Introduction

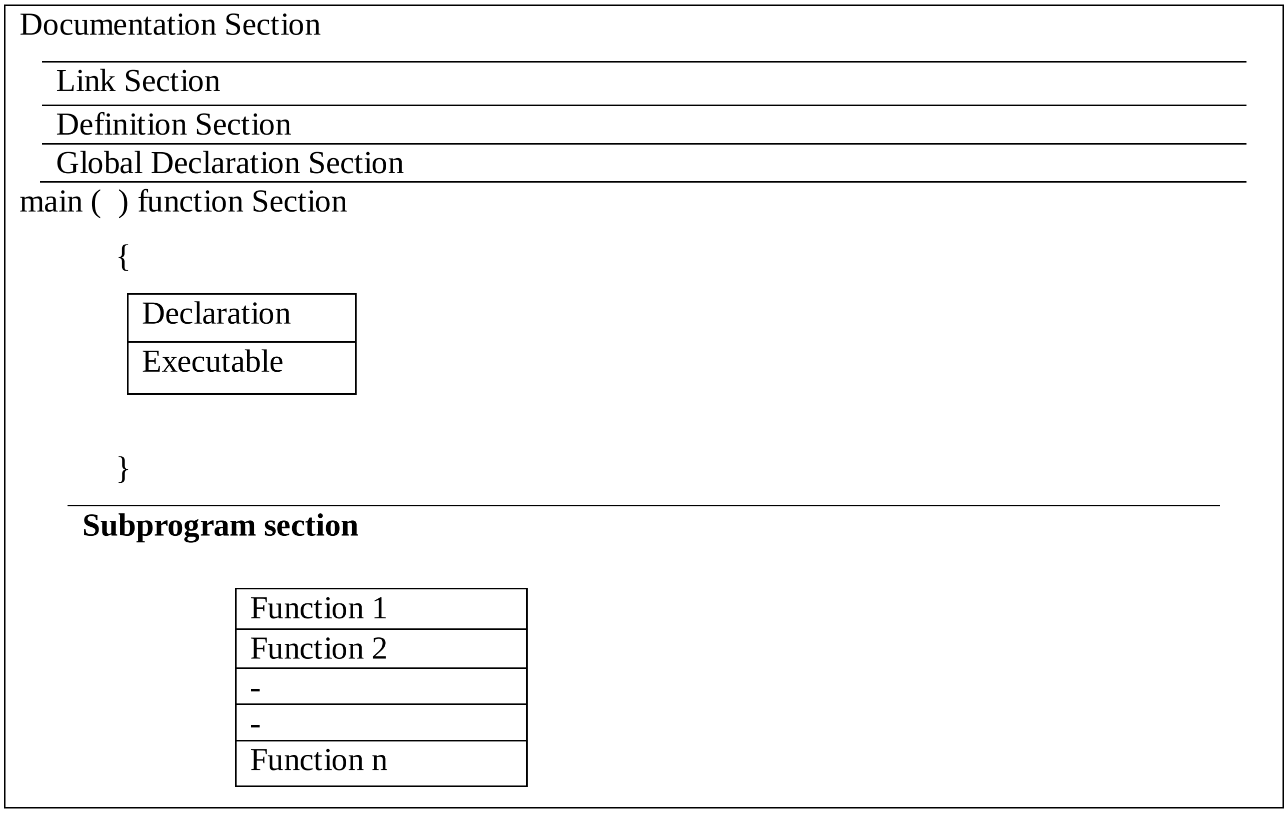
C is one of the most popular computer languages. It is a structural, high-level and machine independent language. C was evolved from the structural languages ALGOL, BCPL and B by Dennis Ritchie at the Bell laboratories in 1972. C uses many concepts from these languages and added the concept of data types and certain other powerful features.

IMPORTANCE OF C

1. The increasing popularity of C language is due to its many desirable qualities.
2. It is a robust language.
3. Its rich set of built-in functions and operators can be used to write any complex program.
4. The C compiler combines the capabilities of assembly language with features of a high level language. Therefore, it is well suited for writing both system software and business packages.
5. Programs written in C are efficient and fast. This is due to its variety of data types and powerful operators.
6. There are only 32 key words.
7. It has several built-in functions (library functions).
8. C is highly portable. This means that C programs written for one computer can be run on another with little or no modification.
9. C language is well suited for structured programming. This modular structure makes program debugging, testing and maintenance easier.
10. C has the ability to extend itself. Therefore, we can add our own functions to C library.

BASIC STRUCTURE OF C PROGRAMS

A C program contains one or more sections of the following.



