Contents

Unit 3
Introduction to C Programming, Variables, and Constants

Structure of the Unit

3.1. Unit Outcomes

After the successful completion of this unit, the student will be able to:

1. Explain the structure of a C program.
2. Describe the header files, main() function in a C program.
3. Discuss the commands to execute and debug a C program.
4. Discuss variables, constants, and data types in a C program.

3.2 Characteristics of C

C is a high-level programming language developed by scientist Dennis Ritchie in 1972 in AT & T's Bell laboratories in the USA. C programming language is based on the older language Basic Cambridge Programming Language (BCPL), or B. Originally C was written for the UNIX operating system, but today it is available on all operating systems such as Windows, Linux, MAC, etc. The C programming language became extremely popular due to its reliability, simplicity, and ease of use. Even today C programming serves as a base to learn other languages namely C++, Java, C#, etc. It is still in use and popular today. Learning C programming helps in developing an understanding of basic programming skills. With a good knowledge of C, one can get a hold of OOP and other advanced coding methods. Most of the parts of operating systems like Microsoft Windows, Linux, UNIX, etc. are developed in C and software supporting different devices is written in C. The space and time requirements of C programs are less. The different mobile devices such as smartphones, and various home appliances such as microwave ovens, digital cameras, washing machines, etc. are embedded with software developed in C. The different 3D game frameworks are developed using C programs with their speed and simple implementation. C is also referred to as middle-level language as it combines the structures of high-level language, with the power of assembly language. C uses high-level programming constructs such as arrays, pointers, structures, etc. to manipulate bits, registers, memory locations, etc. on a machine**.** The C language is a choice of programmers for implementing operations of hardware devices. The important characteristics of C are as follows.

1. C was originally developed for system software such as operating systems, compilers, device drivers, etc., but now it is also effectively used in all other applications.
2. C is called structured programming as it is well structured with control statements as if, if-else, loops, and blocks. Being structured it is easy to understand and maintain computer programs in C.
3. C is also known as procedural-oriented language as it consists of several procedures. Procedure refers to a block of statements performing tasks of a subproblem. With the use of procedures, C programs are easy to understand and error handling is better.
4. C programs are portable. A C program written on one type of computer can be executed or used on another type of computer with very little modification. A C program can be ported easily to another hardware or operating system.
5. C programming language supports re-usability with library files. C supports library functions and user-defined functions to use multiple times whenever there is a need.
6. C programming language is extensible, where a user can add his function to the library.
7. C supports several standard functions with limited 32 keywords.
8. C program is translated into machine language using a translator known as a compiler. The compiler checks for the structure, reports errors if any, and generates the output. The program written in high-level language is source code and the byte code or machine language program is called object code.

3.3 Structure of C Program

Before discussing the structure of a C program, some important rules for C programs are:

1. Every C program includes a set of instructions, each instruction is a separate statement.
2. All the instructions are executed in serial order. If there are any control structures as if-else conditions or loops, then there is a change in the flow of execution.
3. Blank spaces are used between two words, but black or white space is not permitted in variables, keywords, or constants.
4. Each statement in C ends with a semicolon (;), it is known as a statement terminator.
5. C Programming is case-sensitive. Every statement is written in small letters.

There are approximately 6 fundamental sections in the C program as given below.

1. Documentation using comments
2. Preprocessor for including header files
3. Definitions of constants

4. Global Declaration
5. main() function
6. User-defined functions or subprograms

The basic structure of a C program is shown in Program 3.1.

```
// C Program to Find Area of a Circle  ──────►      Documentation using
Comments
#define pi 3.14 ──────────────────────►              Preprocessor for
constants
#include <stdio.h> ───────────────────►       Preprocessor for library
files
float calculate_area(float radius); ──►    Global Declaration

int main(void) ───────────────────────►    main function
{
float r = 12.5;
float a= calculate_area(r);
printf("%f",a);
return 0;
}
/* subprogram to calculate the area of a circle */
float calculate_area(float radius) ──────►    Subprogram
{
 float area;
area = pi * radius * radius;
return(area);
}
```

