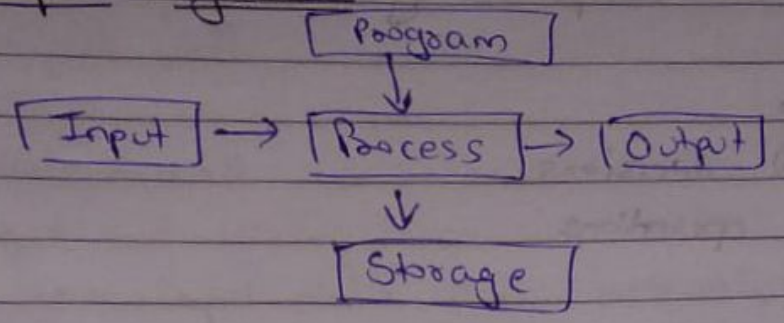
Unit 1

Page No.
Date :

Computer System

Computer System



A Computer System is a programmable electronics device that can be programmed to accept some inputs in terms of data then process this data and provides the output in desired format.

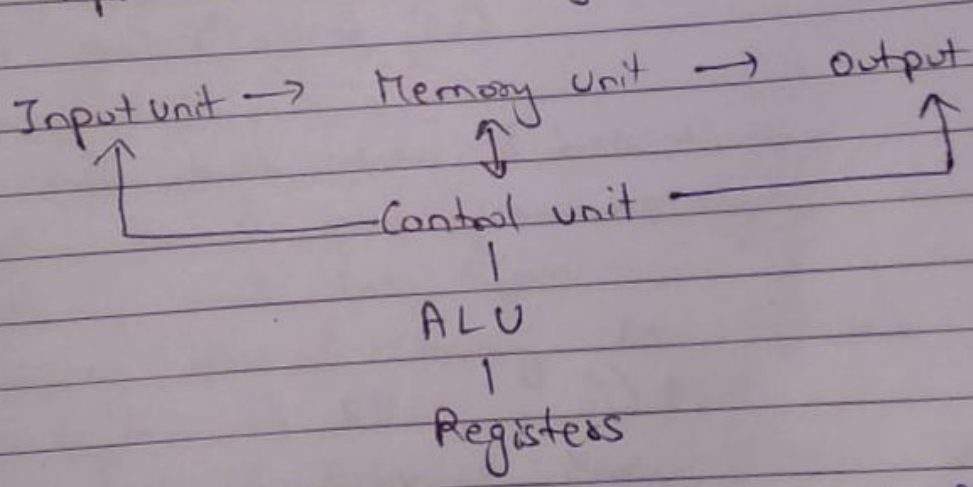
Computer System consist of Two Main components

a) Hardware Components → Physical component like keyboard etc
b) Software Components → like MS word which we cannot touch

Computer Software components

a) Operating System
b) System Software
c) Application Programs

Input → Processing → Output



Memory unit → Slow → Memory access speed → fast
like Harddisk

CPU → Brain & Provide computational & dicision Making Power

Page No.
Date :

Characteristics of computer system

a) Speed
b) Data storage
c) Fast logical decisions
d) Accuracy of operations

Features of computer system

a) Computer is Programmable
b) Understand only Binary (0 & 1)

Program Compilation

High level — Low level

Stage 1 — Stage 2 — Stage 3 — Stage 4

~~Processing~~ Preprocessing — Compilation — Assembly — Linking

Assembly Code

Page No.
Date :

