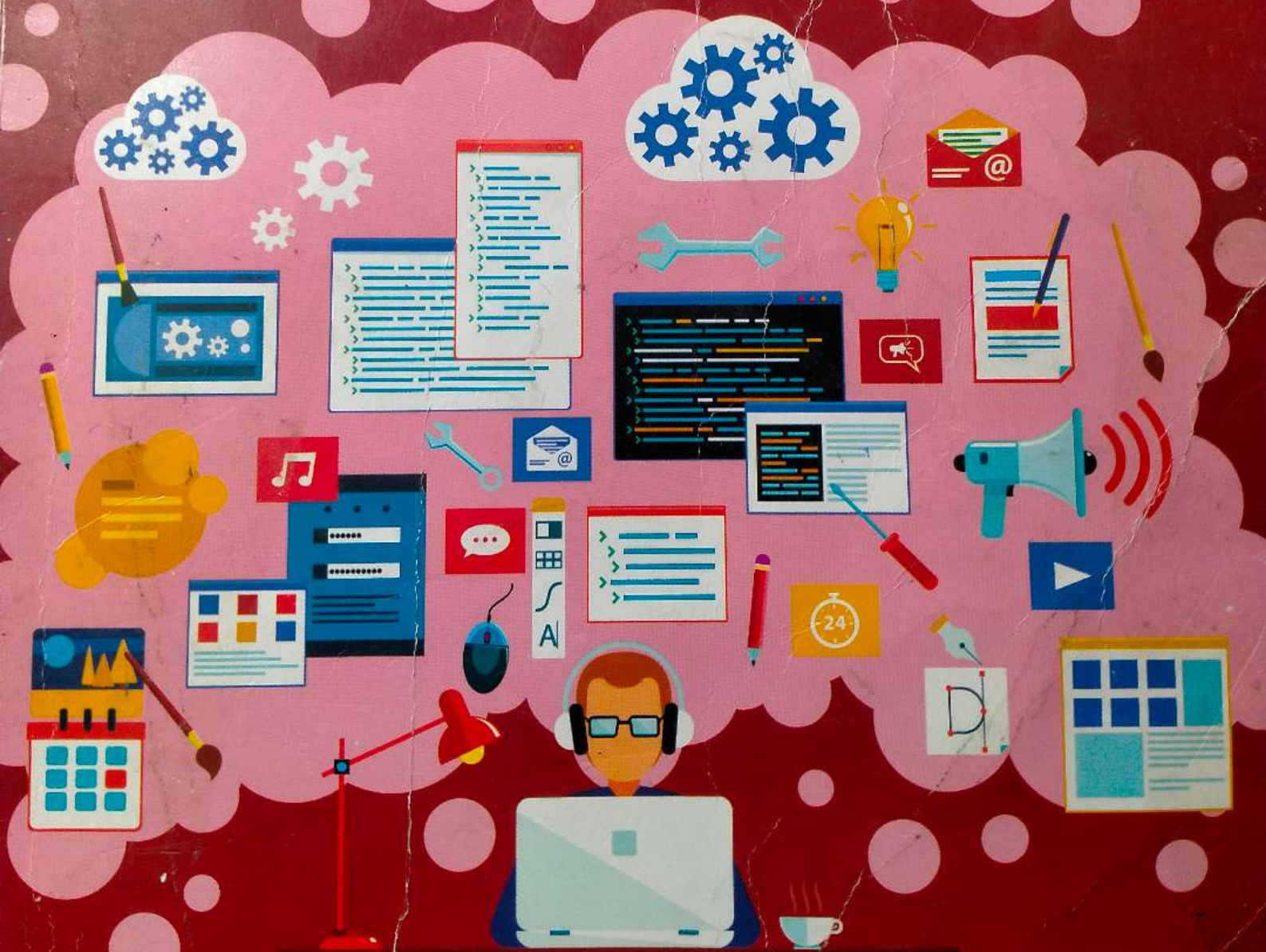
Sarthak

According to the New Syllabus of UPBTE

Concept of Programming Using 'C'

Rajiv Prasad



Jai Prakash Nath Publications
Meerut

Sarthak

According to New N.S.Q.F. Syllabus approved by UPBTE

Concept of Programming Using 'C'

For the Students of

First Year (Second Semester)
Diploma CS & IT

Name ⇒ Anurag Kumar
Subject ⇒ Concept of Programming using 'C'
Anurag

Rajiv Prasad
B.Tech. (IT)
HOD, Information Technology
Km. Mayawati Govt. Girls Polytechnic
Badalpur, G.B. Nagar

Jai Prakash Nath Publications
Meerut

Published by
Jai Prakash Nath Publications
41/5, Jagriti Vihar (Behind Hero Showroom)
Garh Road, Meerut-250 004 (U.P.)
Tel. : (Off.) : 0121-2762403, 4056123
email : jpnpmrt@hotmail.com
: info@jpnpbooks.com
Web : www.jpnpbooks.com

Second Edition, **2021-22**

Price : ₹ **250.00**

Typesetter
adhar
the graphics
MEERUT

Printed at :
Saraswati Press,
Meerut

Concept of Programming Using 'C'

प्रस्तावना

कम्प्यूटर प्रोग्रामिंग भाषा, कम्प्यूटर और प्रोग्रामर के बीच कम्यूनिकेशन का एक बहुत-ही effective माध्यम है। प्रोग्रामिंग भाषा की सहायता से कम्प्यूटर के लिए निर्देश और प्रोग्राम लिखकर कम्प्यूटर को किसी कार्य को करने या संचालित करने के लिए निर्देशित किया जाता है। C प्रोग्रामिंग भाषा एक बहुत-ही प्रचलित और सहज भाषा है। C प्रोग्रामिंग भाषा का प्रयोग करके विभिन्न प्रकार के सिस्टम सॉफ्टवेयर, एप्लीकेशन सॉफ्टवेयर, डेटाबेस, डिवाइस ड्राइवर तथा ग्राफिक्स डिजाइन इत्यादि का डेवलपमेंट कर सकते हैं। C प्रोग्रामिंग भाषा को समझना और लिखना बहुत-ही आसान है इसलिए यह अत्यधिक लोकप्रिय कम्प्यूटर प्रोग्रामिंग भाषा है।

इस पुस्तक में नवीन पाठ्यक्रम के अनुसार, सम्पूर्ण syllabus को 8 अध्यायों (chapters) में विभाजित किया गया है। सभी अध्यायों में Basic Concept को तथ्यों और प्रयोगात्मक उदाहरण द्वारा समझाने का प्रयास किया गया है।

पुस्तक को त्रुटिहीन बनाने में हर समभव प्रयास किया गया है, फिर भी पुस्तक को और अधिक उपयोगी बनाने के लिए सभी प्रकार के सुझाव आमंत्रित हैं।

–लेखक

ACKNOWLEDGMENT

First of all I would like to thank God for giving me the strength, knowledge and ability to spread my knowledge through the book.

I would like to thank all the staff members of Kumari Mayawati Govt. Girls Polytechnic, Badalpur, Gautam Budh Nagar, for encouraging and supporting me to write the book.

I would like to express my sincere gratitude to my Parents.

I am also very thankful to my wife Dr. Namita Gupta and son Mr. Atherva Gupta for their valuable support.

Finally I express my sincere thanks to M/s Jai Prakash Nath Publications, Meerut to give me opportunity to write and their cooperation in the publication of book.

—Rajiv Prasad

A parent's love
is whole
no matter
how many times
Divided
This Book is Dedicated to

My Parents

UPBTE SYLLABUS

Concept of Programming Using C

RATIONALE

Computer plays a vital role in present day life, more so, in professional life of technician engineers. People, who are working in the field of computer industry, use computers in solving problems more easily and effectively. In order to enable the students use the computers effectively in problem solving, this course offers the modern programing language C along with exposition to various applications of computers. The knowledge of C language will be reinforced by the practical exercises.

LEARNING OUTCOMES

- Identify the problems and formulate an algorithm for it.
- Identify various control structures and implement them.
- Identify various types of variables.
- Use pointer in an array and structure.
- Use structures and union for handling data.
- Explain the concepts of C programming language.
- Explain and implement the language construct concepts.
- Install C software on the system and debug the programme.
- Describe and implement array concept in C programming.
- Describe and execute pointers.
- Expose File System using File Handling.

DETAILED CONTENTS

1. Algorithm and Programming Development

Steps in development of a program, algorithm development, concept of flowcharts, programming and use of programming, various techniques of programming. Structured Programming, Preprocessors, Debugging, Compiling.

2. Program Structure

Structure of C program. Writing and executing the first C program, Translator Assembler, interpreter. Compiler. I/O statement, assign statement, keywords, constants, variables and data types, storage classes, operators and expressions. Unformatted and Formatted IOS, Data Type Casting.

3. Control Structures

Introduction, decision making with If-statement, If-else and Nested If, Ladder if-else, Loop : While, do-while, for, Break, Continue, goto and switch statements.

4. Functions

Introduction to functions, Global and Local Variables, Function Declaration, Function Call and Return, Types of Functions, Standard functions, Parameters and parameter Passing. Call - by value/reference, recursive function, function with array, function with string.

5. Arrays and Strings

Introduction to Arrays, Array Declaration, Length of array, Manipulating array elements, Single and Multidimensional Array, Arrays of characters. Passing an array to function, Introduction of Strings, String declaration and definition, String Related function i.e. strlen, strcpy, strcmp.

6. Pointers

Introduction to pointers, Static and dynamic memory allocation, Address operator and pointers, Declaring and initializing pointers, Single pointer, Pointers to an array.

7. Structures and Unions

Declaration of structures, Accessing structure members, Structure initialization, array of structure variable, Pointer to a structure, Union, Declaration of Union.

8. File Handling

Basics of File Handling, opening and closing of File, reading and writing character from a file.

LIST OF PRACTICALS

1. Programming exercises on executing and editing a C program.
2. Programming exercises on defining variables and assigning values to variables.
3. Programming exercises on arithmetic, logical and relational operators.
4. Programming exercises on arithmetic expressions and their evaluation.
5. Programming exercises on formatting input/output using printf and scanf and their return type values.
6. Programming exercises using if statement.
7. Programming exercises using if – else.
8. Programming exercises on switch statement.
9. Programming exercises on while and do – while statement.
10. Programming exercises on for – statement.
11. Simple programs using functions and recursive function.
12. Programs on one-dimensional array.
13. Programs on two-dimensional array.
14. (i) Programs for concatenation two strings together.
    (ii) Programs for comparing two strings.
15. Simple programs using pointers.
16. Simple programs using structures.
17. Simple programs using union.
18. Simple programs for File Handling.

CONTENTS

●●●

Unit

1 Algorithm and Programming Development

1.1 कम्प्यूटर प्रोग्रामिंग लैंग्वेज (Computer Programming Language)

प्रोग्रामिंग लैंग्वेज (language) बहुत सारे निर्देशों (instructions) का set है जिसका प्रयोग कम्प्यूटर पर कुछ विशेष कार्य करने या किसी problem के solution प्राप्त करने के लिए अथवा कम्प्यूटर से बात (Communicate) करने के लिए किया जाता है। कम्प्यूटर द्वारा instruction को executes किया जाता है और instructions के अनुसार कम्प्यूटर किसी problem को sovle करता है। कम्प्यूटर एक Electronic मशीन है जो बाइनरी (binary) सिस्टम पर आधारित है। कम्प्यूटर human language नहीं समझता है। कम्प्यूटर से कार्य कराने के लिए या कम्प्यूटर को कुछ समझाने के लिए एक माध्यम एक (language) जो कम्प्यूटर और user समझ सके Develop किया गया जिसे programming language (प्रोग्रामिंग भाषा) कहते हैं। Programming language (प्रोग्रामिंग भाषा) एक माध्यम है जिसका उपयोग हम कम्प्यूटर को कुछ समझाने कम्प्यूटर से बात (Communicate) करने के लिए करते है। प्रोग्रामिंग लैंग्वेज अनेक प्रकार के keywords variables, identifier, operator, symbols और instructions के सेट (set) से बनते हैं।

▮ कम्प्यूटर प्रोग्रामिंग भाषा के प्रकार (Types of Programming Language)

(a) निम्न स्तरीय प्रोग्रामिंग भाषा (Low Level Programming Language)

(b) उच्च स्तरीय प्रोग्रामिंग भाषा (High Level Programming Language)

▮ (a) निम्न स्तरीय प्रोग्रामिंग भाषा को दो भागों में विभाजित किया जा सकता है–

(i) मशीनी भाषा (Machine Language)

कम्प्यूटर (computer) बाइनरी सिस्टम पर आधारित हैं अत: कम्प्यूटर (computer) केवल बाइनरी भाषा 0 और 1 ही समझता है, इसे ही Machine Language (मशीनी भाषा) भी कहते हैं। यह programming language का सबसे lowest level है। Program जो High Level language में लिखे जाते हैं, पहले Assembly या Machine language में Translate होते हैं उसके बाद execute होते हैं। सभी CPU का अपना unique machine language होता है। Machine language को computer द्वारा आसानी से execute कराया जा सकता है। Machine language में Binary Number 0 से 1 का प्रयोग किया जाता है। Symbol electric pulse के absence को show करता है। Symbol 1 electric pulse के presence को show करता है करता है। कम्प्यूटर electric pulse को recognize करता है इसलिए machine language को समझ पाता है। Machine language हमेशा ही machine dependent होता है।

(ii) असेम्बली भाषा (Assembly Language)

Assembly language एक प्रकार का low level programming language है जो एक विशेष प्रकार के processor के लिए designe किया जाता है। Assembly language को assembler की सहायता से machine language में बदला जाता है। Assembly language प्रयोग और समझने में machine language से आसान है। विभिन्न प्रकार के CPU के लिए मशीनी भाषा और असेम्बली भाषा अलग-अलग होते हैं। इसलिए किसी एक CPU के लिए design किया गया Assembly language दूसरे CPU पर run नहीं हो सकता है। Assembly language में symbol का प्रयोग किया जाता है जो समझने में आसान होता है। Assembly language में लिखे प्रोग्राम के error को आसानी से दूर कर सकते हैं तथा modify कर सकते हैं।

(b) उच्च स्तरीय प्रोग्रामिंग भाषा (High Level Programming language)

High Level Programming language एक प्रकार का user friendly programming language है। यह Computer harware और Architecture पर depend नहीं होता है। यह समझने में सबसे आसान होता है तथा आसानी से इसको modified किया जा सकता है। High level language कम्प्यूटर ऑपरेटर और प्रोग्रामर के प्रयोग करने के लिए designe किया गया है। High level language में syntax, English alphabet arithmetic expression, number इत्यादि का प्रयोग किया जाता है।

C, C^{++}, JAVA, PHP, Visual basic इत्यादि High level progamming language हैं।

High Level language दो प्रकार के होते हैं—

(i) तीसरी पीढ़ी की प्रोग्रामिंग भाषा (Third generation programming language)

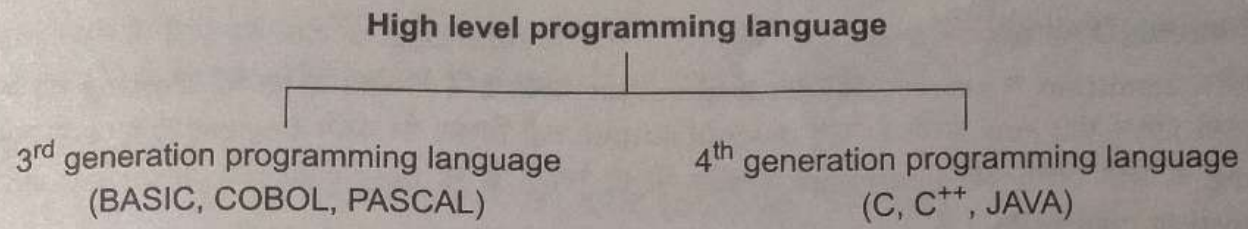
Third generation language (3 GL) एक प्रकार का High level programming language है जो Assembly language से ऊपर और (4 GL) Fouth generation language से नीचे का language है। यह एक user friendly programming language है। 3GL सबसे पहले सन् 1950 में develop किया गया। 3GL में English alphabet, symbols और digits का प्रयोग किया गया। 3GL को ही आगे modify करके 4GL में बदला गया जो आजकल सभी प्रोग्रामर के द्वारा प्रयोग किया जा रहा है। Translator के द्वारा Third generation language को object code में translate किया जाता है।

FORTRAN, BASIC, COBOL, PASCAL, PROLOG इत्यादि Third generation Programming language के उदाहरण हैं।

(ii) चौथी पीढ़ी की प्रोग्रामिंग भाषा (Fourth generation programming language)

Fourth generation language एक बहुत ही user friendly compuer language है जो Portable और machine independent language है। इसे आसानी से समझा जा सकता है तथा modify किया जा सकता है। आजकल के अधिकांश software को develop करने में fourth generation language का प्रयोग किया जाता है। Fourth generation language किसी भी software development में overall cost, effort और समय को बचाता है। Fourth generation language में English alphabet, digit, symtax, symbol, number इत्यादि का प्रयोग किया जाता है।

C, C^{++}, JAVA, PHP, visual basic इत्यादि fourth generation language है।



1.2 Steps to Develop a Computer Program (प्रोग्राम डेवलप्मेंट के चरण)

Computer Program का development किसी goal या task को achieve करने के लिये Computer programmer या developer के द्वारा किया जाता है। Computer Program को develop करने के लिये मुख्यत: कुल 7 Systematic Steps को follow किया जाता है। Systematic Steps follow करने से Program में Syntactical या Logical किसी भी प्रकार के Errors होने की सम्भावना कम रहती है और Program को आसानी से Manage, Maintain और Upgrade कर सकते हैं।

(i) Problem Definition

प्रोग्राम को develop करने का यह inital step होता है। इस step में समस्या को अच्छी तरह से समझ कर उसे define करना होता है। प्रोग्राम को जितनी अच्छी तरह define करेंगे उसका solution उतना ही अच्छा निकलेगा। यह Step किसी भी Computer Program का सबसे महत्वपूर्ण Step होता है, क्योंकि पूरे Program का Result इसी बात पर निर्भर करेगा कि समस्या को कितना बेहतर तरीके से define किया गया है। यदि Problem को सही तरीके से define नहीं किया गया है तो हमारा प्रोग्राम हमें accurate result नहीं देगा।

(ii) Problem Design

इस step में समस्या (Problem) को कई छोटे-छोटे भागों में devide कर उसे algorithm या Logic के अनुसार लिखा जाता है। Algorithm लिखने के लिये flowchart और अन्य logic का भी प्रयोग किया जाता है। पहले Step में Problem से सम्बंधित जिन जरूरतों को अपने Program द्वारा achieve करना चाहते हैं उन जरूरतों को कैसे पूरा किया जाए, इसका decision इसी चरण में लिया जाता है। यह कार्य System Designer के द्वारा किया जाता है।

(iii) Program Coding

Problem Design के बाद उसे किसी Programming language में implement करने के लिये Program Coding का प्रयोग करते हैं। कम्प्यूटर प्रोग्रामर अपनी और clients की सुविधानुसार किसी हाई लेवल भाषा का प्रयोग कर Coding करता है। Systematic Coding कम्प्यूटर प्रोग्रामर के द्वारा ही की जाती है।

(iv) Program Execution

इस step में प्रोग्राम को run किया जाता है और यह insure किया जाता है कि प्रोग्राम desired output दे रहा है या नहीं। Run कराने के बाद अगर प्रोग्राम में error है तो desired output नहीं मिलता है। प्रोग्राम में logical या synax error हो सकता है। Errors को remove करके desired output प्राप्त किया जा सकता है।

(v) Program Debugging

प्रोग्राम को run कराने के बाद उसमें कई तरह के logical या synax error हो सकता है जिसके कारण desired output नहीं मिलता है। प्रोग्राम में उपस्थित अनेक प्रकार के Error या Bugs को search करना व उन्हें correct करने की process को ही Debugging कहते हैं। यह कार्य Debugger करता है।

(vi) Program Testing

प्रत्येक condition में correct और desired output प्राप्त करने के लिए प्रोग्राम की Testing की जाती है। कभी-कभी प्रोग्राम सही run करता है परंतु desired output नहीं मिलता है। उसमें Logical Errors हो सकता है। Testing के लिये अलग-अलग software Team भी हो सकती है जो user side और developer side दोनों तरफ testing insure करती हैं।

(vii) Program Maintenance

Program Maintenance का मतलब, आवश्यकतानुसार Program को moidified करना होता है। Maintenance सॉफ्टवेयर डेवलपमेंट का एक बहुत ही महत्वपूर्ण भाग होता है। Maintenance के द्वारा हम प्रोग्राम में correction और alteration कर सकते हैं। जब Hardware या software की configuration बदल जाती है तो उस पर execute होने वाले Program को भी configuration के अनुसार modified करना होता है।

Maintenance के प्रकार—

a. Corrective maintenance
b. Adaptive maintenance
c. Preventive maintenance
d. Perfective maintenance

Pragram development के चरण

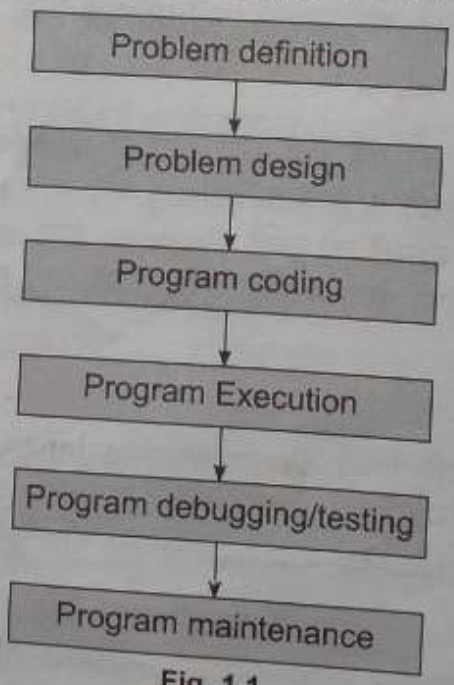


Fig. 1.1

1.3 Algorithm (एल्गोरिथ्म)

Algorithm एक well defined और चरणबद्ध प्रक्रिया है जो problem को solve करने में प्रयुक्त होता है। Algorithm का प्रयोग calculation और data processing में किया जाता है। किसी भी problem को solve करने के लिए program बनाते हैं जो Algorithm पर आधारित होता है। Algorithm के द्वारा ही हम प्रोग्राम को design करते हैं। Algorithm ही किसी problem की complexity को define करता है। किसी एक problem

को solve करने के लिए कई तरह के algorithm हो सकते हैं। प्रत्येक algorithm में निश्चित चरण और definite procedure होते हैं जो एक निश्चित चरण के बाद समाप्त होते हैं और result देते हैं।

Algorithm के गुण

(i) Algorithm कुछ निश्चित चरण के बाद हमेशा terminate होने चाहिए।
(ii) Algorithm के प्रत्येक चरण अच्छी और सही define होने चाहिए।
(iii) किसी भी problem को solve करने के लिए सही और eeffective algorithm का selection करना चाहिए।

Algorithm के प्रकार

(i) Simple algorithm
(ii) Recursive algorithm
(iii) Divide and conquer algorithm
(iv) Randomized algorithm

Algorithm का उपयोग

(i) Algorithm मुख्यत: calculation और data processing में उपयोग होते हैं।
(ii) Algorithm का प्रयोग computer सम्बन्धित problem को solve करने में करते हैं।
(iii) Algorithm का प्रयोग program design में करते हैं।

Algorithm के उदाहरण

(i) उदाहरण : दो संख्याओं के घटाव का एल्गोरिथ्म

Step 1 : Start.
Step 2 : Read two numbers A and B.
Step 3 : Difference = A – B.
Step 4 : Display difference.
Step 5 : Stop.

(ii) उदाहरण : दो संख्याओं के addition का एल्गोरिथ्म

Step 1 : Start.
Step 2 : Read two numbers A and B.
Step 3 : SUM = A + B.
Step 4 : Display SUM.
Step 5 : Stop.

(iii) उदाहरण : किसी number का फ़ैक्टोरियल (factorial) प्राप्त करने का एल्गोरिथ्म

किसी नंबर का फ़ैक्टोरियल (factorial) निकालने के लिए उस नंबर को उससे 1 घटाकर multiply करते जाते हैं जब तक कि उससे 1 घटाते-घटाते अंतिम नंबर 1 ना हो जाए। उदाहरण के लिए अगर हमें 5 का factorial निकालना है तो उसे निम्नलिखित तरीके से निकालें :

$5! = 5 * (5 - 1) * (4 - 1) * (3 - 1) * (2 - 1) * (1 - 1)!$

5! = 5 * 4 * 3 * 2 * 1

0 का factorial (0!) = 1 होता है और 1 का factorial (1!) = 1 ही होता है।

Algorithm :

1. Start
2. Read the number n
3. i = 1, fact = 1
4. Repeat Step 5 to 6 until i = n
5. fact = fact * i
6. i = i + 1
7. Display : fact
8. Stop

(iv) उदाहरण : किसी दिये गये तीन numbers से सबसे largest number प्राप्त करने का एल्गोरिथ्म।

1. Start
2. Input the three values a, b, c.
3. Assign biggest = a.
4. If b > biggest, then assign biggest = b.
5. If c > biggest, then assign biggest = c.
6. Display biggest.
7. Stop.

(v) उदाहरण : किसी दिये गये number N का Fibonacci series प्राप्त करने का एल्गोरिथ्म।

Fibonacci series : Fibonacci series वह series है जिसमें series का प्रत्येक नया number, अपने दो previous numbers का addition होता है।

Example of Fibonacci series (for N = 7)

0, 1, 1, 2, 3, 5, 8

Algorithm :

Step 1 : Start

Step 2 : Declare variables N, N_1, N_2, N_3, i.

Step 3 : Read value of N.

Step 4 : यदि N < 2, display "Enter number > 2" और go to step 9.

Step 5 : Initialize variables $N_1 = 0$, $N_2 = 1$, i = 0

Step 6 : Print : N_1, N_2

Step 7 : $N3 = N_1 + N_2$

Step 8 : Repeat following statements until i < N – 2

Print : N_3

$N_1 = N_2$

$N_2 = N_3$

$N_3 = N_1 + N_2$

$i = i + 1$

Step 9 : Stop

(vi) उदाहरण : किसी दिये गये numbers के digits के sum प्राप्त करने का एल्गोरिथ्म।

Step 1 : Input N

Step 2 : Sum = 0

Step 3 : While (N ! = 0)

Rem = N % 10;

Sum = Sum + Rem;

N = N / 10;

Step 4 : Display : Sum

(vii) किन्हीं पाँच नम्बरों का औसत (Average) प्राप्त करने का एल्गोरिथ्म (Algorithm)

Step 1 : Start

Step 2 : Read a, b, c, d, e avg

Step 3 : Calculate avg = (a + b + c + d + e) / 5

Step 4 : Print avg

Step 5 : Stop.

(viii) Profit और Loss का algorithm

Step 1 : Input Income

Step 2 : Input Cost

Step 3 : if Income > Cost

Step 4 : Profit

Step 5 : else loss.

Step 6 : Stop.

(ix) दो नम्बरों के Swap का एल्गोरिथ्म

Algorithm : Thrid variable का प्रयोग करके)

Step 1 : Start

Step 2 : Read n_1, n_2

Step 3 : temp = n_1

Step 4 : $n_1 = n_2$

Step 5 : n_2 = temp

Step 6 : display n_1, n_2

Step 7 : Stop.

(x) दो नम्बरों के Swap का एल्गोरिथ्म (Without third variable)

Step 1 : Start

Step 2 : Read n_1, n_2
Step 3 : $n_1 = n_1 + n_2$
Step 4 : $n_2 = n_1 - n_2$
Step 5 : $n_1 = n_1 - n_2$
Step 6 : Display n_1, n_2
Step 7 : Stop.

(xi) किसी वर्ग (Square) का क्षेत्रफल (Area) प्राप्त करने का एल्गोरिथ्म

Step 1 : Start
Step 2 : Read X
Step 3 : Square = X * X
Step 4 : Print : Square
Step 5 : Stop

(xii) किसी Rectangle का क्षेत्रफल (Area) प्राप्त करने का एल्गोरिथ्म

Step 1 : Start
Step 2 : Read X, Y
Step 3 : Area = X * Y
Step 4 : Print Area
Step 5 : Stop

(xiii) किसी वृत्त (cicrle) का Area प्राप्त करने का एल्गोरिथ्म

Step 1 : Start
Step 2 : Read r
Step 3 : Area = 3.14 * r * r
Step 4 : Print : Area
Step 5 : Stop.

(xiv) Quadratic equation $ax^2 + bx + c = 0$ के Root प्राप्त करने का Algorithm लिखिये।

Step 1 : Start
Step 2 : Read a, b, c, D, x_1, x_2 real p, img p
Step 3 : Calculate $D = b^2 - 4ac$
Step 4 : If (यदि) $D \geq 0$

$$r_1 = \frac{-b + \sqrt{D}}{2a}$$

$$r_2 = \frac{-b - \sqrt{D}}{2a}$$

Print : r_1 and r_2
else calculate Real part और Imaginary part

real p. = b / 2a

imgp. = $\sqrt{(-D)}$ / 2a

Print : realp +j (imgp), realp–j (imgp)

Step 5 : Stop

▌एल्गोरिथ्म (Algorithm) के लाभ

(i) एल्गोरिथ्म (Algorithm) से problem को सही तरीके से समझा जा सकता है।

(ii) एल्गोरिथ्म (Algorithm) definite procedures को follow करता है।

(iii) Programming Knowledge के बिना भी algorithm से problem को आसानी से समझा जा सकता है।

(iv) किसी भी Problem को solve करने के लिए एक से ज्यादा algorithm के option होते हैं उसमें से हम सबसे सटीक algorithm का चुनाव कर सकते हैं।

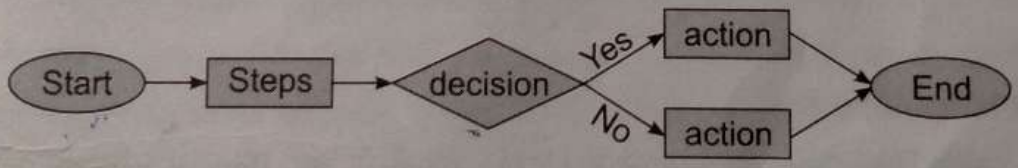
(v) एल्गोरिथ्म से हम Program की designing भी करते हैं।

(vi) एल्गोरिथ्म किसी Programming language पर depend नहीं करता हैं।

(vii) एल्गोरिथ्म के सभी चरण well defined होते हैं इसलिए किसी भी error को आसानी से identify कर सकते हैं।

1.4 फ्लोचार्ट (Flowchart)

किसी भी Algorithm का symbolic अथवा graphical रूपान्तरण Flowchart कहलाता है। Flowchart में process और उसके बहाव को अलग-अलग symbol से प्रदर्शित करते हैं। Flowchart के द्वारा हम problem को आसानी से समझ कर solve कर सकते हैं। Flowchart के द्वारा हम किसी problem को आसानी से analyze, design और manage कर सकते हैं। Flowchart मुख्यत: architectural, engineering, manufacturing education, science, technology इत्यादि में problem solving के रूप में प्रयुक्त होता है।



BASIC FLOW CHART
Fig. 1.2

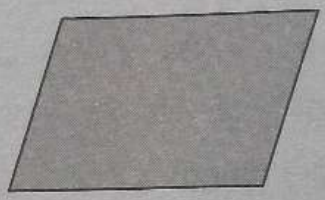
▌फ्लोचार्ट (Flowchart) के प्रकार

a. Document flowchart
b. Diagram flowchart
c. System flowchart
d. Data flowchart

▌Symbols of Flowchart

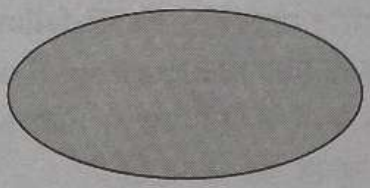
(i) इनपुट/आउटपुट (Input/Output)

यह समानान्तर चतुर्भुज (Parallel quadrilateral) का Symbol होता है जो इनपुट (Read) या आउटपुट (Print) के लिए प्रयोग किया जाता है।



(ii) टर्मिनल (Terminal)

यह अण्डाकार सिंबल है जो process को Start, End और Pause करने के लिए प्रयोग होता है। किसी फ्लो चार्ट में पहला और अंतिम Symbol टर्मिनल होता है।

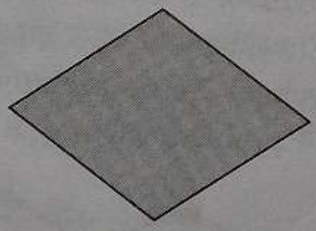


(iii) प्रक्रिया (Processing)

Processing के लिये rectangle के सिंबल का प्रयोग करते हैं जिसमें प्रोसेसिंग के Instructions होते हैं।

(iv) निर्णय (Decision)

Decision लेने और condition show करने के लिये Decision सिंबल का प्रयोग किया जाता है।



(v) फ्लो लाइन (Flow line)

फ्लो लाइन्स, फ्लोचार्ट के flow का डायरेक्शन (दिशा) बताती है और flow of control को भी समझाने में सहायक होती है।

(vi) कनेक्टर (Connector)

कनेक्टर (Connector) की सहायता से flow chart के अलग-अलग पार्ट को connect किया जाता है। फ्लो चार्ट अगर एक पेज से बड़ा है और वह अगले पेज पर भी आ रहा है तो कनेक्टर की सहायता से उसे जोड़ा जाता है। पिछले पेज के अंत में अंदर आती हुई रेखा (line) और अगले पेज के शुरू में बाहर जाती हुई रेखा (line) से प्रदर्शित किया जाता है।

(i) ऑन पेज-कनेक्टर

(ii) आफ पेज कनेक्टर

1.5 फ्लोचार्ट (Flowchart) और एल्गोरिथ्म में अंतर

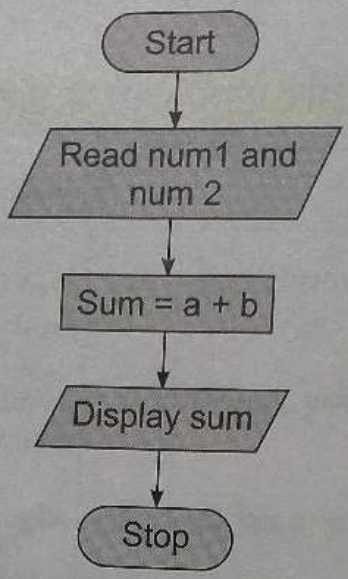
	Flowchart		Algorithm
1.	Algorithm के graphical conversion को flow chart कहते हैं।	1.	Algorithm में हम process को step by step analysis करते हैं।
2.	Process और Input/Output के लिए symbol का प्रयोग होता है।	2.	Symbol का प्रयोग नहीं करते हैं, केवल text का प्रयोग करते हैं।
3.	Flow chart को आसानी से समझा जा सकता है।	3.	Algorithm को समझना कठिन होता है।
4.	Flow chart को बनाने में symbol का प्रयोग होता है।	4.	Algorithm को लिखने में Text का प्रयोग होता है।
5.	Error को debuge करना difficult होता है।	5.	Error को debuge करना आसान होता है।
6.	Flow chart का उपयोग हम process को show करने के लिए कई क्षेत्रों (fields) में कर सकते हैं।	6.	Algorithm का उपयोग Mathematical calculation और Computer science के क्षेत्र (field) में करते हैं।

1.6 फ्लोचार्ट (Flowchart) के लाभ

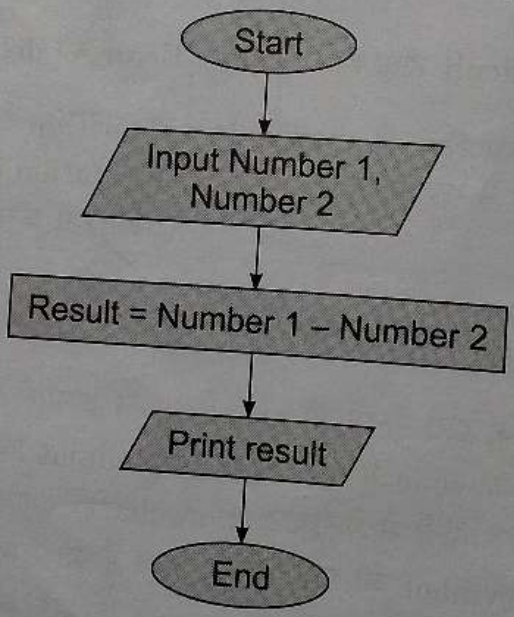
1. फ्लोचार्ट (Flowchart) के द्वारा किसी भी problem को आसानी से समझा जा सकता है।
2. फ्लोचार्ट (Flowchart) को code में आसानी से Implemnet किया जा सकता है।
3. Process को समझ कर आसानी से फ्लोचार्ट (Flowchart) बनाया जा सकता है।
4. Flow chart में केवल symbol का प्रयोग किया जाता है जो समझने में आसान होता है।
5. Flow chart के द्वारा हम system और उसके process को सही और सटीक तरह से analysis कर सकते हैं।
6. Flowchart के द्वारा हम clear और complete documentation तैयार कर सकते हैं।

1.7 फ्लोचार्ट (Flowchart) के उदाहरण

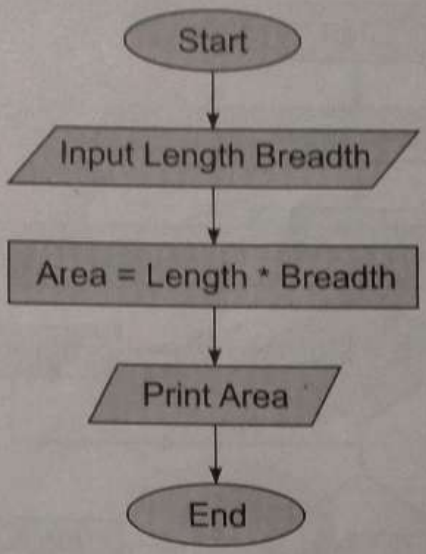
(i) किसी दो नम्बर को Add करने कर फ्लोचार्ट (Flowchart)



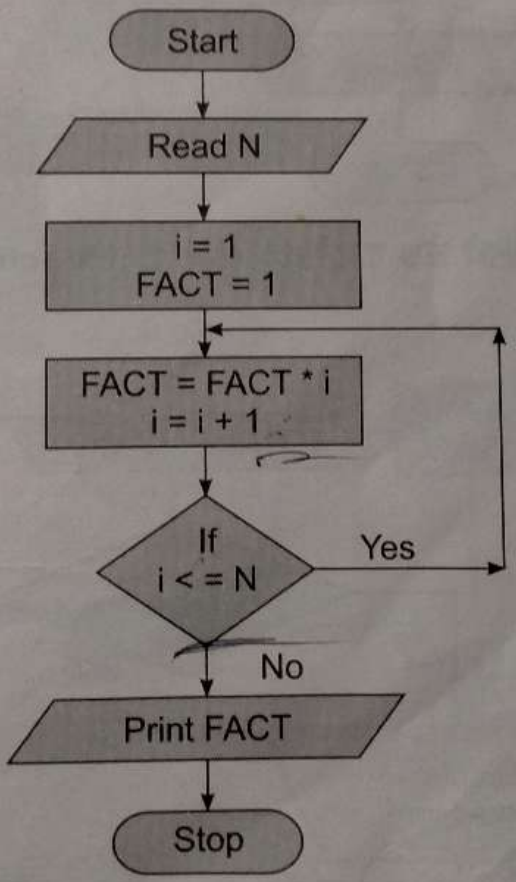
(ii) किन्हीं दो नम्बरों को SUBTRACT करने का फ्लोचार्ट (Flowchart)



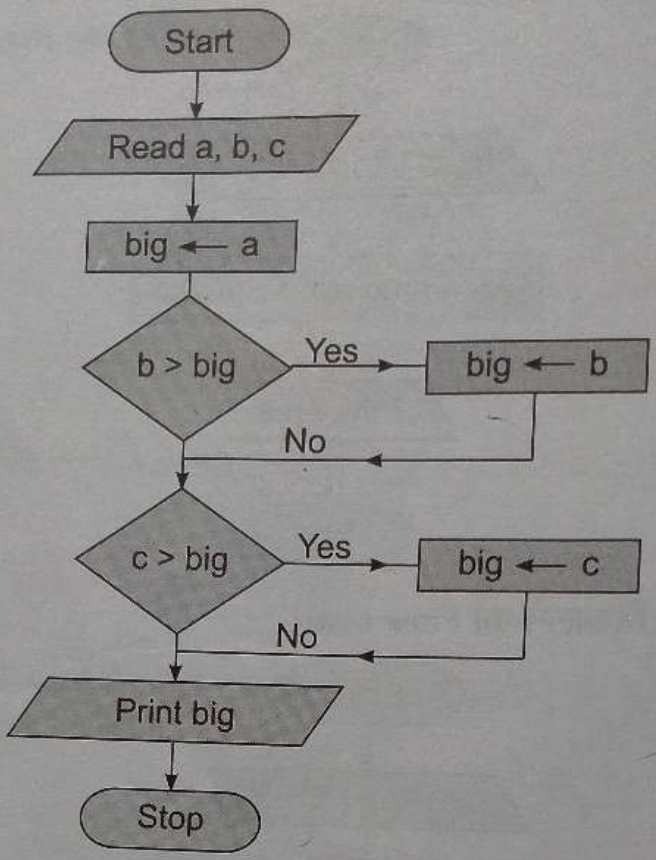
(iii) किसी Rectangle का Area calculate करने का फ्लोचार्ट (Flowchart)



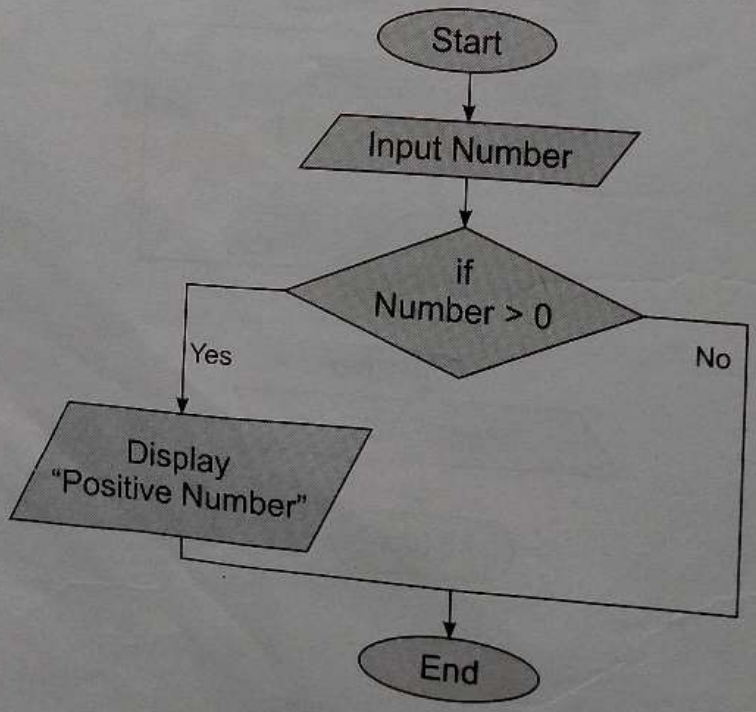
(iv) किसी नंबर के factorial निकालने का Flow chart