Program 3.1 Structure of a C Program

1) **Documentation using comments**: Writing programs with comments is good practice. Comments indicate the problem definition, the purpose of the program, author details, date, time, and place on which the program was written. There can be any number of comments. A well-commented program is useful to make understand the progress of the project in a team. There are two ways of writing comments in C. Comments are not executed but ignored by the C compiler. Examples are shown below.
   - `/* Banking Software Program`
   - `Developed by Robert Date: 1/1/2024, India */`
   - `// C Program banking`
   - `// Number of Functions = 40`
   - `// Tasks are related to Savings accounts, Fixed Deposits, and Loans`

- `// Simple Interest = (p * t * r)/100`

2) **Preprocessor for including header files**: Preprocessor is a program that processes the source program before it is passed to the compiler. A common name for preprocessor commands is preprocessor directives. There are two preprocessors as given below.
   a) Macro Substitution: Used for declaring constants as #define pi 3.14
   b) File Inclusion: Used for linking library header files as #include <stdio.h>
3) **Definitions of constants**: Use of preprocessor directive for declaring constant values.
4) **Global Declaration**: Section for declaring variables and user-defined functions.
5) **main() function**: `main()` is a collective name given to a set of statements enclosed in a pair of curly braces {} as shown in Program 3.2. It is declared with the argument `void` and return type as `int`. The keywords int and void will be explained in the coming sections.

```
return_type main() {
// Statement 1;
// Statement 2;
// and so on.
   return;
}
```

```
#include <stdio.h>
int main(void)
{
  printf("welcome to the main section of a C program);
  return 0;
}
```

Program 3.2 `main()` Function

6) **User-defined functions or subprograms**: A complex problem is divided into subproblems, based on the divide, and conquer approach. Each subproblem is implemented using a subprogram or user-defined function in this section.

3.4. Commands to Run a C Program

C program can be run or executed on Windows, UNIX, Linux, etc. The two commonly used methods to run the C program on Windows and Linux are explained in this section. Running a C program means identifying and correcting errors, generating a machine language program, and showing the results. Steps in compiling and running a C program are depicted in Figure 3.1.

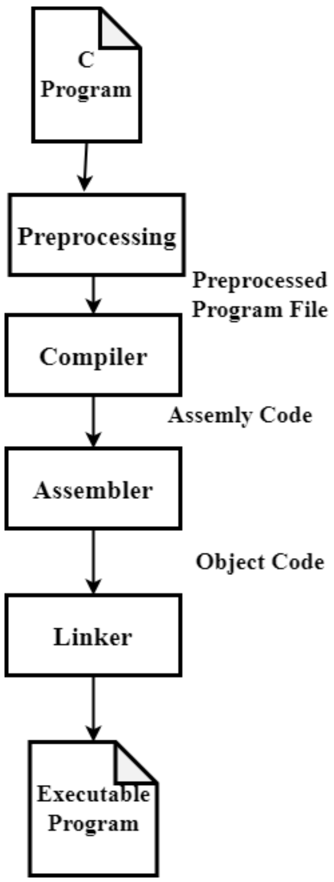


Figure 3.1 Compilation Process for a C Program

Commands to run a C program on Linux or UNIX Operating System are as follows.

1. Prepare a C program using any editor e.g., vi, gedit, vim, notepad, etc., and save the file as filename.c
2. The C program file is run using the following command
```
$ gcc filename.c –o filename
```
3. The option -o is used to specify the output file name. If we do not use this option, then an output file with the name a.out is generated.
4. The generated executable file can be run using the command
```
$ ./filename
```

Commands to run a C program on Windows Operating System are as follows.

