Module 3: Data types

Learning objectives:

1. To know the C tokens and character set of c.
2. To learn about identifiers and keywords.
3. To understand constants and variables.
4. To study the rules for writing variable names.

1. Introduction

Every C program basically has five main parts: preprocessor commands, functions, variables, statements & expressions and comments. In this module, we shall understand several aspects of C languages related to character set, tokens, identifiers, keywords, variables and constants.

Before we receive the input, it is necessary to have a place to store that input. In programming language, input and data are stored in variables. There are several types of variables. One needs to declare a variable to tell the compiler about the data type and the name of the variable. Several basic types like int, float, char are present in C language.

A variable of type int stores integers i.e. numbers without decimal point, a variable type float stores numbers with decimal places and a variable type char stores a single character.

2. Character Set of C

While writing any program in C one needs to use different statements. Every C program is a set of statements and each statement is made out of some valid characters allowed by the language. A **character** denotes any alphabet, digit or special symbols used to represent information. The character set is the fundamental element of any programming language and they are used to represent information.

Like any natural language, C programming language also have well defined character set, which is useful to design any program. C does neither use nor does it requires the use of every character found on a modern computer keyboard. The characters in C are grouped into the following categories as given in table -1.

Table-1: Character Set of C

Types	Character Set
1. **Lowercase letters**	a to z
2. **Uppercase letters**	A to Z
3. **Digits**	0 to 9
4. **Special characters**	+ - * / \ ! @ # $ & ( ) { }
5. **White spaces**	Tab or newline or space

The valid C special characters are listed in table-2.

Table-2: Special characters in C

Symbol	Meaning	Symbol	Meaning
~	Tilde	!	Exclamation mark
#	Number sign	$	Dollar sing
%	Percent sign	^	Caret
&	Ampersand	*	Asterisk
(	Left parenthesis	)	Right parenthesis
_	Underscore	+	Plus sign
=	Equal to sign	|	Vertical bar
\	Backslash	‘	Apostrophe
-	Minus sign	“	Quotation mark
{	Left brace	}	Right brace
[	Left bracket	]	Right bracket
<	Opening angle bracket	>	Closing angle bracket
:	Colon	;	Semicolon
.	Period	/	Slash

3. C tokens

After learning the character set of C language, it will be easy to understand the other building blocks of C programming language. In any language, the individual words and punctuation marks are called tokens or lexical units. In a C source program, the ***smallest individual unit recognized by the compiler is the "token".*** Fig.1 indicates the classification of tokens.

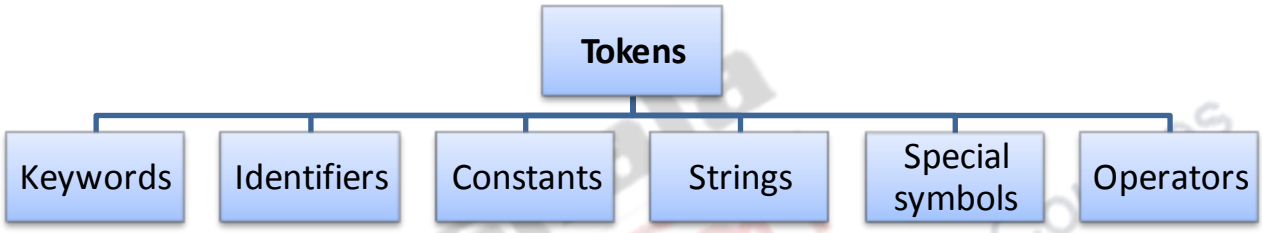

Figure 1: Classification of Tokens

C tokens are the basic building blocks in C language which are constructed together to write a C program. Table-3 indicates different tokens along with their meaning.

Table-3: Types of C Tokens

Token	Meaning	Example
Keyword	A set of reserved words with meaningful names	int, float, for, while
Identifier	Term normally used for variables	main, a, b
Constants	Fixed values assigned to variable	5, 10
Strings	Sequence of characters	"Hello"
Special symbols	Symbol representing characters other than alphabets and digits.	(,),{,}
Operators	Mathematical or nonmathematical symbol	+,-,*,/, ++

3.1 Keywords

In c language, programs are written from a set of reserved words. C keeps a small set of keywords for its own use. Keywords have standard and predefined meaning in C language. C keywords are the words that convey special meaning to the C compiler. These words provide control and perform special function with the help of libraries supported by compiler.

Each keyword is meant to perform a specific function in a C program. These keywords cannot be used as identifiers or variable name in the program. This is a common restriction with programming languages hence these are also called as **reserved words**. Keywords are not written in upper-case letters. Fig.2 shows the list of keywords used in Standard C language.