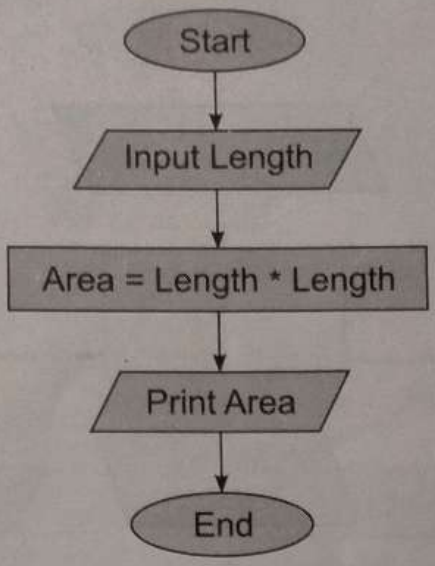
(v) तीन नंबरों से Largest number प्राप्त करने का Flowchart



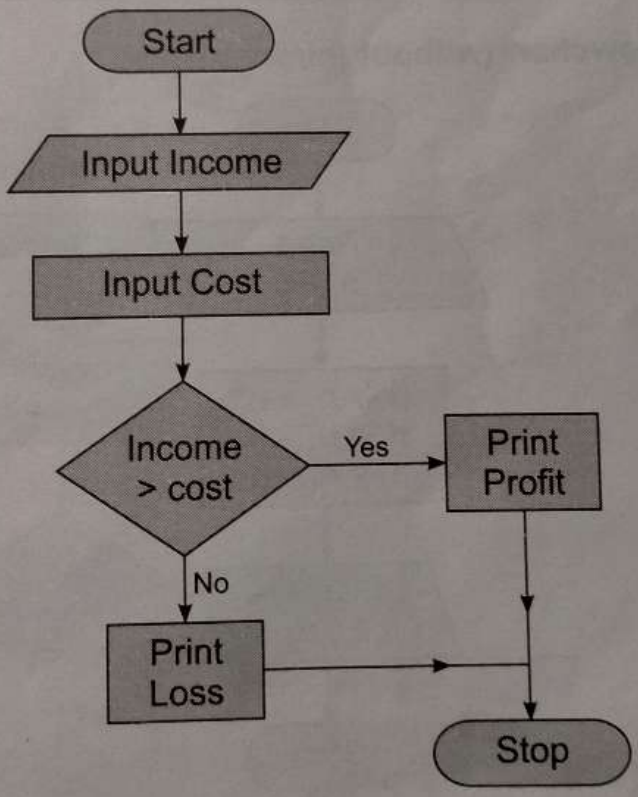
(vi) किसी नम्बर का positive के लिये चेक करने का फ्लोचार्ट (Flowchart)



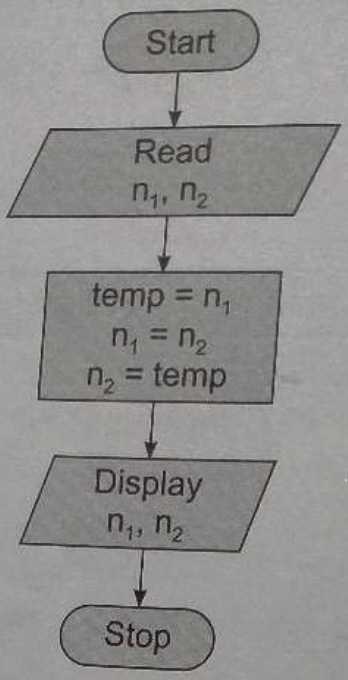
(vii) किसी square का Area calculate करने का फ्लोचार्ट (Flowchart)



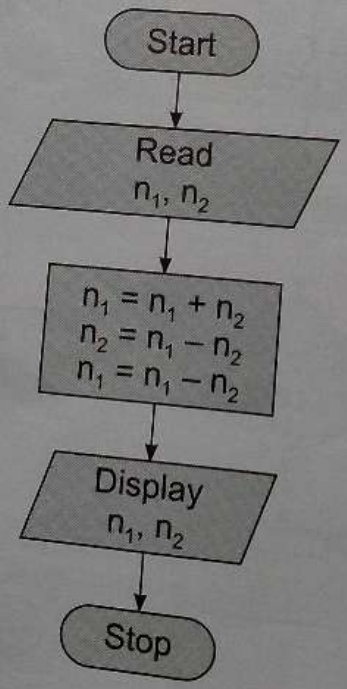
(viii) Profit और loss का Flowchart



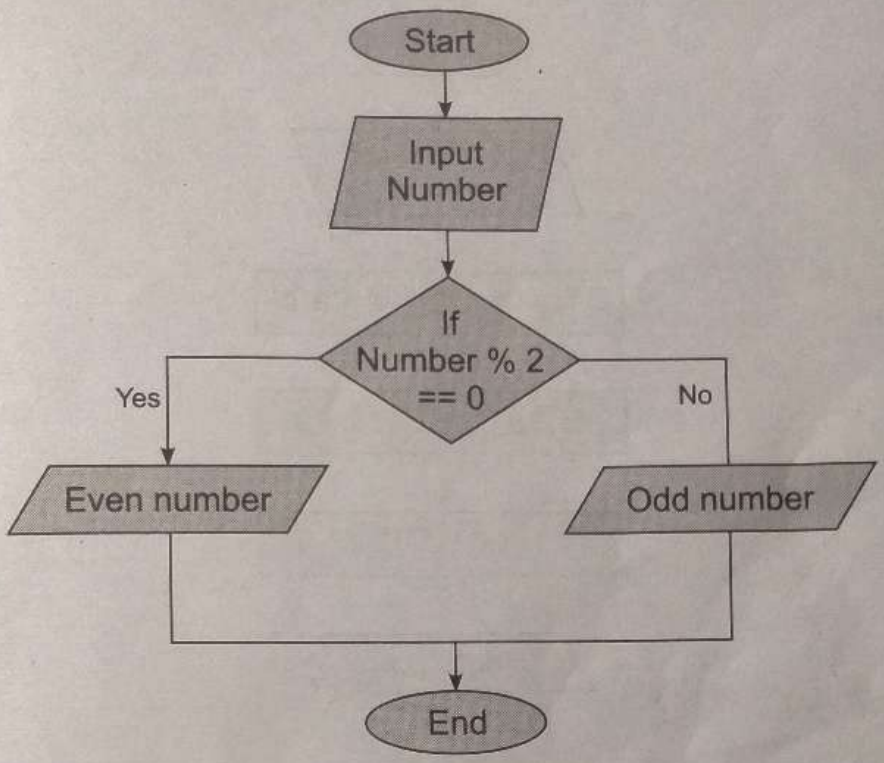
(ix) दो नम्बरों के Swap का Flowchart (Third variable का प्रयोग करके)।



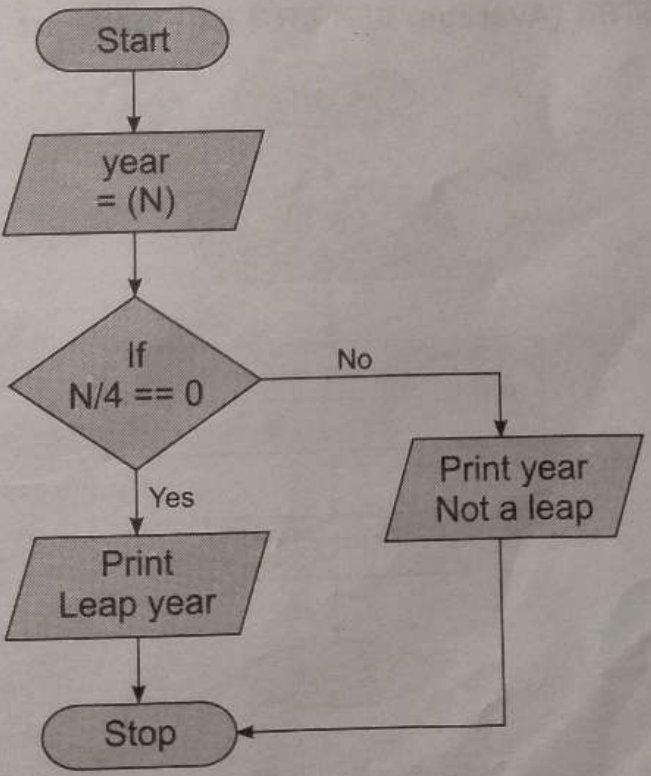
(x) दो नम्बरों के swap का Flowchart (without third variable)



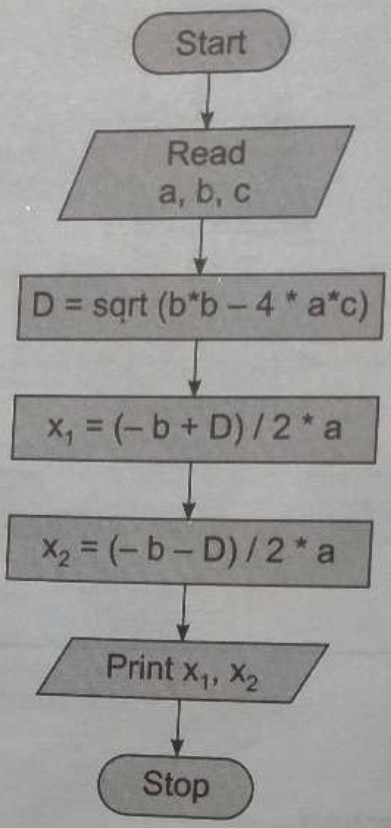
(xi) Odd या even नम्बर प्राप्त करने का Flowchart



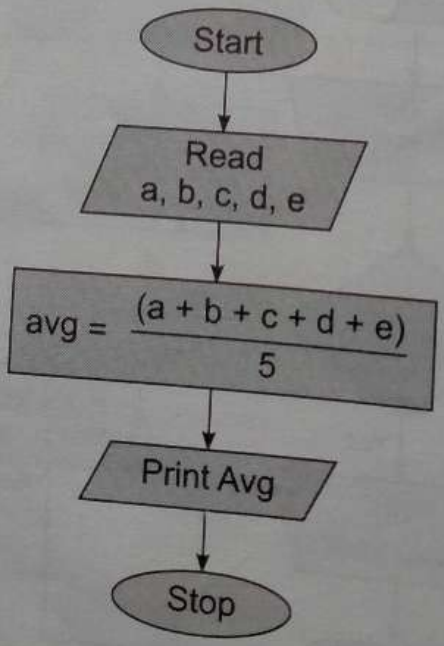
(xii) Leap year ज्ञात करने का Flowchart



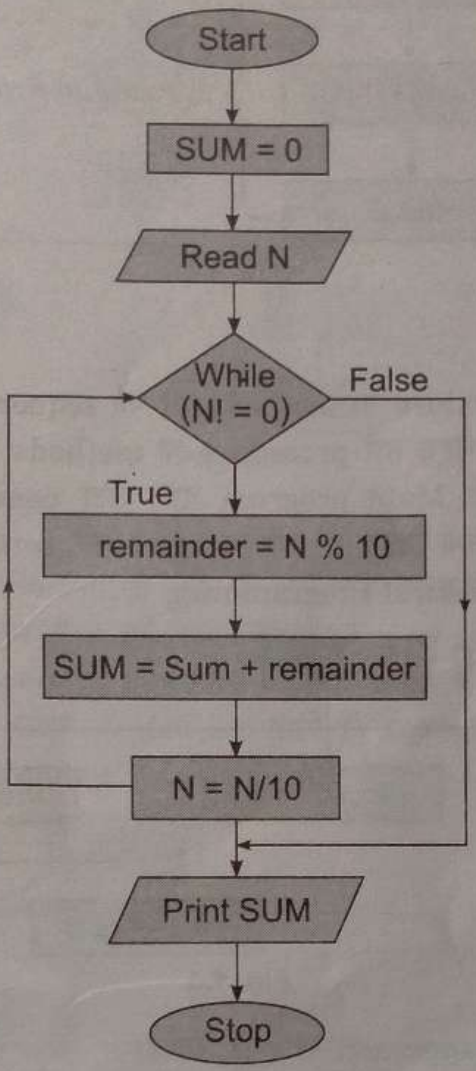
(xiii) Quadratic equation के Roots प्राप्त करने का Flowchart (यदि D ≥ 0)



(xiv) किन्हीं पाँच नम्बरों का औसत (Average) प्राप्त करने का Flowchart



(xv) किसी दिये गये numbers के digits के sum प्राप्त करने का Flowchart



1.8 प्रोग्रामिंग की विभिन्न तकनीक (Various Programming Techniques)

- Structural or Modular Programming
- Procedural Programming
- Top Down Designing
- Bottom up Designing
- Object Oriented Programming

(i) Structured Programming

Structured Programming को Modular Programming भी कहा जाता है। इस तकनीक में program बहुत सारे छोटे भागों में बाँट (divide) दिया जाता है जिन्हें modules, subprogram, subroutines या procedures कहा जाता है। सभी modules किसी specific कार्य को पूरा करने के लिये design किया जाता है। सभी modules एक दूसरे को affect किये बिना अपना कार्य perform करते हैं। Module पूरा होने पर और execution सही होने पर उसे अन्य module से जोड़कर पूरे program का structure तैयार कर लिया जाता है।

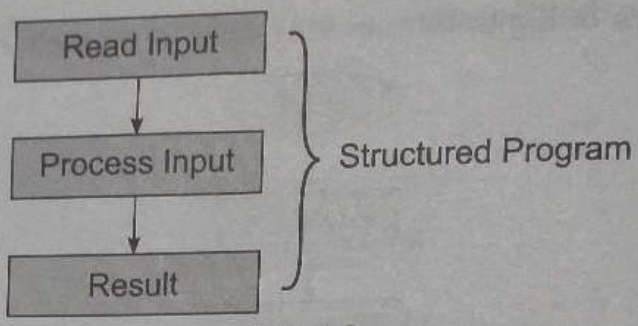


Fig. 1.3

(ii) Procedural Programming

इस प्रोग्रामिंग में प्रोग्राम एक procedure या method call का sequence होता है। यह प्रोग्रामिंग procedure के call पर निर्भर करता है। प्रोग्राम में बहुत सारे procedure या methods हो सकते हैं जो main program द्वारा या एक दूसरे के द्वारा call होते हैं। Main program द्वारा डेटा pass किया जाता है। Procedures द्वारा arguments के रूप में input लिया जाता है और output को return values के रूप में देता है। C programming, Pascal एक Procedural Programming है।

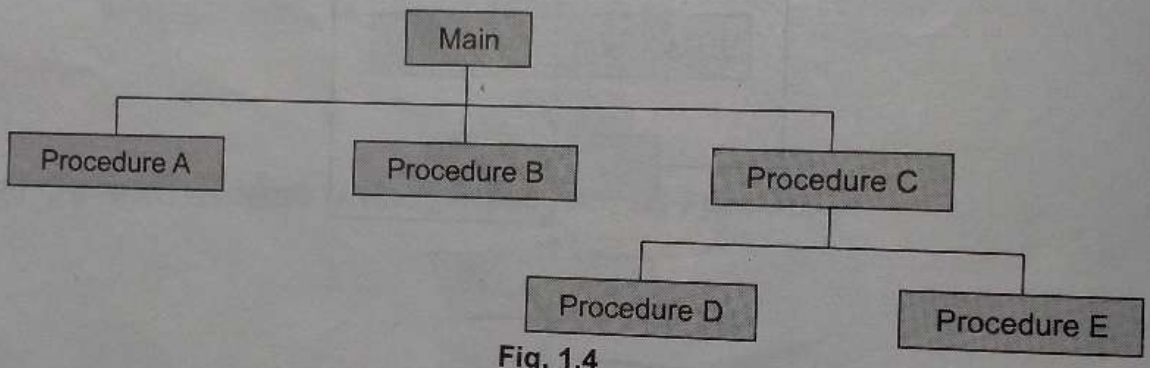


Fig. 1.4

1. यह प्रोग्रामिंग top down approach पर आधारित है।
2. प्रोग्राम को छोटे-छोटे procedure/module में रखा जाता है।
3. Data की तुलना में procedure को ज्यादा महत्व दी जाती है।
4. मुख्यत: Global data share किया जाता है।

(iii) Top Down Approach

Top down programming में प्रोग्राम को ऊपर से नीचे की ओर छोटी-छोटी sub part या sub module में divide कर दिया जाता है। प्रत्येक sub-program को solve करने के लिए अलग-अलग function लिखा जाता है। सबसे पहले Top module को test किया जाता है। उसके बाद सभी sub-modules को combine करके test किया जाता है।

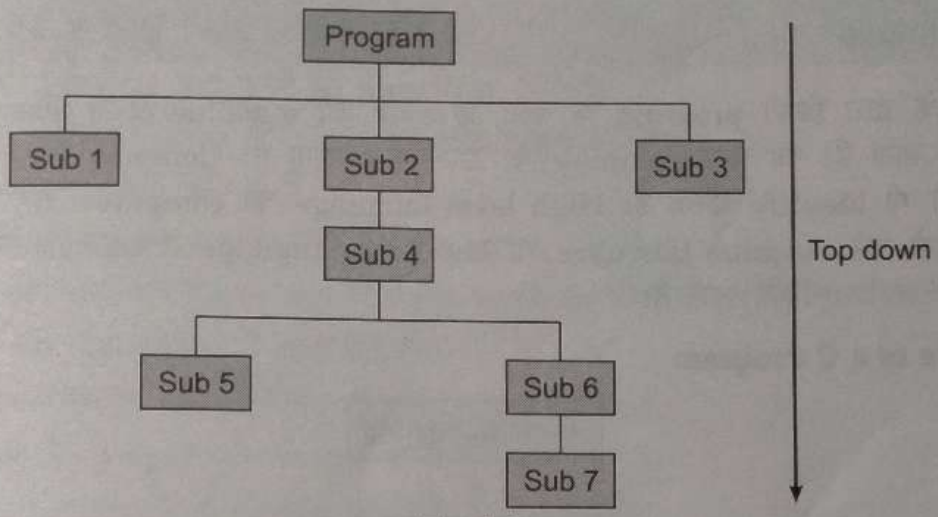


Fig. 1.5

(iv) Bottom up Approach

Bottom up programming में Bottom से top की ओर शुरुआत करते हैं। सबसे पहले Bottom layer के सभी modules की design करके test करते हुए ऊपर की ओर proceed करते हैं और उसके next layer के module को test करके फिर दोनों layers के modules को combine करके text किया जाता है। अंत में सभी modules को combine करके final प्रोग्राम को develop करते हैं। प्रोग्राम development और testing नीचे से ऊपर की तरफ होता है इसलिये इसे Bottom up Approach कहते हैं।

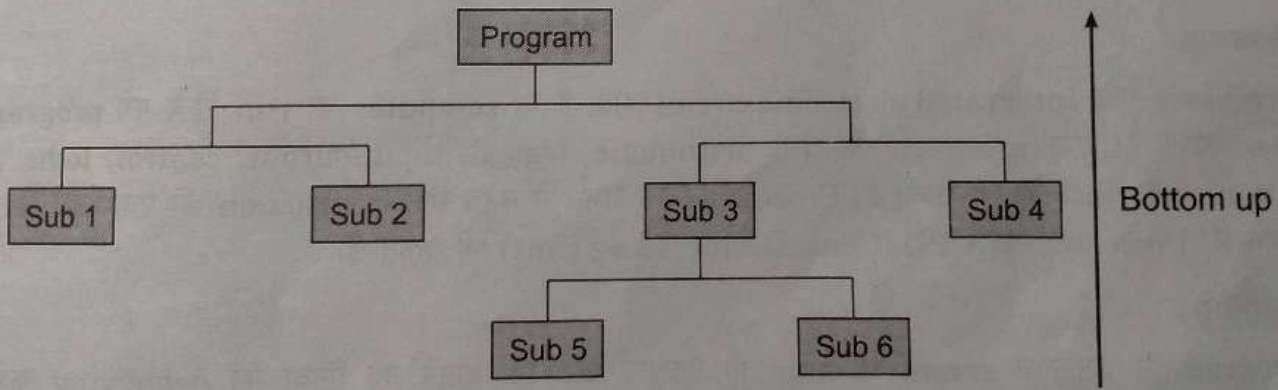


Fig. 1.6

(v) Object oriented programming

Object oriented programming में प्रोग्राम को बहुत सारे objects में decompose करते हैं। objects ही प्रोग्राम के मुख्य बिंदु होते हैं और सभी activities को object के रूप में ही लेते हैं। इसमें process की तुलना में objects को ही महत्व दिया जाता है। सभी data और functions को एक सिंगल entity के रूप में combine करते हैं। Object oriented प्रोग्रामिंग Bottom up Approach को support करता है। इसमें class create किया जाता है और ये class reusable होते हैं जिनको अलग प्रोग्राम के द्वारा reuse कर सकते हैं।

Object oriented programming निम्न concept को follow करता है—

(i) Class
(ii) Abstraction
(iii) Polymorphism
(iv) Encapsulation
(v) Inheritance
(vi) Overloading

1.9 Compiling

Compiling के द्वारा किसी program के source code को machine code (executable code) में transform किया जाता है। यह कार्य compiler के द्वारा किया जाता है। Compiler program में present syntax error को भी identify करता है। High level language को computer नहीं समझता है। इसलिए execution के पहले उसे machine language या low level language में convert करना पड़ता है। यह process compiler के द्वारा किया जाता है।

Compiling Stage of a C Program

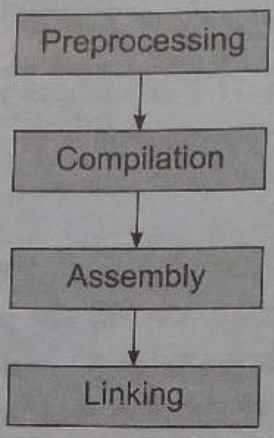


Fig. 1.7

Processors

Processor एक Integrated electronic circuit होता है जो computer पर run किये गये program को process करता है। एक processor के द्वारा arithmetic, logical, input/output, control, logic इत्यादि operation perform किये जा सकते हैं। Processor एक समय में बहुत सारे Instruction को एक साथ execute कर सकते हैं। Processor को CPU (Central Processing Unit) भी कहते हैं।

Debugging

Program में उपस्थित errors या bug को पहचान कर उसे हटाने की क्रिया को debugging कहते हैं। Debugging के द्वारा bug को identify करके उसे remove किया जाता है। Debugging process को dubugger tool (एक program) की सहायता से perform किया जाता है।

▶ प्रश्नावली (Exercise) ◀

1. Programming language क्या है? इसकी क्या आवश्यकता है?
2. Programming language कितने प्रकार के होते हैं? वर्णन कीजिए।
3. Machine language क्या होता है? उदाहरण के साथ समझाइए।
4. Low level programming language और High level programming language में अन्तर स्पष्ट कीजिए।
5. Third generation और fourth generation programming language क्या होते हैं?
6. Computer program के development के विभिन्न चरणों को विस्तार से समझाइए।

7. Algorithm क्या होता है? इसकी उपयोगिता क्या है? (UPBTE 2016)
8. किसी नम्बर के factorial प्राप्त करने का algorithm लिखिए।
9. किसी नबर के digits का sum प्राप्त करने का algorithm लिखिए।
10. Flowchart के लाभ कौन-कौन से हैं? (UPBTE 2008)
11. किन्हीं 5 नम्बरों का average प्राप्त करने का algorithm लिखिए।
12. Algorithm क्या होता है? Flowchart में प्रयुक्त symbols को उदाहरण के साथ समझाइए।
13. Flowchart और algorithm में अन्तर लिखिए।
14. Flowchart के लाभ का वर्णन कीजिए। (UPBTE 2008)
15. किन्हीं दो नम्बरों के swap का flowchart बनाइए।
16. विभिन्न Programming Techniques का संक्षेप में वर्णन कीजिए। (UPBTE 2008)
17. Top down और Bottom up programming तकनीक का वर्णन कीजिए।
18. Quadratic equation के roots प्राप्त करने का flowchart बनाइए।
19. Structured programming क्या होता है? (UPBTE 2018)
20. Procedural programming को समझाइए।

❑

Unit

2

Program Structure

Program Structure

2.1 C Language

C एक structured computer programming language है जो program, system software या application software के development में प्रयोग किया जाता है। C computer language को सन् 1972 में Dennis Ritchie ने develop किया था। यह एक बहुत ही popular और user friendly computer language है जो initial में Unix operating system को develop और support के लिये विकसित किया गया। C एक Base programming है जिसका प्रयोग operating system और विभिन्न प्रकार के Application software बनाने में किया जाता है। C Language कई प्रकार के data type को भी support करती है जो प्रोग्राम बनाने में सहायक होते हैं। C Language ANSI (American National Standards Institute) और ISO (International Organization for Standardization) के द्वारा भी प्रमाणित है। C Language विभिन्न प्रकार के Compiler और DBMS (Data Base Management System) develop करने के लिए प्रयोग में आता है। C Language एक Portable language है जिसका program एक machine पर लिख कर दूसरे machine पर execute करा सकते हैं। इस language में High level language और low level language दोनों के गुण हैं। यह एक system independent language है। C Language में 32 keywords हैं जिसका meaning compiler को पहले से ही defined है जो program को develop करने में प्रयोग किये जाते हैं।

2.2 C Language की विशेषता

C एक बहुत ही popular और user friendly computer language है। बहुत सारी विशेषताओं के कारण आज भी C काफी लोकप्रिय है।

(i) C Language एक structured programming language है। C Language में हम किसी भी program को आवश्यकता के अनुसार छोटे-छोटे modules में बना सकते हैं जिससे किसी बड़े problem को आसानी से समझ कर, उसमें उपस्थित error को remove करके solve कर सकते हैं।

(ii) C Language बहुत ही fast है और तुरन्त ही execute हो जाती है।

(iii) C Programming Language को आवश्यकता के अनुसार extend भी कर सकते हैं। जरूरत पड़ने पर उसमें नये function या module आसानी से add कर सकते हैं।

(iv) C Language में बहुत सारे built in function और operators होते हैं जो किसी program (प्रोग्राम) को लिखने में काम आता है।

(v) C Language का प्रयोग करके embedded system को भी develop करते हैं जो hardware और software को एक साथ support करता है।

(vi) C Language में बहुत सारे inbuilt library functions होते हैं। हम अपना function बना कर C library मे add कर सकते हैं।

(vii) C Language Portability को support करता है। हम C Language में लिखे program को एक machine से दूसरे machine में port करा के execute करा सकते हैं।

(viii) C Language में लिखे प्रोग्राम को आसानी से modify भी कर सकते हैं।

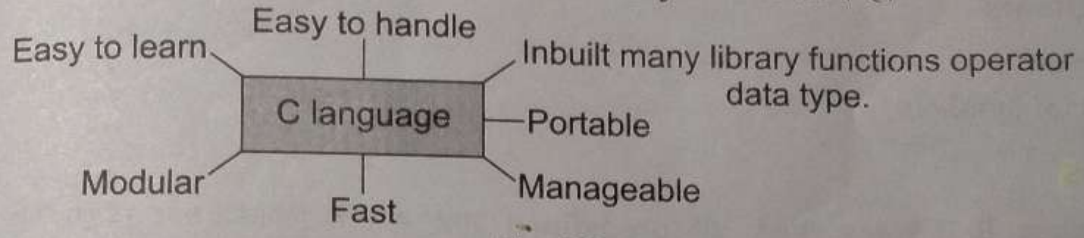


Fig. 2.1

2.3 C Language का प्रयोग

(i) C Language का प्रयोग करके विभिन्न प्रकार के system या application software develop कर सकते हैं।

(ii) C Language का प्रयोग करके compiler भी बना सकते हैं जो किसी प्रोग्राम को compile करने के लिए प्रयोग किया जाता है।

(iii) C Language का प्रयोग करके हम data को manage करने के लिए database management system (DBMS) भी बना सकते हैं।

(iv) C Language का प्रयोग network device के software बनाने में भी किया जाता है।

(v) C Language का प्रयोग graphical related application और computer या mobile games के software बनाने में भी किया जाता है।

2.4 C Language का Limitation

OOPS

(i) C Language object oriented concept को support नहीं करता है जो इसकी flexibility को कम करता है।

(ii) C Language namespace को support नहीं करता जिसके कारण हम दो variables को एक ही नाम से declare नहीं कर सकते हैं।

(iii) C Language data security के मामले में कम reliable है।

(iv) C Language में constructor और destructor का प्रयोग नहीं कर सकते है।

(v) C Language run time type checking को नहीं support करता है यह केवल compile time type checking को ही support करता है।

(vi) C Language inheritance और polymosphism को support नहीं करता है।

 POP

2.5 C TOKENS

Token प्रोग्राम का सबसे smallest unit होता है। यह C Language का basic building block होता है जिनको compiler पहचान (recognize) करता है। Tokens ही combined होकर C program को construct करते हैं। C Language में विभिन्न प्रकार के tokens का प्रयोग किया जाता है।

- Keywords
- Identifiers
- Operators
- Constants
- Strings
- Special symbols

Keywords

C Language में कुछ reserved और predefined शब्द होते हैं जिनका कुछ विशेष अर्थ होता है जिन्हें Keywords कहते हैं। ये syntax के parts होते हैं। प्रोग्रामिंग करते समय इन्हें variable अथवा function name या identifier की तरह use नहीं कर सकते हैं। C programming में 32 keywords होते हैं। compiler को सभी Keywords की meaning पहले से ही define रहता हैं।

C Programming के keywords

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

Fig. 2.2

उदाहरण

```
# include < stdio.n >
main ( )
{
int m, n;
Printf (" keyword in C")
return 0;
}
```

दिये गये उदाहरण में int और return keywords हैं। int के द्वारा variables m और n को declare किया गया है।

Identifiers

किसी variables, constants, functions या user define डेटा के दिये गये नाम को ही Identifier कहते हैं।

प्रोग्राम में identifier यूनिक होते हैं। ये user के द्वारा define किये जाते हैं। Identifier में alphanumeric character और underscore हो सकते हैं। Identifier कभी भी कोई digit के साथ शुरुआत नहीं हो सकता है।

Valid Identifier — abc, _ x, m _, t _ 1.

Invalid Identifier — 1u, char, int

उदाहरण : int abc = 89

इसमें abc एक identifier है जो int टाइप variable को denote करता है जिसकी value 89 है।

Identifier बनाने के नियम

1. एक valid identifier letters (uppercase A-Z और lowercase letters a-z), digit (0-9) या underscores (_) से बना होता है।
2. Identifier का पहला अक्षर letter या underscores होना चाहिये।
3. Identifier case sensitive होते हैं। उदाहरण के लिये sing और Sing दो अलग-अलग Identifiers हैं।
4. Keyword को Identifier की जगह पर use नहीं कर सकते हैं।

Keyword और Identifier में अन्तर

S.No.	Keyword	Identifier
1.	Keyword पहले से ही defined होते हैं।	Identifier User के द्वारा defined होते हैं।
2.	Keyword केवल lowercase में लिखे जाते हैं।	Identifier दोनों lowercase और uppercase में लिखे जा सकते हैं।
3.	इसका meaning पहले से ही compiler को define रहता हैं।	इसका meaning पहले से compiler को नहीं define रहता है।
4.	ये alphabetic characters के Combination से बनते हैं।	ये alphabetic alphanumeric के combination से बनते हैं।

Operators

Operator एक प्रकार का operational symbol होता है जिसका meaning पहले से ही compiler को define होता है। यह mathematical या logical functoin की तरह कार्य करता है। यह operand पर किसी specific कार्य को perform करने के लिए apply होता है। सभी operator का meaning अलग-अलग होता है।

उदाहरण — +, –, *, /, % etc.

Operators के प्रकार

- Arithmetic Operators

- ❖ Increment and Decrement Operators
- ❖ Logical operator
- ❖ Relational Operators
- ❖ Assignment operator
- ❖ Bitwise Operators
- ❖ Conditional operator
  - ❖ Size of Operator
  - ❖ Ternary Operator
- ❖ Comma operator

Arithmetic operators

Arithmetic operator का प्रयोग numerical values (operands) पर mathematical operation (add, subtract, multiply or division) करने के लिये किया जाता है।

उदाहरण — addition operator (+), दोनों operands को add करने के लिये प्रयोग किया जाता है।

Operator	Meaning
+	addition
–	Subtraction/minus
*	multiplication
/	division
%	remainder ater division (modulo division)

Fig. 2.3

Arithmetic operators का उदाहरण

```
// C Program to explain the working of arithmetic operators
#include < stdio.h >
int main()
{
int a = 12, b = 6, c;
c = a + b;
printf("a + b = %d \n", c);
c = a – b;
printf("a – b = % d \n",c);
c = a * b;
printf("a * b = % d \n", c);
c = a / b;
```

```
printf("a / b = %d \n", c);
c = a % b;
printf("Remainder when a divided by b = %d \n", c);
return 0;
}
```

Output

a + b = 18
a – b = 6
a * b = 72
a / b = 2
Remainder when a divided by b = 0

Explanation :

f गये उदाहरण में (+) operator दो operands a और b को add करता है और result को print करता है।

(–) operator दो operands a और b पर apply होकर b को a से subtract करता है।

(*) operator दोनों operands a और b को multiply करके Result को Print करता है।

/ (operator) operand a को operand b से divide करता है।

(%) operator operand a को operand b से divide करके remainder को print करता है।

Increment and Decrement Operators

Increment और decrement operators एक तरह का unary operator है जो एक operand पर apply होता है। यह operator उसी operand में value (increase) add करता है या उस operand से value को subtract (decrease) करता है।

Increment और Decrement operator के दो forms होते हैं—

(i) Postfix – Operator, operand के बाद आता है जैसे a ++, a – –

(ii) Prefix – Operator, operand के पहले आता है जैसे + + a, – – a

Increment and Decrement Operators का उदाहरण :

```
// C Program to Explain the working of increment and decrement operators
(i)#include <stdio.h>
   int main()
   {
   int m = 99, n = 96;
   float o = 90.5, p = 99.5;
   printf (" + + m = % d \n", + + m);
   printf ("– – n = % d \ n", – – n);
   printf (" + + o = % f \n", + + o);
   pfintf (" – – p = % f \n", – – p);
   return 0; }
```

Output

++m = 100
--n = 95
++o = 91.500000
++p = 98.500000

Explanation

दिये गये program में (pre-increment और pre-decrement) operator का प्रयोग हुआ है।

++m,

पहले m की value बढ़ेगी तब प्रिन्ट होगा, 99 से बढ़कर 100 हो जायेगी।

--n

पहले n की value घटेगी तब प्रिन्ट होगा, 96 से घटकर 95 हो जायेगी।

(ii)

```
# include < stdio.h >
 void main()
 {
 int m, n, x = 96, y = 96
 m = x + + ;
 n = + + y ;
 printf ("m will be : % d", m);
 printf ("n will be : %d", n);
 }
```

Output

m will be : 96
n will be : 97

Explanation

दिये गये उदाहरण में Pre-increment और post-increment operator का प्रयोग हुआ है।

Post increment x ++ पहले x execute होगा फिर उसका मान बढ़ेगा।

परन्तु Pre-inrement में पहले मान बढ़ेगा फिर execute होगा इसलिए

m = 96
n = 97

Logical Operators

Logical operator का प्रयोग logical operations को perform करने के लिए किया जाता है। C language में तीन प्रकार के logical operators हैं—logical AND (&&), logical OR (||) और logical NOT (!) Logical operator का प्रयोग मुख्यत: program के flow को control करने के लिये किया जाता है।

&& और || दोनों Binary operators हैं और ! एक unary operator है।

Operator (symbol)	Opeator का मतलब	उदाहरण
&&	Logical AND. True only if all operands are true	If a = 4 and b = 3, then expression ((a == 4) && (b == 3)) equals to 1
||	Logical OR. True only if either one operand is true	If a = 4 and b = 2, then expression ((a == 4) || (b > 5)) equals to 1.
!	Logical NOT. True only if the operand is 0	If a = 4, then expression ! (a ==4) equals to 0.

Logical Operators का उदाहरण

(i)
```
// C Program to explain the working of logical operators
#include <stdio.h>
int main()
{
int m = 75, n = 75, c = 85, result;
result = (m == n) && (c > n);
printf ("(m == n) && (c > n) equals to %d \n", result);
result = (m == n) && (c < n);
printf ("(m == n) && (c < n) equals to %d \n", result);
result = (m == n) || (c < n);
prinftf ("(m == n) || (c < n) equals to %d \n", result);
result = (m ! = n) || (c < n);
printf ("(m! = n) || (c <n) equals to %d \n", result);
return 0;
}
```

Output

```
(m == n) && (c > n) equals to 1
(m == n) && (c > n) equals to 0
(m == n) || (c < n) equals to 1
(m ! = n) || (c < n) equals to 0
```

Logical Operators का Explanation

- (m == n) && (c > 5) evaluates to 1 because both operands (m == n) and (c > n) is 1 (true).
- (m == n) && (c < n) evaluates to 0 because operand (c < n) is 0 (false).
- (m == n) || (c < n) evaluates to 1 because (m = n) is 1 (true).
- (m! = n) || (c < n) evaluates to 0 because both operand (m! = n) and (c < n) are 0 (false).

Relational Operators

Relational Operator का प्रयोग दो variables के values को compare करने के लिये किया जाता है। इसका प्रयोग दो variables के बीच relation जानने के लिये किया जाता है। Relational operator expression के true होने पर 1 return करता है और expression के false होने पर 0 return करता है।

Operator /symbol	Operator का मतलब	उदाहरण
==	Equal to	87 == 87 returns 1
>	Greater than	87 > 78 returns 1
<	Less than	87 < 78 returns 0
!=	Not equal to	87 != 78 returns 1
>=	Greater than or equal to	87 >= 78 returns 1
<=	Less than or equal to	87 <= 78 returns 0

(i) उदाहरण

```
# include <stdio.h>
int main()
   {
      int a = 95, b = 75;
      if ( a == b)
        {
      printf ("a and b are equal");
        }
          else
          {
          printf("a and b are not equal");
          }
          }
```

Output :

a and b are not equal.

इस प्रोग्राम में relational operator (==) का प्रयोग दो variables a और b के value को compare करने के लिये किया गया है। यदि दोनों के values एक ही (equal) हैं तो स्क्रीन पर a and b are equal display होगा। यदि दोनों variables का value equal नहीं है तो स्क्रीन पर a and b are not equal display होगा।

(ii) उदाहरण

```
# include <stdio.h>
int main()
{
int m = 94, n = 88;
printf ("m is % d\n", m);
printf ("n is % d\n", n);
printf ("m > n : % d\n", m > n);
printf ("m < n : % d\n", m< n);
return 0;
}
```

Output

```
m is 94
n is 88
m > n : 1
m < n : 0
```

इसमें m > n True है इसलिए 1 return होगा।

इसमें m < n False है इसलिए 0 return होगा।

Assignment operator

Assignment operator एक ऐसा operator है जो right side के operand के value को left side operand में assign करता है।

Assignment operator का प्रयोग करके एक ही variable को program execution के different चरण में अलग-अलग value दे सकते हैं। assignment operator को equal (=) sign से denote करते हैं।

Operator /symbol	Opeator का मतलब	उदाहरण
=	assignment operator (यह Operator right side operands के values को left side operand में assign करता है।	M = n * o (n * o value M में assign होगी।)
+=	यह Operator right side operands के values को left side operand में Add करके परिणाम को left side operand में assign करता है।	M += N is same as M = M + N
−=	यह Operator right side operands के values को left side operand के values से Subtract करके परिणाम को left side operand में assign करता है।	M −= N is same as M M = N − N

*=	यह Operator left side operands के values को right side operand के values से multiply करके परिणाम को left side operand में assign करता है।	M *= N is same as M = M * N
/=	यह Operator right side operands के values को left side operand से divide करके परिणाम को left side JeÛecelebokeâ में assign करता है।	M /= N is same as M = M / N
%=	यह Operator दो operands के modulus values को left side operand में assign करता है।	M %= N is same as M = M % N

Assignment operator के उदाहरण

(i)
```
# include <stdio.h>
  int main()
  {
  int abc ;
  abc = 97 ;
  return (0);
  }
```

इस प्रोग्राम में हम 97 को abc में assign कर रहे हैं।

(ii)
```
# include <stdio.h>
  int main()
  {
  int a;
  a = 95
  a + = 2;
  printf ("\n result 1 = % d", a);
  a = 95
  a – = 2
  printf ("\n result 2 = % d", a);
  return (0);
  }
```

Output

Result 1 = 97

Result 2 = 93

Bitwise Operator

Bitwise Operator bit पर apply होते हैं। Decimal value पहले bit के sequence में convert होते हैं तब bitwise operator उन पर apply होते हैं।

C language में निम्न bitwise operator हैं—

& (bitwise AND), << (Left shift)
| (bitwise OR), >> (Right shift)
Λ (XOR), ~ (bitwise not)

Conditional Operator

C program में conditional operator का प्रयोग condition check करने के लिए किया जाता है। इसको निम्न नामों से भी जानते हैं—

a. Ternary operator
b. ? : operator

Conditional operator का syntax :

expression 1 ? expression 2 : expression 3

- Syntax में question mark " ? " if part को represent करता है।
- expression 2 और expression 3 का execution expression 1 के execution के return value पर निर्भर करता है।
- यदि (expression 1), execution के बाद true return करता है तो (expression 2), execute होगा।
- यदि (expression 1) execution के बाद false return करता है तो (expression 3) execute होगा।

(i) Conditional Operator का उदाहरण

```
# include < stdio.h >
int main()
{
int m = 5, n;
n = (m == 5 ? 98 : 0);
printf ("m value is % d\n", m);
printf ("n value is % d", n);
}
```

Output

```
m value is 5
n value is 98
```

Special Operator

Operator	Opeator का मतलब	उदाहरण
*	किसी variable का pointer	* m, m variable का pointer
sizeof	किसी variable का size return करता है।	sizeof (m), m variable का size return करता है।
&	किसी variable का address return करता है।	&m, variable m का Address return करता है।

(i) * (operator) का उदाहरण

```
# include < stdio.h >
int main()
{
int * ptr, m;
m = 98
ptr = &m;
printf ("% d", * ptr);
return 0;
}
```

Output 98

(ii) & (operator) का उदाहरण

```
# include < stdio.h >
int main()
{
int m = 98;
printf ("value : % d\n", m);
printf ("Address : % u\", & m);
return 0;
}
```

Output

```
value : 98
Address : 75684894
```

(iii) size of (operator) का उदाहरण

```
# include < stdio.h >
int main()
{
printf ("% d\n", sizeof (int));
printf ("% d\n", sizeof (char));
printf ("% d", sizeof (float));
return 0;
}
```

Output

```
2
1
4
```

Comma Operator

C program में comma operator comma (,) एक separator और operator दोनों की तरह प्रयोग होता है।

Comma as a separator

उदाहरण (i) int m, n, o;

इस declaration में comma एक separator है और compiler को बताता है कि m, n और o अलग-अलग तीन variables हैं।

Comma as a operator

(ii) m = 10, 20, 30; → इसमें m को 10 assign होगा क्योंकि assignment operator (=) की priority comma (,) से ज्यादा होता है।

(iii) Comma as an operator : n = (10, 20, 30); इसमें n को 30 assign होगा क्योंकि right-most value expression मे assign होता है।

2.6 CONSTANT

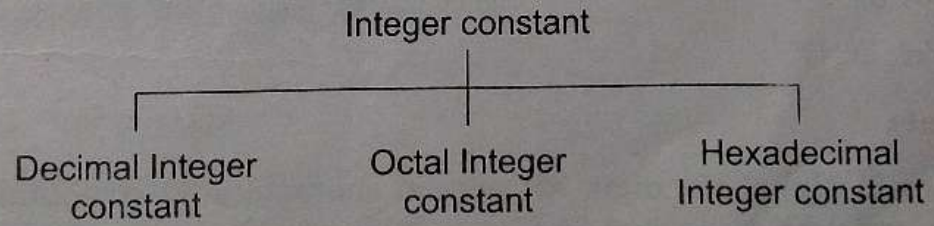
Constant fixed values को refers करता है। इसका value प्रोग्राम execution में कभी नहीं बदलता है। इसको literals भी कहते हैं। किसी भी datatypes का constant हो सकता है। C program में निम्न प्रकार के constant होते हैं।

TYPES OF CONSTANT :

1. Integer constants
2. Real or floating point constants
3. Character constants
4. String constants
5. Backslash character constants

Integer constants

Integer constant में केवल integer value (–ve or +ve) होते हैं। किसी भी integer constant में कम से कम एक digit अवश्य होने चाहिए। इसमें decimal part नहीं होता है।



Integer constant तीन प्रकार के होते हैं—

(i) Decimal Integer constant

(ii) Octal Integer constant

(iii) Hexadecimal Integer constant

Decimal Integer constant

Decimal constant में 0 से 9 तक के digits या digits का sequence हो सकता है Decimal constant का Base 10 होता है।

उदाहरण : 2
4, 7
965, 35792

Octal Integer constant

Octal Integer constant में 0 से 7 तक के digits या digits का sequence हो सकता है जिसमें zero (0) prefix लगा होता है। अर्थात् पहले 0 और उसके बाद 0 से 7 तक का digit रहता है। इसका base 8 होता है।

उदाहरण : 012,
036,
047

Hexadecimal Integer constant

Hexadecimal constant 0 से F तक के कोई भी combination से बनता है जिसके पहले prefix की तरह ox या OX लगा रहता है। इसका base 16 होता है।

उदाहरण : ox69, OX7A, OX8F

Real या Floating point constants

Floating point constant में integer part, एक decimal point, एक fractional part और एक exponent part होता है। Real या floating point constant में कम से कम एक digit अवश्य होना चाहिए।

Real या Floating point constant दो तरीके से लिखे जा सकते हैं—

1. Fractional form — 5.96, – 99.75
2. Exponential form — 5.9 e 6, – 6.8 e – 9

Character constants

Character constant में केवल एक alphabet, digit या special character हो सकता है जो 'single quotes' के अन्दर रहते हैं।

उदाहरण : 'M', 'n', '9', '7'.