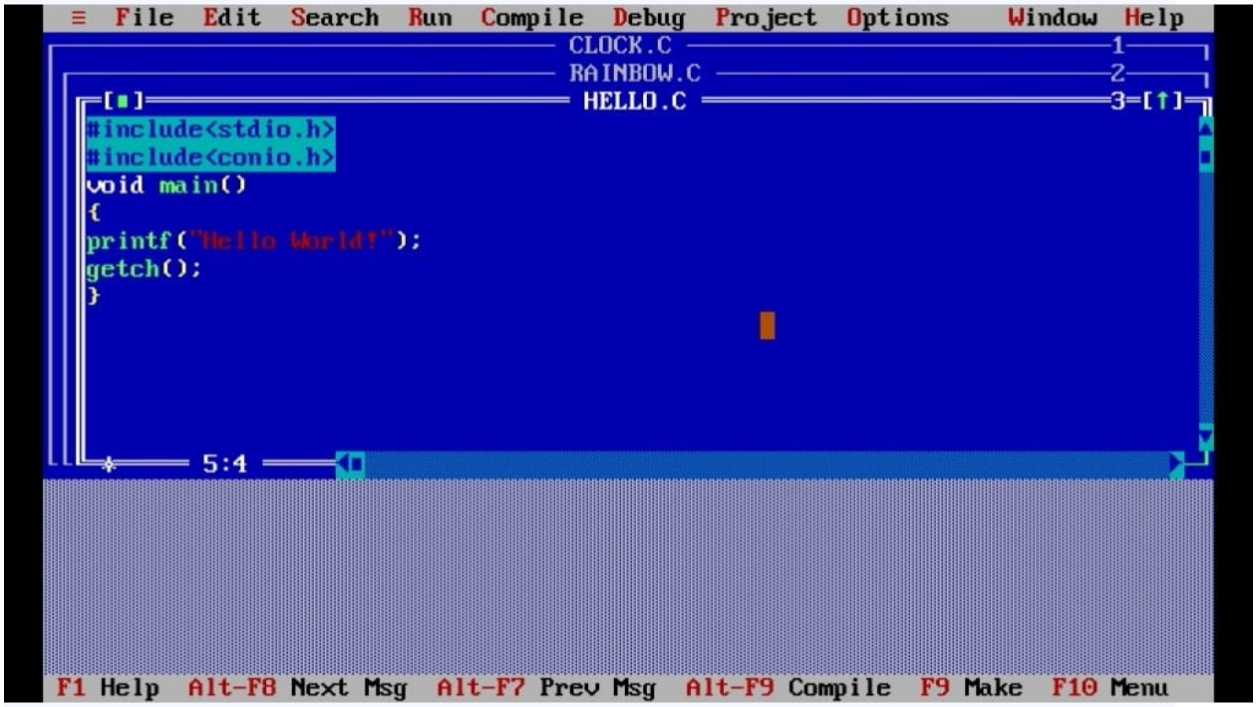
1. An Integrated Development Environment (IDE) such as Eclipse, Netbeans, Dev-C++, etc. can be downloaded.
2. A new file for saving the C program can be created and saved with the extension as **.c**
3. Compile and Run options are in the IDE submenu.

The commonly used C Compilers are:

- Turbo C/C++ Compiler
- Visual Studio
- Eclipse
- Netbeans
- Sublime Text

- GCC and Clang for Linux and MAC operating Systems

A screenshot of a Turbo- C compiler is shown in Figure 3.2.


≡ File Edit Search Run Compile Debug Project Options Window Help
CLOCK.C
RAINBOW.C
HELLO.C
5:4
F1 Help Alt-F8 Next Msg Alt-F7 Prev Msg Alt-F9 Compile F9 Make F10 Menu

