the program. That way, the compiler knows what the variables are called and what type of variables they are (what values they can contain). Officially, this process is known as declaring your variables.

For example:

```
int count;

char key;
```

Two variables are declared here: an integer variable, count; a character variable, key; Doing this at the beginning of the program tells the compiler several things.

First, it says, **"These things are variables!"**

Second, the declarations tell the compiler which type of variable is being used. The compiler knows that integer values fit into the count variable.

Third, the compiler knows how much storage space to set aside for the variables. This can't be done "on the fly" as the program runs. The space must be set aside as the compiler creates the program.

• Declare your variables near the beginning of your program.

• Obviously, you won't know all the variables a program requires before you write it. So, if you need a new variable, use your editor to declare it in the program.

• If you don't declare a variable, your program does not compile. The proper authorities issue a suitable complaint message.

• Most C programmers put a blank line between the variable declarations and the rest of the program.

C Keywords

C makes use of only **32 keywords or reserved words** which combine with the formal syntax to form the C programming language. Note that all keywords in C are written in lower case. A keyword may not be used as a variable name.

auto	double	int	struct
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

Identifiers are names of **variables, functions, and arrays.** They are user-defined names, consisting of sequence of letters and digits, with the letters as the first character. Lower case letters are preferred. However, the upper case letters are also permitted. The (_) under score symbol can be used as an identifier.

Examples:1. #define N 10 2. # define a 15

Here 'N' and 'a' are user-defined identifiers

They consist of a combination of characters and digits. You can then use letters, digits, or underscores for all subsequent characters. Letters include all uppercase characters, A through Z, and all lowercase characters, a through z. Digits include the characters 0 through 9.

Reserved Words

Reserved words are identifiers that you might not use as names for variables, functions, objects, or methods. This includes keywords along with identifiers that are set aside for possible future use.

Data Types

All C compilers support a variety of data types. This enables the programmer to select the appropriate data type as per the need of the application. **Which type of data is storing in a variable is known as data type**. Generally data is represented using numbers or characters. The numbers may be integers or real.

A C language programmer has to tell the system before-hand, the type of numbers or characters he is using in his program. These are data types. There are many data types in C language. A C programmer has to use appropriate data type as per his requirement.

C language data types can be broadly classified as

*** Primary data type**

*** Derived data type**

*** User-defined 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 below

* 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 spaces.

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 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.

Data types and their control strings

Data Type	Size(bytes)	Range	Control String
Char	1	-128 to 127	%c
Unsigned Char	1	0 to 255	%c
Short or int	2	-32,768 to 32,767	%i or %d
Unsigned int	2	0 to 655355	%u
Long	4	-2147483648 to 2147483647	%ld
Unsigned long	4	0 to 4294967295	%lu
Float	4	3.4e-38 to 3.4e+38	%f or %g
Double	8	1.7e-308 to 1.7e+308	%lf
Long Double	10	3.4e-4932 to 1.1e+4932	%lf

In addition, there are a number of qualifiers that can be applied to these basic types. Short and long apply to integers:

```
short int sh;

long int counter;
```

The word int can be omitted in such declarations, and typically it is.

The intent is that short and long should provide different lengths of integers where practical; int will normally be the natural size for a particular machine. short is often 16 bits long, and int either 16 or 32 bits. Each compiler is free to choose appropriate sizes for its own hardware, subject only to the the restriction that shorts and ints are at least 16 bits, longs are at least 32 bits, and short is no longer than int, which is no longer than long.

The qualifier signed or unsigned may be applied to char or any integer. unsigned numbers are always positive or zero, and obey the laws of arithmetic modulo 2n, where n is the number of bits in the type. So, for instance, if chars are 8 bits, unsigned char variables have values between 0 and 255, while signed

chars have values between -128 and 127 (in a two's complement machine.) Whether plain chars are signed or unsigned is machine-dependent, but printable characters are always positive.

The type long double specifies extended-precision floating point. As with integers, the sizes of floating-point objects are implementation-defined; float, double and long double could represent one, two or three distinct sizes.