- The **documentation section** contains a set of comment lines giving the name of the program.
- The **link section** provides instructions to the compiler to link functions from the system library.
- The **definition section** defines all symbolic constants.
- The global variables are declared in the **global declaration section** that is outside of all the functions.
- Every C program must have one **main ( ) function section.** This section contains two parts: Declaration part and executable part. The declaration part declares all the variables used in the executable part.
- The **subprogram section** contains all the user - defined functions that can be called in the main function.

CHARACTER SET

- The characters in C are grouped into the following categories:
  1. Letters 2. Digits 3. Special characters 4. White space

Letters

Uppercase A…..Z
Lowercase a…...z

Digits

All decimal digits 0…….9

Special characters

+ plus sign	! exclamation mark	: colon	^ upward arrow
- minus sign	? question mark	; semicolon	\ back slash
* asterisk	& ampersand	(left parenthesis	~ tilde
/ slash	¦ pipe line	) right parenthesis	# hash
% percent sign	. full stop	[left bracket	_ underscore
< less than	, comma	] right bracket	$ dollar sign
> greater than	‘ apostrophe	{ left brace	b̸ blank space
= equal	“ quotation mark	} right brace	

KEYWORDS AND IDENTIFIERS

Keywords are the basic building blocks of program statements. All keywords must be written in lowercase.

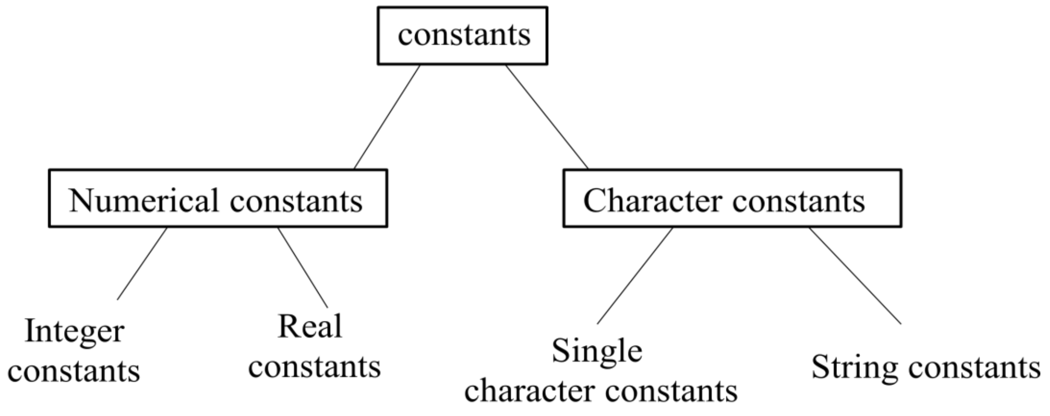
keywords

auto	double	continue	int	signed	static
break	else	default	long	Struct	union
case	extern	do	register	switch	unsigned
char	float	goto	return	type def	void
const	for	if	short	Sizeof	while

***Identifiers** refer to the names of variables, functions and arrays. These are user-defined names and consist of a sequence of letters and digits, with a letter as a first character.* Both uppercase and lowercase letters are permitted, although lowercase letters are commonly used. e.g. x, sum, book name, avg.

CONSTANTS

Constants in C refer to fixed values that do not change during the execution of a program. C supports several types of constants as given below:



Integer constants:

An integer constant refers to a sequence of digits. There are three types of integers, *viz.*, **decimal, octal and hexadecimal.**

Decimal integers consist of a set of digits, 0 through 9, preceded by an optional – or + sign. Valid examples of decimal integer constants are

123 -321 0 86587 +876

Embedded space, commas, and non-digit characters are not permitted between digits.

For example, the following are invalid integers,

15 750 20,000 $1000

An **octal integer** constant refers to any combination of digits from the set 0 through 7, with a leading 0. Some examples of octal integers are

037 0 0435

A sequence of digits preceded by 0x or 0X is considered as hexadecimal integer. Some examples of hexadecimal integers are 0X2 0x9F 0x

Real constants:

The quantities represented by numbers containing fractional parts like 17.548 are called real constants. Some examples of real constants are

0.00045 -0.35 355.54 +43.65

A real number may also be expressed in exponential notation. For example, the value 215.65 may be written as 2.1565e2 in exponential notation. e2 means multiply by 10^2. The general form is

mantissa exponent

Single character constants:

A single character constant contains a single character enclosed within a pair of single quote marks. Examples of character constants are

‘5’ ‘X’ ‘;’ ‘+’

The character constant ‘5’ is not the same as the number 5.

String constants:

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

Examples

“well done” “1989” “4+6” “X”

VARIABLES

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

A variable name can be chosen by the programmer in a meaningful way so as to reflect its function or nature in the program. Some examples of such names are:

sum average count class _ strength

Variable names may consist of letters, digits, and the underscore character, subject to the following conditions:

1. They must begin with a letter.
2. ANSI standard recognize a length of 31 characters. However, the length should not be normally more than eight characters, since only the first eight characters are treated as significant by many compilers.
3. Uppercase and lowercase are significant. That is, the variable ‘**Total**’ is not the same as ‘**total’** or **TOTAL.**
4. The variable name should not be a keyword.
5. White space is not allowed.