String constants

String constant में alphabets, digit और special characters का combination हो सकता है जो "double quotes" के अन्दर होते हैं। यह किसी भी length के हो सकते हैं।

उदाहरण : "C langulage" "std001p"

Backslash character constants

Backslash character (\) का प्रयोग escape sequence के लिए किया जाता है। सभी escape sequence का एक unique ASCII value होता है। ये non-printable character होते हैं।

उदाहरण : '\t' → (Horizontal Tab)

'\n' → (new line)

'\b' → (control back to one position)

Backslash character constants

(i) इसमें सभी characters back slash (\) से start होते हैं।

(ii) ये Non-printable characters होते हैं।

(iii) सभी escape sequence का अपना unique ASCII value होता है।

प्रोग्राम के साथ Backslash character का उदाहरण

(i) Tab : '\t' character

यह एक Horizontal Tab है।

```
# include < stdio.h >
int main()
{
printf (" C program\t");
return (0);
}
```

printf execution के बाद cursor की position

C program —

(ii) New line character : '\n'

यह एक New line character है।

```
# include < stdio.h >
int main()
{
printf (" C program\n");
return (0);
}
```

printf execution के बाद cursor की position

C program —

(iii) Back space : '\b' character

यह एक Backspace character है।

```
# include < stdio.h >
int main ()
{
printf (" C program\b");
return (0);
}
```

printf execution के बाद cursor की position

C program —

Constant बनाने के नियम

Integer Constant

(i) Integer constant में कम से कम एक digit अवश्य होना चाहिए।
(ii) Integer constant negative या positive हो सकता है।
(iii) इसमें decimal point नहीं होने चाहिए।
(iv) Integer Constant में कोई भी comma या blanks नहीं होने चाहिए।
(v) इसकी Range – 32768 से 32767 तक ही होनी चाहिए।

Real constant या Float Point constant

(i) Real constant में कम से कम एक digit अवश्यक होना चाहिए।
(ii) यह negative या positive हो सकता है।
(iii) इसमें एक decimal point होना चाहिए।
(iv) अगर कोई sign नहीं है तो इसे positive मानना चाहिए।
(v) Real constant में कोई भी comma या blanks नहीं होने चाहिए।

Character constant

(i) Character constant में एक single alphabet या single digit या single special symbol किसी single quotes या inverted commas के अन्दर होने चाहिए।
(ii) दोनों inverted commas लेफ्ट (Left) की ओर होने चाहिए। जैसे 'A'
(iii) अधिकतम characters की संख्या one हो सकती है।

String constant

(i) String constant एक double quotes के अन्दर character का sequence होता है।
(ii) String constant, digit, letters, escaped sequences और space का combination हो सकता है।
(iii) सभी string constant Null character के साथ end होता है जो compiler द्वारा अपने आप assign किया जाता है।

Special Characters या symbols

C langage में कुछ special characters या symbols होते है। इन सभी का C program में special purpose होता है और यह प्रोग्राम के parts होते हैं।

उदाहरण : #, &, =, % etc.

Special Characters in C Programming

,	<	>	.	_
(	)	;	$	:

%	[	]	#	?
->	&	{	}	"
^	!	*	/	|
-	\	~	+	@

Fig. 2.4

▌Solves Exampled

(i) C Program to demonstrate the working of arithmetic operators

```
#include < stdio.h>
int main()
{
int m = 88, n = 7, result;
result = m + n
printf("m + n = %d \n", result);
result = m – n;
printf("m – n = % d \n", result);
result = m * n;
printf("m * n = % d \n", result);
result = m / n;
printf("m / n = %d \n", result);
result = m % n;
printf("m % n = %d \n", result);
return 0;
}
```

Output

```
m + n = 95
m – n = 81
m * n = 616
m / n = 12
m%n = 4
```

(ii) C Program to explain the working of assignment operators

```
#include < stdio.h>
int main()
{
```

```
int m = 95, result;
result = m;
printf("result = %d \n", result);
result += m;
printf("result = % d \n", result);
result – = m;
printf("result = % d \n", result);
result * = m;
printf("result = %d \n", result);
result/=m;
printf(" result = %d\n", result);
result % = m;
printf("result = %d \n", result);
return 0;
}
```

Output

```
result = 95
result = 190
result = 95
result = 9025
result = 95
result = 0
```

(iii)

```
#include <stdio.h>
int main()
{
int p = 1, q = 1;
while (p < = 7 || q < = 7)
{
printf (" % d % d\n", p, q);
p++;
q++;
}
return 0;
}
```

Output

```
1 1
```

```
2 2
3 3
4 4
5 5
6 6
7 7
```

(iv) C Program to find greatest number using logical operator

```
#include <stdio.h>
int main()
{
int n1, n2, n3;
printf ("\n Enter value of n1, n2 and n3:");
scanf ("%d %d %d", &n1, &n2, &n3);
if ((n1 > n2) && (n2 > n3))
printf ("\n Number 1 is greatest");
else if((n2>n3)&&(n2>n1))
printf("\n Number2 is greatest");
else
printf ("\n Number 3 is greatest");
return 0;
}
```

Output

```
Enter value of n1, n2 and n3 : 75 99 191
Number 3 is greatest.
```

(v) C Program to explain the working of increment and decrement operators

```
#include < stdio.h>
int main()
{
int m = 98, n = 87;
printf ("++m = %d \n", ++m);
printf ("--n = %d \n", --n);
return 0;
}.
```

Output

```
++m = 99
--n = 86
```

(v) C Program to explain the working of logical operators

```
#include < stdio.h>
int main()
{
int m = 75, n = 75; o = 100, result;
result = (m == n) && (o > n);
printf ("(m == n) && (0 > n) = %d \n", result);
result = (m == n) && (o < n);
printf ("(m == n) && (0 < n) = %d \n", result);
result = (m == n) || (o < n);
printf ("(m == n) || (0 < n) = %d \n", result);
result = (m != n) || (o < n);
printf ("(m != n) || (0 < n) = %d \n", result);
return 0;
}
```

Output

```
(m == n) && (o > n) = 1
(m == n) && (o < n) = 0
(m == n) || (o < n) = 1
(m != n) || (o < n) = 0
```

2.7 Variables

Variable एक memory location का नाम है जो डेटा के value को store करता है। Variables का value प्रोग्राम execution की अवधि में change कर सकते हैं। सभी Variables को प्रयोग करने से पहले उसका declaration और definition जरूरी होता है जो यह बताता है कि Variables का data type क्या है और कितना memory space लेगा।

Variables के Data Type

- int → int variable integer value को store करता है।
- char → char variable character को store करता है।
- float → float variable foat type के डेटा को hold करता है।
- double → double variable double value को hold करता है।

Variable Definition

Variable definition कम्पाइलर को यह बताता है कि किसी Variable के लिए कहाँ और कितना space create करना है। यह Variable के datatype को भी specify करता है।

syntax : datatype variables;

उदाहरण : int a, b, c ;

```
char c, y
float f, t ;
```

उदाहरण में int a, b, c इस लाइन के द्वारा variable define किया गया है जो compiler को instruct करता है कि a, b, c तीन variables int type के create करने हैं।

Variable को initialize भी किया जा सकता है।

उदाहरण : int a = 98, b = 76 ;

Variable a की initial value 98 और b की intial value 76 होगी।

▌Variable declaration

Variable declaration कम्पाइलर को यह बताता है कि Variable का नाम क्या है, Variable का data type क्या है। extern keyword का प्रयोग करके हम Variable को main () function के बाहर declare कर सकते हैं।

```
int a
int c
char b
float d
```

Simple C Program

Example 1 : Displaying Hello World

```
#include <stdio.h>
int main ( )
{
printf ("Hello World");
return 0;
}
```

Output

Hellow World

Explanation

इस प्रोग्राम में different प्रकार के token का प्रयोग हुआ है।

❖ Preprocessor

include एक preprocessor है जो प्रोग्राम को इसके headerfiles से link करता है।

❖ Headerfile

Headerfile में inbuilt function होते हैं जो प्रोग्राम में directly use किये जाते हैं।

❖ main ( )

यह एक function है जो program execution के समय call होता है। int main ( ) - main ( ) के पहले int यह दर्शाता है कि main ( ) का return type int है।

❖ Printf ( )

यह एक output function है जो screen पर output को display करता है।

❖ **return 0**

return 0 यह दर्शाता है कि program successfully execute कर गया है या integer constant 0 return किया है।

Variable बनाने के नियम

(1) Variable का नाम किसी letter या underscore से शुरुआत होने चाहिए।

(2) Variable case sensitive होते हैं।

(3) प्रोग्राम में प्रयोग करने से पहले Variable को declare और define करना जरूरी होता है।

2.8 C program Execution Steps

Writing या Editing

Writing या Editing यह C program execution का पहला स्टेप होता है। सबसे पहले Text editor का प्रयोग करके C program लिखा जाता है।

इसके बाद प्रोग्राम को .C extension का प्रयोग करके save किया जाता है। इसे source program भी कहा जाता है। यह source कोड होता है जिन्हें अगले स्टेप में compile किया जाता है। C program को अलग-अलग Text editor में लिखा जा सकता है।

Compilation

Source program को compiler के द्वारा compile करके इसे machine instruction में बदला जाता है। Compiler error को भी check करता है। Error को correct करने के बाद re-compile करके object file generate किया जाता है। Compiler किसी भी प्रकार का syntactical या structural error को check करता है और .obj exention के साथ object code generate करता है।

Linking

Linking एक execution का process है जो linker के द्वारा perform किया जाता है। Linker final executable file (.exe) extension के साथ produce करता है। Multiple object file जो scattered हो सकते हैं उनको link करके एक single final executable कोड में conversion का कार्य linker ही करता है। Linker library files को भी include करता है। Program को दूसरे libraries या headerfiles के साथ link किया जाता है।

Execution

Execution process C program का final stage होता है। Linking के बाद code को loader के पास भेजा जाता है।

Executable code को पहले loader memory में load करता है फिर execute करता है। execution के बाद output console के पास भेज दिया जाता है।

C Program के Execution Steps

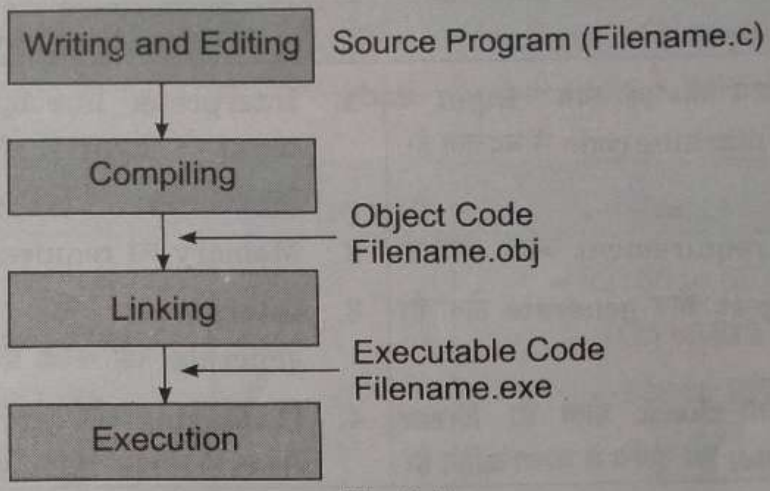


Fig. 2.5

2.9 Translator

Translator एक प्रोग्राम होता है जो source language को input की तरह लेता है और इसको target language में translate करता है। यह higher level language को machine language मे भी बदलता है। C language में निम्न 3 प्रकार के translator होते हैं।

- Compiler
- Assembler
- Interpreter

Compiler

Compiler एक computer program होता है जो high level language को low level language में बदलता है। यह प्रोग्राम में उपस्थित error को भी check करता है। Compiler source program को machine कोड में बदलता है जो processor पर run करता है। Compiler पूरे प्रोग्राम को एक साथ convert करता है।

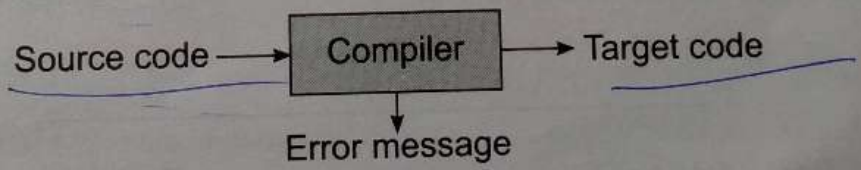


Fig. 2.6

Interpreter

Interpreter एक computer प्रोग्राम है जो program को line by line इनपुट की तरह लेकर machine language में बदलता है। जब किसी कोड की line में कोई error आता है वहाँ पर interpreter stop हो जाता है। Interpreter line by line या statement by statement translation करता है। इसका execution slow होता है। Debugging या analysis of error आसान होता है।

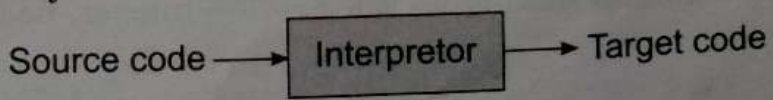


Fig. 2.7

Compiler और Interpreter में अंतर

Compiler	Interpreter
1. Compiler पूरे प्रोग्राम को एक साथ Input के form में लेता है और machine code में बदलता है।	1. Interpreter line by line source कोड को input के form में लेता है और line by line translate करता है।
2. इसमें memory की requirement ज्यादा होती है।	2. Memory की requirement कम होती है।
3. Intermediate object कोड generate होता है।	3. Interpreter कोई intermediate कोड generate नहीं करता है।
4. पूरा प्रोग्राम एक साथ check होता है। Error analysis interpreter की तुलना में ज्यादा कठिन है।	4. Debugging और error checking आसानी से हो सकता है क्योंकि line by line checking होता है।

Assembler

Assember एक Translator है जो Assembly language को मशीनी भाषा (binary Instruction) में convert करता है। High level language या assembly language को computer नहीं समझता है।

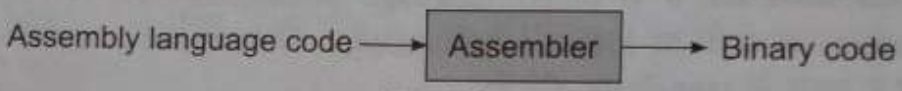
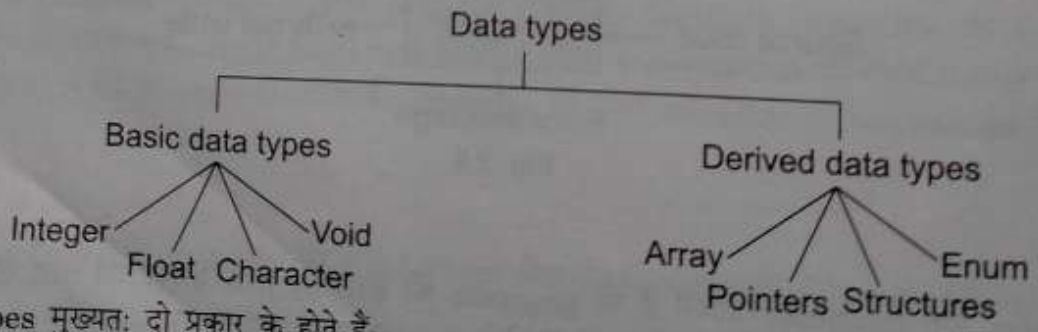


Fig. 2.8

मेमोरी में run कराने से पहले assember का प्रयोग करके assembly language को binary instruction (मशीनी भाषा) में बदला जाता है जिसको processor प्रोसेस कर सकता है।

2.10 Data types

Data types के द्वारा हम किसी variable या function को specify या declare करते हैं। Variable declaration यह बताता है कि variable कितना space मेमोरी में occupy करेगा। C programming language में कई प्रकार के Data types का प्रयोग किया जाता है।



Data types मुख्यत: दो प्रकार के होते हैं—

Basic या Primary data types

C language के ये fundamental data types होते हैं; जैसे—Integer, float, character, void etc.

Derived data types

ऐसे data types जो fundamental data types से बनते है उन्हें Derived data types कहते हैं।

जैसे— array, structures, pointers, enum.

Basic/Primary data types

Integer Type : Integer Type का प्रयोग whole number (without fraction) को store करने के लिये किया जाता है।

Type	Sitz (bytes)	Range
int	2	– 32768 to 32767
unsigned int	2	0 to 65535
short int	2	– 128 to 127
unsigned short int	2	0 to 255
long	4	–2147483648 to 2147483647
unsigned long	4	0 to 4296967295

Floating Type : Floating Type का प्रयोग floating point numbers को store करने के लिये करते हैं।

Type	Size (bytes)	Range
float	4	3.4 E – 38 to 3.4 E + 38
double	8	1.7 E – 308 to 1.7E + 308
long double	10	3.4 E – 4932 to 1.1 E + 4932

Charater Type : Character Type का प्रयोग character को store करने के लिये करते हैं।

Type	Size (bytes)	Range
char or signed char	1	– 128 to 127
unsigned char	1	0 to 255

Void Type : जब कोई function कोई value return नहीं करता है वहां पर void का प्रयोग होता है।

Derived data types

Derived data types किसी भी fundamental data type से ही derived होते हैं। ये User द्वारा defined और create किये जाते हैं।

जैसे—array, structure, union, enumeration, pointer etc.

Array

Array किसी भी एक ही तरह के डेटा के combination से बनता है। Array में एक ही तरह के data type के element होते हैं।

Structure

Structure में different types के डेटा होते हैं। इसमें int, char अथवा float type के डेटा हो सकते हैं।

Pointer

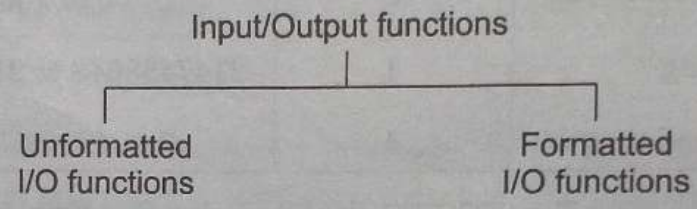
Pointer एक प्रकार का variable होता है जो किसी दूसरे variable के address को store करता है।

Enumeration

Enumeration को enum keyword के द्वारा लिखते हैं। यह एक user defined data type है। Enum के द्वारा हम किसी constant को name assign करते हैं।

2.11 Input Output Statement

Input and Output Statement डेटा को read और write के लिये प्रयोग किये जाते हैं। ये stdio.h (standard input/output) header file में inbuilt होते हैं। C programming में मुख्यत: दो प्रकार के input/output function का प्रयोग किया जाता है।



Input output function

(i) Unformatted I/O functions

(ii) Formatted I/O functions

Unformatted I/O functions

Unformatted I/O function एक Input/Output का Basic भाषा है। Unformatted I/O function डेटा के Internal binary representation को directly मेमोरी और फाइल्स में स्थानान्तरित करता है। Unformatted I/O function को directly पढ़ा नहीं जा सकता है। यह terminal screen पर type या edit नहीं किया जा सकता है। यह केवल character data type के साथ ही होता है।

Formatted I/O functions

Formatted I/O function पहले internal binary representation डेटा को ASCII character में बदलता है जो output file में लिखा जाता है। यह बहुत ही ज्यादाportable होता है। Formatted files human readable होता है और terminal screen पर type या edit किया जा सकता है।

C language में six unformatted I/O functions होते हैं।

(a) getchar ()

(b) putchar()

(c) gets()

(d) puts()

(e) getch()

(f) getche()

(a) getchar ()

getchar () function इनपुट से केवल charater type के डेटा को read करता है। getchar () function केवल एक समय में एह ही character को read करता है।

उदाहरण : (i)
```
#include <stdio.h>
int main( ) {
char input ;
input = getchar ( ) ;
printf ("Enter the character : \n");
print ("Entered character is : % C", input);
return 0;
}
```

Output : Enter the character : Y
Entered character is : Y

(ii)
```
#include <stdio.h>
#include <conio.h>
int main( )
{
char a;
printf ("Enter the character : ");
a - getchar ( ) ;
printef ("Entered character is : % C", a);
return 0;
}
```

Output : Enter the character : X
Entered character is : X

यहाँ getchar (), user के द्वारा दिये गये input को read करके आउटपुट के रूप में display करता है।

(b) putchar()

putchar() function का प्रयोग standard output या screen पर character को लिखने के लिये किया जाता है।

putchar() function C के library function stdio.h में specified होता है। यह एक समय में एक single character को display करता है।

putchar() का प्रोग्राम

```
#include <stdio.h>
void main( )
{
int a;
```

```
printf (" Enter the character ");
a = getchar ( );
putchar (a);
}
```

Program run होने के बाद एक value input लेगा और वही value output के रूप में display करेगा।

(c) gets()

gets() function standard input device से character या line को read करता है और उसको string में store करता है। जब यह newline character read करता है तो यह terminate हो जाता है।

gets() का प्रोग्राम

```
#include <stdio.h>
int main( )
{
char str [ 95 ];
printf (" Enter the string : ");
gets (str );
printf (" Entered string is : %s", str);
return 0;
}
```

Output :

```
Enter the string : C programming language
Entered string is : C programming language
```

(d) puts()

puts() function का प्रयोग किसी string को write या print करने के लिए करते हैं। puts() function mix variable को print कर सकता है। puts() / character stream के बाद newline character (\) को अंत में add करता है।

puts() का प्रोग्राम

```
#include <stdio.h>
int main( )
{
printf (" C language ");
printf (" C++ language ");
puts (" C language ");
puts (" C++ language ");
return 0;
}
```

Output :

C language, C++ language
C language
C++ language

यहाँ printf () लाइन को change नहीं करता है परन्तु puts() line को change करता है।

(e) getch()

getch() function एक non-standard function है जो conio.h header file में define है। यह keyboard से केवल single character को ही read करता है। यह buffer का प्रयोग नहीं करता है इसलिए enter किया गया character यह तुरन्त ही return कर देता है। यह enter key का wait नहीं करता है।

Syntax getch();

(f) getche()

getche() एक input function है जो keyboard से एक single character को read करता है और बिना enter key को दबाये ये शीघ्र output screen पर display होता है। यह एक non-standard library function है जो conio.h में defined होता है।

Syntax : getche ()

Formatted I/O functions

(a) scanf()

(b) printf()

(a) scanf()

scanf() एक input function है। यह keyboard से टाइप या enter किये हुए data को input के रूप में लेता है। यह एक inbuilt library function है जो character, string, numeric data या अन्य प्रकार के data को keyboard से read करता है। value के type को read करने के लिए scanf() के साथ specifier का प्रयोग करते हैं जो read किये हुए data का type बताता है। scanf() जो कि एक library function है यह "stdio.h" headerfile में declare होता है।

उदाहरण :

```
#include <stdio.h>
int main( )
{
int number1;
printf (" Enter the number ");
scanf (" % d, & number 1 ");
printf (" number = % d", numbers ");
return 0;
}
```

Output :

Enter the number : 95
number = 95

scanf() format code

Format Code	Meaning
%c	To read a single character
%d	To read a signed decimal integer (short)
%e	To read a float value exponential
%f	To rad a float (short) or a single precision value
%1f	To read a double precision float value
%g	To read a double float value
%i	To read an integer (decimal, octal, hexadecimal)
%o	To read an octal integer only
%x	To read a hexadecimal integer only
%u	To read an unsigned decimal integer (used in pointer)
%s	To read a string

scanf() का syntax - scanf ("format string", & arg 1, arg 2, ...);

(b) printf()

यह एक inbuilt library function है जो character, string, float, integer, octal और hexadecimal values को output के रूप में print करता है। यह program के output को display करने के लिए प्रयोग किया जाता है। printf() एक library function है जो "stdio.h" headerfile में declare होता है।

उदाहरण :

```
#include <stdio.h>
int main()
{
printf ("procedural program");
return 0;
}
```

Output :

procedural program.

यहाँ printf() जो कि एक library function है output के रूप में procedural program को display करता है।

Format Code	Meaning
%d	To read a decimal integer
%c	To read a single character
%1d	To read a single long decimal integer
%s	To read a string

%f	To read a float
%g	To read a double float value
%u	To read an unsigned decimal integer

printf () का syntax — printf ("format string", arg 1, arg 2, ...)

or

printf (" message ");

उदाहरण : C program में Variable का उदाहरण

```
#include <stdio.h>
#include <conio.h>
void main()
{
int m, n, mult // variable declaration
m = 9
n = 8
mult = m * n;
printf ("multiply is : % d, mult);
getch();
}
```

Output :

```
multiply is : 72
```

प्रोग्राम में m, n और mult तीन variables को declare किया गया है और $\begin{matrix} m = 9 \\ n = 8 \end{matrix}$ values initialize किया गया है।

2.12 Storage Class

C language में variables के value को किसी location पर store किया जाता है। ये locations अलग-अलग हो सकते हैं। इसे storage class कहते हैं। Storage class किसी भी variable का scope और limit निर्धारित करते हैं।

Storage class यह भी बताता है कि variable को कहाँ store करें।

Storage class के प्रकार—

1. Automatic storage class
2. External storage class
3. Register storage class
4. Static storage class

1. Automatic storage class

Automatic storage class सभी local variables का default storage class होता है। यदि variable का कोई storage class नहीं है तो वह Automatic storage class हो जाता है। इसका scope उसी function के लिए होता है जिसमें इसको define किया गया है।

उदाहरण - auto int a

int a

2. External storage class

External storage class का प्रयोग करके हम एक global variable को define कर सकते हैं। इस प्रकार के variable को main() function के statements से बाहर delcare करना होता है। इन्हें एक बार define और declare करने के बाद इन्हें कहीं भी प्रयोग में लाया जा सकता है। इसका प्रयोग प्रोग्राम में कोई भी function या user define function कर सकता है।

उदाहरण - extern int a

3. Register storage class

जिस variables को register में store कराना है उन variables को register keyword से लिखते हैं। इन्हें resigter storage class कहते हैं। Variable का maximum size register के size के बराबर हो सकता है। इस पर unary या '&' operator को apply नहीं कर सकते हैं।

उदाहरण - register int a;

4. Static storage class

किसी variable को static declare करने के लिए keyword static का प्रयोग किया जाता है। static storage class से declare variable पूरे प्रोग्राम में execution की अवधि में consistent रहते हैं। ये बार-बार create या destroy नहीं होते हैं।

Syntax - static int a

2.13 Type casting/Type conversion

Type casting के द्वारा हम प्रोग्राम में किसी variable का data type बदल सकते हैं। इसके लिए हम cast operator का प्रयोग करते हैं। सामान्यत: lower data type से higher data type में conversion किया जाता है। Type casting के द्वारा हम विभिन्न data type को एक variable के साथ प्रयोग कर सकते हैं। Type casting दो तरीके से होती है—

Implicit Type Conversion

इसे automatic type conversion भी कहते हैं। इसमें compilter अपने आप ही प्रोग्राम के अनुसार type को convert करता है। इसमें user को कुछ convert करने की जरूरत नहीं पड़ती है।

उदाहरण - Implicit type conversion

```
#include <stdio.h>
int main()
{
int g = 95 // g is integer
char b = 'a' ; // character
g = g +b ; // b implicitly converted to int
          // ASCII value of 'a' is 97
printf ("g = % d", g);
return 0;
```

Output :

g = 192

Explicit Type Conversion

यह type casting प्रोग्रामर या user के द्वारा की जाती है यह user defined होता है। इसमें particular type conversion के लिए user type casting करता है।

Syntax - (type) expression

उदाहरण -

```
#include < stdio.h>
int main()
{
float m = 1.9;
int sum = (int) m + 2; // explicit conversion
printf ("sum = % d", sum);
return 0;
}
```

Output : sum = 3

प्रोग्राम में variable m को int m से declare करके यहाँ explicit conversion किया गया है।

▌उदाहरण

Programming exercises on executing and editing a C program.

```
#include <stdio.h>
int main()
{
printf ("Hello World");
return 0;
{
```

Output :

Hello World

Program to find the average of two numbers

```
#include <stdio.h>
int main()
{
    int n1, n2;
    float avg;
    printf ("Enter first number : ");
    scanf (" % d", & n1);
    printf("Enter second number: ");
```

```
scanf("%d",&n2);
avg = (float) (n1 + n2) / 2;
printf ("Average of %d and %d is : %.f', n1, n2, avg);
return 0;
}
```

Output :

```
Enter first number : 78
Enter second number : 73
Average of 78 and 73 is : 75.5
```

C Program to print Integer enter by user

```
# include < stdio.h >
int main()
{
int m;
printf (" Enter a number : ");
scanf (" %d ", & m) ;
printf ("\number is : %d", m)'
return ( 0 ) ;
}
```

Output :

```
Enter a number  : 89
Number is : 89
```

C Program to find Quotient and Remainder

```
# include < stdio.h >
int main ()
{
int m, n, q, d ;
printf (" Enter the Number : ") ;
scanf ("\%d", & m);
printf (" \nEnter the divisor : ");
scanf ("\n%d ", &d);
n = m % d ;
q = m/ d ;
printf (" Quotient is : ") ;
printf (" %d", q);
printf (" Remainder is : ") ;
```

```
printf (" %d ", n) ;
return ( 0 ) ;
}
```

Output :

```
Enter the Number : 87
Enter the divisor : 6
Quotient is : 14
Remainder is : 3
```

C Program to swap two numbers

```
# include < stdio.h >
int main( )
{
int m, n, temp ;
printf (" first number : ") ;
scanf (" %d ", & m) ;
printf ("\n second number : ") ;
scanf (" %d ", & n) ;
printf ("\n number Before swaping : ") ;
printf ("\n %d \t %d ", m, n) ;
temp = m ;
m = n ;
n = temp ;
printf ("\n number After swapping : ") ;
printf ("\n %d \t %d ", m, n) ;
return ( 0 ) ;
}
```

Output of program :

```
first number : 87
second number : 67
number Before swapping :
87  67
number After swapping :
67   87
```

C Program to find the area of circle

```
# include < stdio.h >
int main( )
{
int radi ;
```

```
float area ;
printf (" enter the radius of circle : ") ;
scanf (" %d ", & radi) ;
area = 3.14 * radi * radi ;
printf ("\n area of the circle : ") ;
printf (" %f ", area) ;
return ( 0 );
}
```

Output of Program :

```
enter the radius of circle : 5
area of the circle : 78.5
```

C Program to find the simple interest

```
# include < stdio.h >
int main( )
{
float m, n, s, p ;
printf (" enter the principle amount : ") ;
scanf (" %f ", &n) ;
printf ("\n enter the rate of interest : ") ;
scanf (" %f ", &s) ;
printf ("\n enter the time period : ") ;
scanf (" %f ", &p) ;
sa = (n * s * p) / 100 ;
printf ("\n simple interest : ") ;
printf (" %f ", m) ;
return ( 0 );
}
```

```
enter the principle amount : 75
enter the rate of interest : 5
enter the time period : 20
simple interest : 75
```

C Program to reverse a number

```
# include < stdio.h >
int main( )
{
int number, remainder, reverse = 0 ;
printf (" enter the number : ") ;
scanf (" %d ", & number) ;
```

```
while ( number > = 0)
{
remainder = number % 10 ;
reverse = (reverse * 10) + remainder ;
number = number / 10 ;
}
printf ("\n reverse of the number is : ") ;
printf (" %d ", reverse) ;
return ( 0 );
}
```

Output :

```
enter the number : 7568
reverse of the number is : 8657
```

Fahrenheit से Celsius में बदलने का C प्रोग्राम

```
# include < stdio.h >
int main( )
{
float fo, ce;
printf (" Enter the value of Fahrenheit : ") ;
scanf (" %f ", &fo) ;
ce = (fo – 32) * 0.556;
printf ("\Celsius is : %.2f ", ce) ;
return 0;
}
```

Output :

```
Enter the value of Fahrenheit : 98
Celsius is : 36.67
```

Celsius से Fahrenheit में बदलने का C प्रोग्राम

```
# include < stdio.h >
int main( )
{
float fo, ce;
printf (" Enter the value of Celsius : ") ;
scanf (" %f ", & ce) ;
fo = ce * (9/5) + 32;
printf ("\Fahrenheit is : %.f ", fo) ;
```

```
return 0;
}
```

Output :

```
Enter the value of Celsius : 32
Fahrenheint is : 89.6
```

C Program to swap two numbers without using third variable

```
# include < stdio.h >
int main( )
{
int m, n, s;
printf (" Enter first number : ") ;
scanf (" %d ", & m) ;
printf (" Enter second number : ") ;
scanf (" %d ", & n) ;
m = m + n;
n = m – n;
m = m – n;
printf ("\n result After Swap\nm : %d n : %d n : %d", m, n) ;
return 0;
}
```

Output :

```
Enter first number : 4
Enter second number : 5
result After Swap : 5  4
```

C Program to find whether the given number is divisible by 5

```
# include < stdio.h >
int main( )
{
int number;
printf (" Enter first number : ") ;
scanf (" %d ", &number) ;
if (number % 5 == 0)
{
printf ("%d is divisible by 5", number) ;
}
else
{
printf ("%d is not divisible by 5", number) ;
```

```
}
return 0;
}
```

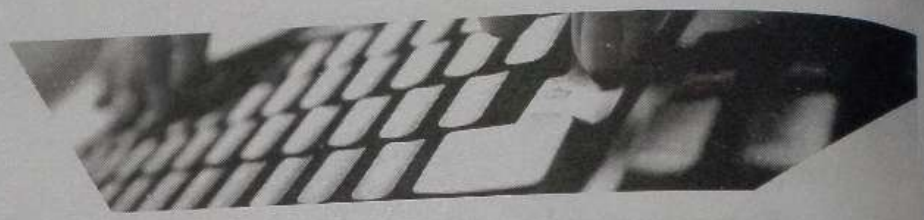
Output :

```
Enter the number : 85
85 is divisible by 5
```

▶ प्रश्नावली (Exercise) ◀

1. C language के बारे में विस्तार से समझाइए।
2. C language की विशेषता का वर्णन कीजिए।
3. C language का प्रयोग कहाँ-कहाँ होता है ? वर्णन कीजिए।
4. C Tokens क्या होते हैं ? उल्लेख कीजिए।
5. Key words क्या होते हैं ? उदाहरण के साथ समझाइए।
6. Identifier क्या होते हैं ? समझाइए।
7. Operators क्या होते हैं ? यह कितने प्रकार के होते हैं ?
8. Logical operator को उदाहरण के साथ समझाइए।
9. Increment और decrement operator को उदाहरण के साथ समझाइए।
10. Relational operator को उदाहरण के साथ समझाइए।
11. Assignment operator को उदाहरण के साथ समझाइए।
12. Conditanal operator क्या होते हैं ? वर्णन कीजिए।
13. Comma operator को उदाहरण के साथ समझाइए।
14. Constant क्या होते हैं ? यह कितने प्रकार के होते है ?
15. C में constant बनाने के नियम क्या है ?
16. Variables क्या होते हैं ? इसकी क्या आवश्यकता है ?
17. C program को write और execution करने के steps का वर्णन कीजिए।
18. निम्नलिखित पर संक्षिप्त टिप्पणी लिखिए—
    (a) Compilation (b) Linking (c) Execution
19. Translator क्या होते हैं ? इसका उपयोग कहाँ होता है ?
20. निम्नलिखित पर संक्षिप्त टिप्पणी लिखिए—
    (a) Compiler (b) Assembler (c) Interpretor
21. Data type क्या होते है ? उदाहरण के साथ वर्णन कीजिए। **(UPBTE 2018)**
22. C में प्रयुक्त होने वाले Input और Output statements का वर्णन कीजिए।
23. Storage class कितने प्रकार के होते हैं ? वर्णन कीजिए। **(UPBTE 2017, 2018)**
24. Type casting क्या होते हैं ? उदाहरण के साथ समझाइए। **(UPBTE 2018)**
25. Compiler और Interpreter में क्या अन्तर है ?