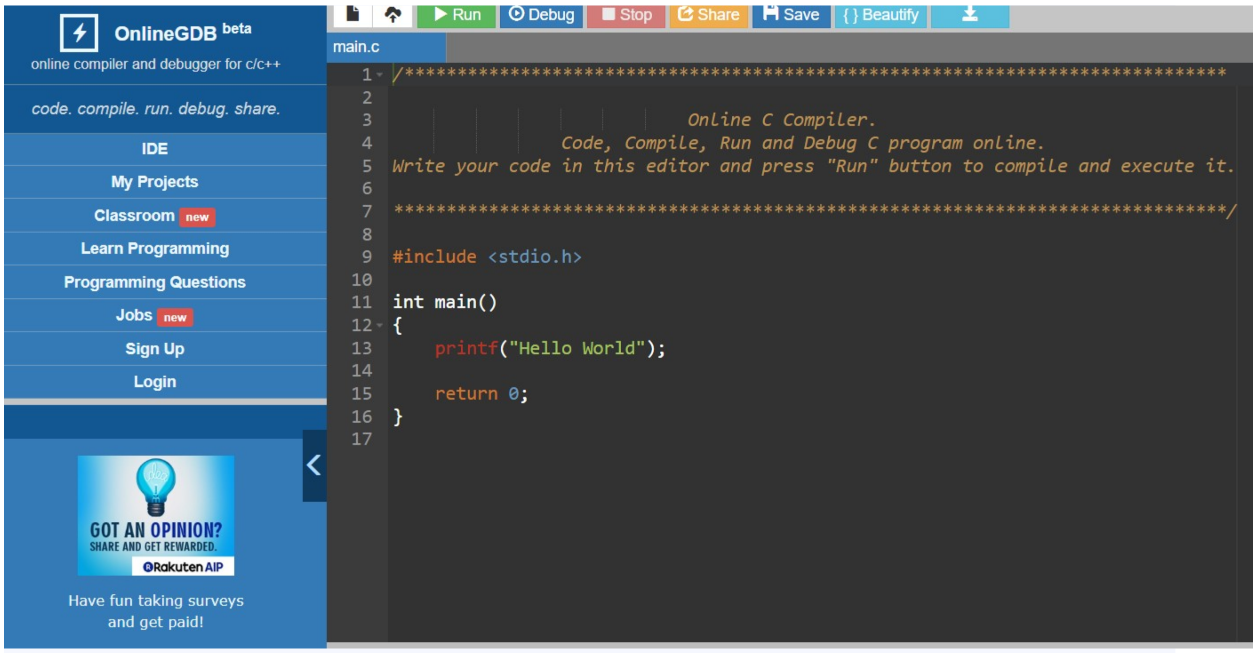
```
#include<stdio.h>
#include<conio.h>
void main()
{
printf("Hello World!");
getch();
}
```

Figure 3.2 Turbo C Compiler IDE

C programs can be compiled using online compilers. The popular C compilers freely available online are:

- https://www.onlinegdb.com/online_c_compiler
- https://www.programiz.com/c-programming/online-compiler/
- https://www.tutorialspoint.com/compile_c_online.php
- https://onecompiler.com/c
- https://www.codingninjas.com/studio/online-compiler/online-c-compiler

A screenshot of the online GDB compiler for running a C program is shown in Figure 3.3.



```
/******************************************************************************

                            Online C Compiler.
                Code, Compile, Run and Debug C program online.
Write your code in this editor and press "Run" button to compile and execute it.

*******************************************************************************/

#include <stdio.h>

int main()
{
    printf("Hello World");

    return 0;
}
```

Figure 3.3 Online GDB Compiler

3.5 Programming Errors

When there is a problem or an issue in a C program and the results obtained are wrong, or the program stops compilation, then there exists an error. The presence of an error will not do the compilation, the program hangs while running, and may generate garbage values or incorrect output. The errors are classified into five classes as given below.

1. Syntax error
2. Run-time error
3. Linker error
4. Logical error
5. Semantic error

Syntax Error: Syntax refers to the general format used for writing C programs. If the programmer does not follow the rules of the programming language, then the compiler will raise syntax errors. If the rules of the language or violated, for example, the use of uppercase letters for the C++ reserved word, parenthesis, etc. are not followed, then there are syntax errors. Some more syntax errors are given below.