Components of computer System

3 component
a) Input unit
b) CPU
c) Output unit

a) Input unit
A user input data and instructions through input devices such as keyboard, mouse etc.
The input unit is used to provide data to the Processor for further processing

b) Central Processing unit
CPU is Main unit of computer
It control all the internal & external devices of a computer & performs arithmatics & logical operators
The Microprocessor chip contains Million of transistors
The transistors work together to store & ~~Manu~~ Manipulate data so Microprocessor Perform a wide variety of functions.
CPU & its components
i) Memory Unit ii) Arithmetic and Logical unit (ALU)
iii) control Unit (CU)

i) Memory unit
It store both data & instructions
Data stored ~~per~~ permanently
Memory unit two types
1) Primary / internal Memory Data stored temporarily
two types i) RAM ii) ROM
2) Secondary / External Memory → Data Stored permanently
non volatile

ii) ~~Arithmetic & logical unit~~

Page No.
Date :

ii) Arithmetic & Logical Unit

* Logical operation AND, OR, NOT, <, >, <=, >=
* + - * /

iii) Control Unit (CU)

CU called Central Nervous System or brain of computer

It tell Computer's ALU and input and output devices how to respond to the instructions

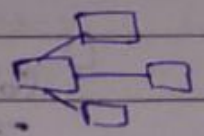
c) Output unit

It helps for displaying or printing the processed data in a user readable form

Page No.
Date :

Computing Environments

1st generation computer — Vaccume
Next generation computer — Transistor and IC are used
3rd generation computer — Microprocessors are used
Then we use the small computer
Computing environment

a) Personal Computing Environment
b) Time sharing Environment  (Time slot given)
c) client server Comp. Environment
d) Distributed computing Environment → (Internet used)

[client] ⟶ [Server] (Process the data which is sent by client

give reply to client

b/w client & server only

Defination Computing Environment is a collection of computer which are used to process & exchange information to solve various type of computing problems.

Computer languages

Machine language
0, 1
Binary digit

High Level language
A programming language

Assembly language
Collection of mnemonic
low level language

Assembler

Compiler

Features of High level lanaguage

a) Fast programming process
b) Instruction are written in almost like every day english

Linker → A program which takes one or more object files & combines into a single executable file.

Page No.
Date :

Creating and Running Program

Program → Set of instructions
written in a programming language

Creating involves 4 steps

a) Writing and editing the program
b) Compiling the ~~pro~~ program → (High level language to Machine language)
c) Linking the program → add library functions & Mathematical library
d) Executing the program

a) Writing & editing program
i) Text editors are used to write program
ii) User can enter, change character data using text editor
iii) After writing Program the file saved to disk it is known as source file

Programer ⟶ Text editor $\xrightarrow[\text{file}]{\text{source}}$ [#include<stdio.h
void main()] → Source file

b) Compiling the program
High level language to Machine language
C compiler has 2 separate programs
Preprocessor ⟶ Translator

read source code & prepares it for translator

It does actual work of converting program into Machine language program

c) Linker
Assembles Mathematical library & some function with Source program for to be executing the Program

$\underbrace{\text{Source program + library}}_{\text{we need linker}}$ ⟶ executable

.obj file ⟶ [Linker] ⟶ .exe file

d) Executing Program
loader is a Software that gets the program that is ready for execute into memory.

Page No.
Date :

Preprocessor Directive

preprocessing → compiling → linking

(Preprocessor program है जो Preprocessing करता है)

~~Preprocessor directive के द्वारा~~

जब Program Run होता है Actual compilation के पहले Preprocessing face में Preprocessor द्वारा source code को examine करा जाता है Preprocessor directive के लिए Preprocessor directive तो ~~ही~~ नहीं है अगर है तो उनहें operate करा जाता है then compilation process take place

include <Stdio.h>

library को Program में excess करवाता है

define
if
else

Preprocessor directive

- Macro Substitution
- File Inclusion
- Compiler Control

include <Stdio.h>

including separate file into our program into our source code

include <string.h>

again include string file into our source code

Page No.
Date :

Compilation Process in C

The compliation is a process of converting the source code into object code It is done with the help of compiler. The compiler checks the source code if it is error free then it generates the object code

The compilation process can be divided into 4 types

a) Pre processing
b) Compilation
c) Assembling
d) Linking

Preprocessing → It removes all the comments from source code
It will attached the file of stdio.h
It will send to the compiler

Compiler → High level language to ~~machine language~~ Assembly language

Assembler → Assembly language into object code by using Assembler
Extension of object file in DOS .obj
~~hello~~ C hello.c ⟶ hello.obj

<u>Linker</u> library added करता है

hello.c
↓ ⟵ Preprocessor
hello.i
↓ ⟵ Compiler
hello.s
↓ ⟵ assembler
hello.obj
↓ ⟵ linker
hello.exe

Page No.
Date :

Algorithm

~~Ate~~ Defination → It is a process or ~~set~~ step by step instruction for solving a problem.