❑



Unit

3

Control Structures

Introduction, decision making with IF - statement, IF - Else and Nested IF, Ladder if-else, Loop : While, do-while, for, Break, Continue, go to and switch statements

3.1 Control structures

C Programming (प्रोग्रामिंग) में, Control structure का प्रयोग विभिन्न computations (गणना) या actions (क्रियाओं) को पूरा करने के लिए किया जाता है। Control structure में Conditional Statements का प्रयोग करके किसी condition के true होने पर उससे सम्बन्धित किसी विशेष कार्य को complete किया जाता है। Control strucure का प्रयोग करके प्रोग्राम के flow का execution को भी बदला या नियंत्रित किया जाता है।

Programming Language में किसी विशेष condition के true या false होने पर जब हम कुछ action चाहते हैं तब हम conditional statements का प्रयोग करते हैं।

उदाहरण के लिए हम ऐसा program बनाना चाहते हैं जिससे अगर User का (उम्र) 18 वर्ष से अधिक हो तो उसे Output में "eligible for voting" लिखा हुआ आये और अगर User का Age (उम्र) 18 वर्ष से कम हो तो उसे Output में "not eligible for voting" Message लिखा हुआ आये। तो इस Program में इस निर्णय के लिए कि User का Age (उम्र) 18 वर्ष से अधिक है या नहीं, Conditional statementsका प्रयोग करेंगे।

The Control Structure को निम्नलिखित भागों में विभाजित किया जा सकता है—

(i) Sequential control structure

(ii) Selection control structure

(iii) Loop or repetition control structure

(i) Sequential control structure

Sequential control structure में प्रोग्राम के statements एक specific sequence या order में execute होते हैं। कोई भी statement skipped नहीं होता है। इस structure में code के statements एक के बाद दूसरा और फिर तीसरा इत्यादि करके execute होता है। कोई भी repetition नहीं होता है। यह एक प्रोग्राम का बहुत ही साधारण और आसान control structure है। प्रोग्राम को समझना और modify करना आसान होता है।

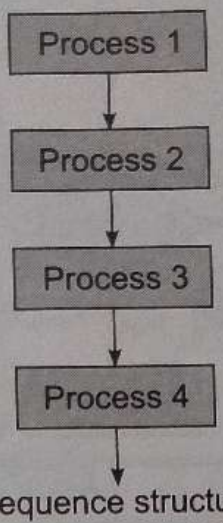


Fig. 3.1 : Flowchart of sequential control structure

(ii) Selection control structure

C program में Selection control structure मुख्यत: decision making के लिये प्रयोग किया जाता है। प्रोग्राम में जब execution के कई सारे paths होते हैं उस समय किसी specific condition पर किसी specific path को select करने के लिए Selection control structure का प्रयोग किया जाता है। ऐसे structure में decision के द्वारा path को selection के लिए if, else, if else, switch इत्यादि key words का प्रयोग किया जाता है।

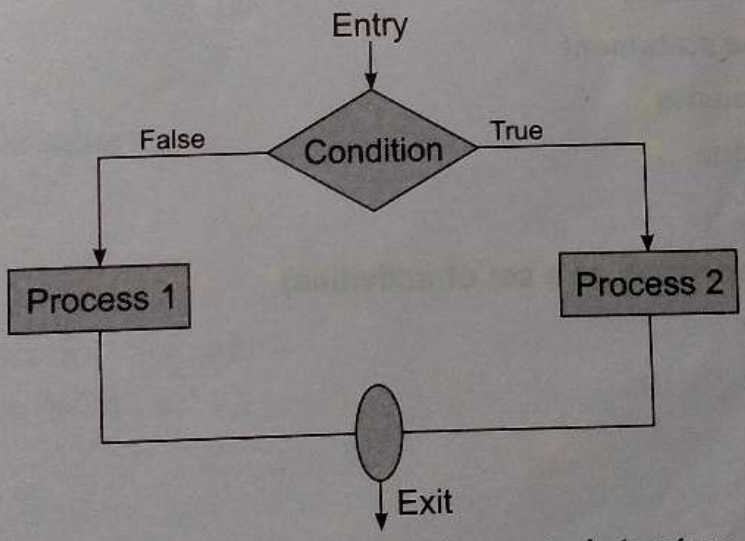


Fig. 3.2 : Flowchart of selection control structure

(iii) Loop or repetition control structure

Loop या repetition contol structure का प्रयोग looping operation या repetition के लिये किया जाता है। जब तक दिया हुआ condition true होता है तब तक repetition होता रहता है और condition false होने पर control loop से बाहर आ जाता है। इस loop statement का प्रयोग करके किसी statement या statement के group को अनेकों बार execute करा सकते हैं।

C program में 3 प्रकार के loop control statements का प्रयोग किया जाता है।

1. for

2. while
3. do-while

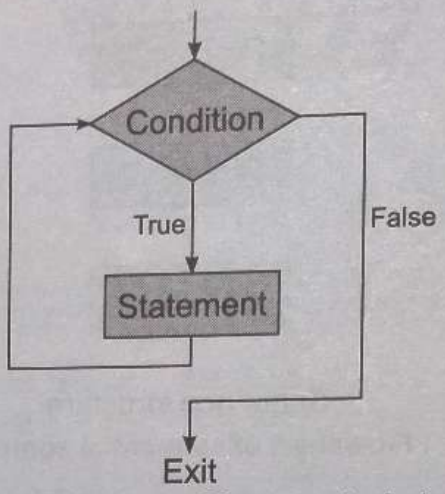


Fig. 3.3 : Flowchart of loop control structure

3.2 Selection Structure

a. If statement
b. If-else statement
c. Nested If Statemnet
d. Ladder if-else statement
e. switch statements
f. Break Continue
g. go to

2. Loop structure (repetition of a set of activities)

a. while
b. do-while
c. for

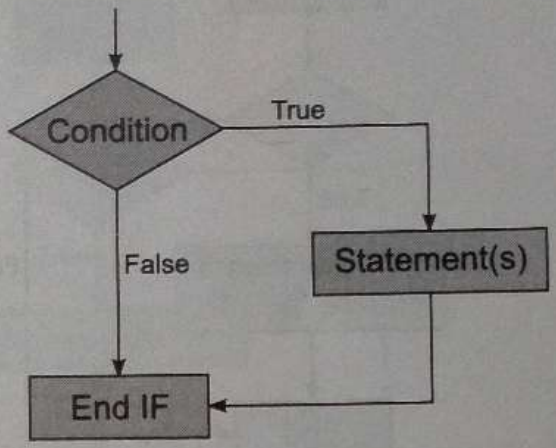
If statement

IF statement का प्रयोग किसी Boolean expresson के True या False होने पर उससे related statement execute कराने में किया जाता है। इसका प्रयोग decision making में किया जाता है। यदि condition true होता है तो If body का statement execute होता है। यदि statement false होता है तो If body के statement skipped होता है और If body के बाद का statement execute होता है।

The Syntax (format) of a simple if statement

```
if (condition)
{
Statements;
... ... ...
```

}



Flowchart for if statement
Fig. 3.4

If statement के C प्रोग्राम

▌उदाहरण

(i) C program to print the square of a number if it is less than 5.

```
# include <stdio.h>
int main()
{
    int n;
    printf ("Enter a number :");
    scanf ("%d", &n);
    if (n < 5)
    {
    printf ("%d is less than 5\n", n);
    printf ("Square = %d\n", n * n);
    }
    return 0;
}
```

Output

```
Enter a number : 4
4 is less than 5
Square = 16
```

यह प्रोग्राम if statement का प्रयोग करके बनाया गया है जिसमें user द्वारा इनपुट किया गया number यदि 5 से कम है तो इसका square होगा अन्यथा प्रोग्राम terminate हो जायेगा।

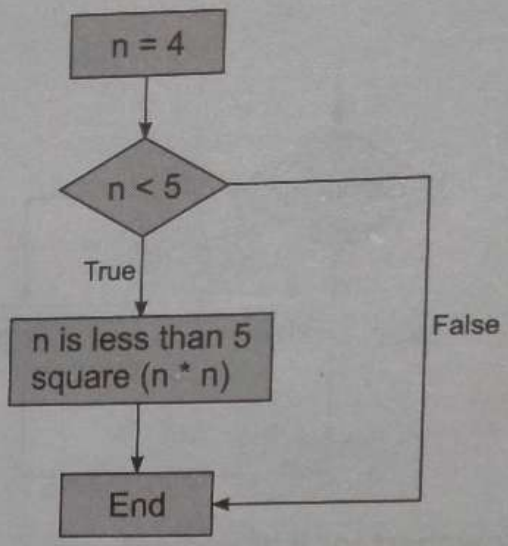


Fig. 3.5

(ii)

```
# include <stdio.h>
    int main()
    {
    int m = 98;
    if (m < 100)
    {
    printf ("m is less than 100\n");
    }
    printf ("m is : %d\n" m);
    }
    return 0;
    }
```

Output :

```
m is less than 100;
m is : 98
```

इस प्रोग्राम में m की value 98 assign किया गया है। If के अन्दर (m < 100) condition check हो रहा है। यदि m < 100 true है तो if body के अन्दर का statement execute होगा अन्यथा if body के बाद का statement execute होगा।

(m < 100) condition true है तो

Output = " m is less than 100 "

m is : 98"

Flowchart

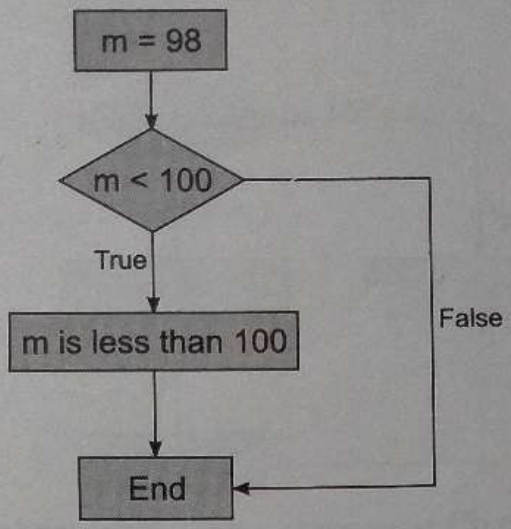


Fig.3.6

(iii) C प्रोग्राम लिखिये यदि user द्वारा गया integer 1 है तो output "UP" होगा, यदि integer 2 है तो output " Punjab" यदि 3 है output "Gujarat" होगा।

```
# include
void main()
{
int num;
printf (" Enter a number");
scanf ("%d", &num); // number from user
if (num == 1)
{
printf ("UP");
}
if (num == 2)
{
printf ("Punjab");
}
if (num == 3)
{
printf ("Gujarat");
}
```

इस प्रोग्राम में यदि input नम्बर num == 1 है तो इस if body के अन्दर का statement "UP" print होगा।

input यदि num == 2 है तो output "Punjab" print होगा।

input यदि num == 3 है तो output "Gujarat" print होगा।

(iv) C program to find the greater number of two numbers.

```
#include <stdio.h>
int main()
{
int p, q;
printf ("Enter the number");
scanf ("%d", &p);
printf ("Enter the number");
scanf ("%d", &q);
if (p > q)
{
printf ("p is a greater than q");
}
if (p < q)
{
printf ("p is less than q\n");
}
```

Output :

```
Enter the number : 98
Enter the number : 75
p is greater than q.
```

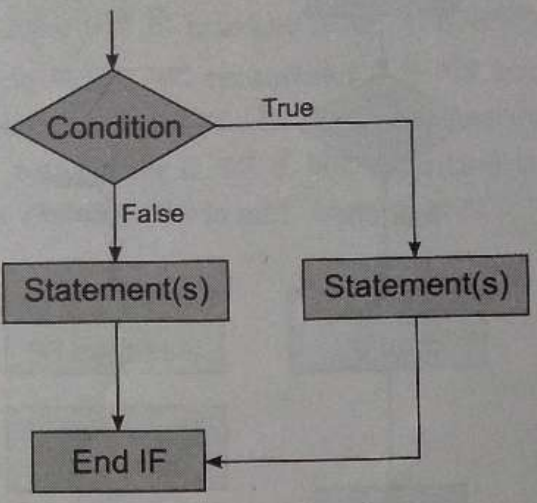
if ... else statement

if ... else statement, two way branching statement होता है। इसमें statements के दो blocks होते हैं। एक statement if block में होता है, दूसरा statement else block में होता है। यदि if block का condition true होता है तो इसका statement execute होगा अन्यथा else block का statement execute होगा।

syntax of if ... else statement

```
if (condition)
{
    statements;
    ... ... ...
}
else
{
    statements;
    ... ... ...
}
```

Flowchart of if ... else statement



Flowchart for if ... else statement
Fig. 3.7

उदाहरण

(i)

```
#include <stdio.h>
    int main()
    {
    int m = 98
    if (m < 96)
    {
    printf (" m is less than 96\n");
    }
else
    {
    printf (" m is not less then 96\n");
    }
    printf (" The value of m is : %d\n", m);
    return 0;
    }
```

Output :

```
m is not less then 96
The value of m is : 98
```

इस प्रोग्राम में condition (m < 96) check हो रहा है। condition false हो रही है क्योंकि m का value 98 है। इसलिए else body का statement execute होगा।

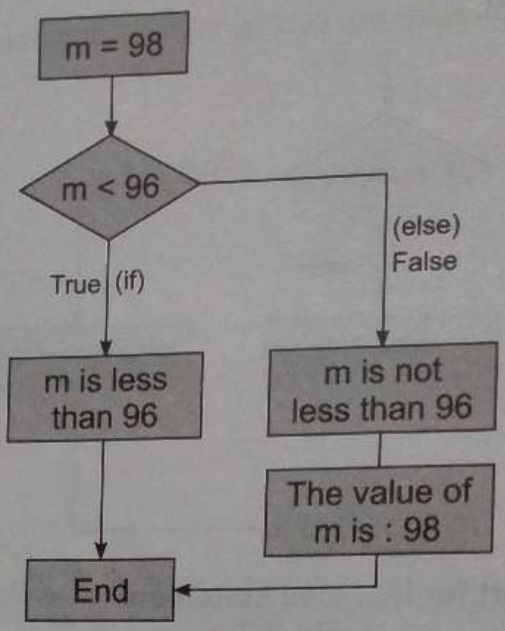


Fig. 3.8

(ii) C program to find if a number is odd or even.

```
# include <stdio.h>
int main()
{
    int n;
    printf ("Enter a number : ");
    scanf ("%d", & n);
    if (n%2 == 0)
        {
        printf ("%d is Even", n);
        }
    else
        {
        printf ("%d is Odd", n);
        }
  return 0;
}
```

Output :

```
Enter a number : 12
12 is even
Enter a number : 35
35 is odd
```

यह प्रोग्राम if-else statement का प्रयोग करके बनाया गया है जिसमें User द्वारा इनपुट किया गया number यदि 2 से divide होने के बाद remainder 0 है तो number Even होगा अन्यथा Odd होगा।

12 input किया गया। 2 से divide होने के बाद remainder 0 है और condition (n%2 == 0) True हो रहा है इसलिए If body का statement execute होगा और Output 12 is even होगा। पुन: 35 input किया गया। चूँकि 2 से divide होने के बाद remainder 0 नहीं है और condition (n %2== 0) false हो रहा है तो else body का statement execute होगा और "35 is odd" output होगा।

(iii)

```
#include <stdio.h>
int main()
{
int m;
printf (" Enter any number ");
scanf ("%d", &m);
if (m > 0)
{
printf ("\n number is greater than 0");
}
else
{
printf ("\n number is less than or equal to 0");
}
return 0;
}
```

Output :

```
Enter any number : 5
Number is greater than 0
```

(iv) C program to check the number is positive, negative or zero

```
#include <stdio.h>
int main()
{
int p;
printf ("Enter the number");
scanf ("%d", &p);
if (p > 0)
printf ("Number is positive");
else
if (p < 0)
printf ("Number is negative");
```

```
else
printf ("Number is equal to zero");
return 0;
}
```

Output :

```
Enter the number : 85
Number is positive.
```

Nested if statements

यदि एक if statement के अंदर कोई दूसरा if statement हो तो यह Nested if statements कहलाता है। जब किसी एक condition के अंदर दूसरा sub-condition test होता है तो वहाँ पर Nested if का प्रयोग किया जाता है।

Syntax of nested if statement

```
if (condition 1)
{
    statements;
    if (subcondition 1)
    {
        statements;
    }
        statements;
}
else if (condition 2)
{
    statements;
    if (subcondition 2)
    {
        statements;
    }
    statements;
}
... ... ...
... ... ...
else
    {
    statements;
    if (sub-conditions n)
    {
```

```
        statements;
    }
    statements;
}
```

nested if statement का Flowchart

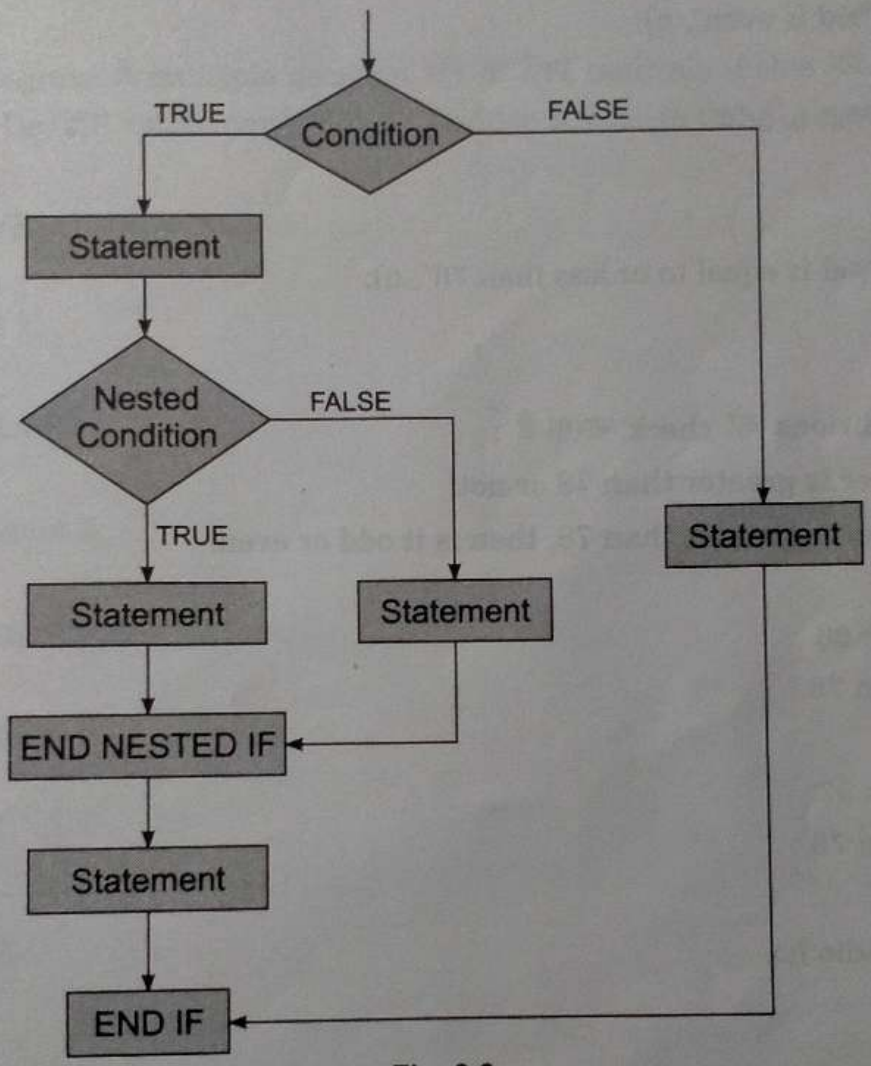


Fig. 3.9

Nested if statement के C प्रोग्राम

▌उदाहरण

(i) C program to check if a number is greater than 78 or not. If it is greater than 78, then check if it is odd or reven.

```
#include <stdio.h>
int main()
{
    int n;
    printf ("Enter a number:");
```

```
    scanf ("%d", &n);
    if (n > 78)
    {
        printf ("%d is greater than 78\n", n);
        if (n % 2 == 0)
        printf ("%d is even", n);
        else
        printf ("%d is odd", n);
    }
    else
        printf ("%d is equal to or less than 78", n);
    return 0;
}
```

यह प्रोग्राम दो conditions को check करता है :

1. if the number is greater than 78 or not.
2. if the number is greater than 78, then is it odd or even.

Output :

```
Enter a number : 86
86 is greater than 78
86 is even
Enter a number : 97
97 is greater than 78
97 is odd.
```

(ii)

```
#include <stdio.h>
int main()
{
    int a = 98, b = 96;
    if (a > b)
  {
    printf ("a is greater than b");
  }
  else if (a < b)
  {
    printf ("a is less than b");
  }
  else
```

```
    {
        printf ("a is equal to b");
    }
}
```

Output : a is greater than b.

Ladder if else

जब किसी C program में multiple decision लेने के लिए multiple if else का प्रयोग एक साथ करते हैं तो उसे ladder if else द्वारा implement करते हैं। Ladder if else में top से down की तरफ condition evaluate होता है।

```
if (condition 1)
{
    statement 1;
}
    else if (condition 2)
    {
        statement 2;
    }
    else if (condition n)
    {
    statement n;
    }
    else
{
    statement;
}
```

उदाहरण

(i) Ladder if else का प्रयोग करके किन्हीं दो नम्बरों का comparison का C प्रोग्राम (equal, greater या less)

```
#include <stdio.h)
int main()
{
    int n1, n2;
    printf ("Enter the two numbers :");
    scanf ("%d %d", & n1, & n2);
    if (n1 == n2)
    {
```

```
            printf ("result : %d = %d\n", n1, n2);
        }
        else if (n1 > n2)
        {
            printf ("result : %d > %d\n", n1, n2);
            }
        else
        {
            printf ("result : %d < %d\n", n1, n2);
        }
        return 0;
    }
```

Output : Enter the two numbers : 98 99
result 98 < 99

(ii)

```
#include < stdio.h>
#include < conio.h>
int main()
{
    int marks;
    printf ("Enter the marks\n");
    scanf ("%d", & marks);
    if (marks >= 80);
    {
        printf ("Grade is A\n");
    }
        else if (marks >= 70 && marks < 80)
    {
        printf ("Grade is B\n");
    }
        else if (marks >= 45 && marks < 70)
    }
        printf ("Grade is C\n");
    }
    else
    {
        printf ("Grade is D\n");
}
```

```
    getch()
    return 0;
Enter the marks : 91
Grade is A
Enter the marks : 69
Grade is C
```

Ladder if else का Flowchart

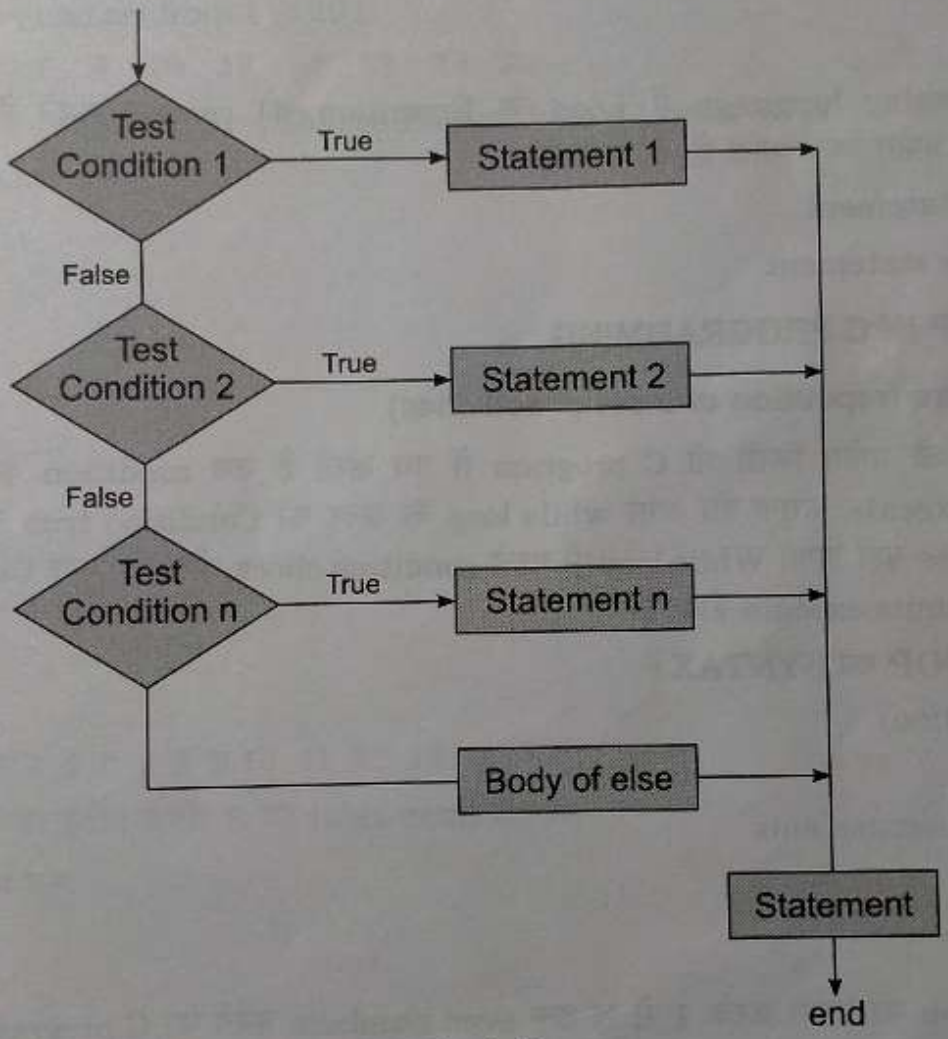


Fig. 3.10

3.3 LOOP

Loop in C Programming

Loop का प्रयोग program के किसी statement या code के block को बार-बार execute करवाने के लिये करते हैं या प्रोग्राम के किसी एक हिस्से को कई बार execute करने के लिए C Programming में Loop का प्रयोग किया जाता है। ये repeat होता रहता है जब तक condition True रहता है या satisfied होता है।

Loop को दो भागों में विभाजित कर सकते हैं—

(1) Loop statement : यह conditional statement होता है जो यह बताता है कि loop कितनी बार repeat होगा।

(2) Loop body : यह executed statement होता है जो loop के प्रत्येक cycle के साथ execute होता है।

उदाहरण : अगर हमें 1 से 100 तक के numbers को print करना है तो LOOP का प्रयोग करके इसे बहुत आसानी से किया जा सकता है। Loop का प्रयोग करके एक ही statement को बार-बार Print कर सकते हैं।

C Programming language में निम्न प्रकार के Loop होते हैं—

LOOP के प्रकार

(a) while

(b) do while

(b) for

C Programming language में Loop के Execution को control करने के लिए Conditional Statements का प्रयोग किया जाता है जो निम्न है—

(i) break statement

(ii) continue statement

WHILE LOOP IN C PROGRAMMING

1. Loop structure (repetition of a set of activities)

while loop का प्रयोग किसी भी C program में तब करते हैं जब condition के true होने पर किसी statement को execute कराना हो। अगर while loop के अंदर का Condition true नहीं है तो while loop एक बार भी execute नहीं होगा। While loop में पहले condition check होता है; अगर Condition true है फिर उसके बाद statements execute होता है।

WHILE LOOP का SYNTAX :

```
while (condtion)
{
Block of code/statements
}
```

उदाहरण

(i) While loop का प्रयोग करके 1 से N तक even numbers करने का C program ।

```
#include <stdio/h)
int main()
{
    int num, n;
    num = 1;
    printf ("enter the value of n :");
    scanf ("%d", & n);
    printf ("even numbers from 1 to %d : \n", n);
    while (num <= n)
    {
```

```
        if (num % 2 == 0)
        printf ("%d", num);
        num ++;
    }
    return 0;
}
```

Output : enter the value of n : 20
even numbers from 1 to 20 :
2 4 6 8 10 12 14 16 18 20

(ii) while loop का प्रयोग करके first 15 natural numbers को print करना।

```
#include <stdio.h>
void main()
{
    int m;
    m = 1;
    while (m <= 15)
    {
        printf ("%d\t", m);
        m ++;
    }
]
```

Output : 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

(iii) while loop का प्रयोग करके 5 का table print करना।

```
#include <stdio.h>
int main()
{
    int i = 1;
    while (i <= 10);
    {
        printf ("5* %d = %d\n", i, 5 * i);
        i ++;
    }
    return 0;
}
```

Output : 5 * 1 = 5
5 * 2 = 10
5 * 3 = 15

```
5 * 4 = 20
5 * 5 = 25
5 * 6 = 30
5 * 7 = 35
5 * 8 = 40
5 * 9 = 45
5 * 10 = 50
```

यह प्रोग्राम 5 के multiplication table को print करता है। while loop का प्रयोग करके table को print किया गया है। condition $i \leq 10$ के लिए test किया गया है। Loop execution के बाद प्रत्येक बार i की value 1 increase हो जाती है। जब value 11 होती है तब condition false होता है और loop terminate हो जाता है।

(iv) while loop का प्रयोग करके 1 से N तक के odd numbers को print करना।

```
#include <stdio.h>
int main()
{
    int num;
    int n;
    num = 1;
    printf ("Enter the vlalue of n :");
    scanf ("%d", &n);
    printf ("odd number from 1 to % d : \n", n);
    while (num <= n)
    {
        if (num %2 ! = 0)
        printf ("%d", num);
    }
    num ++;
    return 0;
}
```

Output :

```
Enter the value of n : 20
Odd numbers from 1 to 20 :  1 3 5 7 9 11 13 15 17 19
```

(v) Find the sum of natural numbers using while loop.

```
#include <stdio.h>
int main()
{
    int n, i, sum = 0;
    prinf ("Enter the value of n");
```

```
    scanf ("%d", & n);
    i = 1;
    while (i <= n)
    {
        sum = sum + i;
        i++;
    }
    printf ("Sum of first %d natural numbers is : %d", n, sum);
    ruturn 0;
}
```

Output :

```
Enter the value of n : 6
Sum of first 6 natural numbers is : 21
```

First iteration में i =1 यह 6 से कम है इसलिए while के अन्दर का statement execute होगा। sum = 0 + 1 = 1 फिर i की value i + 1 = 1 + 1 = 2 होगी, फिर 2 < 6 चेक होगा। यह true है तो sum = 1 + 2 = 3, इस तरह loop चलता रहेगा।

और sum = 0 + 1 = 1 → 1^{st} iteration

sum = 1 + 2 = 3 → 2^{nd} iteration

sum = 3 + 3 = 6 → 3^{rd} iteration

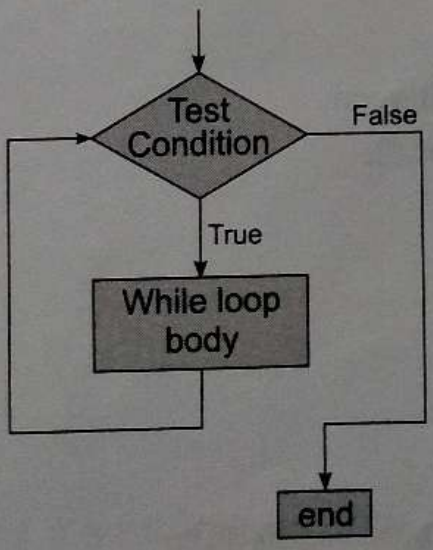
sum = 6 + 4 = 10 → 4^{th} iteration

sum = 10 + 5 = 15 → 5^{th} iteration

sum = 15 + 6 = 21 → 6^{th} iteration

जब i = 7 होगा तो वह loop से बाहर होगा और sum = 21 print होगा।

While loop का Flowchart



Fog. 3.11

(vi) #include <stdio.h>

```
int main()
{
    int i = 0;
    while (i < 8)
    {
        printf ("%d\n", i);
        i++;
    }
}
```

Output :

```
0
1
2
3
4
5
6
7
```

दिये गये प्रोग्राम में while loop के अंदर का statement बार-बार execute होगा, जब तक while loop का condition true रहेगा, अर्थात् जब तक i का value 8 से कम रहेगा तब तक while loop के अंदर का statement बार-बार execute होगा।

(viii) C program to print the sum of digits of a number.

```
#include <stdio.h>
int main()
{
    int p, number, result = 0; remi;
    printf ("Enter the number :");
    scanf ("%d", &p);
    while (p > 0)
    {
    remi = p % 10;
    result = result + remi;
    p = p/10 ;
    }
    printf ("sum of digits of %d is %d", number, result);
    return 0;
```

```
}
```

Output : Enter the number : 98
Sum of digits of 98 is 17.

▌do while loop

do while loop भी while loop की तरह कार्य करता है केवल एक अन्तर यह है कि condition check करने से पहले एक बार do while loop के अन्दर का statement execute होता है जबकि while loop में पहले condition check होता है। True होने पर while loop का statement execute होता है। do while loop में condition false होने पर भी कम से कम एक बार statement का execution होता है।

while : पहले condition check होता है फिर statement execute होता है।

do while : पहले statememt execute होता है फिर condition check होता है।

do while loop का Flowchart

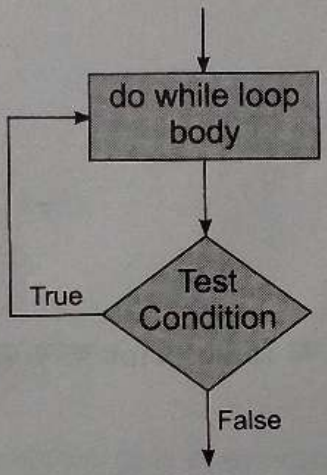


Fig. 3.12

do while loop का Syntax

```
do
{
    statement 1;
    statement 2;
    statement 3;

}
    while (condition);
```

do while loop के C program

▌उदाहरण

(i) do while loop का प्रयोग करके 2 से 8 तक के numbers को print करना।

```
#include <stdio.h>
int main()
```

```
{
    int i = 2
    do
    {
        printf ("%d\n", i);
        i++;
    }
        while (i ≤ 8)
    return 0;
}
```

Output :

2
3
4
5
6
7
8

(ii) do while loop का प्रयोग करके 9 का table print करने का C program.

```
#include <stdio.h>
int main()
{
    int i = 1;
    do
    {
        printf ("9 * %d = %d\n", i, 9 * i);
        i++;
    }
    while (i <= 10);
    return 0;
}
```

Output :

```
9 * 1 = 9
9 * 2 = 18
9 * 3 = 27
9 * 4 = 36
9 * 5 =45
9 * 6 = 54
9 * 7 = 63
9 * 8 = 72
9 * 9 = 81
9 * 10 = 90
```

(iii) do while का प्रयोग करके किसी भी नम्बर का factorial प्राप्त करने का C program.

```
#include <stdio.h>
#include <conio.h>
void main()
{
    int i, n, fact = 1;
    printf ("Enter the value of n\n");
    scanf ("%d", &n);
    i = 1
    do
    {
        fact = fact * i;
        i++;
    }
    while (i <= n);
    printf ("Factorial : %d\n", fact);
    getch();
}
```

Output : Enter the value of n = 5
Fatorial : 120

(iv) do while का प्रयोग करके 1 से 100 तक के 5 के multiple नम्बरों को print करने का C program.

```
#include <stdio.h>
int main()
{
    int i = 1;
    do
    {
        if (i % 5 == 0)
    {
        printf ("%d", i);
        i++;
    }
    while (i < 100);
    return 0;
}
```

Output : 5 10 15 20 25 30 35 40 45 50 55 60 65 70 75 80 85 90 95 100

For Loop

For Loop का प्रयोग किसी भी C program में condition के true होने पर किसी statement को बार-बार execute कराने के लिये करते हैं। जब तक for Loop में Condition true होती है तब तक forloop की statemetn execute होता रहता है।

for Loop का Syntax

```
for (initialization; condition; iteration)
{
    statement 1;
    statement 2;
    statement 3;

}
```

for Loop का Working

for loop में सबसे पहले initialization statements execute होता है, फिर condition statement evaluate होता है, यदि conditions, TRUE है तब curly braces के अंदर का statements execute होता है। Curly braces के अंदर का statement execute होने के बाद control iteration statement के लिये move होता है। Iteration statements execute होने के बाद control पुन: condition statements के पास वापस आता है। Condition statements फिर (TRUE or FALSE) के लिये evaluate होता है। यदि conditions TRUE है तब curly braces के अंदर का statement execute होता है। Condition FALSE होने तक इस process की repetition होती रहती है।

For loop का Flowchart

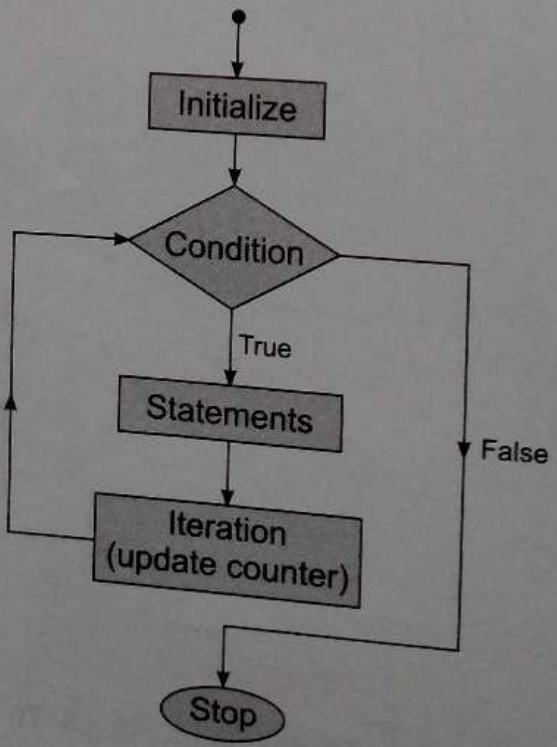


Fig. 3.13

▌उदाहरण

(i) Program to find sum of natural numbers using for loop.

```
#include <stdio.h>
int main()
{
    int n, i, sum = 0;
    printf ("Enter the value of n");
    scanf ("%d", &n);
    for (i = 1; i <=n; i++)
    {
        sum = sum + i;
    }
    printf ("Sum of first %d natural numbers is : %d", n, sum);
    return 0;
}
```

Output :

```
Enter the value of n : 5
Sum of first 5 natural numbers is : 15
```

(ii) Program to find Fibonacci Series in C using for loop

```
#include <stdio.h>
int main()
{
    int count, term_1 = 0, term_2 = 1, term_next, i;
    printf ("Enter the number of terms : \n");
    scanf ("%d", &count);
    printf ("First %d terms of Fibonacci series :\n", count);
    for (i = 0 ; i < count ; i++)
    {
        if (i <= 1)
            term_next = i;
        else
        {
        term_next = term_1 + term_2;
            term_1 = term_2;
            term_2 = term_next;
        }
```

```
            printf ("%d\n", term_next);
        }
        return 0;
    }
```

Output :

```
Enter the number of terms : 6
First 6 terms of Fibonacci series :
0
1
2
3
4
5
```

(iii) for loop का प्रयोग करके 1 से 6 तक की संख्याओं को प्रिन्ट करना।

```
#include <stdio.h>
int main()
{
    int i;
    for (i = 1 ; i <= 6 ; i++)
    {
        printf ("%d\n", i);
    }
    return 0;
}
```

Output :

```
1
2
3
4
5
6
```

(iv) for loop का प्रयोग करके 12 का table प्रिन्ट करने का C program.

```
#include <stdio.h>
int main()
{
    int n, i;
    printf ("Enter an integer :");
```

```
    scanf ("%d", & n);
    for (i = 1 ; i <= 10 ; i++);
    {
        printf ("%d * %d = %d\n", n, i, n * i);
    }
    return 0;
}
```

Output : Enter an integer : 12

12 * 1 = 12	12 * 2 = 24
12 * 3 = 36	12 * 4 = 48
12 * 5 = 60	12 * 6 = 72
12 * 7 = 84	12 * 8 = 96
12 * 9 = 108	12 * 10 = 120

(v) for loop का प्रयोग करके 1 से 10 तक की संख्याओं का average प्राप्त करना।

```
#include <stdio.h>
int main()
{
    float average = 0;
    int sum = 0
    int m = 0
    int num [10];
    for (m = 0 ; m < 10 ; m++)
    {
        printf ("Enter the number %d\n", (m + 1));
        scanf ("%d", &num[m]);
    }
    for (m = 0 ; m < 10 ; m++);
    {
        sum = sum + num [m];
    }
    average = sum/10;
    printf ("average of entered number is : %f avg);
    return 0;
}
```

Output : Enter number 1 : 88

Enter number 2 : 76

```
Enter number 3 : 79
Enter number 4 : 78
Enter number 5 : 75
Enter number 6 : 64
Enter number 7 : 87
Enter number 8 : 76
Enter number 9 : 98
Enter number 10 : 82
average of entered number is : 80.3
```

(vi) triangular star pattern print करने का C program.

```
#include <stdio.h>
int main()
{
    int i, j, m;
    printf ("Enter the number of row :");
    scanf ("%d", &m);
    prinft ("\n patterns are : \n\n");
    for (i = 1 ; i <= m ; i++)
    {
        for (j = 1  ; j <= i ; j++)
    {
    printf (" * ");
    }
    printf ("\n");
    }
    return (0);
}
```

Output : Enter the numer of row : 7

```
*
* *
* * *
* * * *
* * * * *
* * * * * *
* * * * * * *
```

3.4 Goto statement

goto statement एक प्रकार का jump statement है। goto statement का प्रयोग करके हम प्रोग्राम में एक जगह से दूसरी जगह control को transfer करा सकते हैं। goto एक keyword है। यह एक प्रकार का unconditional jump है।

Syntax : goto label ;

```
... ... ...
... ... ...
... ... ...
label statement;
```

label एक valid identifier होता है।

goto का प्रयोग करके दो प्रकार से control को tansfer किया जा सकता है।

(i) down to top

(ii) top to down

down to top

```
label :
statements
    ... ... ...
    ... ... ...
goto label;
```

इसमें control down से top पर transfer हो रहा है।

उदाहरण : (i) goto statement का प्रयोग करके 2 से 8 तक के number को print करने का C program.

```
#include <stdio.h>
int main()
{
    int num;
    num = 2;
    display :
    printf ("%d\n", num);
    num++;
    if (num < 8)
    goto dispay;
    return 0;
}
```

Output :

```
2
```

3
4
5
6
7
8

top to down

goto label;
statements
... ...
... ...
label :

इसमें control top से down transfer हो रहा है।

Disadvantages of goto statement

(1) goto statement प्रोग्राम के logic को ज्यादा complex बना देता है।
(2) goto statement वाले प्रोग्राम को modify करना कठिन होता है।
(3) अगर प्रोग्राम complex है तो goto के flow कंट्रोल का monitor करना कठिन होता है।

Advantages

goto statement का प्रयोग करके प्रोग्राम के normal execution sequence को बदल कर हम प्रोग्राम के किसी भी भाग पर जा सकते हैं।

goto का flow chart

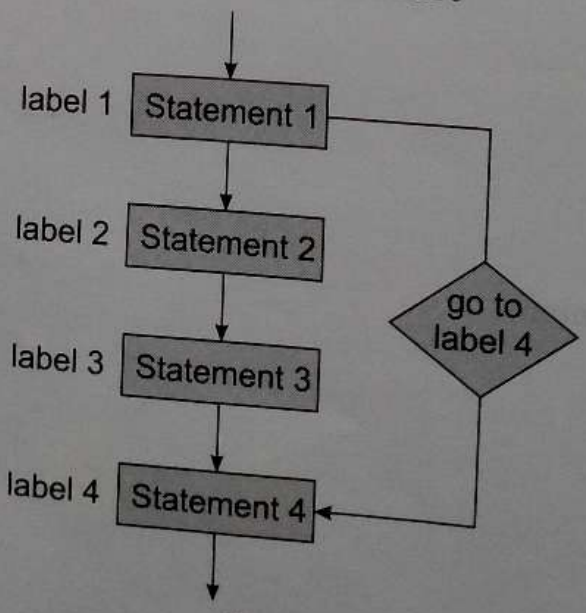


Fig. 3.14

इस flowchart में statement 1 में goto आने के बाद control label 4 पर transfer हो रहा है।

3.5 Break statement

Break statement का प्रयोग loop को terminate करने के लिये किया जाता है। जैसे ही loop के अन्दर break statement आता है loop का iteration बन्द हो जाता है और control, loop से return हो जाता है तथा loop के बाद first statement के पास जाता है। Break का प्रयोग C programming के while, do while या for loop तीनों के साथ कर सकते हैं। Break statement सामान्यत: condition के साथ प्रयोग किया जाता है।

उदाहरण : मान लीजिए हमें 0 से 99 तक के नम्बर में कोई नम्बर search करना है। इसके लिये हम loop का प्रयोग करेंगे। Loop के execution के fifth iteration में में नम्बर मिल जाता है तो हमें बाकी 95 नम्बरों को search या traverse की जरूरत नहीं है। इसमें loop से बाहर आने के लिए break statement का प्रयोग करेंगे।

(i)
```
#include <stdio.h>
int main()
{
    int i;
    for (i = 0 ; i ≤ 8 ; i++)
    {
        if (i == 6)
    {
        break;
    }
        printf ("i = %d\n", i);
    }
    return 0;
}
```

Output :

```
i = 0
i = 1
i = 2
i = 3
i = 4
i = 5
```

प्रोग्राम के 7th iteration में (i == 6) true होने के बाद break statement execute हो रहा है और control loop के बाहर आ जाता है। अगर Break tatement नहीं होता तो loop 9 times execute होगा।

(ii) Switch और Break का प्रयोग करके सप्ताह के दिन को display करने का C program.

```
#include <stdio.h>
int main()
{
    int num;
```

```
        printf ("Enter number for week :");
        scanf ("%d", &num);
        switch (num)
        {
            Case 1 :
            printf ("Monday");
            break ;
            Case 2 :
            printf ("Tuesday");
            break ;
            Case 3 :
            printf ("Wednesday"):
            break ;
            Case 4 :
            printf ("Thursday");
            break ;
            Case 5 :
            printf ("Friday");
            break ;
            Case 6 :
            printf ("Saturday");
            break ;
            Case 7 :
            printf ("Sunday");
            break ;
        }
        return 0;
}
```

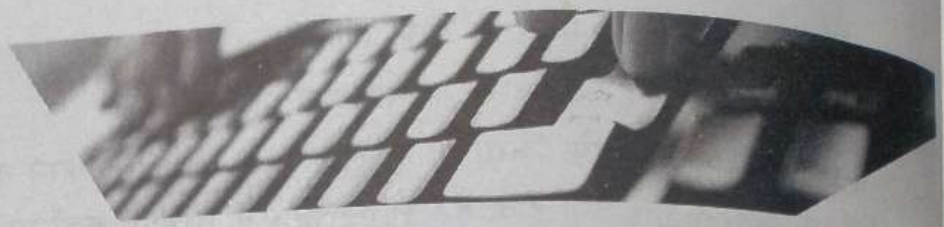
Output : Enter numer for week : 7
Sunday

▶ प्रश्नावली (Exercise) ◀

1. C में control structure क्या होते हैं? विस्तार से समझाइए।
2. If statement का प्रयोग कहाँ होता है ? उदाहरण के साथ समझाइए।
3. If-else statement का प्रयोग कहाँ होता है? उदाहरण के साथ समझाइए।
4. Loop क्या होता है? यह कितने प्रकार के होते हैं?