•Missing Parenthesis (})
•Unknown variable
•Missing semicolon
•Lvalue required

An example of an unknown variable syntax error is given below.

```
#include <stdio.h>
int main(void)
{
 int x, y;
```

Error: Z is not Declared

```
 x = 10, y = 20;
 z = x + y;
 printf(“%d”,z);
 return 0;
}
```

Runtime Errors: When a program is successfully compiled, there may be errors during execution or run-time. Divide by zero, errors due to the wrong usage of the data type are run-time errors. These errors are not detected by the compilation but occur during program execution (run-time). These errors are not pointed out by the compiler; thus, it is difficult to detect these errors. The division by zero error is the most common runtime error as given below.

```
#include <stdio.h>
int main(void)
{
 int x, y, z;
 x = 10, y = 0;
 z = x / y;
 printf(“%d”,z);
 return 0;
}
```

Error: Divide by Zero

Linker Error: Linker error occurs due to wrong header file, or incorrect function declaration. A source program cannot be translated into machine language due to a linker error. An executable file of a source program will not be generated due to linker errors as given below.

```
#include <studio.h>
int main(void)
{
 int x, y, z;
 x = 10, y = 5;
 z = x / y;
 printf(“%d”,z);
 return 0;
}
```

Error: No such Header File

Logical Error: If the logical thinking done by the programmer is incorrect, the output of the program is wrong. The program gets compiled successfully without any errors, but the outcome is not as expected. These errors are known as logical errors. A missing bracket in the mathematical expression can lead to wrong results as given below.

```
#include <stdio.h>
int main(void)
{
 int x, y, z, avg;
 x = 10;
 y = 20;
 z = 30;
 avg = x + y + z/3;
```

Error: Missing bracket in the formula

```
 z = x / y;
 printf(“%d”,z);
 return 0;
}
```

Semantic Errors: If the program statements are used incorrectly, there are semantic errors. Some examples of semantic errors are the wrong spelling of keywords, wrong operators, or incorrect order of operations. Incorrect data type is another example of semantic error as given below.

```
#include <stdio.h>
int main(void)
{
 int x, y, z, avg, choice;
 x = 10;
 y = 20;
 z = 30;
 x + y = z;
 choice = “Yes”;
 choice = choice – 1;
 printf(“%d”,z);
 return 0;
}
```

Error: Wrong Expression

Error: choice is a string data type, which means set of characters, it can not be declared as int

3.6 Variables and Constants

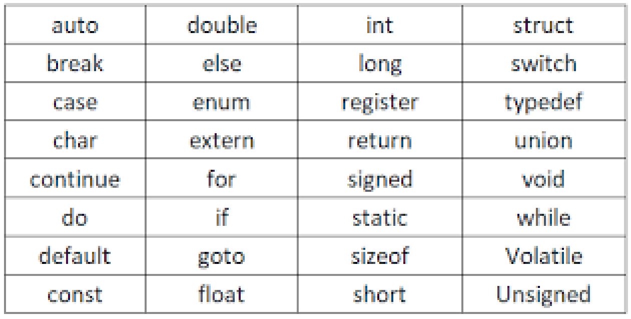
1. **C Character Set**

   The C programming language provides support for the following types of characters.

   a) Digits: The c program supports 10 digits from 0 to 9.
   b) Alphabets: 26 lowercase and uppercase characters are used separately.
   c) Special Characters: Special characters in the C language are used for logical operations, mathematical operations, checking of conditions, backspaces, white spaces, etc. These characters are: **` ~ @! $ # ^ * % & ( ) [ ] { } < > + = _ - | / \ ; : ‘ “ , . ?**

2. **The C keywords**

   The C keywords are reserved words by the compiler. All the C keywords have been assigned a fixed meaning. The list is given below. The keywords cannot be used as variable names because it is like changing the meaning of the keywords.



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

Table 3.1 Keywords in C++

All of these serve a different set of purposes, and we use them in different contexts in the C language.

