ONLINE SUPPROT SERVICES

ignou
THE PEOPLE'S UNIVERSITY

CERTIFICATE IN INFORMATION TECHNOLOGY

NCERT
व्यक्ति निर्माण–राष्ट्र निर्माण

NIRMAN CAMPUS OF EDUCATION RESEARCH & TRAINING

Run & Managed by NASO

IGNOU SC-2281

Jakhepal-Ghasiwala Road, Sunam

For more information visit us at: nirmancampus.co.in

Call us at: 9815098210, 9256278000

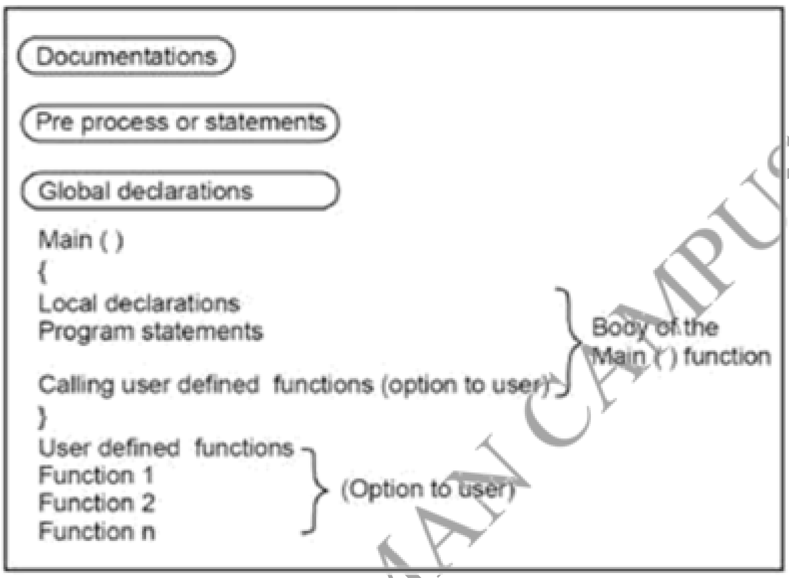
BASIC BUILDING BLOCKS OF C LANGUAGE

HISTORY OF C LANGUAGE

C language is very popular programming language. It was developed in the early 1970s. It was developed by Dennis M. Ritchie at Bell Labs (AT&T). It is developed from the BCPL (Basic Combined Programming Language). C Language is also called the Middle Level Programming language. It is because, it can be used to develop System and Application Programs.

STRUCTURE OF A C PROGRAM

The structure of a C program is a protocol (rules) to the programmer. These rules guide the programmer how to create the structure of the C program. The general basic structure of C program is shown in the figure below.



<u>Documentations</u>

The documentation section consists of a set of comment lines. It allows programmer to define some description of the program.

<u>Preprocessor Statements</u>

The preprocessor statement begins with # symbol. They are also called the **preprocessor directive**. These statements instruct the compiler to include header files or to define symbolic constants etc. before compiling. Some of the preprocessor statements are listed below.

```
# include <stdio.h>   |                   # define P L 3.1412.  |
# include <math.h>    |                   # define TRVE 1       | Symbolic constants
# include <stdlib.h>  | header files      # define FALSE Ø      |
# include <CONIO.h>   |
```

<u>Global Declarations</u>

Global declarations can eb variables or functions. They are declared before the main ( ) function. These global variables can be accessed by all the user defined functions in the program.

<u>The main ( ) function</u>

Execution of C program starts with main () function. No C program is executed without the main function. Every C program should contain only one main ( ) function.

<u>Braces</u>

The left braces after main() indicates the beginning of the main ( ) function and the right braces indicates the end of the main ( ) function.

<u>Local Declarations</u>

Variables declared inside main() are local declarations of main() functions. They can be used only within the main() function.

Program statements

These statements are the instructions of program. They are used to perform a specific task (operations). An instruction may contain an input-output statements, expression, control statements, simple assignment statements etc. Each executable statement should be terminated with semicolon.

User defined functions

These are subprograms. A subprogram is also a function. Subprograms are the logical grouping of statements to perform some specific task. These functions are written by the user. Therefore, they are called user defined functions.

CHARACTER SET:

The set of allowed characters in the language is called the character set. These character are used to form the vocabulary of the language, i.e. identifiers, keywords, constants, variable etc. Character set of C consists of:

- Letters – Alphabets in Upper and Lower Case, i.e. A - Z and a - z
- Digits – From 0 to 9
- Special characters: e.g. &, *, +, $, =, !, %, >, ?, (,), {, } etc.