DATA TYPES

C language is rich in its data types.

C supports three classes of data types

1. Primary (or fundamental) data types
2. Derived data types

3. User-defined data types

Primary data type / Fundamental data types

All C compilers support five fundamental data types, namely

1. integer(**int**),
2. character (**char**),
3. floating point(**float**),
4. double-precision floating point (**double**) and
5. **void.**

Integer	**Character**	**Floating point types**	**void**
Int	Char	Float	
short int	signed char	Double	
long int	unsigned char	long double	
unsigned int			
unsigned short int			
unsigned long int			

Integer types

Integers are whole numbers with the range of values supported by a particular machine. If we use a 16 bit word length, the size of the integer value is limited to the range -32768 to +32767 (that is, -2^{15} to $+2^{15}$ -1).

C has three classes of integer storage, namely **short int, int,** and **long int,** in both **signed** and **unsigned** forms

Floating point types

Float point (or real) numbers are stored in 32 bits with 6 digits of precision. Floating point numbers are defined in C by the keyword float. When the accuracy provided by a float number is not sufficient, the type double can be used to define the number. A double date type number uses 64 bits giving a precision of 14 digits. These are known as double precision numbers.

Void Types

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

Character Types

A single character can be defined as a character (char) type data. Characters are usually stored in an eights (one byte) of internal storage.

DECLARATION OF VARIABLES:

Declaration of variables does the following

1. It tells the computer what the variable name is
2. It specifies what type of data the variable will hold.

The declaration of variables must be done before they are used in the program.

Type Declaration

The syntax for declaring a variable:

data-type v1,v2.....vn ;

v1,v2,.....vn are the names of variables. Variables are separated by commas. A declaration statement must end with a semicolon. Examples:

int count;

int mark1, mark2, sum, ;

float average;

ASSIGNING VALUES TO VARIABLES

Assignment statement

Values can be assigned to variables using the assignment operator = as follows:

variable _name = constant;

Examples:

sum = 0;

count = 1;

balance = 75.84;

yes = 'x';

An assignment statement implies that the value of the variable on the left of the 'equal sign' is set equal to the value of the quantity(or expression) on the right. The statement

count= count + 1;

means that the 'new value' of count is equal to the 'old value' of count plus 1.

OPERATORS

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

C operators can be classified into a number of categories. They include:

1. Arithmetic operators

2. Relational operators
3. Logical operators
4. Assignment operators
5. Increment and decrement operators
6. Conditional operators
7. Bitwise operators
8. Special operators

Arithmetic operators:

Operator	**Meaning**
	Addition
-	Subtraction
*	Multiplication
/	Division
%	Modulo division

Relational operators:

Operator	**Meaning**
<	less than
<=	less than or equal to
>	greater than
>=	greater than or equal to
= =	equal to
!=	not equal to

Logical operators:

Operator	**Meaning**
&&	AND
::	OR
!	NOT

Assignment operators:

Assignment operators are used to assign the result of an expression to a variable.

Operator	
=	*=
+=	/ =
- =	%=

Increment and Decrement operators:

C has two very useful operators not generally found in other languages. These are the increment and decrement operators:

+ + and - - Example : ++x;   --x;

Conditional operators:

?: is a ternary operator.

General usage: exp1? exp2: exp;

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

Bitwise operators:

Bit wise operators are used for manipulating data at bit level.

Operator	Meaning
&	Bitwise AND
|	Bitwise OR
^	Bitwise exclusive OR
<<	Shift left
>>	Shift right
~	One's complement

Comma operators:

The comma operator can be used to link the related expressions together. A comma-linked list of expressions are evaluated left to right and the value of right –most expression is the value of the combined expression. For example, the statement

Z = (x = 10 , y = 5 , x + y);

First assigns the value 10 to x, then assigns 5 toy, and finally assigns 15 (i.e. 10 + 5) to Z.

EXPRESSION

An arithmetic expression is a combination of variables, constants, and operators arranged as per the syntax of the language.

Expressions are evaluated using an assignment statement of the form

variable = expression;

ARITHMETIC EXPRESSIONS

An arithmetic expression is a combination of variables, constants and operators arranged as per the syntax of the language.

Algebraic expression	**C expression**
a × b – c	a * b – c

(m+n) (x+y)	(m+n) * (x+y)
$\left(\dfrac{a+b}{c}\right)$	(a + b)/c

Evaluation of Expressions

Expressions are evaluated using an assignment statement of the form

Variable = expression:

When the statement is encountered, the expression is evaluated first and the result 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.