5. while और do while में अन्तर स्पष्ट कीजिए। **(UPBTE 2018)**
6. for loop क्या होता है? उदाहरण के साथ समझाइए।
7. for loop का प्रयोग करके किसी भी नम्बर का table print करने का C program लिखिए।
8. Break statement क्या होता हैं? Program में Break कहाँ पर प्रयोग किया जाता है?
9. Continue statement को उदाहरण के साथ समझाइए।
10. Switch case() को उदाहरण के साथ समझाइए। **(UPBTE 2018)**
11. for loop का प्रयोग करके किसी भी नम्बर का factorial प्राप्त करने का C program लिखिए।
12. while loop का प्रयोग करके किसी भी नम्बर का table print करने का C program लिखिए।
13. for loop का प्रयोग करके natural number के sum का C program लिखिए।
14. do while का प्रयोग करके किसी भी नम्बर का table point करने का C program लिखिए।
15. Goto statement क्या होता है? इसके disadvantages क्या-क्या होते हैं?
16. for loop का प्रयोग करके 1 से N तक के सभी even numbers को print करने का C program लिखिए।
17. Nested-if क्या होता है? उदाहरण के साथ समझाइए।
18. Ladder if-else को उदाहरण के साथ वर्णन कीजिए।
19. for loop का प्रयोग करके Fibonacci series प्राप्त करने का C program लिखिए।
20. while loop का प्रयोग करके किसी number का factorial प्राप्त करने का C program लिखिए।

❑



Unit

4 Functions

4.1 Function

Function, statement या code का group होता है जो किसी specific task को performs करता है। किसी program मे जब कोई statements या codes का प्रयोग बार-बार करना पड़ता है या एक ही process बार-बार execute कराना पड़ता है तब हम उस statements या codes को अलग-अलग function बना लेते हैं। प्रोग्राम में जरूरत पड़ने पर बनाये हुये function को reuse कर लेते हैं। इस कारण code या statements को हमें बार-बार नहीं लिखना पड़ता है। बार-बार एक ही code नहीं लिखते हुये हम एक ही function को प्रोग्राम में कही भी प्रयोग कर सकते हैं।

4.2 Functions की आवश्यकता

1. एक ही code या statement को हमें बार-बार नहीं लिखना पड़ता है इसलिये code की साइज reduce हो जाती है।
2. Code की debugging आसानी से हो जाती है।
3. Code की reusability बढ़ जाती है।
4. Code की readability अच्छी हो जाती है।

4.3 Function के प्रकार

C Program में दो तरह के functions होते हैं—

(i) Pre-defined या Built-in Functions

(ii) User-defined Functions

(i) Pre-Defined या Built-in standard library Functions

C programming language में पहले से defined कुछ Functions होते हैं। इन functions को Pre-defined या Built-in function कहते हैं। इनका definition पहले से ही header files में define होते हैं जैसे puts(), gets(), printf(), scanf() इत्यादि। इन Pre-defined functions का प्रयोग program में हम Directly कर सकते हैं या आवश्यकता के अनुसार call कर सकते हैं।

उदाहरण—Printf(), scanf(), putsf(), gets(), strlen, strcpy, strcmp, fopen, fprintf etc.

Printf() : Printf() एक library function है जो data को screen पर print करता है। यह Predefined function है जो stdio.h Headerfile में include होता है। Printf() कई प्रकार के arguments को print करा सकता है।

Syntax :

⇒ Printf("user defined message");

⇒ Printf("Format specifiers, value 1, value 2, value 3...);

उदाहरण

```
Printf("C programming");
Printf("% f% c% d", a, b, c);
```

Scanf() : Scanf() एक Predefined/library function है जो stdio.h header file में include होता है। यह keyboard से input value को read करता है।

Syntax : scanf("Format specifier", & values, & value 2,...);

उदाहरण—scanf("%d%d", & a, & b);

Strlen() : strlen() function किसी भी string के length को calculate करता है। यह string.h header file में defined होता है।

Strcmp() : strcmp() दो strings को compare करने के लिए प्रयोग किया जाता है। यह दो strings को input के रूप में लेता है और compare करके integer value को return करता है।

▌(ii) User Defined Functions

User-defined Functions वे functions हैं जो प्रोग्रामर द्वारा प्रोग्राम में जरूरत के अनुसार बनाये जाते हैं। जब हम किसी program को लिखते हैं तो कई बार हमें एक ही तरह के statement के block को बार-बार लिखना पड़ता है। हम बार-बार statement के block को नहीं लिखकर उसका एक function बनाकर declare और define कर देते हैं और आवश्यकता के अनुसार call कर सकते हैं।

User defined Functions के प्रकार—

1. Function with no arguments and no return value
2. Function with no arguments and a return value
3. Function with argument and no return value
4. Function with argument and return value

4.4 Syntax of a Function

```
return_type function_name (argument list)
{
  statements (Block of codes)
}
```

▌return_type :

return_type उस return value को indicate करता है जो Function के द्वारा call और execution के बाद produce होता है। किसी Function का return value किसी भी प्रकार का हो सकता है। यह character, integer, float या double भी हो सकता है।

▌ argument list :

Argument list एक specific input होता है जो function में pass किया जाता है। argument किसी भी function में एक से ज्यादा हो सकते हैं। ये function definition में parentheses के अन्दर होते हैं। जब function call होते हैं तब इनकी value को pass किया जाता है।

▌ function_name :

जब भी किसी function को define और declare करते हैं तो इसके पहले हम उसे एक unique नाम देते हैं इसी नाम से उसे call करते हैं।

Syntax : void function_name().

Function के नाम से उस program में function की calling होती है।

▌ Block of codes :

Function definiton के अन्दर जो statement होते हैं उसे block of codes कहते हैं। Function के execution पर ये statements execute होते हैं और specific कार्य करते हैं।

उदाहरण—किन्हीं दो नम्बरों को add करने का user defined function.

```
#include <stdio.h>
int sum(int n1, int n2)          // Function prototype
int main()
{
   int m, n;
   printf("Enter first number\n ");
   scanf("%d",&m);
   printf("Enter second number:");
   scanf("%d",&n);
   int fcall = sum(m, n);        // Function call
   printf ("output: %d", fcall);

   return 0;
}
int sum(int n1, int n2)          // Function definition
{
   int result;
   result = n1+n2;
   return result;
}
```

Output :

Enter first number :68

Enter second number :78

Output: 146

इस उदाहरण में sum() नाम को एक Function बनाया गया है जो कि user defined है। n1 और n2 दो integer type के argument को pass किया गया है और Function का return भी Integer type का है। Function call के द्वारा दो integers को Argument के रूप में लेकर integer type का value return हो रहा है।

4.5 Function Prototype

Function Prototype, Function का declaration होता है जो Function का नाम और इसका return type को specify करता है।

Function Prototype निम्नलिखित सूचना देता है।

1. यह Function द्वारा return data के type को बताता है।
2. कितने Arguments pass होंगे इसको भी बताता है।
3. यह प्रत्येक argument के data type को बताता है।
4. यह Argument के order को भी बताता है।
5. यह Function name को भी specify करता है और declare करता है।

▌Syntax of function prototype :

returnType functionName(type1 argument1, type2 argument2,...);

Function name—यह Function का वास्तविक नाम होता है। Program में Function name से ही Function को call किया जाता है। यह unique होता है।

▌Function Call :

Function call एक expression होता है जो अपना control और argument दोनों को (if any) Function को transfer करता है।

Function call दो प्रकार से होता है—

1. Call by value
2. Call by reference

Syntax of Function Call :

functionName(argument1, argument2, ...);

Function call का उदाहरण

```
#include<stdio.h>
void mux();    //Function Declaration
void main()
{
  mux();       //Function call
}
void mux()
{
  int mux, p = 11, q = 9;
  result = p * q;
```

```
    printf("\n multiply of p = %d and q = %d is %d", p, q, result);
}
```

Output :

multipleof 11 and 9 is 99.

इस Program में mux() function call हो रहा है और इसमें input के रूप में p और q Parameter pass हो रहे हैं।

Function definiton :

Function Prototype के बाद Function को define किया जाता है। यह Statements का group होता है जो Function call होने पर execute होते हैं। Statements यह specify करता है कि Function execute होने पर क्या कार्य करेगा।

Syntax of Function definition

```
returnType functionName(type1 argument1, type2 argument2, ...)
{
  body of the function
}
```

Function को main के पहले और main के बाद दो तरीके से define कर सकते हैं।

(i) Function definition before main :

Function definition का उदाहरण (main के पहले)

```
#include<stdio.h>
#include<conio.h>
int add(int m, int n)
{                          // Function Definiton before main
  return m+n;
}
int main()
{
  int m, n, sum;
  printf("Enter two numbers\n");
  scanf("%d %d", &m, &n);
  sum = add(m, n);                  // function call
  printf("sum = %d", sum);
  getch();
  return 0;
}
```

Output :

```
Enter two numbers
67 68
sum = 135
```

(ii) Function definition का उदाहरण (main के बाद)

function definition after main

```
#include<stdio.h>
#include<conio.h>
int add(int m, int n);
int main()
{
  int m, n, sum;
  printf("Enter two numbers\n");
  scanf("%d %d", &m, &n);
    sum = add(m, n);          //function call
  printf("sum = %d", sum);

  getch();
  return 0;
}

int add(int m, int n)
{
  return m+n; .              // Function Definition
}
```

Output :

Enter two numbers
67 68
sum = 135

4.6 Passing Arguments to a Function

Function call के समय Function में दो तरीके से argument को pass कर सकते हैं। argument एक तरह का variable है जो एक specific input होता है। किसी भी function declaration में यह एक optional भाग होता है। इसकी संख्या एक से ज्यादा हो सकती है।

▌1. Call by value :

Call by value में Function, actual argument value को Function के Formal parameter में copy करता है। First actual argument की value, First Formal Parameter में copy होता है और second actual argument की value, second Formal Parameter में copy होती है।

Actual Parameter – Actual parameter, Function call में appear होते हैं।

Formal Parameter – Formal parameter, Function declaration में appear होते हैं।

उदाहरण : Function में Call by value का उदाहरण

```
#include <stdio.h>
int mux (int m);
void main()
{
   int p = 4;
   printf("\n value of 'p' before the calling function is = %d", p);
   p = mux(p);
   printf("\n value of 'p' after calling the function is = %d", p);
}
int mux (int m)
{
   m = m*18;
   printf("\n value of 'p' in the called function is = %d", p);
   return n;
}
```

Output :

```
value of 'p' before the calling function is = 4
value of 'p' in the called function is = 72
value of 'p' after calling the function is = 72
```

▌2. Call by reference

Call by reference में Function, argument के address को Formal parameter में copy करता है। Address का प्रयोग करके actual argument को access किया जाता है। reference value को pass करने के लिए pointers का प्रयोग किया जाता है। Parameter में कोई भी बदलाव passed argument में भी बदलाव लाता है।

उदाहरण—

```
#include <stdio.h>
int mux (int *m);
void main()
{
   int p = 4;
   printf("\n value of 'p' before the calling function is = %d", p);
   mux(&p);
   printf("\n value of 'p' after calling the function is = %d", p);
}
int mux (int *m)
{
   *m = *m* 18;
   printf("\n value of 'p' in the called function is = %d", m);
}
```

Output :

value of 'p' before the calling function is = 4
value of 'p' in the called function is = 63723562
The value of 'p' after calling the function is = 72

इस Program में argument के रूप में m का address pass हो रहा है।

4.7 Return Statement

Return statement का कार्य Function के execution को terminate करके program के control को वापस Calling Function को देना होता है। यह ये भी specify करता है कि Function का Return क्या होगा। return statement दो तरीके से प्रयोग किये जाते हैं—

Syntax :

1. return;
2. return (expression)

1. return : यह statement no value return को दर्शाता है, अर्थात् Function call के बाद कोई value return नहीं हो रहा है।

2. return (expression) : Return statement का यह form, Function के return value को दर्शाता है। return value किसी भी प्रकार का constant, variable इत्यादि हो सकते हैं। return (expression) में expression evaluate होने के बाद value return होता है। expression इसमें optional होता है।

उदाहरण—

```
return a;
return result;
```

4.8 User-defined Function के प्रकार

(i) Function with no arguments and no return value
(ii) Function with no arguments and a return value
(iii) Function with arguments and no return value
(iv) Function with arguments and a return value

▌(i) Function with no arguments and no return value

जब किसी Function में कोई argument नहीं होता है तो यह calling function से कोई भी data या argument नहीं receive करता है। और जब इसमें कोई return value नहीं होता है तो यह called function से कोई भी data receive नहीं करता है। ऐसे Function, no argument और no return value की श्रेणी में आते हैं।

Syntax : Function declaration : void function();
Function call : function();
Function definition : void Function()

```
{
    Statements;
}
```

```
/*Function with No argument and No Return value Example */
#include<stdio.h>
void sum();        // Function Declaration
void main()
{
   sum();   //function call
}
voidsum()          //function definition
{
   int result, m = 75, n = 87;
   result = m+ n;

   printf("\n sum of m = %d and n = %d is %d", m, n, result);
}
```

OUTPUT :

Sum of m and n is 162

(ii)
```
#include<stdio.h>
  void smallnum();       // function declaration
  int main()
  {
     smallnum.();        //function call
     return 0;
  }
  void smallnum()       //function definition
  {
     int m, n;
     printf("Entertwo numbers");
     scanf("%d%d", &m, &n);
     if(m> n)
  {
        printf(" smaller number is: %d", n);
     }
     else
  {
        printf(" smaller number is: %d", m);
     }
  }
```

(iii) C program using function to input a number and find Fibonacci series.

```
#include <stdio.h>
void fibonacci();    /* Function declaration */
```

```
int main()
{
  fibonacci ();
  return 0;
}
void fibonacci ()   /* Function definition */
{
  int p, q, s, i, n;
printf("Enter number of terms: ");
  scanf("%d", &n);

    p = 0;
    q = 1;
    s = 0;
printf("Fibonacci terms: \n");
  for(i=1; i<=n; i++)
  {
    printf("%d, ", s);
    p = q;
    q = s;
    s = p + q;
  }
}
```

Output :

Enter number of terms: 6

Fibonacci terms:

0, 1, 1, 2, 3, 5

▌(ii) Function with no arguments and a return value :

ऐसे Function में कोई भी argument नहीं होता है पर value.return होता है।

Syntax :

Function declaration : int function();

Function call : function();

Function definition : int function()

```
{
statements;
return x;
}
```

उदाहरण—Function with no arguments and with Return value

(i) Sum of two numbers using function with no arguments and with Return value

```
#include<stdio.h>
```

```
int sum();                  //function declaration
int main()
{
   int result;
   result = sum();          //function call
   printf("\n sum of p and q is = %d \n", result)
   return 0;
}
int sum()                   //function definition
{
   int sum, p = 75, q = 98;
   result = p+q;
   return result;
}
```

OUTPUT :

```
sum of p and q is = 173
```

(ii)

```
#include <stdio.h>
   int smallnum();      // function declaration
   int main()
{
   int output;
   output = smallnum();      // function call
   printf("The smaller number is : %d", output");
   return 0;
}
int smallnum()      // function definition
{
   int m, n, smallnum;
   printf("Enter the numbers");
   scanf("%d%d", &m, &n);
   if(m >n)
{
      smallnum = n;
   }
   Else
{
   smallnum= m;
   }
   return smallnum;
}
```

▌(iii) Function with arguments and no return value

जब function में argument होता है तो यह Calling function से data या Parameter receive करता है। इसमें value return नहीं होता है तो ऐसे function argument and no return value की श्रेणी में आते हैं।

Function declaration : void function (int);

Function call : function (x);

Function definition : void function (int x)

```
{
statements;
}
```

उदाहरण—Function with arguments and no Return value

Sum of two numbers using function with arguments and no Return value

```
#include<stdio.h>
void sum(int, int);          // function declaration
void main()
{
  int p, q;
printf("\n Enter two numbers \n");
  scanf("%d %d",&p, &q);
  sum(p, q);          //function call
}
void sum(int p, int q}      //function definition
{
  int result;
  result = p + q;
  printf("\n sum of p and q is = %d \n", result);
}
```

OUTPUT :

```
Enter two numbers
67  78
```

▌(iv) Function with arguments and a return value

ऐसे function में Argument pass होता है और value भी return होता है। function, calling function से data या Parameter प्राप्त करता है और Called function को data return करता है।

Syntax :

Function declaration : int function (int);

Fuction call : function (x);

Function definition : int function (int x)

```
{
statement 1
statement 2
```

```
                    statement n
                      return x;
                      }
```

उदाहरण—Function with arguments and Return value

(i) Multiplication of two numbers using function with arguments and Return value

```
#include<stdio.h>
int mux(int, int);              /* Function declaration */
int main()
{
   int p, q, result;
   printf("\n enter two numbers \n");
   scanf("%d %d",&p, &q);
result = mux(p, q);             /* Function call */
printf("\n Multiplication of p and q is = %d \n", p, q, result);
  return 0;
}
int mux(int p, int q)       /* Function definition */
{
  int result;
  result = m * n;

  return result;
}
```

OUTPUT :

```
Enter two numbers
8 9
Multiplication of p and q is = 72
```

(ii) Smaller number प्राप्त करने का C program

```
#include <stdio.h>
int smallnum(int m, int n);     // function declaration
int main()
{
  int m, n, output;
  printf("Enter the numbers ");
  scanf("%d%d", &m, &n);
  output = smallnum(m, n); // function call
  printf("The smaller number is: %d", output);
  return 0;
}
```

```
int smallnum(int m, int n)   // function definition
{
  if(m> n)
{
    return n;
  }
  Else
{
    return m;
  }
}
```

(ii) Function का प्रयोग करके दो नम्बरों को add करने का C program

```
#include <stdio.h>
    int Addition(int n1,int n2)
   {
    printf("%d", n1);
printf("%d", n2);
    return nl + n2;
    }
    int main()        {
    int result, m1, m2;
scanf("%d", &m1);
    scanf("%d", &m2);
    result = Addition(m1,m2);
    print(f' result = %d\n", result);
    return 0;
}
```

(iv) C program using function to check even or odd

```
#include <stdio.h>
int find(int n);       /* Function declaration */
int main()
{
    int n, fcall;
  printf("Enter the number:");
    scanf("%d", &n);
  fcall = find(n);          /* Function call */
  if(fcall = = 0)
    printf(" number is even");
  else
    printf(" number is odd");
```

```
    return 0;
}
find(int n)        /* Function definition */
{
    if(n%2==0)
    return 0;
  else
  return 1;
}
```

Output :

Enter a number: 4
number is even.

4.9 Recursion

जब किसी प्रोग्राम में किसी Function के अन्दर same function call होता है तो इस Process को recursion कहते हैं। recursion का प्रयोग करते समय Programmer को exit condition define करना होता है जो iteration को terminate करता है। अन्यथा यह infinite loop में iterate करता रहेगा।

Syntax :

```
function1()
{
  //function1 body
  function1();
  //function1 body
}
```

Recursion का प्रयोग करके किसी number का Factorial प्राप्त करना

```
#include<stdio.h>
int fact(int m);     //declaring the function
void main()
{
    int p, q;
  printf("Enter the number");
  scanf("%d", &p);
  q = fact(p);     // function call
  printf("%d", q);
}
int fact(int p) // function definition
{
  int f = 1;
  if (p = = 0)
    return 1;
```

```
else
return p*fact (p–1);
}
```

4.10 Local variables

Local variable किसी function block के अन्दर declare होते हैं। इसका scope केवल उस function तक होता है जिसके अन्दर इसका declaration होता है। local variable केवल उस function के statements के द्वारा ही access हो सकते हैं। उस Program के दूसरे function से access नहीं हो सकते हैं। इसका life केवल उस function के शुरुआत और execution के end तक ही रहता है।

Global Variable

Global variable किसी भी function के बाहर Declare होता है और ये Program में किसी भी function में use हो सकते हैं। इसका scope पूरे Program में होता है। Global variables को program के सभी functions के द्वारा access किया जा सकता है। इसका life प्रोग्राम के execution तक बना रहता है।

4.11 Local और Global Variable में अन्तर

Local variable	Global variable
1. Local variable किसी भी function के अन्दर declare होता है।	1. Global variable function के बाहर declare होते हैं।
2. Local variable का scope उस function के अन्दर होता है जिसमें उसका declaration हुआ है।	2. Global variable program के किसी भी function के द्वारा access हो सकते हैं।
3. Local variable का life उस function के शुरुआत और exit के duration में ही होता है।	3. Global variable का life पूरे Program के execution तक बना रहता है।

Local और global variables के C प्रोग्राम

(i)
```
#include <stdio.h>
 int m,n;          /*global variables*/
 void  values1(void)
{
        m=89;
        n=98;
}
int main()
{
        int p,q;   /*local variables*/
        p = 78;
        q=87;
        values1();
        printf("m=%d, n=%d\n",m,n);
```

```
        printf("p=%d, q=%d\n",p,q);
        return 0;
}
```

Output :

```
m=89, n=98
p=78, q=87
```

(ii)

```
#include <stdio.h>
int m,n;                    //global variables
void setValues(void)
{
        m=95;
        n=78;
}
int main()
{
        int p,q;        //local variables
        p=96;
        q=81;
        setValues();
        printf("m=%d, n=%d\n",m,n);
        printf("p=%d, q=%d\n",p,q);
        return 0;
}
```

Output :

```
m=95, n=78
p=96, q=81
```

Global variables are: m and n
Local variables are: p and q

(iii)

```
#indude<stdio.h>
int m=98;
void output();
void main()
{
int m=89;
 {
 int m = 75;
 printf("%d ",m);
 }
```

```
printf("%d ",m);
output();
}
void output()
{
printf("%d ",m);
}
```

Output :

75 89 98

4.12 Array and Function

C program में array का एक single element या entire array, Formal parameter के रूप में function में pass किया जा सकता है।

Passing One-dimensional Array to a Function

Passing single element of an array to function

```
#include <stdio.h>
void print(int num)
{
  printf("%d", num);
}

int main()
{
  int number[] = {1,2, 3, 8,5,6};
  print(number[3]);          //Passing array element number[3]
  return 0;
}
```

Output :

8

Passing an entire array to a function

(i) //C Program for addition by passing an array to a function

```
#include <stdio.h>
int addition(int num[]);
int main()
{
        int add, num[] = {78, 87, 67, 89, 65};
        add = addition(num);              // name of an array is passed as an argument
        printf("addition is = %,d", add);
        return 0;
}
```

```
int addition(int num[])
{
        int m;
        int add, sum = 0;
        for (m = 0; m <= 4; m++)
    }
            sum = sum + num[m];
        {
            return sum;
}
```

Output :

sum = 386

(ii) //C program to find Average by passing entire array to function

```
#include <stdio.h>
#include <conio.h>
  float avg(float a[])
    {
      int i
    float avg, sum = 0;
    for (i=0; i≤5; i++)
      {
        sum = sum + a[i];
          }
          avg = sum/6;
            return avg;
        }
int main()
      {
        float m, n[] = {80, 70, 69.5, 78, 68, 75}
          m = avg(n);
    printf("average of number is : %f', avg);
        retur0;
    }
```

output = 73.4

Solved Examples

C program to find the addition of two number using function.

```
#include <stdio.h>
void Add();
```

```
void main()
{
  Add();
}
voidAdd()
{
int sum, x= 68, y = 88;
sum = x + y;
printf("\n sum of x= %d and y = %d is = %d", x, y, sum);
}
```

Output :

sum of 68 and 88 = 156

C program to find the addition of two numbers passing as parameter using function.

```
#include<stdio.h>
int Addition(int num1,int num2 )
{
printf("%d", num1);
printf("%d", num2);
return num1 + num2;
        }
int main()
{
int result, n1, n2;
scanf("%d", &n1);
scanf("%d", &n2);
result = Addition(n1,n2);
printf(" answer = %d\n", result);
return 0;
}
```

C program to find multiplication of two numbers using function with No arguments and with Return value .

```
#include <stdio.h>
int mux();
int main()
{
  int multi;
  multi = mux();
  printf("\n Multiplication of m and n is %d \n", multi );
return 0;
}
```

```
int mux()
{
   int multi, m = 14, n = 6;
      multi = m * n;
   return multi;
}
```

OUTPUT :

Multiplication of m and n is 84

C program to find multiplication of two numbers using function with arguments and with Return value

```
#include <stdio.h>
int mux(int, int);
int main()
{
   int m, n, multi;
   printf("\n Enter two numbers\n");
   scanf("%d %d",&m, &n);
   multi = mux(m, n);
   printf("\n multiplication of %d and %d is %d \n", m, n, multi);
   return 0;
}

int mux(int m, int n)
{
   int multi;
multi = m* n;
return multi;
}
```

OUTPUT :

Enter two numbers
14 6
multiplication of m and n is 84.

C program to create a user defined function addition()

```
#include <stdio.h>
int add(int n1, int n2)
{
   int sum;
   sum = n1+n2;
   return sum;
}
```

```
int main()
{
  int m, n;
  printf("Enter number first :");
  scanf("%d",&m);
  printf("Enter number second:");
  scanf("%d",&n);
  int result = add(m, n);
  printf ("output is : %d", result);
  return 0;
}
```

Output :

```
Enter number first: 88
Enter number second: 97
Output: 185
```

C program to print table of any integer number using User Define Functions.

```
#include <stdio.h>
  void table(int);
  int main()
{
  int n;
  printf("enter an integer number:");
  scanf("%d",&n);
  printf("table of %d is:\n",n);
  table(n);
  return 0;
}
void table(int num)
{
  int i;
  for(i=1; i<=10; i++)
  printf("%d\n",(num*i));
}
```

```
Enter an integer number: 11
Table of 11 is:
  11
  22
  33
  44
  55
```

```
66
77
88
99
110
```

C Program to convert binary number to decimal using function

```
#include <stdio.h>
#include <math.h>
int conversion(long n);
int main()
{
  long n;
  printf("Enter the binary number: ");
  scanf("%ldd", &n);
  printf("%ldd binary = %d decimal", n, conversion(n));
  return 0;
}

int conversion(long n)
{
  int decinum = 0, i = 0, remi;
  while (n!=0)
  {
    remi= n%10;
    n /= 10;
    decinum += remi*pow(2,i);
    ++i;
  }
  return decinum;
}
```

Output :

```
Enter a binary number: 1111
1111 binary = decimal 15
```

C program to find sum and average of two numbers using User Define Functions.

```
#include <stdio.h>
int sum(int,int);
float average(int,int);
int main()
{
  int n1, n2;
```

```
    int sum;
    float avg;
    printf("Enter the first number: ");
    scanf("%d",&n1);
    printf("Enter the second number: ");
    scanf{"%d",&n2);
    sum=sum(n1,n2);
    avg=average(n1,n2);
    printf("Numberl: %d, Number2: %d\n",n1,n2);
    printf("Sum: %d, Average: %f\n",sum,avg);
      return 0;
}
int sum(int x,int y)
{
  int sum;
  sum=x+y;
  return sum;
}
float average(int x,int y)
{
  float average;
  return ((float)(x)+(float)(y))/2;
}
  Enter the first integer number: 78
  Enter the second integer number:99
  Number1: 78, Number2: 99
  Sum: 177, Average: 88. 50
  return 0;
  result = 125
```

C program to reverse a sentence entered using recursion

```
#include <stdio.h>
void reversestring();
int main()
{
  printf{"Enter the string: ");
  reversestring();
  return 0;
}
void reversestring()
{
```

```
    char c;
    scanf("%c", &c);

    if(c != '\n')
    {
      reversestring();
      printf("%c", c);
    }
}
```

Output :

```
Enter the sentence: datastructure
erutcurtsatad
```

C program example to Passing a complete One-dimensional array to a function

```
#indude <stdio.h>
float average(int marks[]);
int main{)
{
  float avg;
  int marks[] = {95, 92, 99, 91 };
  avg = average(marks);
  printf("average marks = %.2f", avg);
  return 0;
}

float average(int marks[])
{
  int i, sum = 0;
  float avg;
  for (i = 0; i <= 3; i++)
{
    sum =sum+ marks[i];
  }
  avg = (sum/ 4);
  return avg;
}
```

Output :

```
94.25
```

▶ प्रश्नावली (Exercise) ◀

1. Function क्या होता है? वर्णन कीजिए।
2. Function Prototype क्या होता है? उदाहरण के साथ समझाइए।
3. Function को कैसे define करते हैं? समझाइए।
4. Function कितने प्रकार के होते हैं? उदाहरण के साथ समझाइए।
5. Function में Parameter कैसे पास करते हैं?
6. Call by value और Call by reference में क्या अन्तर है? **(UPBTE 2017, 2018)**
7. User defined function और library function में अन्तर स्पष्ट कीजिए।
8. किसी भी number का factorial प्राप्त करने का C program लिखिए।
9. Local veriable और Global variable में अन्तर स्पष्ट कीजिए।
10. Recursion क्या होता है? उदाहरण के साथ समझाइए।
11. Recursion का प्रयोग करके किसी number का factorial प्राप्त करने का C program लिखिए।
12. One dimensional array को Function में कैसे pass करते हैं? Program के साथ समझाइए।
13. Fibonacci series प्राप्त करने का C program लिखिए।
14. Main function क्या होता है? समझाइए।
15. दो नम्बरों को swap करने का C program लिखिए। **(UPBTE 2018)**
16. Function का प्रयोग करके किसी array का Largest number प्राप्त करने का C program लिखिए।
17. Function को कैसे call करते हैं? उदाहरण के साथ समझाइए।
18. किसी Two dimensional array को function के द्वारा कैसे access करते हैं? समझाइए।
19. Even और odd नम्बर check करने का C program लिखिए।
20. Function की क्या आवश्यकता है? समझाइए।

❑

Unit

5 Array and String

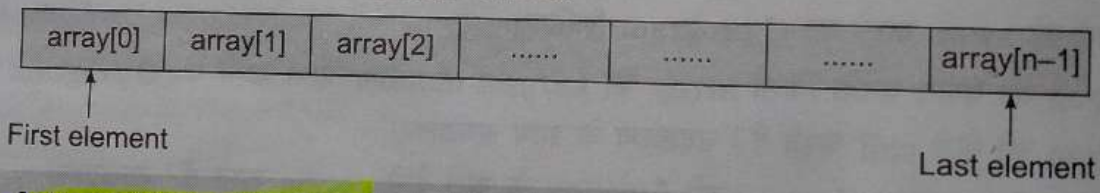
5.1 Array

Array एक linear data structure होता है जो एक ही तरह के Fixed साइज के data को sequentially store करता है। Array में संग्रहित डेटा का साइज और टाइप एक ही होता है।

उदाहरण—यदि आप 100 integers करना चाहते हैं जो कि एक sequence में हों तो एक array create कर सकते हैं जिसका data type integer हो और 100 integer values को store कर सके—

```
int array[100]
```

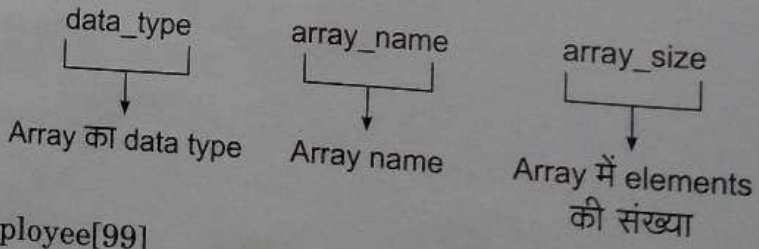
array एक contiguous memory को allocate करता है। Lowest address पहले element को और highest address अन्तिम element को denote करता है।



5.2 Array का Declaration

Array declaration यह decide करता है कि array का data type क्या है। यह array का नाम और array size को भी बताता है।

Syntax :



Example : int employee[99]

इसमें employee नाम का एक array बनाया गया है जिसमें 99 elements हैं जो integer type के हैं।

⇒ Char name[78]

name नाम का array जिसकी length 78 है जो Character type के हैं।

5.3 Array का Initialization

Array को दो प्रकार से initialize कर सकते हैं—

1. Static array Initialization : इस प्रकार के array initialization में array declaration के समय initialize होता है।
2. run time या dynamic : इस प्रकार के array initialization में program execution के समय array को initialize करते हैं।

▌Static array initialization

```
int age[6] = {85, 76, 81, 91, 93, 88};
int age[] = {85, 76, 81, 91, 93, 88};
```

array initialization में array का साइज optical होता है।

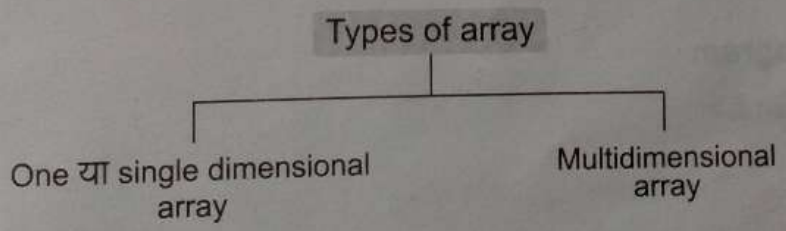
▌Memory allocation

```
int age[6] = {85, 76, 81, 91, 93, 88};
```

age[0]	age[1]	age[2]	age[3]	age[4]	age[5]
85	76	81	91	93	88

age[0] = 85
age[1] = 76
age[2] = 81
age[3] = 91
age[4] = 93
age[5] = 88

5.4 Array का प्रकार



▌Single Dimensional Array

One dimensional array में data एक linear form में store होता है। इसमें केवल एक ही subscript variable होता है। इसे Linear array भी कहते हैं।

Syntax :

```
data type array_name[array_size]
```

उदाहरण— `int age[75]`
`char employee[86]`

उदाहरण—

Reading और Printing element

```
(i) #include<stdio.n>
  int main()
    {
      int age[6], i;
  for (i = 0; i < 6; i ++)
      {
  printf('Enter age [%d]: " i);
      scanf("%d", &age[i]);
      }
  printf("\n printing element of array :/");
      for (i = 0; i < 6; i ++)
        {
      printf("%d", age[i]);
        }
    return 0;
    }
```

Output :
```
Enter age[0] : 85
Enter age[1] : 76
Enter age[2] : 75
Enter age[3] : 91
Enter age[4] : 88
Enter age[5] : 92
Printing element of array :
85  76  75  91  88  92
```

Array Access का Program

```
(ii) #include <stdio.h>
  int main()
    {
      int i;
  int age[6] = {85, 79, 76, 82, 92, 78};
    for (i = 0; i < 6; i ++)
        {
  printf("value of age[%d] : %d\n", i, age[i]);
    }
    }
```

Output :
```
value of age[0] : 85
value of age[1] : 79
value of age[2] : 76
```

```
value of age[3] : 82
value of age[4] : 92
value of age[5] : 78
```

(iii) Largest element प्राप्त करने का C program

```
#include <stdio.h>
  int main()
    {
  int arr[75], n, i, largest;
printf("Enter the number of element : ");
  scanf("%d", &n);
printf("\n Enter element of array" : );
for (i = 0; i < n; i ++)
  scanf("%d", &arr[i]);
    largest = arr[0];
  for (i = 1; i < n; i ++)
      {
      if (largest < arr[i])
        largest = arr[i];
        }
  printf("\n largest element is : %d, largest);
      return 0;
      }
```

Output :

```
Enter the number of element : 6
Enter element of array :
    72
    86
    69
    75
    88
    91
largest element is : 91
```

▌Multidimensional array

Array जिनको दो या दो से ज्यादा dimension होते हैं, उसे multidimensional array कहते हैं। Arrays के array को भी multidimensional array कहते हैं। इसमें data row और column के रूप में store होते हैं। Multidimensional array में data tabular form में arrange होते हैं। Two dimensional array एक array का array होता है। Three dimensional array एक arrays के arrays का array होता है।

Types of multidimensional array :

(i) Two dimensional array

(ii) Three dimensional array

declaring multidimensional array

data_type array_name[size1[[size2] ... [size n]

उदाहरण—Two dimensional array :

int array[3] [4]

Three dimensional array

int array[5] [3] [4]

▌Multidimensional array का size

Multidimensional array के सभी dimensions को Multiply करने पर प्राप्त होने वाली संख्या के बराबर मान को multidimensional array store कर सकता है।

उदाहरण—

int array[5] [14]

$5 \times 14 = 70$ elements (size)

int array[6] [5] [3]

$6 \times 5 \times 3 = 90$ elements (size)

5.5 Two Dimensional Array

Two dimensional array को Matrix भी कहते हैं। इसमें data का row और column के order में store होता है। row major order और column major order से किसी Data का address प्राप्त कर सकते हैं।

Syntax :

data_type array_name[m] [n];

m denotes row

n denotes column

उदाहरण— int array[3] [3]

	Column 1	Column 2	Column 3
Row 0	array[0] [0]	array[0] [1]	array[0] [2]
Row 1	array[1] [0]	array[1] [1]	array[1] [2]
Row 2	array[2] [0]	array[2] [1]	array[2] [2]

▌Two dimensional array का initialization

int array[3] [3] = {0, 1, 2, 3, 4, 5, 6, 7, 8};

or int array[3] [3] = {{0, 1, 2,}, {3, 4, 5}, {6, 7, 8}};

Two dimensional array में data को store और access करना

उदाहरण—

```
#include <stdio.h>
   int main()
     {
```

```
    int array[3] [3]      //array declaration
    int m, n;
  for (m = 0; m < 3; m ++)
        {
  for (n = 0; n < 3; n ++)
      {
  printf("Enter the data in two dimensional array\n");
  scanf("%d", & array[m] [n]); }}
printf("Two dimensional array elements : \n");
    for (m = 0; m < 3; m ++)
        {
    printf("\n")
    for (n = 0; n < 3; n ++)
  printf("%d", array[m] [n]);
      }
    printf("\n");
    return 0;
    }
```

Enter the data in two dimensional array :

1 2 3 4 5 6 7 8 9

Two dimensional array elements :

1 2 3

4 5 6

7 8 9

Static या compile time array initialization का उदाहरण

```
#inciude<stdio.h>
void main()
{
  int m;
  int array[] = {1, 2, 3, 4,5,6};    // Compile time array initialization
  for(m = 0 ; m < =5 ; m++)
  {
    printf("%d\t",array[m]);
  }
}
```

Output :

1 2 3 4 5 6

Dynamic या run time array initialization का उदाहरण

```
#include<stdio.h>
void main()
{
  int array[7];
  int m, n;
  printf("Enter element in the array");
  for(m = 0; m <= 6; m++)
  {
    scanf("%d", &array[m]);   //Run time array initialization
  }
  for(n= 0; n<= 6; n++)
  {
    printf("%d\t", array[n]);
  }
}
```

Output :

Enter element in the array: 0 1 2 3 4 5 6

Array के elements को read और display का C प्रोग्राम

```
#include <stdio.h>
int main()
{
int a[85], m, p ;
printf("Numbers of Element:");
scanf("%d ",&m);
printf ("\n Enter the Element: \n");
for ( p = 0;p< = m–1; p++)
{
scanf("%d ",& a[p]);
}
printf("\n Elements are : \n ");
for ( p= 0; p< = m–1 ; p++)
{
printf("\n%d",a[p]);
}
return ( 0 );
}
```

Output:

Numbers of Element:

7

```
Enter the Element:
6
3
4
5
2
1
0
Elements are :
6
3
4
5
2
1
0
```

Array के elements को delete करने का C प्रोग्राम

```
#indude <stdio.h>
int main()
{
  int a[85], p, i, m;
  printf("Enter number of elements \n");
  scanf("%d", &m);
  for ( i = 0 ; i <= m; i++ )
{
  scanf("%d",&a[i]);
}
printf("Enter the location for deletion \n");
  scanf("%d", &p);
  if ( p >= m+l )
  printf("Deletion not possible, \n")
  else
  {
    for ( i = p – 1 ; i< =m – 1 ; i++ )
      a[i] = a[i+1];
  printf("result is\n");
  for( i = 0 ; i<= m – 1 ; i++ )
  printf("%d\n", a[i]);
  }
  return 0;
}
```

Output :

```
Enter number of elements : 6
6
4
2
5
1
3
Enter the location for deletion:
3
Result is
6
4
5
1
3
```

C programming to find out the average of 6 integers using array

```
#include <stdio.h>
int main()
{
  float avg = 0;
  int sum =0;
  int m=0;
  int num[6];
for (m=0; m<=5;m++)
{
    printf("Enter number %d \n", (m+1));
    scanf("%d", &num[m]);
  }
  for (m=0; m<=5;m++)
  {
    sum = sum+num[m];
  }

  avg = sum/6;
  printf("Average of entered numbers is: %f", avg);
  return 0;
}
```

Output :

```
Enter number 1
80
Enter number 2
70
Enter number 3
70
Enter number 4
60
Enter number 5
80
Enter number 5
82
Average of entered numbers is: 73.6
```

C Program to find the average of n numbers using arrays

```
#include <stdio.h>
int main()
{
  int mark[20], i, n;
  float sum = 0, average;
  printf("Enter the value of n;");
  scanf("%d", &n);
  for(i=0; i<n; ++i)
  {
    printf("Enter number %d: ",i+1);
    scanf("%d", &mark[i]);
    sum = sum +marks[i];
  }
  average = sum/n;

  printf("Average = %f", average);

  return 0;
}
```

Output :

```
Enter the value of n: 5
Enter number1: 68
Enter number2: 72
Enter number3: 98
Enter number4: 78
Enter number5: 76
Average = 78.4
```

Array से data को find करना और उसकी position को Display का C प्रोग्राम

```
#include <stdio.h>
int main()
{
   int a[95] ,i ,j,m,data;
   printf("Enter number of elements \n");
   scanf("%d", &m);
   for (i = 0 ; i <= m–1; i++)
{
scanf("%d",&a[i]);
}
   printf("array elements are :\n");
        for(i = 0; i<=m–1; i++)
{
     printf(" %d\n", a[i]);
  }
   printf("Enter number to be found \n");
   scanf("%d", &j);
   while(j<m)
{
    if(a[j] == data )
{
       break;
  }

  j = j + 1;
  }
printf("element found at position %d\n", data, j+1);
}
```

Output :

```
Enter number of elements
6:
Array elements are :
4
3
6
3
1
5
Enter number to be found
1
element found  at position 5
```

C Program to find Sum and average of elements of array

```
#include <stdio.h>
int main( )
{
int a[75], m, p ;
float avg = 0, sum = 0 ;
printf(" Total number of elements in Array: ") ;
scanf("%d ", &m) ;
printf("\n Enter the Element : \n") ;
for ( p = 0 ; p <= m–1 ; p++)
{
scanf("%d",&a[p]);
}
for ( p= 0 ; p<= m–1 ; p++)
{
sum = sum + a[p] ;
}

avg = sum/ m ;

printf("\n Sum of Elements is : %f ",sum);
printf("\n Average of Elements is : %f ",avg) ;
return (0) ;
}
```

Output :

```
Total number of elements in Array:
5
Enter the Element:
67
87
85
76
78
Sum of Elements is: 393
Average of Elements is 78.6
```

Matrix के Transpose प्रिन्ट का C प्रोग्राम

```
# include < stdio.h >
int main( )
{
int arr[5][5], a, b, row, col;
printf(" Enter Numbers of Row : ");
```

```
scanf("%d ",& row);
printf("\n Enter Numbers of Column : ");
scanf("%d ",& col);
printf("\n Enter the Matrix data : \n");
for ( a = 0 ; a <= row–1 ; a++)
{
for ( b = 0 ; b < =col–1 ; b++)
{
printf("\n Enter the element [%d] [%d] : ", a, b);
scanf("%d",&arr[a][b]);
}
}
printf("\n\n Element in the matrix are : \n");
for ( a= 0 ; a<= row–1 ; a++)
{
for ( b = 0 ; j <= col–1 ; b++)
{
printf("\t%d", arr[a][b]);
}
printf(" \n ");
}
printf("\n\n Transpose of a Matrix : \n");
for ( a= 0 ; a<= row–1 ; a++)
{
for ( b = 0 ; b<= col–1 ; b++)
{
printf("\t %d ", arr[b][a]);
}
printf(" \n ");
}
return ( 0 );
}
```

Output :

```
Enter Numbers of Row : 2
Enter Numbers of Column :4
Elements in the matrix are : 5 6 2 4
                             5 1 4 5
Transpose of a Matrix :      5   5
                             6   1
                             2   4
                             4   5
```

Two-dimensional array elements को read और display का C प्रोग्राम

```
#include <stdio.h>
int main()
{
  int matrix[6] [6];
int a, b, row, col;
printf(" Enter Numbers of Row : ");
scanf("%d ", &row);
printf("\n Enter Numbers of Column :");
scanf("%d ", &col);
printf("Enter elements in matrix :);
  for(a=0; a<=row–1; row++)
  {
        for(b=0; b<=col–l; col++)
    {
      scanf("%d", &matrix[a][b]);
  }
}
printf("\nElements in the matrix : \n");
for(a=0; a<=row–1; row++)
{
  for(b=0; b<=col–1; col++)
  {
    printf("%d ", matrix[a][b]);
  }
  printf("\n");
  }
  return 0;
}
```

Output :

```
Erter Numbers of Row: 3
Enter Numbers of column : 4
Enter elements in matrix :
81 90 75 65
66 80 89 76
78 98 76 89
Elements in matrix:
81 90 75 65
66 80 89 76
78 98 76 89
```

5.6 String

Characters के array को string कहते हैं। सभी string एक special symbol '\0' के साथ Terminate होते हैं। यह (null character) '\0' symbol, string की length निर्धारित करते हैं।

Declaration of string

String का declaration सिंगल ऐरे declaration के समान ही होता है।

Syntax :

```
Char str1_name[size];
```

Str1_name - string variable को दिया गया नाम है।

Size – यह string की length को दर्शाता है।

String Initialization

String निम्न तरीकों से initialize हो सकती है—

a. Char Str1[] = "Cprogram";
b. Char Str1[85] = "Cprogram";
c. Char Str1[85] = {'c', 'p', 'r', 'o', 'g', 'r', 'a', 'm', '\0'};
d. Char Str1[] = {'c', 'p', 'r', 'o', 'g', 'r', 'a', 'm', '\0'};

String का Memory representation

	0	1	2	3	4	5	6	7	8
Str :	c	p	r	o	g	r	a	m	\0

उदाहरण—String print करने का C program

(a)
```
#include <stdio.h>
int main()
  {
Char str1[6] = "rajiv"; //declare, initialize string
printf("%s", str1);
  return 0;
}
```

Output :

```
rajiv
```

(b)
```
#include <stdio.h>
int main()
  {
char string1[1] = {'r', 'a', 'j', 'i', 'v', '\0'};
printf("name : %s\n", string1);
  return 0;
  }
```

```
name : rajiv
```

5.7 String Function

C विभिन्न प्रकार के string handling functiosn को support करता है जो "string.h" library function में define होते हैं।

String function और उसके कार्य

String function	String function का विवरण
strcpy()	एक string को दूसरे string में copy करता है।
strcmp()	दो strings को compare करता है।
strlen()	string की लम्बाई बताता है।
stracat()	दो strings को जोड़ता है।
strchr()	string में किसी Character को search करता है।
strstr()	एक string में किसी दूसरे string को search करता है।
strdup()	string को duplicate करने के लिए
strrev()	string को reverse करने के लिए
strupr()	string को upper case में बदने के लिए
strlwr()	string को lower case में बदलने के लिए

(1) दो Strings को जोड़ने का C program

```
#include<stdio.h>
#include<string.h>
  int main()
    {
      Char s1[60];
      Char s2[70];
  printf("First name");
    scanf("%s", s1);
  printf("last name;");
  scanf("%s", s2);
    strcat (s1, s2);
  printf("complete name", s1);
      return(0);
    }
```

Output : First name : rajiv
Last name : prasad
Complete name : rajivprasad.