3. Variables

Computer memory consists of several cells like the human brain and stores the data in each labeled cell. These cells are termed memory locations. Variables are the names given to the memory locations that store data values. Based on the requirements, the values in the variables can be changed. Values in the variables can be stored as shown below.

```
X = 10;
```

In the above example, 10 is stored in a memory location, named `X.` Identifiers are used to name the variables. They are known as functional requirements of any language. The rules for the declaration of the identifier are as given below.

a) Only the alphabet, digits, and underscores are permitted.
b) The identifier cannot be any of the keywords in C.
c) There is a difference between uppercase and lowercase letters for variable names.
d) The length of the identifier can be up to 247 letters.
e) The first letter in the variable name must be alphabetical or underscore.

Variables are declared using the data type and identifier name as shown in the following example.

```
int roll_no, age, day_of_birth; //declaration of 3 integer
variables
age = 30;
float bank_balance, salary; //declaration of 2 real/floating point
variables
char choice;
/*declaration of character type variable holding single alphabet */
```

A set of valid and invalid variable names is shown below.

Valid Names		Invalid Names	
No_account	// Valid	#No	//Invalid character
a	// Valid but poor style	100Players	//First Letter Digit
Int_Max1	// Valid	float	//Keyword
PI	// Valid	Roll NO	//Space
_Name_Student	// Valid	Ex:1	//Invalid Character

4. Constants

Constants are the values that remain constant or of a fixed value. Constants are read-only values. For example, X=10 means X is the name of a constant with a fixed value of 10. Some more examples of constants are as follows.

```
const float pi = 3.14 // const is a keyword used for declaring a
constant
#define max 30 // preprocessor directive for declaring constants
```

Constants are classified as shown in Figure 3.4.

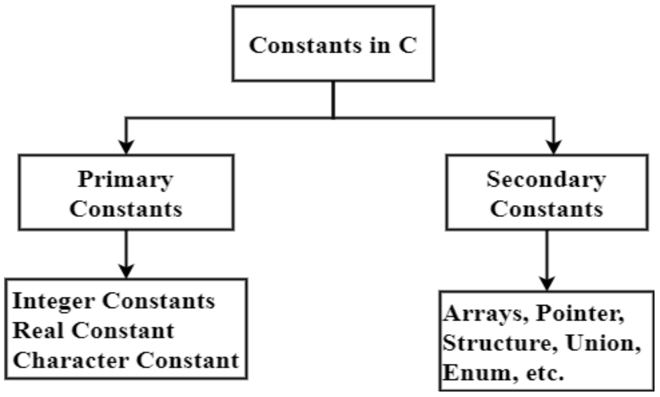


Figure 3.4 Classification of Constants in C

1) **Integer Constants:** An integer constant must have at least one digit and it is without a decimal point. The integer must not contain commas or blank spaces and it can be either positive or negative. One example of a range for integer constants is -32768 to 32767. It depends on the number of bits of a compiler such as 16 or 32-bit.

2) **Real Constants:** Real constants are known as floating point constants and are written in two forms.

   a) **Fractional Form:** A real constant must have at least one digit. It must have a decimal point. It could be either positive or negative. The default sign is positive. Commas or blanks are not allowed within a real constant. Example: `12.5, 123.56577, 234.12,…`

   b) **Exponential Form:** The mantissa part and the exponential part should be separated by the letter `e` or `E.` The mantissa part and the exponential part should be separated by a letter in exponential form. The mantissa part may have a positive or negative sign. The default sign of the mantissa part is positive. The exponent part must have at least one digit, which must be a positive or negative integer. The default sign is positive. The range of real constants expressed in exponential form is `-3.4e38 to 3.4e38`. The number `0.000142` can be represented in exponential form as `1.42e-4.`

   c) **Character Constant:** A character constant is a single alphabet, a single digit, or a single special symbol enclosed within single inverted commas. Point to the left. Example: 'A, 'B'. The maximum length of a character constant can be 1 character. The secondary constants will be discussed in the subsequent sections.