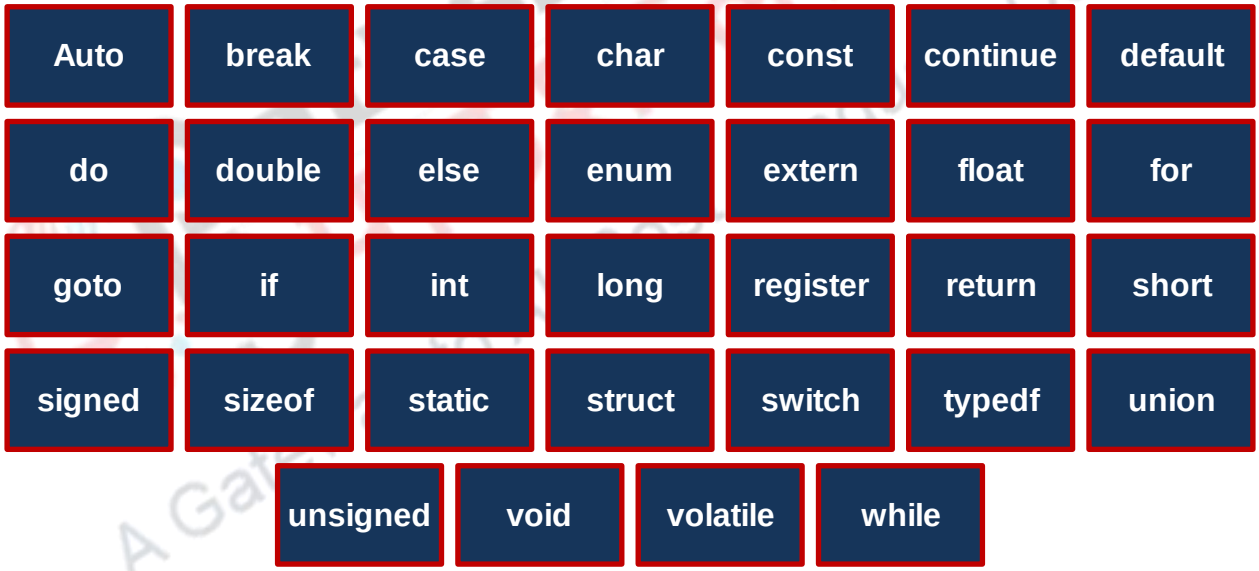

Figure 2: C keywords

All keywords must be written in lower case. There are at least 32 keywords available in C. These keywords are grouped in ten different categories based on their purpose as shown in table 4.

Table 4: Classification of Keywords

	Type	Example
1.	Data types	int, float, char, double
2.	Qualifiers	signed, unsigned, short, long
3.	User defined	typedef, enum
4.	Storage classes	auto, extern, register, static
5.	Loop	for, while, do
6.	Decision	if, else, switch, case, default
7.	Jump	goto, continue, break
8.	Function	void, return
9.	Derived	struct , union
10.	Others	const, volatile, sizeof

Detail description and usage of these keywords would be discussed in other modules whenever necessary.

3.2 Identifiers

Each basic element of C program is given a name called identifier. Names are given to identify variables, functions and arrays. Every data object needs some storage space in the computer memory. To refer these locations we use references, known as identifiers. Rules for naming the identifier are:

1. First character of identifier should be an alphabet or underscore
2. Identifiers can use a-z, A-Z, and 0-9 as succeeding characters.
3. Punctuation and special characters are not allowed except underscore.
4. Identifiers may be 31 to 40 characters long.
5. Identifier should not be a keyword.

Example: Valid identifiers are - temp, average, net_salary etc.

3.3 Constants

C allows you to declare constants. Constants refer to the data that do not change their values during the program execution. A constant is a number, character or character string that can be used as a value in program. Constants may be belonging to any of the data types. You can declare a constant, like a variable declaration but its value cannot be changed. The const keyword is to declare a constant, as shown below:

Syntax:

```
const    data_type  variable_name=value;
```

Example:

```
const float p = 190.5;

const int a =23;
```

We can declare the const before or after the type. Choose any one out of it. It is usual to initialize a const with a value because it cannot get a value using any other way. The preprocessor #define can also be used to define constants in a program. For example, #define k = 113;

Types of C constants

1. Integer constants (e.g. 5, -23 etc.)
2. Floating point constants (e.g. 3.14, 1.6213 etc.)
3. Octal & Hexadecimal constants (e.g. O43, O61, 0x2f etc.)
4. Character constants (e.g. ‘p’,’e’ etc.)
5. String constants (e.g. “Sum”, “Root” etc.)
6. Backlash character constants (e.g. \t, \n etc.)

Detail description and usage of these constants would be discussed in other modules whenever necessary.