(ii) String को Lowercase में बदलने का C program

```
#include<stdio.h>
#include< string.h>
  int main()
    {
    Char str1[75] = "RAJIV PRASAD";
    printf("%s\n", strlwr(str1));
      return 0;
      }
```

Output : rajiv prasad

(iii) String को Uppercase में बदलने का C program

```
#include<stdio.h>
#include<string.h>
  int main()
    {
  char str1[75] = "rajiv prasad";
  printf("\n", strupr(str1));
      return 0;
  }
```

Output : RAJIV PRASAD

(iv) String की Length प्राप्त करने का C program

```
#include<stdio.h>
#include<string.h>
  int main()
    {
    int length;
  Char s2[70] = "rajiv";
      length = strlen (s2);
    printf("\length = %d\n", length);
      } return 0;
```

Oputut : length = 5

Solved Examples

C program to Calculate Length of String without Using strlen() Function

```
#include<stdio.h>
int main()
{
  char s[95];
  int i;
  printf("Enter the string:");
```

```
    scanf("%s", s);
    for(i = 0; s[i] != '\0'; i++)
    printf("length of the string: %d", i);
    return 0;
}
```

Output :

```
Enter the string: datastructures
length of the string: 14
```

C program to Concatenate Two Strings Without Using strcat()

```
#include<stdio.h>
int main()
{
   chars1[80], s2[80], i,j;
   printf("Enter the first string: ");
   scanf("%s", s1);
   printf("Enter the second string: ");
   scanf("%s", s2);
   for(i = 0;s1[i] != '\0'; ++i);
   for(j = 0; s2[j] != '\0'; ++j, ++i)
   {
      s1[i] = s2[j];
   }
   s1[i] = '\0';
   printf("After concatenation: %s", s1);
   return 0;
}
```

Output :

```
Enter first string: data
Enter second string: structure
After concatenation: data structure
```

C program to copy the second string argument to the first string argument using strcpy() function.

```
#include<stdio.h>
#indude<string.h>
int main()
{
   char s1[100];
   char s2[100];
strcpy(s1, "rajivprasad");
strcpy(s2, s1);
   printf("%s\n", s2);
   return(0);
}
```

Output :
rajivprasad

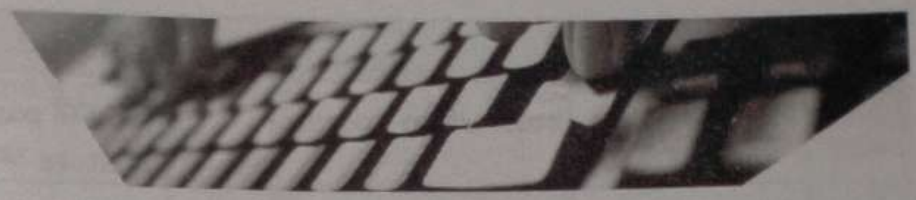
C program to reverse the given string expression using strrev() function.

```
#include<stdio.h>
int main()
{
   char s1[95];
   printf("enter the string; ");
   gets(s1);
   printf("\n reverse string is: %s",strrev(s1));
   return(0);
}
```

Output :
enter the string: rajivprasad
reverse string is: dasarpvijar

▶ प्रश्नावली (Exercise) ◀

1. Array क्या होता है? Array को कैसे access किया जा सकता है?
2. Array declaration क्या होता है? उदाहरण के साथ समझाइये।
3. Array initialization क्या होता है? उदाहरण के साथ समझाइये।
4. Array के किसी element को search करने का program लिखिए।
5. किन्हीं दो Arrays को जोड़ने का Program लिखिए।
6. Array के किसी element को function में कैसे pass करते हैं? Program के साथ वर्णन कीजिए।
7. Two dimensional array क्या होता है? इसे कैसे access किया जा सकता है?
8. Array की क्या आवश्यकता है? वर्णन कीजिए।
9. किसी array से किसी element को delete करने का C program लिखिए।
10. Two dimensional array को function में कैसे pass करते हैं? उदाहरण के साथ समझाइये।
11. किसी array के element को read करने और उसे Print करने का C program लिखिए।
12. String क्या होता है? उदाहरण के साथ समझाइये।
13. किन्हीं दो strings को compare करने का C program लिखिए।
14. किसी string का length प्राप्त करने का C program लिखिए।
15. किसी string को copy करने का C program लिखिए।
16. किन्हीं दो strings को जोड़ने का C program लिखिए।
17. किसी string को input करके और print करने का C program लिखिए।
18. किन्हीं string को reverse करके print करने का C program लिखिए।
19. किन्हीं string के सभी words count करने का C program लिखिए।
20. किन्हीं string में सभी vowels और consonants count करने का C program लिखिए।



Unit

6

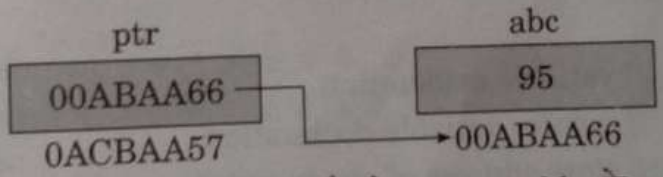
Pointers

6.1 Pointer

Pointer एक विशेष प्रकार का variable है जो किसी दूसरे variable के address को store अथवा point करता है। Pointer का प्रयोग memory को dynamically allocate करने के लिए किया जाता है। Normal variable किसी value को store करता है लेकिन pointer किसी variable के address को store करता है। Pointer variable को * से denote करते हैं जो variable के नाम के आगे लगा होता है। उदाहरण *ptr, इसमें * यह बताता है कि ptr एक pointer variable है। '&' symbol को variable के address प्राप्त करने के लिए प्रयोग करते हैं। उदाहरण : &ptr.

उदाहरण—

```
int abc = 95;
int *ptr;
ptr = &abc;
```



दिये गये उदाहरण में ptr एक pointer variable है जो abc variable के address को denote करता है।

Pointer declaration का syntax

```
data_type *pointer_variable;
```

उदाहरण—

```
int *pt    // integer pointer variable
char*pt    // character pointer variable
float *pt  // float pointer variable
```

6.2 Pointer Operators

Pointer में दो प्रकार के operators का प्रयोग होता है—

(i) * (indirection operator)

(ii) & (address operator)

operator	name	function
*	indirection operator	यह किसी particular address पर value को store करता है।
&	address operator	यह किसी variable का address बताता है।

6.3 Program

Pointer variable का address और value ज्ञात करने का Cprogram

(i)
```
#include <stdio.h>
int main()
{
   int abc = 85;
   printf("\n.address of abc : %u", & abc);
   printf("\n.value of abc : %d", abc);
   printf("\n.value of abc : %d", * (& abc));
}
```

output :

address of abc : 00ABCCBB (3202501620)
value of abc : 85
value of abc : 85

(ii)
```
#include <stdio.h>
int main()
{
   int abc = 85;    // variable declaration
   int *pt;         // pointer variable declaration
   pt = & abc;      // store address of abc in pointer variable
   printf("\addresss of abc variable : %u", & abc);
   printf("\value of *pt variable : %d\n", * pt);
   return0;
```

Output :

address of abc variable : 00CBCFCA
value of *pt variable : 85

(iii)
```
#include < stdio.h>
int main()
{
   int *ptr, a;
   a = 85;
   printf("\n address of a : %u",&a);
   printf("\n value of a: %d", a);
   ptr = &a;
```

```
    printf("\n address of pointer ptr : %u", ptr);
    printf("\n content of pointer ptr : %d", *ptr);
    return 0;
}
```

Output :

address of a : 002676a1
value of a : 85
address of pointer ptr : 002676a1
content of pointer ptr : 85

(iv) Pointer का प्रयोग करके array के element का access करने का C program

```
#include <stdio.h>
int main()
  {
    int array[6], m;
printf("enter the element;");
for (m = 0; m ≤ 5; m++)
  scanf("%d", (array + m));
printf("entered element is : \n");
for (m = 0; m < = 5; m ++)
printf("%d\n", * (array + m));
  return 0;
  }
```

Output :

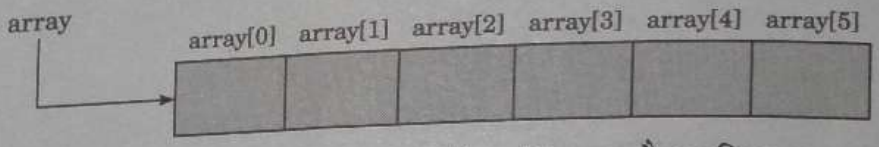
Enter the element : 0 1 2 3 4 5
Entered elements are 0 1 2 3 4 5

(v)

```
#include <stdio.h>
  int main()
    {
      int m;
  int m[6] = {0, 1, 2, 3, 4, 5};
    int *ptr = m
  for (m = 0; m ≤ 5; m++)
      {
    printf("%d", *ptr);
        ptr++;
      }
        return 0;
    }
```

Output : 0 1 2 3 4 5

6.4 Array और Pointer में Relation



दिये गये figure में array और array[0] का एक ही address है इसलिए array और &array[0] दोनों equivalent हैं। array[0] और *array भी equivalent हैं।

इस तरह

- array + 1 और &array[1] equivalent हैं।
- *(array + 1) और array[1] equivalent हैं।

- array + 2 और &array[2] equivalent हैं।
- *(array + 2) और array[2] equivalent हैं।

array + i और &array[i] equivalnet हैं।

*(array + i) और array[i] equivalnet हैं।

6.5 Pointer एक Function argument के रूप में

Pointer को किसी भी function में argument के रूप में pass किया जा सकता है। Pointer उस argument के address को Hold किये रहता है। इसे call by reference भी कहते हैं। इसमें जब भी function को call किया जाता है तो reference variable में किसी प्रकार का बदलाव इसके original variable को बदल देता है।

Pointer एक function argument का प्रयोग करके दो संख्याओं को add करने का C Program

```
# include <stdio.h>
int sum (int*, int*);
int main ()
{
int n1, n2, *a, *b, addition;
printf("Enter the two numbers\n");
scanf("%d %d" &n1, &n2);
addition = sum(&n1, &n2);
printf("result: % d\n"addition);
return 0;
}
int sum (int *a, int *b)
{
int addition;
addition = *a +*b;
```

```
return addition;
}
```

इस उदाहरण में n1 और n2 का address, sum() function में argument के रूप में pass हो रहे हैं तथा pointer variable a और b में store हो रहे हैं। *a और *b के addition value के according n1 और n2 के resultant value भी result में store हो रहे हैं।

Pointer का प्रयोग करके दो Numbers को Swap करने का C Program

```
# include <stdio.h>
void swapping (int*m, int*n);
int main ()
{
int p=85; q=95;
printf("value of p "%d\n",p);
printf("value of q : %d\n",q);
swapping (&p, &q);
printf("result after swapping;\n");
printf("p=%d\n, p");
printf("q=%d\n q);
return 0;
}
void swapping (int *m, int *n)
{
int mid;
mid = *m;
*m=*n;
*n = mid;
}
```

Output :

```
value of p : 85
value of q : 95
result after swapping :
p=95
q=85
```

इस उदाहरण में p और q के address, swapping () function में swapping (&p, &q) के द्वारा pass हो रहे हैं और pointers m और n में store हो रहे हैं। जब m और n की values change होती हैं तभी p और q की values भी बदल जाती हैं। जब swapping function के अन्दर m और n की values बदलती हैं तभी main function के अन्दर p और q की values भी बदल जाती हैं।

Pointer का प्रयोग करके two numbers को add करने का C program

```
# include <stdio.h>
int main ()
{
int n1, n2, *a, *b, sum ;
print f("Enter the two numbers\n");
scanf("%d %d"& n1, &n2);
a=&n1;
b=&n2;
sum= *a+*b;
pritntf("result =%d\n", sum);
return 0;
}
output :
Enter the two numbers
89
78
result =167
```

6.6 Dynamic memory allocation

किसी भी program के run time duration में memory allocation की प्रक्रिया को dynamic allocation कहते हैं। Dynamic memmory allocation से memory space को effectively utilize किया जा सकता है। Compile time allocation में किसी भी variable को पहले से ही memory allocate होता है इसे static memory allocation भी कहते हैं। Static memory allocation में पहले से ही fixed size space का allocation होता है जो program execution के समय स्थिर रहता है। इसको बदला नहीं जा सकता है। Dynamic memory allocation में space का wastage नहीं होता है इसमें space allotment की flexibility होती है। Memory management के लिए निम्न प्रकार के library function का प्रयोग करते हैं जो <stdlib.h> header file में predefined होते हैं।

(i) malloc ()

(ii) calloc ()

(iii) realloc ()

(iv) free ()

(i) malloc() : malloc () function का प्रयोग program के execution के समय memory में space allocation के लिए किया जाता है। malloc() program execution के समय allocated memory को initialize नहीं करता है। यह garbage value को carry करता है। अगर यह requested memory को allocate नहीं कर पाता है तो null pointer को return करता है। malloc () function केवल single argument को लेता है।

syntax : malloc (n*size of (int)); (integer data type) के लिए

जहाँ n= number of required/sized space in bytes

उदाहरण : malloc (100*size of (int)); अगर integer 2 bytes लेता है तो यह statement $100 \times 2 = 200$ bytes allocate करेगा।

(ii) Calloc() : Calloc () function का प्रयोग Program के execution के समय memory में space allocation के लिए किया जाता है। malloc () function allocated memory को zero से initialize करता है। यह दो arguments को लेता है। Calloc () function contiguous memory को allocate करता है। calloc () function, malloc () function की अपेक्षा slower होता है।

syntax : calloc (n, size of (int)); (integer data type) के लिए

where n= number of required/sized space in bytes

(iii) realloc() : realloc () function का प्रयोग allocated memory को modify करने के लिए किया जाता है। realloc () function memory से old object को deallocate करके specified size के new object को allocate करता है। realloc () function दो arguments को accept करता है। पहला argument यह denote करता है कि malloc () या calloc () function द्वारा पहले कितना memory allocate किया गया है। दूसरा parameter modified size या new size memory block को indicate करता है। यह <stdlib.h> header file में defined होता है।

Syntax : realloc (ptr, n*size of (int)); integer data type के लिए

where n= number of required/sized space in bytes

(iv) free() : free () function का प्रयोग memory को dynamically deallocate करने के लिए किया जाता है। malloc () और calloc () function के द्वारा allocated memory स्वत: deallocate नहीं होती है। इसे free () functon के द्वारा deallocate करनी पड़ती है। Program का execution समाप्त हो जाने के बाद memory को free करनी पड़ती है ताकि यह future में प्रयोग हो सके।

Syntax : free (*ptr)

6.7 malloc() और calloc() में अन्तर

	malloc()	calloc()
1.	malloc() function, memory का single block को allocate करता है।	calloc() function memory के multiple blocks को allocate करता है।
2.	malloc() function, allocated memory को initialize नहीं करता है।	calloc() function, zero से allocated memory को initialize करता है।
3.	malloc() function, single argument को accept करता है।	calloc() function, दो arguments को accept करता है।
4.	malloc() function fast होता है।	calloc() function slow होता है।

6.8 Program

(i) Dynamic memory allocation का प्रयोग करके largest number प्राप्त करने का C program.

```
# include <stdio.h>
# include <stdlib.h>
int main ()
{
  int i, m;
  int *number;
  printf("Enter numbers of elements;");
  scanf("%d",&m);
  number = (int*) calloc (m, size of (int));
  if (number ==NULL)
  {
    printf("not space variable");
    exit (0);
  }
    for (i=0, i<=m -1;i++)
    {
      scanf("%d", number +i);
    }
      for (i=0;i<=m-1; i++)
      {
        if(*(number +i)>*number)
        {
          *(number +i)=*number;
        }
      }
      Printf("forgest number is %d", *(number +i));
      return 0;
  }
```

Output :

```
Enter number of elements : 6
9 8 7 2 6 1
Largest number is 9
```

(ii) malloc () और free () function का उदाहरण

```
#include <stdio.h>
#include <stdlib.h>
int main ()
{
```

```
int i, m;
int *pt1;
printf ("Enter the numbers of element :");
scanf("%d";&m");
pt1 = (int*)malloc (m*sizeof (int));
printf ("Enter elements\n");
for (i=0; i<=m–1;i++)
scanf("%d", (pt1+i));
print f("\n elements are :\n");
for (i=0; i<=m–1; i++)
printf("%d",*(pt1+i));
free (pt1);
return 0;
```

Output :

Enter the numbers of element : 6
Enter elements :
5 3 4 2 1 6
Elements are
5 3 4 2 1 6

6.9 Pointer to Structure

Structure और उसके members को pointer का प्रयोग करके access किया जा सकता है। members को दो प्रकार से access कर सकते हैं।

(i) indirection (*)operator और (.) dot operator का प्रयोग करके।

(ii) arrow operator (→) का प्रयोग करके।

```
Struct name1
{
  member_1;
  member_2;

  member_n;
};
  int main()
  {
    struct name1 *ptr;
  }
```

Structure with pointer

Pointer और structure का प्रयोग करके employee के detail को access करने का C program

```
#include <stdio.h>
#include <string.h>
struct employee
{
   int id;
   float salary ;
   char name;
   int age;
};
   int main ()
   {
      struct employee detail = {76, 79.5, "ram", 62};
      struct employee *pt1;
      pt1= &detail;
      printf ("detail of employee 1 :\n");
      printf ("id:%d\n",pt1→id);
      printf ("salary : %f\n", pt1→ salary);
      printf ("name : %s\n",pt1→ name);
      printf ("age : %d\n",pt1→ age);
      return 0;
      detail of employee 1:
      id : 76
      salary : 79.5
      name : ram
      age : 62
```

6.10 Function Pointer

Function pointer को subroutine pointer या procedure pointer भी कहते हैं। Function pointer किसी function को point करता है। यह एक variable होता है जो किसी function के address को store करता है।

Declaration

Syntax : return_type (*pointername) (function argument list)

उदाहरण :

int *ptr1 (int, char)

यहाँ पर int एक return type है।

ptr1, function pointer का नाम है।

(int, char) function का argument list है।

उदाहरण :

(i) Function pointer का प्रयोग करके two numbers का addition का C program

```
# include <stdio.h>
int addition (int num1, int num2);
int main ()
{
  int number1 =75;
  int number2 =98;
  int sum;
  int *(*pt1) (int, int);
  pt1 =& addition;
  sum =(*pt1) (number1, number2);
  printf("addition of two numbers :%d", sum);
  return (0);
}
  int addition (int num1, int num2)
  {
    return (num1 +num2);
  }
```

Output

addition of two numbers :173

(ii) Pointer का प्रयोग करके तीन नम्बरों में से greatest number प्राप्त करने का C program

```
# include < stdio.h >
int main()
{
  int num, m, n, o ;
  int *pt1, *pt2, *pt3 ;
  printf(" Enter how many numbers: ");
  scanf("%d ",&num) ;
  printf(" \nEnter first number : ") ;
  scanf("%d ",&m);
  printf("\n Enter second number: ") ;
  scanf("%d ",&n) ;
  printf("i:\n Enter third number : ") ;
  scanf("%d ",&o),
  pt1 = &m ;
  pt2 = &n ;
  pt3 = &o;
  if((*pt1> = *pt2)&&(*pt1> = *pt3))
  printf("\n greatest Number is : %d ",*pt1) ;
```

```
    else
    {
      if(( *pt2> = *pt1) && ( *pt2> = *pt3))
      printf("\n greatest Number is : %d "*pt2;)
      else
      printf("\n greatest Number is : %d ",*pt3);
    }
      return ( 0 );
}
```

```
Enter how many numbers: 3
Enter first number :78
Enter second number :98
Enter third number :88
greatest Number is : 98
```

(iii) Pointer का प्रयोग करके तीन नम्बरों का sum का C program

```
#include <stdio.h>
int main()
{
  int n1, n2, n3, *m, *n, *o, sum;
  printf("Enter three integers :\n");
  scanf("%d%d%d", &n1, &n2,&n3);
  m = &n1;
  n=&n2;
  o=&n3;
  sum = *m + *n+*o;
  printf("Sum of the numbers : %d\n", sum);
  return 0;
}
```

Output;

```
Enter three integers
67 87 88
Sum of the numbers: 242
```

(iii) C program to subtraction of two numbers using pointers

```
#include <stdio.h>
int main()
{
  int n1, n2, add;
  int *ptr1, *ptr2;
  ptr1 = &n1;
  ptr2 = &n2;
```

```
    printf("Enter two numbers:");
    scanf("%d%d", ptr1, ptr2);
    add = *ptr1 *ptr2;
    printf("addition= %d", add);
    return 0;
}
```

(iv) C program to perform all arithmetic operations using pointers

```
#include <stdio.h>
int main()
{
  float n1, n2;
  float *ptr1, *ptr2;
  float add, sub, mult, div;
  ptr1 = &n1;
  ptr2 = &n2;
  printf("Enter any two numbers:");
  scanf("%f%f", ptr1, ptr2);
  add =(*ptr1) + (*ptr2);
  sub = (*ptr1) - (*ptr2);
  mult= (*ptr1) * (*ptr2);
  div =(*ptr1)/(*ptr2);
  printf("addition = %.1f\n", add);
  printf("subtraction = %.1f\n", sub);
  printf ("multiply = %.1f\n",mult);
  printf("division=%.1f\n",div);
  return 0;
}
```

Output :

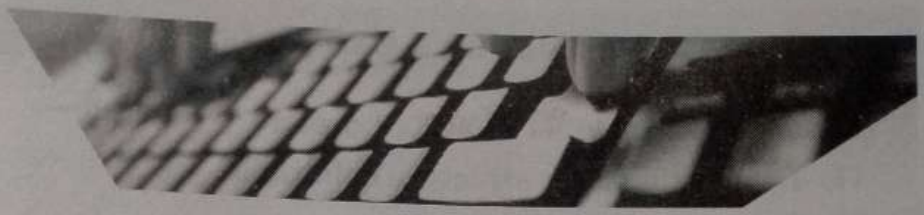
```
Enter any two numbers : 80  40
addition : 120.0
Subtraction : 40.0
mutiply : 3200.0
division : 2.0
```

▶ प्रश्नावली (Exercise) ◀

1. Pointer क्या होता है? इसका प्रयोग कहाँ किया जाता है? **(UPBTE 2017)**
2. Pointer variable को create, initialize और access कैसे किया जाता है?
3. Null pointer क्या होता है? उदाहरण के साथ समझाइए।
4. Dynamic memory allocation का प्रयोग करके array elements के sum प्राप्त करने का C Program लिखिए।
5. Pointer का प्रयोग करके किन्हीं दो नम्बरों को add करने का C Program लिखिए।
6. Pointer का प्रयोग करके किन्हीं दो नम्बरों में से maximum नम्बर प्राप्त करने का C Program लिखिए।
7. Dynamic memory allocation क्या होता है? उदाहरण के साथ समझाइए। **(UPBTE 2016)**
8. N numbers को array में store करने और उन्हें pointer की सहयता से print करने का C Program लिखिए।
9. Pointer का प्रयोग करके array element को access करने का C Program लिखिए।
10. Pointer का प्रयोग करके किसी string का length प्राप्त करने का C program लिखिए।
11. Pointer का function argument के रूप में कैसे pass करते हैं? C program की सहायता से समझाइए।
12. किन्हीं दो नम्बरों को swap करने का (Pointer की सहायता से) C program लिखिए।
13. निम्नलिखित function को उदाहरण के साथ समझाइए।
    (i) malloc()
    (ii) calloc()
    (iii) free()
    (iv) realloc()
14. एक array को दूसरे array में copy करने का C program pointer का प्रयोग करके लिखिए।
15. Pointer का प्रयोग करके array में किसी element को search करने का C Program लिखिए।
16. Pointer का प्रयोग करके किसी दिए गए नम्बर का factorial प्राप्त करने का C program लिखिए।
17. Pointer का प्रयोग करके किसी array को ascending order में sort करने का C program लिखिए।
18. Pointer का प्रयोग करके किसी string को reverse करने का C program लिखिए।
19. Dynamic memory allocation का प्रयोग करके largest number प्राप्त करने का C program लिखिए।

Unit

7



Structure and Union

7.1 Structure

Structure एक user defined data type है जो विभिन्न प्रकार के data types का collection होता है। जिस प्रकार array केवल एक ही तरह के data type को store करता है उसी प्रकार structure अलग-अलग तरह के data types को एक single datatype में store करता है। structure को struct नाम के key words से represent करते हैं। Structure में अलग-अलग data types के variable को member कहते हैं। Members के लिए अलग-अलग memory allocate होता है। structure का प्रयोग मुख्यत: विभिन्न प्रकार के records को store करने के लिए करते हैं।

उदाहरण—

```
struct student
{
  char std_name [85];
  int std_roll;
  float marks;
  int std_id;
};
```

यहाँ पर structure का नाम student है जिसमें char, int और float type के तीन अलग-अलग data types का प्रयोग हुआ है। जिनका प्रयोग student के record को store करने के लिए हुआ है।

Syntax :

```
struct strcuture_name
{
  data_type member_1;
  data_type member_2;
  data_type member_3;
  .....
  .....
  data_type member_n;
};
```

7.2 Structure Variable Declaration

Structure variable को दो प्रकार से declare कर सकते हैं—

(1) Declaration after structure definition

(2) Declaration in main function.

(i) Syntax of structure variable declaration after definition

```
struct structurename
{
  data_type member_1;
  data_type member_2;
  data_type member_3;

  data_type_member n;
}
  variable(s);
```

उदाहरण—

```
strcut student
{
  char stud_name[85];
  int stud_id;
  int stud_roll;
  float stud_marks;
}
  record;
```

इस उदाहरण में student का नाम का structure है जिसमें stud_name, stud_id, stud_roll और stud_marks, members हैं और record नाम का एक variable create किया गया है। इसी variable से हम student के record को access करते हैं।

(ii) Syntax of structure variable declaration in main() function

```
struct structurename
{
  datatype member_1;
  datatype member_2;
  datatype member_3;
  .....
  .....
  datatype member_n;
};
  int main
  {
```

```
    struct structurename varibale(s);
  }
```

उदाहरण—

```
strcut student
{
  char stud_name[85];
  int stud_id;
  int stud_roll;
  float stud_marks;
};
  int main
  {
    struct student record;
  }
```

इस उदाहरण में structure का नाम student है जिसमें stud_name, stud_id, stud_roll और stud_marks, members हैं। record नाम का variable main function में delcare किया गया है।

7.3 Structure Initialization

Structure में variable को initialize करने पर ही memory allocate होता है। Un-initailize variable को memory allocate नहीं होता है। structure के variable को विभिन्न प्रकार से memory allocate किया जा सकता है।

(i) Declare the variable and initialize

(a)
```
sturct student
{
  char student_name[85];
  int stdu_id;
  int stud_roll;
  float stud_marks;
}
  record = {"rajiv", 198, 75, 75.9};
```

इस उदाहरण में declaration के तुरन्त बाद structure के variable record को initialize किया गया है।

record = {"rajiv", 198, 75, 75.9}

(b) Initialization for multiple variables

```
struct student
{
  char student_name[85];
  int stud_id;
  int stud_roll;
  float stud_marks;
```

```
}
  record 1 = {"rajiv", 198, 75, 75.9};
  record 2 = {"rajiv", 175, 85, 69.8};
```

इस उदाहरण में delcaration के तुरन्त बाद structure के variables, record 1, record 2 को initialize किया गया है।

```
record 1 = {"rajiv", 198, 75, 75.9}
record 2 = {"rajiv", 175, 85, 69.8}
```

(c) Structure के किसी single member का initialization

```
struct student
{
  char stud_name[85];
  int stud_id;
  int stud_roll;
  float stud_marks;
}
  record = {"rajiv"};
```

इस उदाहरण में चार members हैं जिनमें से केवल एक मेम्बर initialize हुआ है। इसके बाकी सभी members default value से स्वत: ही initialize हो जायेंगे।

integer के लिए default value = 0

float के लिए default value = 0.0

char के लिए default value = NULL

(ii) main function के अन्दर declaration और initialization

```
struct student
{
  char stud_name[85];
  int stud_id;
  int stud_roll;
  float stud_marks;
    };
int main ()
    }
    struct student record = {"rajiv" 198, 85, 75.9};
}
```

इस उदाहरण में record नाम का variable है जिसका initialization main function के अन्दर हुआ है।

7.4 Structure Member को Access करना

(i) Structure member को dot [.] operator से access किया जाता है। इसको structure member operator भी कहते हैं। इस [.] operator को variable name और member name के बीच में प्रयोग किया जाता है।

Syntax : variable . member

(ii) Structure member को Pointer operator अथवा arrow operator (—>) से भी access कर सकते हैं। जब हम pointer variable का प्रयोग करते हैं वहाँ हम members को access करने के लिए arrow operator (—>) का प्रयोग करते हैं।

उदाहरण—Stucture creation, declaration और initialization का C program

(i)
```
#include <stdio.h>
struct toy
{
  char toy_name[75];          //structure creation
  int toy_price;
  int toy_wheel;
}
  t1 = {"kidtoy", 95, 4};     //structure declaration and initialization
  int main ()
  {
    printf ("\n toy name : %s", t1.toy-name);
    printf ("\n toy price : %d", t1.toy-price);
    printf ("\n toy wheel : %d", t1.toy-wheel);
    return 0;
```

Output :

```
toy name : kidtoy
toy price : 95
toy wheel : 4
```

(ii)
```
#include <stdio.h>
struct student
{                              //structure creation
  char name[68];
  int roll;
  float marks;
};
  int main ()
  {                            //structure declaration and initialization
    struct student std = {"rajiv", 75, 75.8};
    printf ("\n name : %s", std. name);
    printf ("\n roll number : %d", std. roll);
```

```
        printf ("\n marks : %f", std. marks);
        return 0;
    }
```

Output :

```
name : rajiv
roll number : 75
mark : 75.8
```

(iii) 15 students के record को store करना और से display करने का C program.

```
#include <stdio.h>
struct student
{
  char name [85];
  int roll_number;
  int id;
  float marks;
}
  r[15];
  int main ()
  {
    int m;
    printf ("Enter the record of students : \n");
    for (m = 0; m ≤ 14; m++)
    {
      r[m] · roll_number = m + 1;
      printf ("\n roll number %d, \n", r [m] · roll_number);
      printf ("Enter the name:");
      scanf ("% s", r[m] · name);
      printf ("Enter the id");
      scanf ("% d", & r[m] · id);
      printf ("Enter the marks");
      scanf ("%f ", & r[m] · marks");
  }
    printf ("Display information \n");
    for (m = 0; i ≤ 14; m++)
    {
      printf ("\n Roll number : %d \n", m + 1);
      printf ("Name : %s" r[m] · name);
      printf ("id : %d", r[m] · id);
      printf ("marks; %f ", r[m] · marks);
  }
```

```
    return0;
}
```

Output :

```
Enter the record of students
roll number 1
Enter the name : rajiv
Enter the id : 75
Enter the marks : 75.6
roll number 2
Enter the name : ravi
Enter the id : 85
Enter the marks : 67.8
. . . . . .
```

7.5 Nested Structure

जब एक structure के अन्दर कोई दूसरा structure का प्रयोग करते हैं तो वहाँ पर Nested structure होता है। Nested structure को structure within structure भी कहते हैं जिसमें एक structure का declaration किसी दूसरे structure के अन्दर होता है।

```
struct teacher
{
  char name [65];
  int age;
  int salary;
  struct
  {
    char branch_1 [67];
    char branch_2 [75];
    char branch_3 [68];
  }
    branch;
{
  record;
```

इस उदाहरण में teacher एक strcuture है जिसमें चार members (name, age, salary और branch) हैं। इस structure में एक दूसरा structure, branch नाम का है, जिसमें तीन members (branch_1, branch_2, branch_3) हैं।

इस structure में branch, स्ट्रक्चर के अन्दर innter structure है जबकि outer structure के लिए एक member की तरह कार्य करता है।

उदाहरण—Nested structure का C program

```
#include <stdio.h>
```

```
int main ()
{
  struct employee
  {
    char name [65];
    int id;
    struct
    {
      int da;
      int basic;
      int hra;
    }
      salary;
  };
  struct employee e1, e2;
  printf ("\n Enter the name of employee 1");
  scanf (" % s", e1.name);
  printf ("\n Enter the id of employee 1");
  scanf (" % d", & e1.id);
  printf ("\n Enter the DA of employee 1");
  scanf (" % d", &e1.salary.da);
  printf ("\n Enter the basic of employee 1");
  scanf (" % d", &e1.salary.basic);
  printf ("\n Enter the HRA of employee");
  scanf (" % d", &e1.salary.hra);
  printf ("\n Name of employee_1 : s%", e1.name);
  printf ("\n id of employee_1 : %d", e1.id);
  printf ("\n DA of employee_1 : %d", e1 · salary · da);
  printf ("\n Basic of employee_1 : %d", e1 · salary · basic);
  printf ("\n HRA of employee_1 : %d", e1 · salary · hra);
  getch ();
}
```

इसी तरह हम employee 2 के record को भी enter और display कर सकते हैं।

7.6 Structure with Function

किसी भी structure को हम एक function में argument की तरह pass कर सकते हैं। Structure का definition केवल उस function के अन्दर होगा। अगर हम structure variable को global variable की तरह declare करेंगे तो यह program के सभी function के द्वारा access हो पायेगी।

(i) Passing structure to function by value

(ii) Passing structure to function by address

उदाहरण—**(i) Structure member को argument के रूप में function में pass करने का C program**

```
#include <stdio.h>
struct employee
{
  char name [65];
  int id;
  float salary;
};
  void; emp_func (char name [ ], int id, float salary);
  int main ()
  {
    struct employee emp = {"sumit", 75, 79.8};
    emp_func (emp.name, emp. id, emp. salary);
    return 0;
  }
    void emp_func (char name [ ], int id, float salary)
    {
      printf ("Name : % s\n", name);
      printf ("id : % d\n , id);
      printf ("salary : % f\n", salary);
      printf ("\n");
```

Output:

```
Name : sumit
id : 75
salary : 79.8
```

(ii) Stucture variable को argument के रूप में function में pass करने का C program

```
#include <stdio.h>
struct employee
{
  char name [65];
  int id;
  float salary;
};
  void emp_func (struct employee emp);
  int main ()
  {
    struct employee emp = {"sumit", 75, 79.8};
    emp_func (emp);
    return 0;
```

```
}
  void emp_func (struct employee emp)
  printf ("Name : % s\n", emp. name);
  printf ("id : % d\n", emp. id);
  printf ("salary : % d\n", emp. salary)
```

Output:

```
Name : sumit
id : 75
salary : 79.8
```

(iii) structure pointer को argument के रूप में function में pass करने का C program

```
#include <stdio.h>
struct student
{
  char name [65];
  int id;
  char address [85];
};
  void stud (sruct student*);
  int main ()
  {
    struct student st1 = {"rajiv", 85, "mumbai"};
    stud (&st1);
    return 0;
  }
    void stud (struct student *ptr1)
    {
      printf ("Name : % s\n", ptr1 -> name);
      printf ("id :% d\n", ptr1 -> id);
      printf ("address : % d\n", ptr1 -> address)
    }
```

Output :

```
Name : rajiv
id : 85
address : mumbai
```

(iv) array of structure को argument के रूप में function में pass करने का C program.

```
#include <stdio.h>
struct school
{
  char name [65];
  char address[75];
```

```
    int no_of_student;
}
  void display (struct school sch [ ]);
  int main ()
  {
    struct school board [4] = {"Dps", "delhi", 700}
                              {"Dpst", "mumbai", 800}
                              {"Jnu", "Agra", 900}
                              {"spt", "Lucknow", 750}
  };
    display (board);
    return 0;
  }
    void display (struct school sch[ ])
    {
    int m;
    for (m = 0; m ≤ 3; m++)
    {
      printf ("Name : % \s", sch [m] · name);
      printf ("address : %s\n , sch [m] · address);
      printf ("number of student : % d\n", sch [m] · no_of_student);
    }
  }
```

Output :

```
name : Dps
address : delhi
number of students : 700
name : Dpst
address : mumbai
number of students : 800
name : JNU
address : Agra
number of students : 900
name : Spt
address : Lucknow
numbher of students : 750
```

7.7 Pointer to a Structure

Pointer एक variable होता है जो किसी दूसरे variable के address को point करता है। इसी तरह structure का भी pointer हो सकता है जो structure के variable के address को point करता है।

Pointer to a structure में हम arrow —> द्वारा structure के member को access करते हैं। इसे हम indirection operator से भी access करते हैं।