Example

Write a algorithm of adding 3 no.

Step1 → Start
Step2 → Declare variables ~~Result~~ sum num1 num2 num3
Step3 → ~~Result~~ = num1 + num2 + num3
Step4 → Print ~~Result~~ sum
Step5 → Stop

Characteristics

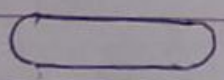
a) Clear & unambiguous — आप algorithm को easy language में लिखेंगे तो बाकी सब भी समझ सके।
b) Effectiveness
c) Definiteness

Advantages

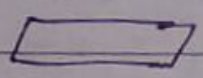
a) You can easily understand the Motive of program
b) Helps in searching errors
c) Helps in correcting errors

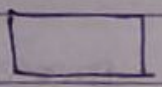
Flowchart

Defination → Graphical representation of algorithm

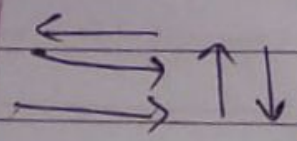

Starting / ending

Connector


Input / output / statement


Processing

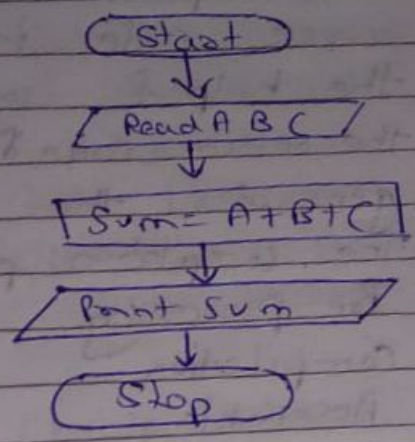
Relation / condition


Flow lines

Page No.
Date :

Algorithm
Start
Read 3 variable
Sum = A + B + C
Print Sum
Stop

Flow chart



Pseudocode
It is an artificial and informal language that helps programmers to develop program
Ex → two no. addition
Start
input no.1 and no.2
Determine Sum = No.1 + No.2
Print Sum
end

Page No.
Date :

C Programming language

language → The way of communication b/w two persons
Programming Language → The way of communication b/w man & Machine.

Program consist of set of ~~ins~~ instructions and written in some language which is called Programming Language

History of C

C Programming → created by Dennis Ritchie in Bell Laboratories in 1972

year	Name	Full Form	used for
1957	FORTAN	Formula Translation	Mathematical & Scientific application
1960	COBOL	Common Business Oriented Language	Business application or softwares
1967	BCPL	Basic combined Programming Language	General purpose but with some issues
1970	B		Some issues
1972	C		General purpose Programming language so got famous

Why C language is still Important even after around 50 years

C Programming language was develop to write unix operating system but later a lots of application and system software were written in C language

Ex → Unix, Linux, Windows, Mac etc
Viros, antivirus, Robotics, IoT
Printer & Scaner.

