

An Autonomous Institution

Accredited by NBA – AICTE and Accredited by NAAC – UGC with ‘A+’ Grade
Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai

DEPARTMENT OF AIML

PROBLEM SOLVING AND C PROGRAMMING

I YEAR - I SEM

UNIT 2 – C Programming Basics

Introduction to C Programming & Fundamental Rules

originally developed in the 1970s, by **Dennis Ritchie** at **Bell Labs**

high level , general – purpose **structured** programming language.

allows software developers to develop programs without worrying
about hardware where they will be implemented.

Standardized by American National Standards Institute (ANSI) & International Organization (ISO).

bs developed C language based on “Basic Combined Program

ions of C consists of terms that are very closely same **to algebra**
ng of certain **English keywords** such as if, else, for ,do and while.

ins certain additional features that allows it to be used at a lowe
between machine language and the high level languages.

lows C to be used for **system programming** as well as
mming

orts Both the low-level and High-Level programming features.



History of C

Developed by

International Group

Martin Richard

Ken Thompson

Dennis Ritchie

ANSI committee

ISO committee

ALGOL

BCPL

B

Traditional C

ANSI C

ANSI/ISO C

1960

ALGOL

International

1967

BCPL

Martin R

1970

B

Ken Tho

1972

C

Dennis R

1978

K&R C

Kernigha

1989

ANSI C

ANSI con

1999

C99

Standard

2011

C11

Standard

LANGUAGE: (STRONG)

most language, whose **rich set of built-in functions** and operators can be used to write any complex program.

AND FAST:

Programs written in C are efficient and fast.

Due to its **variety of data types** and powerful operators.

FUNCTIONS

Only **32 keywords in ANSI C** and its strength lies in its built-in functions.

Standard functions are available which can be used for developing programs.

Very portable.

Such that C programs **written for one computer can be run on another** with little or no modification.

PROGRAMMING:

C is well suited for structured programming, thus requiring the user to think of a problem in terms of functions.

A collection of these modules would make a complete program.

“**Modular structure**” makes program debugging, testing and maintenance easier.

EXTEND:

Important feature of C is its ability to extend itself.

A program is basically a collection of functions that are supported by the C library.

Users can add new user-defined functions to C library.

statements must **end with semi colon (;)**

(;) acts as a **terminator**.

case sensitive i.e., upper case and lower case characters are different

F)

the statements are typed in lower case.

statements can be written in one line or it can split into multiple lines

braces must always **match upon pairs** i.e., every opening brace

must have a corresponding closing brace ({...})

every C program must contain a Main() function

comments **can not** be nested. Example (/* welcome to c/*programming)

spaces can be included between two words to improve the readability

variables must be **declared in the declaration section before they**

```
main( )
{
/*.....printing begins.....*/
    printf("I see, I remember");
/*.....printing ends.....*/
}
```

informs the system that the execution begins at this line.

is a special function used by the C system to tell the computer where

am must have exactly one main function.

ore than one main function, the compiler **cannot** understand which
of the program.

brace “{ ” in the second line marks the **beginning of the function**
e “}” in the last line indicates the **end of the function**.

ginning with /* and ending with */ are known as **comment lines**.

nes are **not executable statements** and therefore anything between

```
main( )
{
/*.....printing begins.....*/
    printf("I see, I remember");
/*.....printing ends.....*/
}
```

) function is the only executable statement of the program.

```
    printf("I see, I remember");
```

redefined standard C function for printing output.

means that it is a function that has already been written and compiled

in our program at the time of linking.

function causes everything between the starting and the ending **qu**

at.

the output will be:

I see, I remember


```
main( )
{
/*.....printing begins.....*/
    printf("I see, I remember");
/*.....printing ends.....*/
}
```

different forms of main statement. Following forms are allowed.

main()

main()

main(void)

and **void** means that the function does not return any information to the caller.

and **int** means that the function returns an integer value to the operator. If the return type is **int** specified, the last statement in the program must be “return 0”.

```

#include<stdio.h>
#include<conio.h>
void main() ← main() Function Must Be There
{
    clrscr();
    printf("Welcome to DataFlair");
    // helps to print the message "Welcome to DataFlair"
    getch(); ← Semicolon After Each Statement
}

```

Including Header Files
 Single Line Comment
 Program Enclosed Within Curly Braces

It states the inclusion of Header files.

`main()` is a special function used by the C system to tell the computer to start and returns no value.

Every program must have exactly one main function.

If there is more than one main function, the compiler cannot understand which one is the main function of the program.

The opening brace “{” in the fourth line marks the beginning of the function and the closing brace “}” in the last line indicates the end of the function.

```
    Including Header Files
#include<stdio.h>
#include<conio.h>
void main() ← main() Function Must Be There
{
    clrscr();
    printf("Welcome to DataFlair");
    // helps to print the message "Welcome to DataFlair"
    getch(); ← Semicolon After Each Statement
}
    Program Enclosed Within Curly Braces
```

beginning with // are known as comment lines (Second Type of Comment). Comment lines are not executable statements and therefore anything starting with // is ignored by the compiler.

clrscr() represents **getting the character from processing**.

clrscr() represents clear screen indication, which processes with **clearing the screen** and **descriptive screen**.

Each statement must end with Semicolon (;)