उदाहरण—Array operator —> द्वारा structure member को access करने का program

```
#include <stdio.h>
struct campus
{
  char name [65];
  int area;
  char address [62];
};
  int main ()
  {
    struct campus camp = {"KMGGP", 81, "Badalpur"};
    struct campus * ptr;
    ptr = & camp;
  print ("name : % s\n", ptr —> name);
  print ("area : % d\n", ptr —> area);
  print ("address : % d\n", ptr —> address);
  return 0;
}
```

Output :

```
name : KMGGP
area = s1
address = Badalpur
```

7.8 Union

Union एक विशेष प्रकार का data type होता है जो विभिन्न प्रकार के data type को एक ही memory space locate करता हैं। Union के members एक ही memory location को share करते हैं। इस data type को 'union' keyword से indicate करते हैं। एक union बहुत सारे member functions के साथ define किया जा सकता है। structure में सभी member का अपना अलग-अलग memory location होता है, परन्तु union के सभी members एक ही मेमोरी को share करते हैं।

Syntax : union union_name

```
{
  datatype member 1;
  datatype member 2;
  datatype member 3;


  datatype member n;
};
```

▌Union variable declaration

Union variable को दो प्रकार से declare कर सकते हैं।

(i) Declaration outside the union definition

```
Union unionname
{
  datatype member_1;
  datatype member_2;
  datatype member_3;


  datatype member_n;
}
  variable (s);
```

(ii) Declaration in main() function

```
Union unionname
{
  datatype member_1;
  datatype member_2;
  datatype member_3;


  datatype member_n;
};
  int main ( )
  {
    union unionname variable;
  }
```

Structure और Union में अन्तर

Union	Structure
(i) Union के सभी members एक ही memory location को share करते हैं।	(i) Structure के प्रत्येक member के लिए अलग-अलग memory allocate होता है।
(ii) Union को define करने के लिए union keyword का प्रयोग करते हैं।	(ii) Structure को define करने के लिए struct keyword का प्रयोग किया जाता है।
(iii) Union केवल एक single value को एक समय में store करता है। एक समय में केवल एक ही member को access किया जा सकता है।	(iii) Structure में multiple value store किया जा सकता है। कोई भी member कभी भी access किया जा सकता है।

(iv) Union का साइज इसके largest size वाले member के बराबर होता है।	(iv) Structure का साइज इसके सभी members के साइज के sum के बराबर होता है।
(v) Vnion में केवल first member एक समय पर initialize होता है।	(v) Structure के सभी members एक साथ initialize हो सकते हैं।

Programs

(i) Union के member को access करने का C program

```
#include <stdio.h>
#include <conio.h>
union customer
{
  char name [65];
  int age;
    char address [72];
  };
    int main ()
    {
      union customer cust_1;
      printf ("Enter the customer name;");
      scanf (" %s", & cust_1.name);
      printf ("Enter the age:");
      scanf (" %d", & cust_1.age);
      printf ("Enter the address");
      scanf (" %s", & cust.address);
      printf ("customer name : %s", cust_1.name);
      printf ("customer age : %d", cust_1.age);
      print ("customer address : %s", cust_1.address);
      getch ();
      return 0;
    }
```

Output :

```
Enter the customer name : sumit
Enter the age : 75
Enter the address : delhi
customer name : sumit
customer age : 75
customer address : delhi
```

(ii) Union members access करने का C Program.

```
#include <stdio.h>
void main()
{
  union display
  {
    int x1;
    float x2;
    char x3;
  };
    union display m;
    printf("Enter the value of x1: ");
    scanf("%d", &m.x1);
  printf("Value of x1 is=%d", m.x1);
  printf("\nEnter the value of x2: ");
  scanf("%f", &m.x2);
  printf("Value of x2 is = %f\n", m.x2);
  printf("Enter the character x3: ");
  scanf("%c", &m.x3);
  prmtf("\Character is = %c", m.x3);
  }
  Enter the value of x1: 90
  Value of x1 is = 90
  Enter the value of x2: 80
  Value of x2 is= 80.0
  Enter the character x3: r
  Character is : r
```

(iii) Union का प्रयोग करके students के record को store करना और access करने का program

```
#include <stdio.h>
union student
{
  char name[85];
  float marks;
  int age;
  int roll;
}
  std1;
  int main()
  {
    printf("Enter the name:\n");
```

```
    scanf("%s", &std1.name);
    printf("Enter the marks: \n");
    scanf("%f', &std1.marks);
    printf("Enter the age:\n");
    scanf("%d", &std1.age);
    printf("Enter the roll number:\n");
    scanf("%d", &std1.roll);
    printf("Name :%s\n", std1.name);
    printf("marks: %.2f\n", std1.marks);
    printf("age :%d\n",std1.age);
    printf("roll number: %d", std1.roll);
    return 0;
  }
```

Output :

```
Enter the name :  rajiv
Enter the marks : 74.50
Enter the age : 72
Enter the roll number : 64
Name : rajiv
Marks : 74.50
age : 72
roll number : 64
```

(iv) Structure का प्रयोग करके तीन नम्बरों के addition का Program

```
#include<stdio.h>
#include<conio.h>
struct addition
{
  int m,n,o;
}
  void main()
  {
    int result1;
    struct addition a;
    printf("Enter three numbers:");
    scanf("%d%d%d",&a.m,&a.n,&a.o);
    result1=a.m+a.n+a.o;
    printf("\nSum=%d", result1);
    getch();
  }
```

Output :

```
Enter the three numbers: 85 75 95
Sum = 255
```

(v) Structure का प्रयोग करके students के records को access करने का program

```
#include <stdio.h>
struct student
{
  char name[98];
  int roll;
  int marks;
  int m[6];
};
  int main()
  {
    int i;
    printf("Enter the detail of students:\n");
    for(i=0; i<=5; i++)
    {
      m[i].roll = i+1;
      printf("\n roll number%d,\n",m[i].roll);
      printf("Enter name:");
      scanf("%s",m[i].name);
      printf("Enter marks: ");
      scanf("%d",&m[i]. marks);
      printf("\n");
    }
    printf("show lnformation:\n\n");
    for(i=0; i<=5; i++)
    printf("\nRoll number: %d\n",i+1);
    printf("Name: ");
    puts(m[i].name);
    printf(" Marks: %d",m[i]. marks);
  }
  return 0;
}
```

Output :

```
Enter the detail of students:
roll number1,
Enter name : Ramanuj
Enter marks : 88
```

```
        roll number2,
        Enter name : Shobhit
        Enter marks : 85
show Information
        roll number1,
        Enter name : Ramanuj
        Enter marks : 88
        roll number2,
        Enter name : Shobhit
        Enter marks : 85
```

(v) C program to read and print average marks of students using structure of array

```
#include <stdio.h>
struct student
{
  char name [70];
  int marks[7];
  int sum;
  float avg ;
}
  int main()
  {
    struct student st1,
    int i;
    printf("Enter name: ");
    scanf("%s",&st1.name);
    printf("Enter marks:\n");
    st1.sum=0;
    for(i=0; i<=6; i++)
  {
    printf("Marks in subject %d : ",i+1);
    scanf("%d",&st1.marks[i]);
    st1.sum= st1.sum+st1.marks[i];
  }
    st1.avg=(float)(st1.sum/7);
    printf("\nName: %s \ntotal_marks: %d \naverage: %f",st1.name,st1.sum,st1.avg);
    return 0;
}
```

Output :

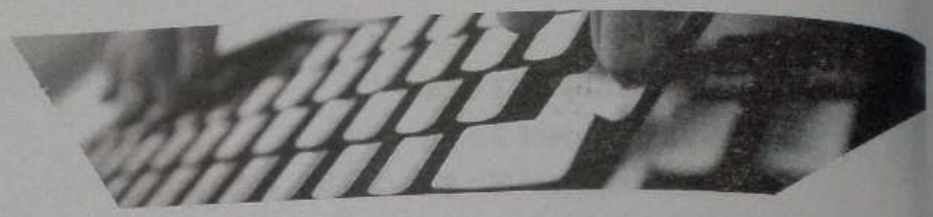
```
    Enter name : Ramanuj
```

```
Enter marks :
Marks in subject 1 : 74
Marks in subject 2 : 92
Marks in subject 3 : 88
Marks in subject 4 : 88
Marks in subject 5 : 76
Marks in subject 6 : 99
Marks in subject 7 : 78
Name : Ramanuj
total_marks : 595
average : 85.0
```

▶ प्रश्नावली (Exercise) ◀

1. Strcuture को कैसे initialize करते हैं? उदाहरण के साथ समझाइए।
2. Strcuture क्या होता है? उदाहरण के साथ वर्णन कीजिए। **(UPBTE 2017)**
3. Strcuture को कैसे declare करते हैं?
4. Strcuture के member को कैसे access करते हैं? उदाहरण के साथ समझाइए।
5. Nested structure क्या होते हैं?
6. Strcuture को एक function के रूप में कैसे pass करते हैं?
7. Union क्या होता है? समझाइए।
8. Strcuture और Union में क्या अन्तर होता है?
9. Union के members को कैसे access करते हैं?
10. C program की सहायता से union member को access करते हुए इसे विस्तार से समझाइए।
11. Structure और array में क्या अन्तर है?
12. Strcuture का प्रयोग करके किसी student के roll number, age, marks, name और stream को store और print करने का C program लिखिए।
13. Structure का प्रयोग करके किन्हीं दो complex number के addition का program लिखिए।
14. Union को कैसे define करते हैं? उदाहरण के साथ समझाइए।
15. Arrays of structure क्या होता है? उदाहरण के साथ समझाइए।
16. किसी structure को create, declare और initialize करने का C program लिखिए।
17. किसी union को delcare, initialize और create करने का C program लिखिए।

❑



Unit

8 File Handling

8.1 File

File, किसी disk पर bytes के sequence को प्रदर्शित (represent) करता है, जहाँ पर related data का group store होता है। File किसी disk पर permanent storage की तरह store होता है। File को handle करने के लिए file handling function का प्रयोग करते हैं।

8.2 Files के प्रकार

(i) Text files

(ii) Binary files

(i) Text files : Text files को किसी भी Text editor का प्रयोग करके create किया जा सकता है। इसे आसानी से पढ़ा जा सकता है। इसको store और save करके modify भी किया जा सकता है। ये .txt (extension) files होते हैं जो alphanumeric, digit, character, space इत्यादि से create किये जाते हैं।

(ii) Binary files : Binary files में केवल 0 और 1 होते हैं। ये .bin files होते हैं जो binary form में computer में save होते हैं। ये आसानी से read नहीं किये जा सकते हैं।

8.3 Files की आवश्यकता

(i) Data को permanently सुरक्षित रखने के लिए files का प्रयोग किया जाता है।
(ii) Files का प्रयोग करके large amount data को आसानी से access किया जा सकता है।
(iii) Files को आसानी से एक computer से दूसरे computer पर transfer किये जा सकते हैं।
(iv) Files का प्रयोग backup के लिए भी किया जाता है।

8.4 File Handling

Files को computer या Hard Disk में सुरक्षित रखने तथा access करने की प्रक्रिया को File handling कहते हैं। File handling में विभिन्न प्रकार के functions का प्रयोग किया जाता है जिन्हें file operation function भी कहते हैं।

8.5 File Operation

File पर मुख्यत: चार प्रकार के operations perform किये जाते हैं—

(i) Creating या opening a file
(ii) Reading a file
(iii) Closing a file
(iv) Writing a file

इन Operations को perform करने के लिए विभिन्न प्रकार के functions प्रयोग किये जाते हैं जो inbuilt functions होते हैं।

8.6 Files handling functions और उनका प्रयोग

Functions	प्रयोग
fopen()	fopen() function का प्रयोग नया file create करने अथवा existing File को open करने के लिए किया जाता है।
fclose()	fclose() function का प्रयोग किसी File को बन्द close करने के लिए किया जाता है।
putc ()	putc() का प्रयोग file पर character को write करने के लिए किया जाता है।
getc ()	getc() function का प्रयोग files से character को read करने के लिए किया जाता है।
getw ()	getw() function का प्रयोग integer को read करने के लिए किया जाता है।
putw ()	putw() function का प्रयोग integer को file पर write पर करने के लिए किया जाता है।
fprintf()	fprintf() function का प्रयोग file में formatted data को write करने के लिए किया जाता है।
fscanf()	fscanf() function का प्रयोग file से formatted data को read करने के लिए किया जाता है।
fputchar()	fputchar( ) function का प्रयोग Keyboard के character input को output screen पर write करने के लिए किया जाता है।
fseek ()	fseek() function किसी दिए गए location पर File pointer को move करने के लिए किया जाता है।
fget char()	fget char() function का प्रयोग Keyboard character को read करने के लिए किया जाता है।
fputs()	fputs( ) function का प्रयोग File पर string को write करने के लिए किया जाता है।
fgets ()	fgets( ) function का प्रयोग Files से string read करने के लिए किया जाता है।
feof()	feof() function का प्रयोग File के end को ज्ञात करने के लिए किया जाता है।
ftell()	ftell( ) function file के current position को बताता है।
rewind ()	rewind( ) function, file pointer position को File के beginning position पर move करता है।
getchar ()	getchar() function का प्रयोग Keyboard से character को read करने के लिए किया जाता है।

8.7 Opening a File

File को open करने या new file create करने के लिए fopen() function का प्रयोग करते हैं। किसी भी file को हम different mode में open कर सकते हैं।

Syntax : FILE pointer_name = fopen ("file_name", "mode");

जहाँ pointer name = user द्वारा दिया गया File pointer का नाम

fopen = file open या new file creation के लिए function

mode = mode यह decide करता है कि file के opening का reason या purspose क्या है?

file_name = जिस File को open करना है उस File का नाम

उदाहरण—

```
#include <stdio.h>
int main()
{
  FILE *ftr
  ftr=fopen ("filename", "mode");
  return 0;
}
```

8.8 File opening mode और उसका उद्देश्य

Mode	Description
r	reading mode में text file open करने के लिए
a	append mode में text file open करने के लिए
w	writing mode में text file create करने के लिए
r+	reading और writing दोनों mode में text file open करने के लिए
w+	reading और writing modes में text file open करने के लिए
rb	Binary file को reading modes में open करने के लिए
wb	Binary file को writing mode में open या create करने के लिए
ab	Binary file को append mode में open करने के लिए
rb+	Binary file को reading और writing दोनों modes में open करने के लिए
wb+	Binary file को reading और writing दोनों modes में open करने के लिए

8.9 Closing a file

File को close करने के लिए fclose() function का प्रयोग करते हैं। कोई भी file, Binary या Text file को close करने के लिए fclose() का प्रयोग किया जाता है।

Syntax : fclose (*file_pointer)

जैसे fclose (*ftr)

उदाहरण—

```
#include <stdio.h>
int main ()
{
  FILE *ftr;
  ftr=fopen ("File1.txt","r");
  fprintf(ftr,"%s", "c language");
  fclose(ftr);
  return (0);
```

प्रोग्राम run करने के बाद File1.txt नाम का file create करेगा जो कि reading mode में होगा और fclose() function से File close होगा।

उदाहरण :

किसी File में character data को write करनें और उसको close करने का C program

```
#include <stdio.h>
#include <conio.h>
void main ()
{
  FILE*ftr;
  char che[85];
  ftr = fopen("File1.txt", "w");
  printf("Enter the text");
  scanf ("%s",&che);
  fprintf(Ftr,"%s, che);
  fclose(Ftr);
  getch();
};
```

8.10 getw() और putw() function

(i) getw() function का प्रयोग file से integer को read करने के लिए किया जाता है। यह एक integer oriented library function है।

Syntax : getw(ftr) जहाँ ftr एक file pointer है।

उदाहरण—

```
#include <stdio.h>
main()
{
  FILE*ftr;
  int a;
  ftr=fopen ("file.txt", "r");
  a=getw(ftr)
```

```
    Printf("%d",a);
    fclose (ftr);
}
```

(ii) Putw() : putw() function का प्रयोग file में integer को write करने के लिए किया जाता है। यह एक library function है।

Syntax : putw(integer, ftr) जहाँ ftr एक file pointer है।

उदाहरण—

```
#include <stdio.h>
main ()
{
    FILE*ftr;
    ftr=fopen("File.txt","w");
    putw (85,ftr);
    fclose (ftr);
}
```

8.11 File से character को Read करना

(i) getc() : File से character को read करने के लिए getc() function का प्रयोग किया जाता है। getc function एक समय में केवल एक character को read करता है। यह एक library function है।

Syntax : getc(ftr); जहाँ ftr एक file pointer है।

(ii) File में character को write करना

putc() : putc() function का प्रयोग किसी file में character को write करने के लिए किया जाता है।

Syntax : putc(char,ftr) जहाँ ftr एक file pointer है।

उदाहरण—

(i) fprintf() function का प्रयोग करके किसी File में Integer data write करने का Program

```
#include <stdio.h>
#include <stdlib.h>
int mian ()
{
    int n1;
    FILE *ftr;
    ftr=fopen ("File1.txt","w");
    Printf("Enter the number");
    scanf ("%d",&n1);
    fprintf(ftr,"%d",n1);
    fclose (ftr);
    return 0;
}
```

Program run होने के बाद user के द्वारा input number, File1.txt File में store करेगा।

(ii) fscanf() function का प्रयोग करके integer data को read करने का C program।

```
#include<stdio.h>
#include <stdib.h>
int main ()
{
  int n1;
  FILE*ftr;
  ftr =fopen ("file1.txt","r");
  fscanf(ftr,"%d",&n1);
  Printf("value : %d", n1);
  fclose (ftr);
  return 0;
}
```

Program, run होने के बाद File1.txt में available integer n1 को read करने के बाद file को close करेगा।

(iii) getc() और putc () function का प्रयोग करके किसी File पर character read और write का C program.

```
#include <stdio.h>
#include <stdlib.h>
int main ()
{
  FILE *ftr;
  char ch1;
  ftr=fopen ("File1.txt","w");
  Printf ("Enter the character");
  while ((ch1=getchar())!=EOF)
  {
    putc (ch1, ftr);
  }
    fclose (ftr);
    ftr =fopen ("File1.txt","r");
    while ((ch1=getc (ftr))!=EOF)
    Printf(("%c",ch1);
    fclose(ftr);
    return 0;
}
```

8.12 fgets() और fputs() function

(i) fgets() : fgets() function का प्रयोग file से string को read करने के लिए करते हैं। यह एक standard library function है।

(ii) fputs() : fputs() function का प्रयोग किसी file में string को write करने के लिए किया जाता है। यह एक standard library function है।

fgets() और fputs() function का उदाहरण।

```
#include <stdio.h>
#include <conio.h>
int main()
{
   FILE *ftr;
   char ch2[82];
   int m=0;
   ftr = fopen ("file1.txt","w");
   Printf("Enter the string:");
   scanf ("%s", ch2);
   fputs(ch2, ftr);
   fclose (ftr);
   ftr =fopen ("file1.txt", "r");
   if (Fgets (ch2, 82, ftr)!=NULL)
   while (ch2[m]!='\0')
   {
      putchar (ch2[m]);
      m++;
   }
      fclose (ftr);
      getch();
}
```

Output :

```
Enter the string : C language
C language
```

8.13 fput char() function

fputchar () एक file handling function है जो standard output या screen पर character को write करने के लिए प्रयोग किया जाता है।

Syntax : fputchar (char);

उदाहरण—

```
#include<stdio.h>
int mian ()
{
  char ch1='d';
  while (ch1<=t)
  {
    fputchar(ch1)
    ch1++;
  }
    return 0;
}
```

Output :

defghijklmnopqrst

fseek(), ftell () और rewind () function

(i) fseek() : fseek() function का प्रयोग File pointer को किसी location पर move करने के लिए किया जाता है। fseek() में तीन constants का प्रयोग किया जाता है।

(a) SEEK_SET : यह file pointer position को File के beginning पर move करता है।

(b) SEEK_CUR : यह file pointer position को किसी location पर move करता है।

(c) SEEK_END : यह file pointer position को file के end पर move करता है।

(ii) ftell() : ftell( ) function का प्रयोग filepointer का current position को प्राप्त के लिए किया जाता है।

Syntax : ftell(ftr) जहाँ ftr एक file pointer है।

(iii) rwind() : rewind () function का प्रयोग file pointer position को file के beginning पर लाने के लिए किया जाता है।

Syntax : rewind (ftr) जहाँ ftr एक file pointer हैं।

▶ प्रश्नावली (Exercise) ◀

1. Files क्या होते हैं तथा कितने प्रकार के होते हैं?
2. किसी existing file को read करने का C program लिखिए।
3. किसी file के words और character को count करने का C program लिखिए।
4. किसी File को किसी दूसरे नाम में copy करने का C Program लिखिए।
5. किन्हीं दो files को compare करने का C Program लिखिए।
6. किसी file को open या create करने का C Program लिखिए।
7. किन्हीं file पर कौन-कौन से operations perform किये जा सकते हैं।
8. fget() और fput() क्या होते हैं?
9. किसी Text file से specific lines को remove करने का C Program लिखिए।

10. किसी file के lines को count करने का C Program लिखिए।
11. किसी file से word को remove करने का C Program लिखिए।
12. *n* employees के name और salary को read करने और उन्हें file में store करने का C Program लिखिए।
13. निम्नलिखित file handlings को उदाहरण के साथ समझाइए।
    (i) fopen() (ii) fclose() (iii) putc () (iv) getc ( )
14. File opening क्या होता है? उदाहरण के साथा समझाए।
15. fprintf() और fscanf() कहाँ प्रयोग किये जाते हैं? उदाहरण के साथ समझाइए।
16. निम्नलिखित (File handling functions) को वर्णन कीजिए—
    (i) fseek()
    (ii) ftell()
    (iii) rewind ()
17. किसी File के size को प्राप्त करने का C Program लिखिए।
18. File में Text को replace करने का C program लिखिए।
19. File handling क्या होता है? किन्हीं दो File handling functions का वर्णन कीजिए।
20. getw() और Putw() functions को समझाइए।

❑

C Program

Programming exercises on formatting input/output using printf and scanf and their return type values.

Example 1: C Output

```
#include<stdio.h>
int main()
{
  printf(" Programming Language");
  return 0;
}
```

Output :

Programming Language

Integer Output का उदाहरण

```
#include<stdio.h>
int main()
{
  int = 95;
  printf("Number = %d", a);
  return 0;
}
```

Output :

Number = 95

Integer Input/Output का उदाहरण

```
#include<stdio.h>
int main()
{
int a;
printf("Enter an integer: ");
scanf("%d",&a);
  printf ("Number = %d",a);
  return 0;
}
```

Output :

Enter an integer: 95
Number = 95

Float Input/Output का उदाहरण

```
#include <stdio.h>
int main()
{
float a;
  printf("Enter a number:");
  scanf("%f",&a);
  printf("float Value = %f", a);
  return 0;
}
```

Output :

Enter a number: 95.85
Float Value = 95.850000

integer और float input/output का उदाहरण

```
#include<stdio.h>
#include<conio.h>
void main();
{
int m;
float n;
printf("Enter any two numbers: ");
scanf("%d %f",&m,&n);
printf("%d %f",m,n);
getch();
}
```

Output :

Enter any two numbers:
99
99.8

Character Input/output का उदाहरण

```
#include<stdio.h>
int main()
{
  char a;
  printf ("Enter a  character:  ");
  scanf("%c",&a);
  printf (" entered character is %c",a);
  return 0;
}
```

Outut :

Enter a character: a
entered character is : a

C Program to find ASCII Value of a Character

```
#include <stdio.h>
int main( )
{
char ch;
printf("\n Enter any character \n");
scanf("%c",&ch);
printf("\n ASCII value of given character =
%d",ch);
return 0;
}
```

Output :

```
Enter any character: A
ASCII value of given character = 65
```

Programming exercises on executing and editing a C program.

```
#include<stdio.h>
int main()
{
   printf("Hello World");
   return 0;
}
```

Output :

```
Hellow World
```

Program to find the average of two numbers

```
#include<stdio.h>
int main()
{
int n1, n2;
float avg;
printf("Enter first number: ");
scanf("%d",&n1);
printf("Enter second number:");
scanf("%d",&n2);
avg=(float)(n1+n2)/2;
printf("Average of %d and %d is : %.f",
n1,n2,avg);
return 0;.
}
```

Output:

```
Enter first number: 78
Enter second number: 73
Average of 78 and 73 is: 75.5
```

C Program to print Integer enter by user

```
#include<stdio.h>
int main()
{
int m;
printf(" Enter a number: ");
scanf("%d",& m);
printf("\number is :%d",m);
return (0);
}
```

Output:

```
Enter a number : 89
Number is : 89
```

C Program to find Quotient and Remainder

```
#include<stdio.h>
int main()
{
int m,n,q,d ;
printf(" Enter the Number:");
scanf("%d ", &m);
printf(" \nEnter the divisor:");
scanf("\n%d ", &d);
n = m % d ;
q = m/ d;
printf("Quotient is : ");
printf(" %d ", q);
printf(" Remainder is ;");
printf(" %d ", n);
return (0);
}
```

Output :

```
Enter the Number: 87
Enter the divisor :6
Quotient is :14
Remainder is :3
```

C Program to swap two numbers

```
#include<stdio.h>
int main( )
{
int m,n,temp;
printf(" first number:");
scanf("%d ",&m);
printf(" \n second number : ");
scanf("%d ",&n);
printf("\n number Before swapping : ");
printf("\n %d \t %d ", m, n) ;
temp = m ;
m= n ;
n= temp ;
printf("\n number After swapping : ");
printf("\n %d \t %d ", m, n);
return (0);
}
```

Output of Program:

```
first number : 87
second number : 67
number Before swapping :
87 67
number After swapping :
67 87
```

C Program to find the area of circle

```
#include<stdio.h>
int main()
{
int radi;
float area;
printf(" enter the radius of circie : ") ;
scanf("%d ",& radi);
area = 3.14 * radi * radi;
printf("\n area of the circle : ");
printf("%f ", area);
return(0);
}
```

Output of Program:

```
enter the radius of circle :13
area of the circle:50.24
```

C Program to find the simple interest

```
#include<stdio.h>
int main()
{
float m, n, s, p;
printf(" enter the principle amount: ");
scanf("%f ",&n);
printf("\n enter the rate of interest: ");
scanf("%f ",&s);
printf("\n enter the time period : ");
scanf("%f",&p);
sa = ( n *s* p)/100;
printf("\n simple Interest :");
printf(" %f", m);
return ( 0);
}
```

```
enter the principle amount: 75
enter rate of interest :5
enter the time period :20
sample Interest :75
```

C Program to reverse a number

```
#include<stdio.h>
int main()
{
int number, remainder, reverse=0;
printf("enter the number: ");
scanf ("%d ",& number);
while { number > = 0 )
{
remainder = number % 10;
reverse = (reverse * 10 ) + remainder;
number = number/ 10;
{
printf("\n reverse of the number is :");
printf("%d", reverse);
return (0);
}
```

Output :

```
Enter the number : 7568
reverse of the number is : 8657
```

C program to convent Fahrenheit to celsius

```
#include<stdio.h>
int main()
{
  float fo,ce;
  printf("Enter the value of Fahrenheit:");
  scanf("%f",&fo);
  ce=(fo-32)*0.556;
  printf("Celsius is: %.2f",ce);
return 0;
}
```

Output:

Enter the value of Fahrenheit : 99
Celsius is: 37.22

C Program to convert Celsius to Fahrenheit

```
#include<stdio.h>
int main()
{
  float fo,ce;
    printf("Enter the value of Celsius:");
    scanf("%f",&ce);
  co=ce*(9/5)+32;
   printf("Fahrenheit is : %.f",fo);
  return 0;
```

Output :

Enter the value of Celsius:30
Fahrenheit is : 86

C Program to swap two numbers without using third variable

```
#include<stdio.h>
int main()
{
  int m,n,s;
printf{"Enter first number:"};
scanf("%d",&m);
printf("Enter second number:");
scanf("%d",&n);
m=m+n;
n=m-n;
m=m-n;
printf("\n result After Swap\nm: %d n: %d",m,n);
return 0;
}
```

Output :

Enter first number: 4
Enter second number: 5
result After Swap: 5 4

C Program to find whether the given number is divisible by 5

```
#indude<stdio.h>
int main()
{
int number;
printf("Enter the number:");
scanf("%d" &number);
if(number%5==0)
{
printf("%d is divisible by 5",number);
}
else
}
printf("%d is not divisible by 5", number);
}
return 0;
}
```

Output:

Enter the number: 85
85 is divisible by 5

Programming exercises on arithmetic, logical and relational operators

C Program to demonstrate the working of arithmetic operators

```
#include <stdio.h>
int main()
{
  int m = 88,n = 7, result;
  result= m+n;
  printf("m+n = %d \n",result);
  result = m-n;
  printf("m-n = %d \n",result);
  result = m*n;
```

```
    printf("m*n = %d \n",result);
    result= m/n;
    printf("m/n = %d \n",result);
    result = m%n;
    printf("m%n = %d \n",result);

    return 0;
}
```

Output

```
m+n = 95
m–n = 81
m*n = 616
m/n = 12
m%n = 4
```

▌C Program to explain the working of assignment operators

(i)
```
#include <stdio.h>
int main()
{
  int m = 95, result;
result = m;
  printf(" result = %d \n", result);
result + = m;
  printf(" result = %d \n", result);
    result–= m
  printf(" result = %d \n", result);
    result*= m;
printf(" result = %d \n", result);
  result/= m;
    printf(" result = %d \n", result);
  result % = m;
printf(" result = %d \n", result);
  return 0;
}
```

Output

```
result = 95
result = 190
result = 95
result = 9025
result = 95
result = 0
```

(ii)
```
#include <stdio.h>
int main()
{
  int = 1, p = l;
  while (p <= 7 | | q<= 7)
  {
      printf("%d %d\n",p, q);
      p++;
      q++;
  }
  return 0;
}
```

Output :

```
11
22
33
44
55
66
77
```

▌C Program to find greatest number using logical operator

```
#include<stdio.h>
int main ()
{
int n1,n2,n3;
printf{"\nEnter value of n1, n2 and n3:");
scanf("%d%d%d",&n1,&n2,&n3);
if((n1>n2)&&(n1>n3))
printf("\n Number 1 is greatest");
else if((n2>n3)&&(n2>n1))
printf("\n Number2 is greatest");
else
printf("\n Number3 is greatest");
 return 0;
```

Output :

```
Enter value of n1, n2 and n3: 75 99 191
 Number 3 is greatest
```

C Program to explain the working of increment and decrement operators

```
include<stdio.h>
int main ()
{
int m = 98, n = 87;
 printf("++m= %d \n", ++m);
printf("--n = %d \n", --n);
return 0;
}
```

Output :

```
++m = 99
--n = 86
```

C Program to explain the working of logical operators

```
#include<stdio.h>
 int main()
{
int m = 75, n = 75, o = 100, result;
result = (m == n) && (o > n);
printf("(m== n) && (o > n) = %d \n",
result);
result = (m == n) && (o < n);
printf("(m == n) && (o < n) = %d \n",
result);
result = (m== n) | | (o < n);
printf("(m == n) | |  (o < n) = %d \n",
result);
result = (m != n) | | (o < n);
printf("(m != n) | | (o < n) =%d \n",
result);
 return 0;
}
```

Output :

```
(m==n)&&(o>n) = 1
(m== n) && (o < n) = 0
(m==n)|| (o<n) = 1
(m != n) || (o<n) = 0
```

C program to explain declaration and initialization of variables

```
#include<stdio.h>
void main()
{
int height; // Declaration of variable height
height = 85; //initialization of the variable
height
printf("%d \n",height);
}
```

Output :

```
85
```

Global and local variables

C Program to demonstrate the global variables

```
#include
int global=85; //global variables
void main()
{
++global;
printf("%d\n",global);
increase ();
return 0;
 }
void increase()
{
++global;
printf("%d\n",global);
}
```

Output :

```
86
87
```

C program to display day of week using switch case

```
#include <stdio.h>
int main()
{
int num;
printf("Enter number for week:");
 scanf("%d", &num);
switch(num)
{
case 1:
printf("Monday");
break;
case 2:
printf("Tuesday");
```

```
break;
case 3:
printf("Wednesday");
break;
case 4:
printf{"Thursday"); .
break;
case 5:
printf("Friday");
break;
case 6:
printf("Saturday");
break;
case 7:
printf("Sunday");
}
return 0;
}
```

Output

```
Enter number for week: 3
Wednesday
```

C program to dispaly vowel or consonant using switch case

```
#include <stdio.h>
int main()
{
char c;
printf("Enter any English alphabet: ");
scanf("%c", &c);
switch(c)
{
case 'a':
printf("vowel");
break;
case 'e':
printf("vowel");
break;
case'i':
printf("vowel");
break;
case o :
printf ("vowel");
break;
case 'u':
printf("vowel");
default:
printf("Consonant"j;
break;
}
return 0;
}
```

Output :

```
Enter any English alphabet: o
vowel
```

C program to find larger number of two numbers using switch case

```
#include <stdio.h>
int main()
{
int n1, n2;
printf("Enter any two numbers: ");
scanf("%d%d", &n1, &n2);
switch(n1 > n2);
{
case 0:
printf("%d is the larger number", n2);
break;
case 1:
printf("%d is the larger number", n1);
break;
default:
printf("\both numbers are equal");
return 0;
}
```

Output :

```
Enter any two numbers: 99 87
99 is the larger number.
```

C program to find whether the number is positive or negative

```
#include <stdio.h>
int main()
{
int number;
printf("Enter the number:");
```

```
scanf("%d", &number);
switch (number >0)
{
case 1:
printf("number is positive ");
break;
case 0:
printf("number is negative ");
break;
}
return 0;
}
```

Output:

```
Enter any number: 99
Number is positive
```

Programming exercises using if statement C program to find greater variable

```
#indude <stdio.h>
int main()
{
int m = 89;
int n = 92;
if(m<n)
{
printf("Variable m is less than n");
}
return 0;
}
```

Output :

```
Variable m is less than n
```

C program to find the square of a number if it is less than 6

```
#indude<stdio.h>
int main()
{
int m;
printf("Enter the number:");
scanf("%d",&m);
if(m<6)
{
printf("%d is less than 6\n",m);
printf("Square = %d\n",m*m);
}
return 0;
}
```

Output :

```
Enter a number : 5
5 is less than 6
Square = 25
```

C program on If statement.

```
#include
void main()
{
int n;
printf(" Enter a number");
scanf("%d",&n);
printf("Entered number is %d\n",n);
if(n==1)
{
printf("INDIA");
}
if(n==2)
{
printf("AMERICA");
}
if(n==3)
{
printf("ENGLAND");
}
if(num==4)
{
printf("UTTAR PRADESH");
  }
```

OUTPUT :

```
Enter a number
Entered number is 4
UTTAR PRADESH
```

C program to find greater number of two numbers

```
#include <stdio.h>
int main()
{
int m, n;
```

```
printf("Enter the value of m:");
scanf("%d", &m);
printf("Enter the value of n:");
scanf("%d", &n);
if(m>n)
{
printf("m is greater than n\n");
}
if(m<n)
{
printf("m is less than n\n");
}
if (m==n)
{
printf("m is equal to n\n");
}
return 0;
}
```

Output :

```
Enter the value of m : 78
Enter the value of n: 87
m is less than n
```

C PROGRAM USING IF ELSE

▌C program to check the number whether it is even or odd.

```
#include<stdio.h>
int main()
{
int a;
printf("Enter the number:");
scanf("%d",&a);
if(a%2 ==0)
printf("%d is even",a);
else
printf("%d is odd",a);
return 0;
}
```

Output :

```
Enter a number:78
78 is even
Enter a number:73
73 is odd
```

▌C program to check the number is positive, negative or zero.

```
#include<stdio.h>
int main()
{
int m;
printf("Enter the number:");
scanf("%d",&m);
if(m>0)
printf("Number is positive");
else if(m<0)
printf("Number is negative");
else
printf("Number is equal to zero");
return 0;
}
```

Output :

```
Enter a number : 85
Number is positive
Enter a number : –86
Number is negative
Enter a number : 0
Number is equal to zero
```

▌C program to print Fibonacci series

```
#include<stdio.h>
int main()
{
int p, term_1 = 0, term_2 = 1, termnext = i;
printf("Enter the number of terms:\n");
scanf("%d",&p);
printf("First %d terms of Fibonacci series
is:\n",p);
for ( i = 0 ; i < p ; i++ )
{
if (i <= 1)
termnext = i;
else
{
term1 = term1 + term2;
term1 = term2;
term2 = termnext;
}
```

```
printf("%d\n",termnext);
}
return 0;
}
```

Output :

```
Enter the number of terms: 5
First 5 terms of Fibonacci series:
0
1
1
2
3
```

C Program to check the year whether it is leap year or not.

```
#include <stdio.h>
int main()
{
int m;
printf("Enter the year: ");
scanf("%d",&m);
if(m%4==0)
{
if( m % 100 == 0)
{
if ( m% 400 == 0)
printf("%d is a leap year", m);
else
printf("%d is not a leap year", m);
}
else
printf("%d is a leap year", m );
}
else
printf("%d is not a leap year", m);
return 0;
}
```

Output:

```
Enter year: 1995
1995 is not a Leap Year
```

While loop

C program to print the number from 1 to 12 using while loop

```
#include <stdio.h>
int main()
{
int m=1;
while(m<=12)
{
printf("%d\n",m);
m++;
}
return 0;
}
```

Output :

```
1
2
3
4
5
6
7
8
9
10
11
12
```

C program to print the number from 0 to 5 using while loop

```
#include <stdio.h>
int main()
{
int count=0;
while (count <= 5)
{
printf("%d ", count);
count++;
}
return 0;
}
```

Output :

```
012345
```

(i) Examples of infinite while loop

```
#include <stdio.h>
int main()
{
int m = 7;
while (m>=6)
{
printf("%d", m);
m++;
}
return 0;
}
```

(ii) #include <stdio.h>

```
int main()
{
int m =98;
while (m <=100)
{
printf("%d", m);
m--;
}
return ;
}
```

C program to find the sum of natural numbers using while loop

```
#include <stdio.h>
int main()
{
int n, m, sum = 0;
printf("Enter the value of n;");
scanf("%d",&n);
m=1;
while(m <= n)
{
sum = sum + m;
m++;
}
printf("Sum of the first %d natural numbers is: %d",n, sum);
return 0;
}
```

Output :

```
Enter the value of n : 6
Sum of the first 6 natural numbers is : 21
```

C program to find the reverse of numbers using while loop

```
#indude<stdio.h>
int main()
{
int number ,remi,reverse_number=0;
printf("\nEnter the number:");
scanf("%d",&number);
while(number>=1)
{
remi = number % 10;
reverse_number = reverse_number * 10 +
remi;
number = number/10;
}
printf("\nReverse of input number is: %d",
reverse_number);
return 0;
}
```

Output:

```
Enter any number: 67898
Reverse of input number is: 89876
```

C program to print the number using while loop

```
#include <stdio.h>
int main ()
{
int m = 75;
while(m<85)
{
printf("value of a: %d\n", m);
m++
}
return 0;
}
```

Output :

```
value of m: 75
value of m: 76
value of m: 77
```

```
value of m: 78
value of m: 79
value of m: 80
value of m: 81
value of m: 82
value of m: 83
value of m: 84
```

C program to print the number from 0 to 9 using do while loop

```
#include <stdio.h>
int main()
{
int m = 0;
do
{
printf("%d\n",m);
m++
}
while(m<10);
return 0;
}
Output :
0
1
2
3
4
5
6
7
8
9
```

For Loop

C program to print the number from 0 to 9 using For Loop

```
#include <stdio.h>
int main()
{
int m;
for(m=0; m<10; m++);
{
printf("%d\n",m);
}
return 0;
}
Output :
0
1
2
3
4
5
7
8
9
```

C program to reverse the input numbers using For Loop

```
#include<stdio.h>
int main()
{
int array[15] = {1, 2, 3, 4, 5, 6, 7,
9,10,11,12,13,14, 0};
int n;
for(n = 14; n >= 0; n--)
printf("%d", array[n]);
return 0;
}
```

Output : –0 14 13 12 11 10 9 8 7 6 5 4 3 2

C program to find the smallest number using for loop

```
#include <stdio.h>
int main()
{
int a[15] = {1, 2, 3, 4, 5, 6, 7, 8,
10,11,12,13,14,15};
int m, smallest;
smallest = array[0];
for(m = 1; m <= 15; m++)
{
if(smallest > a[m] )
smallest = a[m];
}
```

```
printf("smallest number is %d", smallest);
return 0;
}
```

Output :

smallest number is 1

C Program to find sum of natural numbers using for loop

```
#include <stdio.h>
int main()
{
int m, n, sum = 0;
printf("Enter the value of m");
scanf("%d",&m);
for(n=1; n<= m; n++)
{
sum = sum + n;
}
printf("Sum of the first %d natural numbers is: %d",m, sum);
return 0;
}
```

Output : Enter the value of m : 7

Sum of the first 7 natural numbers is: 28

C program using break

```
#include <stdio.h>
int main()
{
int i;
for(i=10;i>0; i--)
{
if(i==6)
{
printf("\n out from for loop where i = %d \n", i);
break;
}
printf(" %d ",i);
  }
}
```

C programming to find out the average of 5 integers using array

```
#include <stdio.h>
int main()
{
int avg = 0;
int sum =0;
int m=0;
int num[5];
for (m=0; m<5; m++)
{
printf("Enter number %d \n", (m+1));
scanf("%d", &num[m]);
}
for (m=0; m<5;m++)
{
sum = sum+num[m];
}
avg = sum/5;
printf("Average of entered number is: %d", avg);
return 0;
}
```

Output :

Enter number 1

80

Enter number 2

70

Enter number 3

70

Enter number 4

60

Enter number 5

80

Average of entered number is : 72

C Program to display the numbers pattern

```
#include <stdio.h>
int main()
{
int i, j, m;
printf("Enter the Number of row : ");
```

```
scanf("%d ",& m) ;
printf("\n Pattern are : \n\n") ;
for ( i = 1 ; i < = m ; i++)
{
for(j=1;j<=i;j++)
{
printf(" %d",j);
}
printf(" \n ");
}
return (0);
}
```

Output:

```
Enter the Number of row : 6
1
12
123
1234
12345
123456
```

C Program to print the triangular star pattern

```
#include <stdio.h>
int main()
{
int i, j, m;
printf("Enter the Number of row : ") ;
scanf("%d ",& m) ;
printf("\n Patterns are : \n\n") ;
for ( i = 1; i< =m; i++)
{
for (j=1; j< = i; j++)
{
printf("*");
}
printf(" \n ");
}
return ( 0 );
}
```

Output :

```
Enter the Number of row: 6
*
**
***
****
*****
******
```

C Program to print the triangular word pattern

```
#include <stdio.h>
int main()
{
int i, j, m;
char ch;
printf(" Enter the Number of row : ") ;
scanf("%d ",& m);
printf("\n Patterns are : \n\n");
for ( i = 1; i < = m ; i++)
{
ch = 65
for( j = 1; j< = i; j++)
{
printf("%c",ch);
ch++ ;
}
printf(" \n ");
}
return (0);
}
```