Features

a) General purpose Programming language
सारे Program, App etc में used होता है

b) Very Fast language
c) Versatile language
d) Modular Programing language → बड़े Program को छोटे में तोड़ कर solve करता है।
e) Simple & small Programming language

Character set of C Program

letters a, b, & etc → character set (A to z, a to z, 0 to 9, + etc, other symbol)
↓
words cat → Tokens
↓
Sentence What is your name? → Instructions by Syntax
↓
Paragraph what → Program

ASCII CODE do from Notes GATE

Tokens

a) Smallest individual unit of a Program
b) We can not further split tokens.
c) Every instruction of a program is collection of ~~token~~ tokens

Page No.
Date :

Tokens

- **keywords**
  - int
  - float
  - char
  - (32 keywords)
  - if
  - else
  - while
  - Keyword written in lowercase
- **identifiers**
  - ram
  - shyam
  - (we given the name by our own)
  - Space not allow
  - — allow
  - identifier can't start with digits (0-9)
- **Constants/ literals**
  - 5
  - 7.5
  - A
  - etc
  - Integer constant — decimal not allow
  - Decimal integer constant — decimal allow
  - Single character constant — 'A', 'a', '<' etc
  - 'ab' ✗ "ab" ✓
  - String constants — It contain any no. of characters enclosed in double quotes (" ")
  - "ABC", "a"
  - " "
- **operators**
  - +
  - −
  - *
  - /
- **special symbol**
  - (), {}
  - ; : , ,
- **string literals**
  - "ABC"
  - "A"
  - etc
  - String(" ") होता है।
- **white spaces**
  - \n
  - etc

%d → integer
%o → Octal
%x → Hexadecimal

NY

```
char ch = 'A';
int i = 48;
```

① char character की ASCII value को print करना है

② char character data type defined करते समय
'A' ✓ "A" ✗

Page No.
Date :

Structure of a C Program

C programming language is a structured Programming language. C programing divided into 7 sections these are

a) Documentation Section This is very first section of any Program. It contains comment lines to give information such as title, Author, Name date, time etc

/* _____ */

This is optional.

Types of comment (//)

i) Single line comment → 1 line comment

we use symbol (//)

eg → // Program to get sum

ii) Multiple line comment (/* */)

eg → /* x y z
— —
— — — */

b) Link Section

To include predefined header file

include < stdio.h >

include < conio.h > → header file

we use printf scanf so it is in stdio.h.

c) Macro (symbolic constant) / Definition section

In this section we define any symbolic constant used in Program then value of symbolic constant will always remain fix throughout the program.

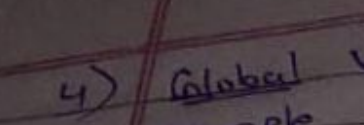
define PI 3.142

preprocessor directive — Macro Symbolic constant Name — value

4) Global ~~Variable(s)~~ Declaration section

example

int size = 10;



its value can be change & use it through out the Program

~~User defined functions (Sub Program) Declaration section~~

5) Main function section

This section is Must to be included in every C Program in its starting point of execution of any program.

```
void main()
{
 Declaration Part;
 Executable Part;
}
```

8) Sub-Program Section

As we know that every ~~function~~ program has some built in & some user defined function.
User defined is defined by the user.

I/O statement

Input & output statement are used to read and write in the data in C programming.
These are embedded in stdio.h
stdio.h is a header file

a) Printf → This function is used for displaying the output on the screen

Syntax

```
printf (" xyz")
```

Output

xyz

b) Scanf we enter data by using an input device

Syntax

```
Scanf (" format specifier", & arg1);
```

c) getch → to display long time

d) clrscr → only current run

e) putchar → A single character is displayed on screen

Syntax

```
putchar (ch);
```

Chapter 1: Variables, Constants & Keywords

Variables

A variable is a container which stores a 'Value'. In Kitchen, we have containers storing Rice, Dal, Sugar etc. Similar to that Variables in C stores value of a constant. Example:

```
a = 3;      // a is assigned "3"
b = 4.7;    // b is assigned "4.7"
c = 'A';    // c is assigned 'A'
```

Rules for naming variables in C

1. First character must be an alphabet or underscore (_)
2. No commas, blanks allowed.
3. No special symbol other than (_) allowed.
4. Variable names are case sensitive.

We must create meaningful variable names in our programs. This enhances readability of our programs.