3.6.1 A First C Program

A first C program using character sets, variables, and constants is given in Program 3.2. The different characteristics of the program are as follows:

- The order of the statements in a program must be the same as per the sequence of execution. The order can change if there are if-else or looping statements.
- The blank space is used for improving the clarity or readability of the statement.
- There is no restriction on the positions of the statement, thus C is a free-form language.
- Every statement ends with a semicolon (;), known as statement terminator.
- The main() function is an important function in the C program, where the execution begins. In a C program, instructions are executed in a sequence and the execution begins at the main() function. The main() contains several other functions including tasks of the subproblem. The man() passes control to other functions and after completing several functions executions, the control returns to the main().

```
#define pi 3.14
int main(void)
{
 int r, a;
 r = 10;
 a = pi * r *r;
 return 0;
}
```

Program 3.2 First C Program

3.7 Data Types in C

Variables and constants are connected to the data type. A variable can be stored as a data type like integer, character, floating, double, etc. The memory requirement of every data type is different and multiple special operations can be performed on these data types. Data types in C are classified as given in Figure 3.3.

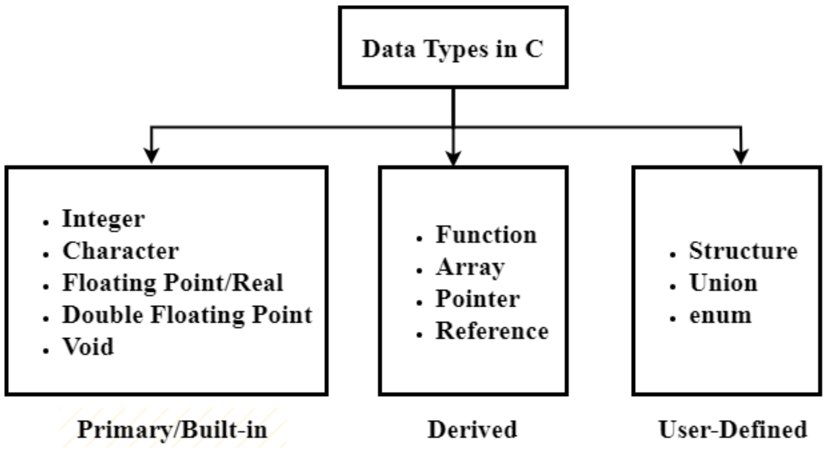


Figure 3.5 Classification of Data Types in C

Built-in Data Types:

The data types that are built-in and used for various direct declarations of variables are called basic, primitive, or built-in data types. The basic data types are int, char, float, double, and void. Each data type is of a fixed size and is available in a range based on a 16-bit or 32-bit compiler.

1) Integer: The int data type refers to the whole number without a decimal point with 4 bytes of

memory space and it is available in the range from -2147483648 to 2147483647. These values are given on a 32-bit compiler. Example: int age; int rollno; int quantity.

2) Character: char is used to store a non-numeric or character type of data. The memory space for char is 1 byte in a range from -128 to 127 or 0 to 255. Example: char choice; char gender
3) Floating Point or Real: Numbers with a decimal point are known as floating points or real numbers. The memory space for float requires 8 bytes. Example: float salary; float price; float bank_balance;
4) void: void refers to an entity without any value. The void type is used when there is a function without a return value.

Except for void, different modifiers can be applied such as long, short, signed, and unsigned. The list of a few data types, size, and range on the 32-bit compiler is shown in Table 3.1.

Table 3.1 Range for Built-in Data Types

Data Type	Number of Bytes	Range
`int`	4	-32768 to 32767
`short`	2	-32768 to 32767
`long int`	4	-2147483648 to 2147483647
`float`	4	3.4E-38 to 3.4E+38
`double`	8	1.7E-308to 1.7E+308
`char`	1	-127 to L27

Derived and user-defined data types will be discussed in the subsequent units.