Output:

```
Enter the Number of row : 6
A
AB
ABC
ABCD
ABCDE
ABCDEF
```

Programs on one-dimensional array

C program to find the average of 5 integer numbers.

```
#include <stdio.h>
int main()
{
int avg = 0;
int sum =0;
int m=0;
int num[5];
for (m=0; m<=4;m++)
{
printf("Enter the number %d \n", (m+1));
scanf("%d", &num[m]);
}
for (m=0; m<=4;m++)
{
sum = sum+num[m];
}
avg = sum/5;
printf("average of the numbers is: %d",
avg);
return 0;
}
```

Output:

```
Enter the number 1
88
Enter number 2
78
Enter number 3
98
Enter number 4
77
Enter number 5
67
average of the numbers is : 81
```

C Program to find the average of n numbers using arrays

```
#include <stdio.h>
int main()
{
int mark[15], i, n, sum = 0, average;
printf("Enter the value of n:");
scanf("%d", &n);
for(i=0; i<n; ++i)
{
printf("Enter number%d: ",i+1);
scanf("%d", &mark[i]);
sum = sum +marks[i];
}
average = sum/n;
printf("Average = %d", average);
return 0;
}
```

Output :

```
Enter the value of n: 4
Enter nurnber 1: 68
Enter number 2: 72
Enter number 3: 98
Enter number 4: 78
Average = 79
```

C program example to Passing a complete One-dimensional array to a function.

```
#include<stdio.h>
float average(int marks[]);
int main()
{
float avg;
int marks[] = {95, 92,99, 91};
avg = average(marks);
printf("average marks = %.2f", avg);
return 0;
}
float average(int marks[])
{
int i, sum = 0;
float avg;
for (i = 0; i <= 3; i++)
{
sum =sum+ marks[i];
}
avg= (sum /4);
return avg;
}
```

output :

```
94.25
```

Function

C program to find the addition of two numbers using function.

```
#include<stdio.h>
void Add ();
void main()
{
Add();
}
void Add()
{
int sum, x= 98, y = 88;
sum = x + y;
printf("\n sum of x= %d and y = %d is =
%d", x, y, sum);
}
```

Output :

sum of x and y =186

C program to find the addition of two numbers passing as parameter using function.

```
#include<stdio.h>
int Addition(int num1, int num2)
{
printf("%d", num1);
printf("%d", num2);
return num1 + num2;
}
int main()
{
int result, n1, n2;
scanf("%d", &n1);
scanf("%d", &n2);
result = Addition(n1,n2);
printf(" answer = %d\n", result);
return 0;
}
```

C program to find multiplication of two numbers using function with No arguments and with Return value .

```
#include<stdio.h>
int mux();
int main()
{
int multi;
multi = mux();
printf("\n Multiplication of m and n is %d
\n", multi );
return 0;
}
int mux()
{
int multi; m = 15, n = 6;
multi = m * n;
return multi;
}
```

OUTPUT :

Multiplication of m and n is 90

C program to find multiplication of two number using function with arguments and with Return value

```
#include<stdio.h>
int mux(int, int);
int main()
{
   int m, n, multi;
   printf("\n Enter two number\n")
   scanf("%d%d",&m, &n);
   multi = mux(m, n);
   printf("\n multiplication of %d and %d
   is %d \n", m, n, multi);
   return 0;
}
int mux(int m, int n)
{
int multi;
multi = m* n;
return multi;
}
```

Output :

Enter the two numbers

15 6

multiplication of 15 and 6 is 90

C program to create a user defined function addition ()

```
#include <stdio.h>
int add (int n1, int n2)
{
int sum;
sum = n1+n2;
return sum;
}
int main()
{
int m, n;
printf("Enter number first :");
scanf("%d",&m);
printf("Enter number second:");
scanf("%d",&n);
int result = add(m, n);
printf ("output is : %d", result);
return 0;
}
```

Output :

```
Enter number first: 98
Enter number second: 97
```

Output : 195

C program to print table of any integer number using User Defines Functions

```
#include<stdio.h>
void table(int);
int main()
{
int n;
printf("enter an integer number:");
scanf("%d",&n);
printf("table of %d is:\n",n);
table(n);
return 0;
}
void table(int num)
{
int i;
for (i=1; i<=10; i++)
printf("%d\n",(num*i));
}
```

```
Enter an integer number : 12
Table of 12 is :
12
24
36
48
60
72
84
96
108
120
```

C Program to convert binary number to decimal using function

```
#include <stdio.h>
#include <math.h>
int conversion(long n);
int main()
{
long n;
printf("Enter the binary number: ");
scanf("%d", &n);
printf("%d binary = %d decimal", n,
conversion(n));
return 0;
}
int conversion(long n)
{
int decinum = 0, i = 0, remi;
while (n!=0)
{
remi= n%10;
n=n/10;
decinum += remi*pow(2,i);
++i;
}
return decinum;
}
```

Output :

```
Enter a binary number : 1111
1111 binary = decimal 15
```

C program to find sum and average of two numbers using User Defined Functions

```
#include<stdio.h>
int sum(int,int);
float average(int,int);
int main()
{
int n1,n2;
int sum;
float avg;
printf("Enter the first number: ");
scanf("%d",&n1);
printf("Enter the second number:");
scanf("%d",&n2);
sum=sum(n1,n2);
avg=average(n1,n2);
printf("Number1: %d, Number2: %d\n",
n1,n2);
printf("Sum: %d, Average: %f\n",sum,
avg);
return 0;
}
int sum(int x,int y)
{
int sum;
sum=x+y;
return sum;
}
float average(int x,int y)
{
float average;
return ((float)(x)+(float)(y))/2;
}
```

Enter the first integer number: 78
Enter the second integer number:99
Number1: 78, Number2: 99
Sum: 177, Average: 88.50

Recursion

```
#include<stdio.h>
int factorial(int m);
void main()
{
int x, y;
printf("Enter a number:");
scanf("%d", &x);
y = factorial(x);
printf("%d", y);
}
int factorial(int x)
{
if(x==0)
return 1;
else
return = x*factorial(x–1);
}
```

Output :

Enter a number : 6
720

C Program to calculate power using recursion

```
#include <stdio.h>
int pow(int n1, int n2);
int main()
{
int base, power, result;
printf ("Enter the base :");
scanf("%d",&base);
printf{"Enter the power:");
scanf("%d",&power);
result = pow(base, power);
printf("%d", result);
return 0;
}
int pow(int base, int power)
{
if (power != 0)
return (base*pow(base, power-1));
else
```

```
return 1;
}
```

Output :

```
Enter the base : 5
Enter power : 3
result = 125
```

C program to reverse a sentence entered using recursion

```
#include <stdio.h>
void reversestring();
int main()
{
printf("Enter the string: ");
reversestring();
return 0;
}
void reversestring()
{
char c;
scanf("%c", &c);
if(c!='\n')
{
reversestring();
printf("%c",c);
}
}
```

Output :

```
Enter the sentence: datastructure
erutcurtsatad
```

C program to find Sum of Natural Numbers using Recursion

```
#include <stdio.h>
int add(int m);
int main()
{
int number, result;
printf("Enter a positive integer: ");
scanf("%d", &number);
result = add(number);
printf("sum=%d", result);
}
int sum(int num)
{
if(num!=0)
retrunnum + sum(num-1);
else
return num;
}
```

Output :

```
Enter a positive integer:
5
15
```

C program to print Fibonacci series using recursion

```
#include <stdio.h>
int fibo(int a)
{
if(a==0)
{
return 0;
}
if(a ==1)
{
return 1;
}
return fibo(a–1) + fibo(a–2);
}
int main()
{
int a;
for (a = 0; a < =5; a++)
{
printf("%d\n", fibo(a));
}
return 0;
}
```

Output:

```
0
1
1
2
3
5
```

C program on array to show compile time initialization.

```
#include<stdio.h>
void main()
{
int a;
int array[] = {1, 2, 3, 4,5}; // Compile time
array initialization
for(a = 0 ; a < =4 ; a++)
{
printf("%d\t",array[a]);
  }
}
```

Output :

12345

C program on array to show run time initialization.

```
#indude<stdio.h>
void main()
{
int array[6];
int a, b;
printf("Enter element in the array");
for(a = 0; a <= 5; a++)
{
scanf("%d", &array[a]);   //Run time array
initialization
}
for(b= 0; b<= 5; b++)
{
printf("%d\t", array[b]);
}
```

Output:

Enter element in the array:

012345

C Program to read and Display the element of an Array

```
# include <stdio.h>
int main();
{
int a[75], m, p ;
printf("Numbers of Element:");
scanf("%d ",& m);
printf("\n Enter the Element :\n");
for ( p = 0 ; p< = n–1; p++)
{
scanf("%d",&a[p]);
}
printf("\n Elements are : \n ");
for ( p= 1 ; p< = m ; p++)
{
printf("\t%d",a[p]);
)
return (0);
}
```

Output:

Number of Element :

5

Enter the Element:

6

3

4

5

2

Elements are:

6

3

4

5

2

C program to delete data item from Array.

```
#include <stdio.h>
int main()
{
int a[85], p, i, m;
printf("Enter number of elements \n");
scanf("%d", &m);
for (i = 0 ; i <= m; i++ )
{
scanf("%d", &a[i]);
}
printf("Enter the location for deletion \n");
scanf("%d", &p);
if (p >= m+1)
```

```
printf("Deletion not possible.\n");
else
{
for (i = p -1; i< =m-1; i++)
a[i] = a[i+1];
printf("result is\n");
for(i = 0 ; i<= m -1; i++)
printf("%d\n", a[i]);
}
return 0;
}
```

Output :

```
Enter number of elements : 6
6
4
2
5
1
3
Enter the location for deletion:
3
Result is
6
4
5
1
3
```

C Program to find highest number using array

```
#include < stdio.h >
int main()
{
int a[75], m, p, high=0 ;
printf(" Enter the number of elements:");
scanf("%d ",& m) ;
printf("\n Enter the Numbers: \n");
for ( p = 0; p< = m-1 ; p++)
{
scanf("%d ",& a[p]);
}
for ( p= 0 ; p< = m-1; p++)
{
if ( a[p] > high )
{
high=a[p];
  }
}
printf("\n The highest Number is : %d",
high);
return ( 0);
}
```

Output :

```
Enter the number of element : 5
Enter the Numbers:
67
89
78
66
88
The highest Number is : 89
```

C program for search a data item and find its position in array.

```
#include <stdio.h>
main()
{
int a[85 ] ,i ,j,m,data;
printf("Enter number of elements \n");
scanf("%d", &m);
for (i = 0 ; i <= m; i++)
{
scanf("%d", &a[i]);
}
printf("array elements are :\n");
for(i = 0; i<=m; i++)
{
printf(" %d\n", a[i]);
}
printf("Enter number to be found \n");
scanf("%d", &j);
while(j<m)
{
if( a[j] == data )
{
break;
```

```
}
j = j + 1;
}
printf("element found at position %d\n",
data, j+1);
}
```

Output :

```
Enter number of elements
6:
Array elements are :
4
3
6
3
1
5
Enter number to be found
6
element found  at position 3
```

C Program to find Sum and average of elements of array

```
#include <stdio.h>
int main()
{
int a[75], m, p ;
float avg = 0, sum = 0 ;
printf(" total number of elements in
Array:");
scanf("%d ",& m) ;
printf("\n Enter the Element: \n");
for (p = 0 ; p <= m-1; p++)
{
scanf("%d",&a[p]);
}
for ( p= 0 ; p<= m-1; p++)
{
sum = sum + a[p] ;
}
avg = sum / m ;
printf("\n Sum of Elements is : %f ",sum);
printf("\n Average of Elements is : %f ",
avg);
return (0);
}
```

Output :

```
total number of element in Array:
5
Enter the Elements :
67
87
85
76
78
Sum of Elements is : 393
Average of Elements is 78.6
```

C Program to print the Transpose of a Matrix

```
#include < stdio.h >
int main( )
{
int arr[5][5], a, b, row, col ;
printf("Enter Number of Rows : ");
scanf("%d ",& row);
printf("\n Enter Number of Columns : ");
scanf("%d ",& col);
printf("\n Enter the Matrix data : \n");
for ( a = 0 ; a <= row-1 ; a++)
{
for ( b = 0; b<=col-1 ; b++)
{
printf("\n Enter the element [%d] [%d] : "
,a, b);
scanf("%d",&a[a][b]);
}
}
printf("\n\n Elements in the matrix are :
\n";
for ( a= 0 ; a<= row-1 ; a++)
{
for ( b = 0 ; j <= col-1 ; b++)
{
printf("\t%d", arr[a][b]);
}
printf(" \n ");
```

```
}
printf("\n\n Transpose of a Matrix : \n");
for ( a= 0 ; a<= row–1 ; a++)
{
for ( b = 0 ; b <= col–1; b++)
{
printf("\t %d ", arr[b][a]);
}
printf(" \n");
}
return (0);
}
```

Output :

```
Enter Number of Rows : 2
Enter Number of Columns : 4
Elements in the matrix are :        5621
                                    5345

Transpose of a Matrix :
5   5
6   3
2   4
1   5
```

C program to read and print two-dimensional array

```
#include <stdio.h>
int main()
{
int matrix[5][5];
int a, b, row, col ;
printf(" Enter Number of Rows :");
scanf("%d ",& row) ;
printf("\n Enter Number of Columns :");
scanf("%d ",& col);
printf("Enter elements in matrix :);
for(a=0; a<=row-1; row++)
{
for(b=0; b<=col-1; col++)
{
scanf("%d", &matrix[a][b]);
  }
}
printf("\nElements in the matrix : \n");
for(a=0; a<=row–1; row++)
{
for(b=0; b<=col–l;col++)
{
printf("%d ", matrix[a][b]);
}
printf("\n");
}
return 0;
}
```

Output :

```
Enter Numbers of Row:3
Enter Numbers of column : 3
Enter elements in matrix :
80 90 75
67 80 89
78 98 76
Elements in matrix :
80 90 75
67 80 89
78 98 76
```

Simple programs using structures

C program to create, declare and initialize structure

```
#include <stdio.h>
struct students .    //structure declaration
{
char name[88];
int roll;
int marks;
}
int main()
{
struct student std={"Rajiv",89,888};
//declare and initialization of structure
variable
printf("\n Name: %s" ,std.name);
printf("\n roll_ number: %d",std. roll);
printf("\n marks: %d\n",std. marks);
return 0;
}
```

Output :

```
Name: Rajiv
roll_number : 89
marks : 888
```

C program to add two numbers using structure

```
#include<conio.h>
struct sum
{
int m;
int n;
}
void main()
{
int total;
struct sum s;
clrscr();
printf("Enter two numbers:");
scanf("%d%d",&s.m,&s.n);
total=s.m+s.n;
printf("\nSum=%d",total);
getch();
}
```

Output:

```
Enter two numbers: 77 87
Sum=164
```

C program to read and display the customer record using structure

```
#include <stdio.h>
struct customer
{
char name[88];
int cusid;
float balance;
};
int main()
{
struct customer cus;
printf("\n enter details :\n");
printf("Name of customer:");
scanf("%s",& cus.name);
printf("customer ID :");
scanf("%d",&cus.cusid);
printf("balance:");
scanf("%f",&cus. balance);
printf("\nEntered detail is:");
printf("Name: %s"  , cus.name);
printf("customer Id: %d"  , cus.cusid);
printf ("balance: %f\n",cus. balance);
return 0;
```

Output :

```
Enter details :
Name of customer : Rahul
customer ID : 888
balance : 88888
Entered detail is:
Name: Rahul
customer Id: 888
balance : 8888
```

C program to sead and display the student record using structure

```
#include <stdio.h>
struct student
{
char name[78];
int roll;
int marks;
int a [6];
};
int main()
{
int i;
printf("Enter the detail of students:\n");
for(i=0; i<=5; i++)
{
a[i].roll = i+1
printf("\n roll number%d,\n",a[i].roll);
printf("Enter name: "};
scanf("%s",a[i].name);
printf("Enter marks: ");
scanf("%d",&a[i]. marks);
printf("\n");
}
printf("show information:\n\n");
```

```
for(i=0; i<=5; i++)
{
printf("\nRoll number: %d\n",i+1);
printf("Name: ");
puts(a[i].name);
printf("Marks: %d",a[i]. marks);
}
return 0;
}
```

Output :

```
Enter the detail of students:
roll number1,
Enter name: Rahul
Enter marks: 78
roll number2,
Enter name: Shyam
Enter marks: 95
.
.
.
show Information
roll number1,
Enter name: Rahul
Enter marks: 78
roll number2,
Enter name: Shyam
Enter marks: 95
```

▌C program to read and print average marks of students using structure of array

```
#include <stdio.h>
struct student
{
char name [30];
int marks[ 7];
int sum;
float avg ;
};
int main()
{
struct student std;
int i;
printf("Enter name: ");
scanf("%s",&std. name);
printf("Enter marks:\n");
std.sum=0;
for(i=0;i< =6;i++)
{
printf("Marks in subject %d : ",i+1);
scanf("%d",&std.marks[i]);
std.sum= std.sum+std.marks[i];
}
std.avg=(float)(std.sum/7);
printf("\nName: %s \ntotal_marks: %d
\naverage: %f",std.name,std.sum,std.avg);
return 0;
}
```

Output :

```
Enter name: Rahul
Enter marks :
Marks in subject 1 : 84
Marks in subject 2: 82
Marks in subject 3 : 78
Marks in subject 4 : 78
Marks in subject 5 : 86
Marks in subject 6 :89
Marks in subject 7 : 78
Name : Rahul
Total_marks: 575
Percentage : 82.1
```

▌C program to add two distances in kilometer and meter using structure

```
#include <stdio.h>
struct distance
{
int kilometer;
int meter;
};
void sum(struct dis d1, struct dis d2)
{
struct dis d3;
d3.kilometer= d1. kilometer + d2.
kilometer;
d3.meter= d1. meter + d2. meter;
```

```
d3.kilometer = d3.kilometer + d3.meter/1000; //1 kilometer is equal to 1000 meter
d3.meter= d3. meter% 1000;
printf("\nTotaI distance- kilometer: %d, meter: %d",d3.kilometer,d3.meter);
}
int main()
{
struct dis d1,d2;
printf("Enter first distance in kilometer & meter:");
scanf("%d%d",&d1.kilometer, &d1,meter);
printf("Enter second distance in kilometer & meter:");
scanf("%d%d",&d2.kilometer, &d2.meter);
sum(d1,d2);
return 0;
}
```

Output :

Enter first distance in kilometer & meter : 78 700
Enter second distance in kilometer & meter; 75 800
Total distance- kilometer: 154, meter: 500

C PROGRAM TO PASSING STRUCTURE TO FUNCTION

```
#include <stdio.h>
#include <string.h>
struct customer
{
int id;
char name[20];
float balance;
};
void cus1(struct customer record);
int main()
{
struct customer record;
record. id=1;
strcpy(record.name, "Rahul");
record. balance= 99.9;
cus1(record);
return 0;
}
void cus1(struct customer record)
{
printf(" Id : %d \n", record. id);
printf(" Name : %s \n", record. name);
printf(" Percentage ; %f \n", record. balance);
}
```

OUTPUT :

Id is: 1
Name : Rahul
Average : 99.9

C program to explain the accessing of structure member

(i)
```
#include<stdio.h>
#include<string.h>
struct employee
{
char name[95];
int age;
}
int main()
{
struct employee e1;
e1.age = 68;
strcpy(e1.name, "Rahul");
printf("Name of empolyee1: %s\n", e1.name);
printf("Age of employee 1: %d\n", e1.age);
return 0;
}
```

Name of employee 1; Rahul
Age of employee 1: 68

(ii)
```
#include<stdio.h>
struct student
{
char sname[88];
int mark;
};
struct student std[6];
int i, j;
```

```
void fun()
{
for(i = 0; i < =5; i++)
{
printf("\nEnter %dst student record;\n",
i+1);
printf("\n student name:\t");
scanf("%s", std[i].sname);
printf("\nEnter marks:\t");
scanf("%d", &std[i].mark);
}
printf("\n student record:\n");
for(i = 0; i < =5; i++)
{
printf("\n student name is %s",
std[i].sname);
printf("\n mark is %d", std[i]. mark);
}
}
void main()
{
fun();
}
```

Output:

```
Enter 1st student record:
student name: Rahul
Enter marks : 87
.  :  .  .  .
```

▌C program to store the Information and Display it Using Structure

```
#include <stdio.h>
struct voters
{
char name[85];
int id;
int age;
}v;
int main()
{
printf("Enter information:\n");
printf("Enter name:");
scanf("%s", v.name);
printf("Enter  id: ");
scanf("%d", &v.id);
printf("Enter age: ");
scanf("%d", &v.age);
printf ("Display lnformation:\n");
printf("Name: ");
puts(v.name);
printf("id : %d\n",v.id);
printf("age: %d\n", v.age);
return 0;
}
```

Output :

```
Enter information:
Enter name : Rahul
Enter id: 876
Enter age: 77
Display Information:
Name: Rahul
id: 876
age: 77
```

Simple programs using union

▌C program to read and print the value of variable using unions

(i)

```
#include <stdio.h>
void main()
{
union data
{
int d1;
float d2;
}
union data d;
printf("Enter the value of d1:");
scanf("%d", &d.d1);
printf("Value of d1 = %d", d.d1);
printf("\nEnter the value of d2: ");
scanf("%f", &d.d2);
printf("Value of d2 = %f\n", d.d2);
}
```

Output :

```
Enter the value of d1: 90
```

```
Value of d1 = 90
Enter the value of d2: 85
Value of d2 = 85.00
```

(ii)
```
include <stdio.h>
union employee
{
char name[62];
int empid;
} emp1;
int main()
{
printf("Enter name:\n"};
scanf("%s", &emp1.name);
printf("Enter id: \n");
scanf("%d", &emp1.empid);

printf("Display\nName :%s\n", emp1.name);
printf("id: %d", emp1.empid);
return 0;
}
```

Output :

```
Enter name
Rahul
Enter id
8799
Display
Name: Rahul
id: 8799
```

C program to display the total memory size occupied by the union

```
#include <stdio.h>
#include <string.h>
union item
{
int i;
float f;
char a[88];
};
int main()
{
union item item1;
printf( "Memory size occupied by item1 :
%d\n", sizeof(item1));
return 0;
}
```

Output :

```
Memory size occupied by item1 :88
```

Note : memory occupied by the union is the memory required for the largest member of the union.

C program to demonstrate accessing of union member

```
#include <stdio.h>
#include <string.h>
union item
{
int i;
float f;
char a[88];
};
int main()
{
union item d1;
d1.i = 90;
printf("d1.i: %d\n", d1.i);
d1.f = 95.5;
printf( "d1.f :%f\n", d1.f);
strcpy( d1.str, " Programming language");
printff "d1.str: %s\n", d1.str);
return 0;
}
```

Output :

```
d1.i: 88
d1.f: 95.5
d1.str: Programming language
```

String

C program to calculate Length of String without Using strlen() Function

```
#include <stdio.h>
int main()
{
char s[100];
int i;
printf("Enter the string:");
```

```
scanf("%s", s);
for(i = 0; s[i] != '\0'; ++i);
printf("length of the string: %d", i);
return 0;
}
```

Output

```
Enter the string: datastructures
length of the string: 14
```

C program to Concatenate Two Strings Without Using strcat()

```
#include <stdio.h>
int main()
{
char s1[80], s2[80], i, j;
printf("Enter the first string:");
scanf("%s", s1);
printf("Enter the second string: ");
scanf("%s", s2);
for(i = 0;s1[i] !='\0';++i);
for(j = 0; s2[j] != '\0'; ++j, ++i)
{
  s1[i] = s2[j];
}

s1[i] = '\0;'
printf("After concatenation: %s", s1);
return 0;
```

Output :

```
Enter first string : data
Enter second string: structure
After concatenation: data structure
```

C program to copy the second string argument to the first string argument using strcpy() function.

```
#include<stdio.h>
#include<string.h>
int main()
{
char s1[100];
char s2[100];
strcpy(s1, "datastructure");
strcpy(s2, s1);
printf("%s\n", s2);
return(0);
}
```

Output

```
Datastructure
```

C program to reverse the given string expression using strrev() function.

```
#include<stdio.h>
int main()
{
char s1[100];
printf("enter the string: ")';
gets(s1);
printf("\n reverse string is: %s",strrev(s1));
return(0);
}
```

Output :

```
enter the string: datastructure
reverse string is: erutcurtsatad
```

Program on pointer

C Program to print the address and value of variable

```
#include <stdio.h>
int main()
{
int m;
int *v ;
printf(" enter the integer:");
scanf("%d ",& m);
v = &m;
printf("\n value of Integer: %d ",m);
printf("\n value of Integer: %d ",*v);
printf("\n value of Integer: %d ",*(&m));
printf("\n Address of Integer: %u ",&m);
return ( 0 );
}
```

Output :

```
enter the Integer :87
value of Integer :87
value of Integer :87
value of Integer :87
Address of Integer :10643284
```

C program to find memory address of a variable

```
#include <stdio.h>
int main()
{
int n = 88;
printf("Value of n  = %d\n", n);
printf("Address of n = %d\n", &n);
return 0;
}
```

Output:

```
Value of n  =88
Address of n = 64538648
#include<stdio.h>
int main ()
 {
int n = 8 ;
int *ptr ;
ptr = &n;
*ptr = *ptr * 9;
print f ( "%d" , n) ;
return 0;
}
```

Output :

72

C Program to print multiplication table using pointer

```
#include <stdio.h>
int main()
{
int n, i ;
int *ptr;
printf(" Enter the number for multiplication Table:") ;
scanf("%d ",&n);
ptr = &n;
printf("\n Multiplicaiton Table of %d = :\n",n) ;
for (i = 1; i <= 10 ; i++ )
{
printf("\n %d ",(*ptr * i));
}
return (0);
}
```

Output of Program:

```
Enter the number for multiplication Table
: 6
Multiplication Table of 6 =
6
12
18
24
30
36
42
48
54
60
```

C program to access Array Elements Using Pointers

```
#include <stdio.h>
int main()
{
int a[4], i;
printf("Enter number:");
for(i = 0; i < 4; i++)
scanf("%d", a + i);
printf("entered number is \n");
for(i = 0; i<4; ++i)
printf("%d\n", *(a + i)),
return 0;
}
```

Output :

```
Enter number :
2
3
4
5
entered number is
2
3
4
5
```

C program to calculate sum number Using Pointers

```
#include <stdio.h>
int main()
{
int i, a[4], sum = 0;
printf("Enter numbers:");
for(i = 0; i < 4; ++i)
{
scanf("%d", a+i);
sum += *(a+i);
}
printf("Sum = %d", sum);
return 0;
}
```

Enter numbers: 26 18 24 26
Sum=94.

C Program to find factorial of an integer using pointer

```
# include <stdio.h>
int main()
{
int n, fact=1;
int *ptr1, *ptr2;
printf(" Enter any Number:");
scanf("%f ",&n);
ptr1 = &n;
ptr2 = &fact;
do
{
*ptr2 = ( *ptr2 ) * ( *ptr1);
*ptr1 = *ptr1-1;
}
while(*ptr1>0);
printf("\n Factorial of number is : %d
",*ptr2);
return (0);
}
```

Output :

Enter any Number: 5
Factorial of number is :120

C program to demonstrate the Passing address to a Function by swapping

```
#include <stdio.h>
void swap(int *p, int *q);
int main()
{
int n1=85, n2= 90;
swap(&n1, &n2);
printf("number1 = %d\n", n1);
printf("number2 = %d", n2);
return 0;
}
void swap(int * p, int * q)
{
int temp;
temp = *p;
*p=*q;
*q = temp;
}
```

output:
number1 = 90
number2 = 85

C program to add two integers using pointer

```
#include <stdio.h>
int main()
{
int first, second, *p, *q, sum;
printf("Enter two integers to add\n");
scanf("%d%d", &first, &second);
p = &first;
q = &second;
sum = *p + *q;,
printf("Sum of the numbers = %d\n",
sum);
return 0;
}
```

C Program to reverse a number using pointer

```
#include <stdio.h>
int main()
{
```

```
int number, remainder, reverse=0 ;
int *ptr1, *ptr2;
printf("Enter the number:");
scanf("%d ",& number);
ptr1 = &number;
ptr2 = &reverse ;
do
{
remainder =( *ptr1) % 10;
*ptr2 = ( *ptr2 * 10 ) + remainder;
*ptr1 = ( *ptr1 )/10;
}
while(*ptr1>0);
printf("\n Reverse is : %d ",*ptr2);
return ( 0 );
}
```

Output :

```
Enter the number: 987654
Reverse is : 456789 .
```

C Program to print the area of square using pointer

```
#include <stdio.h>
int main()
{
float s, a=0 ;
float *ps, *pa ;
printf("Enter the side of square :");
scanf("%f ",&side);
ps = &s ;
pa = &a ;
area = (*ps) * (*ps) ;
printf("\n Area of square is : %f ", area);
return (0);
}
```

Output :

```
Enter the side of square: 4
Area of square is : 16 .
```

C Program to find the greatest number among three numbers using pointer

```
# include <stdio.h>
int main( )
{
int p, q, r ;
int *ptr1, *ptr2, *ptr3;
printf("enter first number :");
scanf("%d",&p);
printf("\n enter second number :");
scanf("%d",&q);
printf("\n enter third number : ") ;
scanf("%d ",& r);
ptr1 = &p;
ptr2 = &q;
ptr3 = &r;
if((*ptr1> = *ptr2) && (*ptr1> = *ptr3))
printf("\greatest Number is: %d ",*ptr1);
else
{
if(( *ptr2> = *ptr1) &&(*ptr2> = *ptr3))
printf("\n greatest Number is: %d ",*ptr2);
else
printf("\n greatest Number is: %d ",*ptr3);
}
return (0);
}
```

Output:

```
enter first number: 87
enter second number:76
enter first number:89
greatest number is :89
```

C program to print a string using pointer

```
#include <stdio.h>
int main()
{
char string[200];
char*ptr1;
printf("Enter the string: ");
gets(string);
ptr1=string;
printf("Entered string is:");
while(*ptr1!='\0')
printf("%c",*ptr1++);
return 0;
}
```

Output :

```
Enter the string: c programming
Entered string is: c programming
```

```
C program to demonestrae access of array
using pointer
#include <stdio.h>
int main()
{
int a[4] = {6, 5, 3, 4,2};
int * ptr1;
ptr1 =&a[1];
printf("*ptr1 = %d \n", *ptr1);
printf("*ptrl+1 = %d \n", *ptr1+1);
printf("*ptrl–1 = %d", *ptr1–1);
return 0;
}
```

Output :

```
*ptr1 =5
*ptr1+1 = 3
*ptr1–1 = 6
*ptr1+2=4
```

▌C program to calculate the sum of n numbers using malloc() and free() function

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
int m, i, *ptr1, sum = 0;
printf("Enter number of elements:");
scanf("%d", &m);
ptr1 = (int*) malloc(m * sizeof(int));
if(ptr1== NULL)
{
printf("memory not allocated.");
exit(0);
}
printf("Enter elements: ");
for(i = 0; i < m; ++i)
{
scanf("%d", ptr1 + i);
sum += *(ptr1 + i);
}
printf("Sum = %d", sum);
free(ptr1);
return 0;
}
```

▌C program to find Largest number Using Dynamic Memory Allocation.

```
#include<stdio.h>
#include <stdlib.h>
int main()
{
int i, number;
float *n;
printf ("Enter total number of elements ");
scanf("%d", &number);
data = (float*) calloc (number, sizeof
(float));
if(n == NULL)
{
printf ("memory not allocated.");
exit(0);
}
printf ("\n");
for(i =0; i <= number; i++)
{
printf ("Enter Number %d: " , i + 1);
scanf ("%f", n + i);
}
for(i = 1; i < number; ++i)
{
if(*n < *(n + i))
*n = (n + i);
}
printf("largest number is =%f", *n);
return 0;
}
```

▌C program to addition of two numbers using pointers

```
#include <stdio.h>
int main()
{
int n1, n2, add;
```

```
int *ptr1, *ptr2;
ptr1 = &n1;
ptr2 = &n2;
printf("Enter two numbers:");
scanf("%d%d", ptr1, ptr2);
add = *ptr1 +*ptr2;
printf("addition= %d", add);
return 0;
}
```

C program to perform all arithmetic operations using pointers

```
#include <stdio.h>
int main()
{
float n1, n2;
float *ptr1, *ptr2;
float add, sub, mult, div;
ptr1 = &n1;
ptr2 = &n2;
printf("Enter any two numbers:");
scanf("%f%f", ptr1, ptr2);
add = (*ptr1) + (*ptr2);
sub = (*ptr1)-(*ptr2);
mult = (*ptr1) *(*ptr2);
div = (*ptr1)/(*ptr2);
printf("addition = %.1f\n", add);
printf("subtraction = %.1f\n", sub);
printf("multiply = %1f\n", mult);
printf("division = %.1f/n",div);
return 0;
}
```

Output :

```
Enter any two numbers : 80 40
addition = 120.0
subtraction : 40.0
multiply : 3200.0
Division : 2.0
```

Simple programs for File Handling

(i) C program to rename a file usig rename ()function.

```
#include <stdio.h>
int main()
{
char oldfile [88];
char new file[88];
printf("Enter the old file path");
scanf("%s",oldfile);
printf("Enter the new file path");
scanf("%s",newfile);
if (rename(oldfile, newfile)==null)
{
printf("File renamed successfully\n");
}
else
{
printf("Unable to rename files\n");
}
return 0;
}
```

(ii) n employees के name और salary को read करने और उन्हें file में store करने का C Program.

```
#include <satdio.h>
int main()
{
char name[85];
int salary, i, n;
printf("Enter the number of employees");
scanf("%d",&n);
FILE *ftr;
ftr=fopen("employee.txt","w");
if(ftr==NULL)
}
printf("Error");
exit(1);
}
for(i=0; i<=n-1; i++)
```

```
{
printf("employee%d\n Enter the name :",
i +1);
scanf("%s", name);
printf("Enter the salary:");
scanf("%d",&salary);
printf(ftr,"\n name:%s\n salary=%d\n",
name,salary);
}
Fclose(ftr);
return 0;
}
```

▌(iii) C program to read a string using fgets() function and write a strintg using fputs() function in a file.

```
#include<stdio.h>
#include<conio.h>
int main();
{
FILE*ft1;
char b[85];
int n=0;
ft1=fopen("File.txt","w");
printf("Enter the string");
scanf("%s",b);
fputs(b,ft1);
fclose (ft1);
ft1=fopen("file.txt","r");
fopen ((b,85,ft1)!=NULL)
while(b[n]!='\0')
{
put char (b[n]);
n++;
}
fclose (ft1);
getch();
}
```

Output :

```
Enter the string : Datastructure
Datastructure
```

▌(iv) Program to create a file using file handling function fopen().

```
#include<stdio.h>
int main()
{
FILE*ftr;
char file1[75];
printf("Enter the file name:");
scanf("%s",file1);
ftr=fopen(file1,"w");
if (ftr==NULL)
{
printf("File not created");
exit(0);
}
printf("File created successfully");
return 0;
}
```

▌(v) C program to read and write file using getc() and putc() function

```
#include <stdio.h>
void main()
{
FILE*ftr1;
char m ;
ftr1 = Fopen ("fileprogram.txt","w")
while ((m=getchar())!=EOF)
{
putc(m,ftr1);
}
fclose (ftr1);
ftr1=Fopen("File program.txt","r")
while ((m=getc (ftr1))!=EOF)
{
printf("%c",m)
fclose (ftr1);
return 0;
}
```

(vi) C program to write integer in a file

```
#inoude<stdio.h>
#include<stdlib.h>
int main()
{
int m;
FILE *ftr1;
ftr1=fopen("File1.txt","w");
printf("Enter the integer number:");
scanf("%d",&m);
fprintf(ftr1,"%d",m);
fclose(ftr1);
return 0;
}
```

(vii) C program to read integer data by using fscanf()

```
#include <stdio.h>
#include<stdlib.h>
int main()
{
int m;
FILE *ftr1;
ftr1=fopen("File1.txt","r");
fscanf(ftr1,"%d",&m);
printf("value:%d",m);
fclose (ftr1);
return 0;
}
```

(viii) C program to write integer data in file by using fprintf()

```
#include<stdio.h>
#include<stdlib.h>
int main()
{
int m;
FILE *ftr1;
Ftr1=fopen("File1.txt","w");
printf("Enter the number");
scanf("%d",&m);
fprintf(ftr1,"%d",m);
fclose(ftr1);
return 0;
}
```

EVEN SEMESTER EXAMINATION, JUNE-2019

C के प्रयोग द्वारा प्रोग्रामन संकल्पना

(Concept of Programming Using C)

Code : 2087

IInd Semester

Time : 2.30 Hours] [Maximum Marks : 50

Notes :

(i) Attempt all questions.

(ii) Students are advised to specially check the Numerical Data of question paper in both versions. If there is any difference in Hindi translation of any question, the students should answer the question according to the English version.

(iii) Use of Pager and Mobile Phone by the students is not allowed.

1. Answer any two of the following : [2 × 5 = 10]
निम्न में से किन्हीं दो के उत्तर दीजिए :

(a) What are various data types in C? Explain.

(अ) C में कौन से data types होते हैं? व्याख्या कीजिए।

(b) What is the difference between a compiler and an interpreter?

(ब) एक Complier एवं एक interpreter में क्या अन्तर है?

(c) What is structured programming? Explain.

(स) Structured programming क्या होती है? व्याख्या कीजिए।

2. Answer any two of the following : [2 × 5 = 10]
निम्न में से किन्हीं दो के उत्तर दीजिए :

(a) Explain the difference between while and do-while loop with example.

(अ) While एवं do-while loop की अन्तर की व्याख्या उदाहरण देकर कीजिए।

(b) Explain recursive function with example.

(ब) Recursive function की उदाहरण के साथ व्याख्या कीजिए।

(c) Discuss the use of switch statement.

(स) Switch statement का क्या use है? Discuss कीजिए।

3. Answer any two of the following : [2 × 5 = 10]
निम्न में से किन्हीं दो के उत्तर दीजिए :

(a) What is use of header files in C?

(अ) C language में header files का क्या use है?

(b) Explain any two string manipulation functions in C language.

(ब) C language में किन्हीं दो string manipulation function की व्याख्या कीजिए।

(c) Explain ledder if-else statement in C.

(स) C के ladder if-else statement की व्याख्या कीजिए।

4. Answer any two of the following : [2 × 5 = 10]
निम्न में से किन्हीं दो के उत्तर दीजिए :

(a) Explain the static and dynamic memory allocation in C.

(अ) C langauage में Static एवं dynamic memory allocation की व्याख्या कीजिए।

(b) Explain parameter passing methods in a function.

(ब) Function में parameter passing के तरीकों की व्याख्या कीजिए।

(c) Write a 'C' program to find factorial of an integer.

(स) एक integer का factorial निकालने के लिए एक C program लिखिए।

5. Write short notes on any two of the following : [2 × 5 = 10]

निम्न में से किन्हीं दो पर संक्षिप्त टिप्पणी लिखिए :

(a) Declaration of union in C.

(अ) C में declaration of union

(b) Opening and closing of files.

(ब) Opening and closing of files

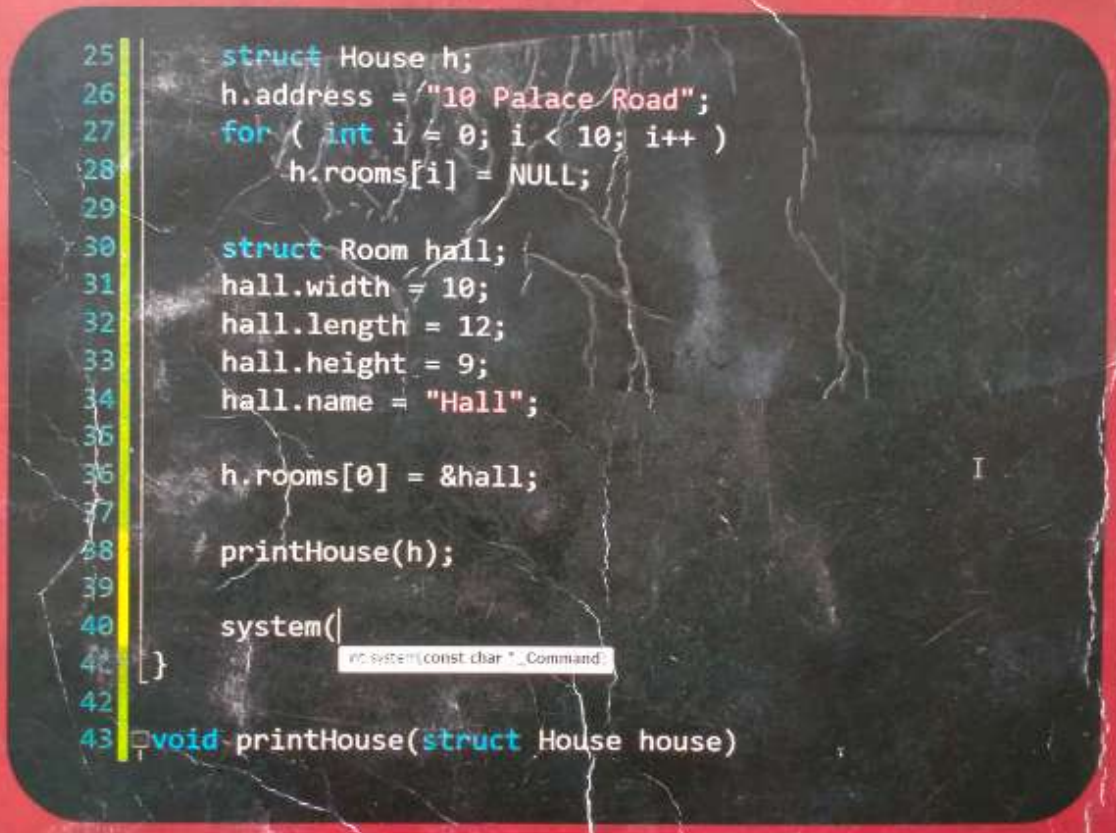
(c) Accessing structure members in C.

(स) C में structure members को access करना

O

Sarthak

Concept of Programming Using 'C'



Jai Prakash Nath Publications

Gandhi Ashram Crossing, Nauchandi Road, Meerut City (U.P.)

Tel.: (O)2762403, 4056123, (R) 4022395 Fax: 0121-2600606

web : www.jpnpbooks.com E-mail : jpnpmrt@hotmail.com

Like us on facebook www.facebook.com/jpnpmrt