3.4 Variables

In every c program, all variables must be declared before their usage. Every variable has a name and a value. A memory location is used to store the value of the variable. Variable is a name given to memory location where a program can store the actual data. As the name indicates the value of variable may change during the program execution. Variables may use any of the data types like int, float, char etc to identify the type of value stored. Variable is one of the basic building blocks of c program which is also called as identifier. The general form of variable declaration is

```
data_type variable_list;
```

Examples:

```
int i,j,k;

float a, b, sum;
```

Variables are initialized with an equal sign followed by a constant expression. Variables may be initialized in the their declaration. The general form of declarations is

```
variable_name = value;

or

data_type variable_name=value;
```

Examples:

```
i= 1;

int count =0;
```

General tip:

- It is good programming practice to declare all the variables together in a program.

Rules for naming variables –

1. Variable names must begin with letter of alphabet. The first character of a C variable cannot begin with a digit.
2. It is legal to start a variable name with underscore character but this is discouraged.
3. Subsequent characters may consist of any combination of alphabets, digits and underscores.
4. Variable name cannot be a reserved keyword.
5. No special characters are permitted in the variable name.
6. Blank spaces are not allowed.
7. Variable name should not be more than thirty-one (31) characters.
8. Before a variable name can be used in computation, it must be assigned a value.

It is conventional to avoid the use of capital letters in variable names. These are used for names of constants. Some old implementations of C only use the first 8 characters of a variable name. Most modern ones don't apply this limit though. The rules governing variable names also apply to the names of functions.

There are three types of variables in C language to specify the scope of the variable. These are shown in table-5.

Table-5: Types of variables

Type of variable	Scope
1. Local variables	• Scope of local variables will be within the function. • These variables cannot be accessed outside the function.
2. Global variables	• Scope of global variables will be throughout the program. Variables can be accessed anywhere in the program. • These variables are defined outside the main function so that these are visible to other functions.
3. Environment variable	• These variables will be available for all C programs and applications. • These variables can be accessed from anywhere in C program without declaring and initializing in the program.

4. Data types

Data types specify how and what type of data is to be entered into the program. C data types are defined as the data storage format that a variable can store a data to perform a specific operation. Data types are used to define a variable before its use in a program.

C language has some predefined set of data types to handle various kinds of data that is normally used in our program. Data types define a system for declaring variables and functions of different types. These data types have different storage capacities. The type of a variable defines how much storage space it occupies. There are four data types in C language as shown in Table-6.

Table-6: C data types

	Types	Data Types
1	Basic data types	int, char, float, double
2	Enumeration data type	Enum
3	Derived data type	pointer, array, structure, union
4	Void data type	Void

4.1 Basic data types

These data types are basically defined for arithmetic type of data. There are four types: integer type, floating point type, double precision type and character type.

4.1.1. Integer data type

Integer data type allows a variable to store numeric values. “int” keyword is used to define integer numbers.

Table -7 provides different standard data types. It also includes storage sizes and range of possible values.

Table-7: Standard integer types

Type	Storage size	Value range
int	2 or 4 bytes	-32,768 to 32,767 or -2,147,483,648 to 2,147,483,647
unsigned int	2 or 4 bytes	0 to 65,535 or 0 to 4,294,967,295
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

Type specifiers - There are 5 different types' specifiers – short, long, long long, signed, and unsigned. For example, if the specifier long is placed before int declaration, the declared integer variable range is extended on some computers i.e. in place of 16-bits integer value it can have 32 bits. Table -7 give the details of these data types. The amount of memory space to be allocated for a variable is derived by modifiers. Modifiers are prefixed with basic data types to modify (either increase or decrease) the amount of storage space allocated to a variable. For example, storage space for int data type is 4 byte for 32 bit processor. We can increase the range by using long int which is 8 byte. We can decrease the range by using short int which is 2 byte.

int is used to define integer numbers. For example,

```
int a=15;
```

This statement declares **a** as integer variable and assigns an integer value to it.

```
{
  int Count;
        Count = 5;
 }
```

4.1.2 Floating data type -

Floating type variables can hold real numbers such as 5.49, -2.73 etc. Usually keywords float or double can be used for declaring floating type variable.

For example

```
float  x1;
double x2;
```

Here x1 and x2 are floating type variables. Float data type allows a variable to store decimal values. Storage size of float data type is 4. This also varies depend upon the processor in the CPU as "int" data type. In float data type, we can use up-to 6 digits after decimal. For example, 21.123456 can be stored in a variable using float data type. Double data type is also same as float data type which allows up-to 10 digits after decimal.

Following table-8 gives you details about standard floating-point types with storage sizes and value ranges and their precision:

Table-8: Floating point types

Type	Storage size	Value range	Precision
float	4 byte	1E-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

The header file float.h defines macros that allow you to use these values and other details about the binary representation of real numbers in your programs.

4.1.3 Character data type:

To declare a variable of type character, a keyword char is used. This data type allows a variable to store only one character. For example,

char c;

To assign or store a character value in a char data type should be declared. A value 'A' can be stored using char data type using following statement:

c='A';

Note that one can assign or store a single character in char variable. To store more than one character, please refer to C-strings in the later modules. Table -9 show different character types along with storage sizes and range.