Constants

An entity whose value doesnt change is called as a constant.

A variable is an entity whose value can be changed.

Scanned with CamScanner

Types of constants

Primarily, there are three types of constants:

1. Integer Constant → -1, 6, 7, 9
2. Real Constant → -322.1, 2.5, 7.0
3. Character Constant → 'a', '$', '@' (Must be enclosed within single inverted commas)

Keywords

These are reserved words, whose meaning is already known to the compiler. There are 32 keywords available in C.

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

Our First C Program

```
#include <stdio.h>

int main() {
printf("Hello, I am learning C with Harry");
return 0;
}
```

File: first.c

URBAN EDGE

Basic Structure of a C Program

All C programs have to follow a basic structure. A C program starts with a main function and executes instructions present inside it. Each instruction is terminated with a semicolon (;)

There are some rules which are applicable to all the C programs :

1. Every program's execution starts from main() function.
2. All the statements are terminated with a semicolon.
3. Instructions are case-sensitive
4. Instructions are executed in the same order in which they are written.

Comments

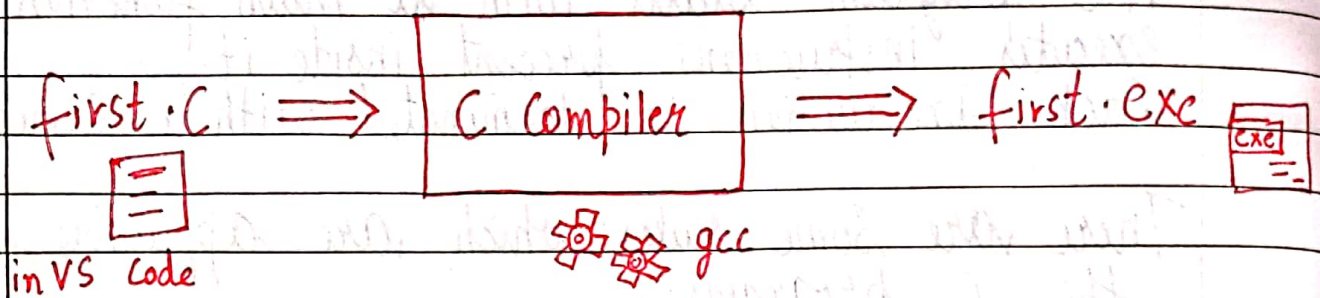
Comments are used to clarify something about the program in plain language. It is a way for us to add notes to our program. There are two types of comments in C.

1. Single line comment : // This is a comment
2. Multi-line comment : /* This is a multi line comment */

Comments in a C program are not executed and are ignored.

Scanned with CamScanner

Compilation and Execution



A compiler is a computer program which converts a C program into machine language so that it can be easily understood by the computer.

A C program is written in plain text. This plain text is combination of Instructions in a particular sequence. The compiler performs some basic checks and finally converts the program into an executable.

Library Functions

C language has a lot of valuable library functions which is used to carry out certain tasks. for instance printf function is used to print values on the screen

```
printf("This is %d", i);
```

%d for integers
%f for real values
%c for characters

Scanned with CamScanner

Types of Variables

1. Integer variables → int a = 3;
2. Real variables → int a = 7.7; float a = 7.7;
3. Character Variables → char a = 'B';

Wrong as 7.7 is real

Receiving input from the User

In order to take input from the user and assign it to a variable, we use Scanf function

Syntax for using scanf :

```
Scanf("%d", &i);
```

This & is important!

& is the "address of" operator and it means that the supplied value should be copied to the address which is indicated by variable i.

Scanned with CamScanner

Chapter 2 : Instructions and Operators

A C program is a set of Instructions. Just like a recepie - which contains Instructions to prepare a particular Dish.