TOKENS

Tokens are like the words and punctuation marks in natural language. These are the basic and the smallest units of C program. Similarly, in C these elements are keywords, constants, identifiers, operators, strings, and special Symbols. These are called tokens of c language. Following figure shows the six types of tokens present in the C language.

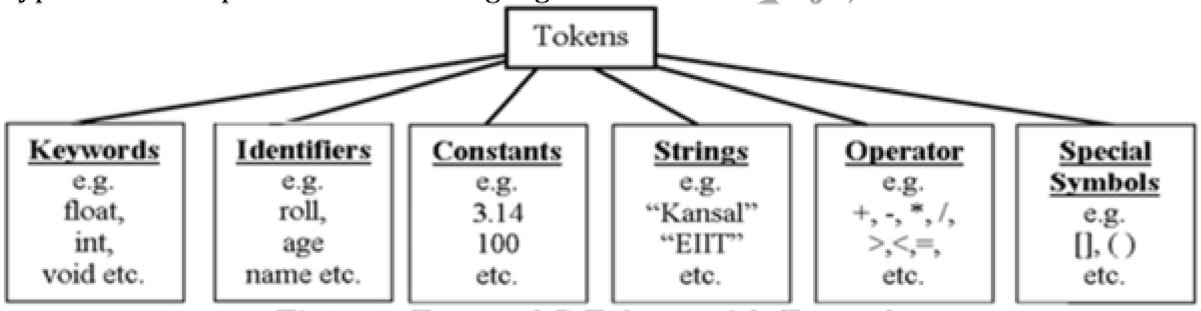

Tokens
Keywords e.g. float, int, void etc.
Identifiers e.g. roll, age name etc.
Constants e.g. 3.14 100 etc.
Strings e.g. "Kansal" "EIIT" etc.
Operator e.g. +, -, *, /, >,<,=, etc.
Special Symbols e.g. [], ( ) etc.

Figure – Types of C Tokens with Examples

Any C program consists of these tokens.

IDENTIFIERS

An Identifier is a name given to a programming element. This program element can be a variable, constant, function, array, structure etc. An identifier consists of a few letters, numbers and a special character (underscore). An identifier consists of maximum of 31 characters. C is a case sensitive language. It means identifiers in upper and lower case are distinct.

Rules for declaring identifiers:

Following are the rules for declaring identifiers:

1. The name of the identifier must begin with a character.
2. The name of identifier can contain maximum of 31 characters.
3. Identifier name cannot be a keyword.
4. Don't use white space in the name of identifier.
5. An identifier name can have letters, digits and valid special characters (underscore).

Following are some examples of valid identifier:

roll, a1, first_name, age, fathername etc.

Following names for identifiers are invalid:

1a	⟶	because first character must be a character
void	⟶	because it is a keyword
first name	⟶	because it contains white space
%age	⟶	because it contains a special character except than underscore

KEY WORDS/RESERVE WORDS

Those words which are predefined in the compiler of C language are called keywords. Keywords are also known as reserve words. We cannot change the meaning of these keywords. All the keywords must be written in the lower case characters only. When we use these keywords in the Turbo C Editor, they are displayed in the white color. Normally, a C compiler has 32 keywords. But new C Compilers have 37 keywords. Following are the C language keyword:

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

VARIABLES

The term ‘variable’ consists of two words vari (vary) + able. It means which can be changed. Thus, a variable allows us to change its value during runtime/execution time of the program. These are used to store the data temporarily in the main memory of computer. This data remain in memory during the execution of program.

To use variables in the program, they must be declared in program. For this, programmers have to give it a name and a data type. For deciding a variable name, following rules should be taken care.

1) The name of variable should not exceed than 31 characters.
2) A variable name can contain alphabets, digits or underscore.
3) The variable name must start with a Character.
4) Special Symbols except Underscore ‘_’ are not allowed in variable name.
5) A keyword cannot be a variable name.
6) Uppercase and lowercase variable names are significant, i.e., roll, ROLL and Roll are the different variables.

Some examples of valid variable names are – a1, first_name, my_float_no, age, rollno etc.