3.8. Self-Assessment Questions

Q1. Explain important features of C programming. [6 marks, L1]
Q2. Discuss the structure of a C program. [5 marks, L1]
Q3. What is a preprocessor? Explain with a suitable example. [5 marks, L1]
Q4. What is the purpose of comments? Discuss in brief the syntax and the uses. [5 marks, L1]
Q5. What are the rules for naming variables? Discuss with examples. [8 marks, L2]
Q6. What are the types of constants? Give examples [6 marks, L2]
Q7. What are the differences between variables and constants? [6 marks, l2]
Q8. Explain the classification of data types in C. [10 marks]
Q9. Discuss the commonly used C Compilers used today. [6 marks, L1]
Q10. Compare syntax, semantic and run-time errors. [5 marks, L1]
Q11. Compare logical and linker errors. [5 marks, L2]

3.9. Self-Assessment Activities

A1. Develop a C program demonstrating constants and Variables.
A2. Discuss the possibility of logical errors while implementing a simple calculator
A3. Describe a pre-processor with an example.

3.10. Multiple-Choice Questions

Q1. C Programming is a ---------- language. [1 mark, L1]
A. Web-based
B. Object-oriented Programming
C. Structured
D. Assembly

Q2. Library functions in C are defined in ———. [1 mark, L1]

A. Header Files
B. Assembler
C. User's Program
D. Executable Files

Q3. The number of keywords in the C programming language is________. [1 mark, L1]
A. 33
B. 32
C. 28
D. 26

Q4. Invalid variable name from the following is ——— symbol. [1 mark, L1]
A. name
B. var
C. variable
D. switch

Q5. Consider PI as a variable name. Which of the following is not a valid variable declaration?

[1 mark, L1]

A. `int PI = 3.14;`
B. `#define PI 3.14;`
C. `float PI = 3.14;`
D. `double PI = 3.142;`

Q6. The size of an integer data type is___. [1 mark, L3]
A. 3 bytes
B. 2 bytes
C. 4 bytes
D. Depends on the compiler/computer

Q7. Which of the following is not a valid variable declaration? [1 mark, L1]
A. `int x;`
B. `float _x123;`
C. `char $name;`

D. `float x_1;`

Q8. ___________ of the following is correct according to the size (< indicates less than). [L1 mark, L3]

A. int<char<double
B. char<int<double
C. float>double>int
D. char>int<double

Q9. All keywords in C Program are——— letters. [1 mark, L1]

A. Uppercase
B. Sentence case
C. Lowercase
D. Toggle case

Q10. ——— of the following is not a data type [1 mark, L1]

A. int
B. float
C. real
D. char

3.11. Keys to Multiple-Choice Questions

Q1. Structured (C).
Q2. Header Files (A).
Q3. 32 (B).
Q4. `switch` (D).
Q5. `#define PI 3.14;` (B).
Q6. Iteration (A).
Q7. `char @name` (C).
Q8. char<int<double (B).
Q9. Lowercase (C)
Q10. real (C).

3.12. Summary of the Unit

C programming is a structured programming language based on top-down design. It is developed by Dennis Ritchie in the 1970s at Bell Laboratories. C programming includes several library files known as header files that support different functions necessary for the user's program. C is used for developing system programs such as operating systems, compilers, loaders, etc., and is also popular in several application software. The execution of a C program begins at a main() function with the entire program enclosed in curly brackets. C programming files are with extension .c and are compiled using compilers such as gcc, turbo C, and various other compilers available online. With the compiler, the C program is checked for errors and the result is displayed using an executable file. C programs include several data types for processing integer, character, and real data values. The data can be volatile with variables and can be kept fixed using constants.

3.13. Keywords:

- Header files
- `main`()
- Variables
- constants
- Algorithm
- Syntax errors
- Logical errors

3.14. Recommended Learning Resources

Herbert Schildt. (2017). *C Programming: The Complete Reference*. 4th ed. USA: McGraw-Hill. [1]