Table-9: Char types

Type	Storage size	Value range
char	1 byte	-128 to 127 or 0 to 255
unsigned char	1 byte	0 to 255
signed char	1 byte	-128 to 127

```
{
  char Letter;
 Letter = 'x';
}
```

4.2 Enumeration data type in C:

Enumeration data type allows user to define their own data type. Keyword **enum** is used to define enumerated data type. Enumeration data type basically defines a set of named integers. A variable with enumeration type stores one of the values of the enumeration set defined by that type. Enumeration data type consists of named integer constants as a list. It start with 0 (zero) by default and value is incremented by 1 for the sequential identifiers in the list. Let us consider general syntax **Enum syntax in C:**

enum identifier {value1, value2,…..value n};

Here enum is the keword defining identifier as the user defined variable of enumeration data type and different values are allotted for identifier are defined. Let us take an example,

```
enum month { Jan, Feb, Mar }; or
/* Jan, Feb and Mar variables will be assigned to 0, 1 and 2 respectively by default */
enum month { Jan = 1, Feb, Mar };
/* Feb and Mar variables will be assigned to 2 and 3 respectively by default */
enum month { Jan = 20, Feb, Mar };
/* Jan is assigned to 20. Feb and Mar variables will be assigned to 21 and 22
respectively by default */
```

The above enum functionality can also be implemented by “#define” preprocessor directive as given below. Above enum example is same as given below.

```
#define Jan 20;
#define Feb 21;
#define Mar 22;
```

C – enum example program:

```
#include <stdio.h>
int main()
{
    enum Month { Jan = 0, Feb, Mar };
    enum Month month = Mar;
    if(month == 0)
       printf("Value of Jan");
    else if(month == 1)
       printf("Month is Feb");
    if(month == 2)
       printf("Month is Mar");
}
```

Output

```
Month is Mar
```

4.3 The Void type

The main reason for void is to declare a function that has no return value. The term void is used in the sense of **empty** rather that invalid. All c functions are considered as the integer type unless specified. Void is an empty data type that has no value. This can be used in functions and pointers. Table -10 indicates three kinds of methods to use void data type.

Table-10: Utility of void function

Type	Description	Example
1. Function returns as void	Function which do not return any value.	void termin(int status);
2. Function arguments as void	Function which do not accept any parameter	int conver(void);
3. Pointers to void	A pointer of type void * represent the address of an object	Void *fetch(bsize size);

4.4 Derived type:

The basic data types studied so far are primitive data types where the type is not defined in terms of other data types. The derived data types are those that are defined in terms of other data types. Derived data types are made of simpler data types and made of integers or characters or strings etc. There are five derived data types as shown below

1. Array types
2. Pointer types
3. Structure types
4. Union types
5. Function types

Detail description and usage of these data types are discussed in other modules in the ePG pathshala.

5. Summary

Every C program basically has five main parts: preprocessor commands, functions, variables, statements & expressions and comments. Every C program is a set of statements and each statement is made out of some valid characters allowed by the language. A **character** denotes any alphabet, digit or special symbols used to represent information. The character set is the fundamental element of any programming language and they are used to represent information.

C tokens are the basic building blocks in C language which are constructed together to write a C program. These building blocks are keywords, identifiers, constants, strings, special symbols and operators.

In c language, programs are written from a set of reserved words. C keeps a small set of keywords for its own use. Keywords have standard and predefined meaning in C

language. Each basic element of C program is given a name called identifier. Names are given to identify variables, functions and arrays.

Constants refer to the data that do not change their values during the program execution. A constant is a number, character or character string that can be used as a value in program. In every c program, all variables must be declared before their usage. Every variable has a name and a value. A memory location is used to store the value of the variable. Variable is a name given to memory location where a program can store the actual data. As the name indicates the value of variable may change during the program execution.

Data types specify how and what type of data is to be entered into the program. Data types define a system for declaring variables and functions of different types. These data types have different storage capacities. There are four data types: Basic data types, Derived data types, Enumerated data types, void data types. Further the basic data types have four types: integer type, floating point type, double precision type and character type. Enumeration data type allows user to define their own data type. Keyword **enum** is used to define enumerated data type. The main reason for void is to declare a function that has no return value. The term void is used in the sense of **empty** rather that invalid.

Development Team

Principal Investigator:	Dr. Arvind D Shaligram, Savitribai Phule Pune University, Pune
Paper Coordinator:	Dr. Vijay P Labade, Fergusson College, Pune
Content Writer:	Dr. Vijay P Labade, Fergusson College, Pune
Content Reviewer:	Dr. Ajay Kumar (Director), Jayawant Institute of Business Studies Tathwade, Pune