<u>Type Declaration/ Variable Declaration</u>

C language is a strongly typed language. It means all the variables must be declared before using them. Variable declaration contains two things:

1. The data type of variable, and
2. The variable name

The syntax for declaring a variable is:

```
Data_typevariable_name;
```

Here, Data_type tells the type of data to be stored in the variable. The variable_name is any valid name for the variable.

For example: int age; float percent; char grade;

In the above examples, int, float and char are the data types and age, percent and grade are the valid variable names.

<u>Storing/Assigning Value To A Variable</u>

Assigning values to a variable means storing a value in the variable. For this we use assignment operator, e.g.

```
int a;              char grade;
a = 20;             grade = ‘A’;
```

In the above example, 20 value is assigned to an integer variable ‘a’ with the help of assignment operator. Here, ‘=’ is the assignment operator.

<u>Variable Initialization</u>

It is an assignment to variable at declaration time. For example:

```
int age = 20;
float pi = 3.14;
```

CONSTANTS:

Those identifiers which do not allow to change their value during execution time of program are called constants. So, constants have fixed values. C supports several types of constants. We can classify the constants as follows-

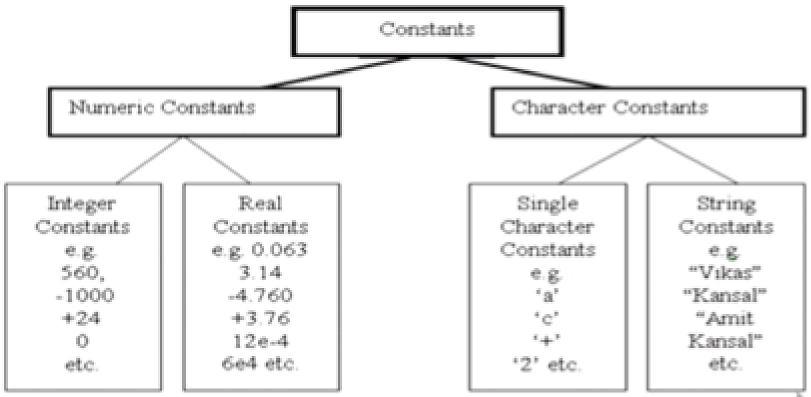

Constants
Numeric Constants
Character Constants
Integer Constants e.g. 560, -1000 +24 0 etc.
Real Constants e.g. 0.063 3.14 -4.760 +3.76 12e-4 6e4 etc.
Single Character Constants e.g. 'a' 'c' '+' '2' etc.
String Constants e.g. "Vikas" "Kansal" "Amit Kansal" etc.

Figure: Types of Constants in C

<u>**Numeric Constants:**</u>

These constants consist of digits from 0 to 9. They can be with or without decimal points in them. These numbers may be preceded with +ve or –ve sign. There are two types of numeric constants:

(1) Integer Constants
(2) Real or Floating Point Constants

1. **Integer Constants:** It is a signed or unsigned whole number. C supports Integer constants in the form of decimal, octal and hexadecimal numbers. Examples of integer constants are- 550 (Decimal), 043 (Octal), 0xA3(Hexadecimal).
2. **Real Constants**: Real constants are also known as floating point constants. Any number with fractional part is called real or floating point constant. This number may be preceded by the +ve or –ve sign. A real constant can be written in decimal or exponential form. Examples of real constants are-

Real constant in Decimal form is	:	0.2569
Real Constant in Exponential form is	:	+2.64e3

The exponential form is used when value is too small or too large. In this form, number is divided in two parts-mantissa and exponent. Mantissa parts appears before 'e' and the exponent part is after 'e'.

<u>**Character Constants**</u>

There are two types of character constants:

(1) Single Character Constants, and
(2) String Constants.

1. **Single Character Constants:** These constants must be enclosed within single quote. It can either be a single alphabet, a single digit or a single special symbol. Example of character constants are- 'v', '2', '*' etc.
2. **String Constants**: The combination of characters is called a string. Any string may consist of alphabets, digits and symbols. The string must be enclosed within double quotes. Example of string constants are- "Param Kansal", "Phone no. 9501010979" etc.

<u>**Constant Declaration:**</u>

To define the constant we use **const keyword**. Its syntax is:

```
const<data type><constants name>;
```

For example;

```
const float pi=3.14;
```

The alternative method of declaring constants is **Symbolic Constants**. For this, '#define' preprocessor directive is used.

For Example:

```
#define PI 3.14
#define MAX 100
```