Types of Instructions

1> Type declaration Instruction
2> Arithmetic Instruction
3> Control Instruction

Type declaration Instruction

```
int a;
float b;
```

Other Variations :

```
int i=10; int j=i; int a=2
int j= a + j - i;
```

```
float b = a+3; float a=1.1
```
=> ERROR! as we are trying to use a before defining it.

```
int a, b, c, d;
a = b = c = d = 30;
```
=> Value of a, b, c & d will be 30 each.

Scanned with CamScanner

Arithmetic Instructions

operators

int i = (3 * 2) + 1

operands

Operands can be int/float etc.
+ - * / are arithmetic operators

```
int b=2, c=3;
int z; z = b*c;   ✓ legal
int z;  b*c = z;  ✗ Illegal (Not allowed)
```

% → Modular division operator
% → Returns the remainder
% → Cannot be applied on float
% → Sign is same as of numerator (-5%2=-1)

5%2 = 1 -5%2 = -1

Note :-

1) No operator is assumed to be present

int i = ab → Invalid
int i = a*b → Valid

2) There is no operator to perform exponentiation in C
However we can use pow(x, y) from <math.h> (More later)

Scanned with CamScanner

Type Conversion

An Arithmetic operation between

Int and Int → Int
Int and float → Float
Float and float → Float

5/2 → 2 5.0/2 → 2.5
2/5 → 0 2.0/5 → 0.4

} Important!!

Note :

int a = 3.5; In this case 3.5 (float) will be demoted to 3(int) because a is not able to store floats.

float a = 8; a will store 8.0
8 → 8.0 (promotion to float)

Quick Quiz :

Q int k = 3.0/9 Value of k? and why?

S 3.0/9 = 0.333.. but since k is an int, it cannot store floats & value 0.33 is demoted to 0.

Scanned with CamScanner

Operator precedence In C

$3 * x - 8y$ is $(3x) - (8y)$ or $3(x - 8y)$?

In C language simple mathematical rules like BODMAS, no longer applies.

The answer to the above question is provided by operator precedence & associativity.

Operator precedence :- The following table lists the operator priority in C

Priority	Operators
1st	* / %
2nd	+ -
3rd	=

Operators of higher priority are evaluated first in the absence of parenthesis.

Operator Associativity :- When operators of equal priority are present in an expression, the tie is taken care of by associativity.

$x * y / z \Rightarrow (x * y) / z$

$x / y * z \Rightarrow (x / y) * z$

*, / follows Left to right associativity

Scanned with CamScanner

Control Instructions
Determines the flow of Control in a program
Four types of Control Instructions in C are:

1. Sequence Control Instruction
2. Decision Control Instruction
3. Loop Control Instruction
4. Case Control Instruction.

Scanned with CamScanner

Chapter 3 - Conditional Instructions

Sometimes we want to watch comedy videos on YouTube if the day is sunday.
Sometimes we order junk food if it is our friend's birthday in the hostel.
You might want to buy an Umbrella if its raining and you have the money.
You order the meal if dal or your favorite bhindi is listed on the menu.

All these are decisions which depends on a condition being met.
In C language too, we must be able to execute instructions on a condition(s) being met.

Decision Making Instructions in C
→ If - else Statement
→ Switch Statement

If - else Statement

The syntax of an If - else Statement in C looks like:

```
if (condition to be checked) {
        Statements - if - condition - true;
    }
else {
        Statements - if - condition - false;
    }
```

Code example:

```
int a = 23;

if (a > 18){
    printf(" You can drive \n");
}
```

Note that else block is not necessary but optional.

Relational Operators in C

Relational operators are used to evaluate conditions (true or false) inside the if statements. Some examples of relational operators are:

==	>=	>	<	<=	!=
equals	greater than or equal to				not equal to

Important note: '=' is used for assignment where as '==' is used for equality check.

The condition can be any valid expression. In C a non-zero value is considered to be true.

Logical Operators

&&, || and ! are three logical operators in C. These are read as "AND", "OR" and "NOT". They are used to provide logic to our C programs.

Scanned with CamScanner

Usage of Logical Operators:

(i) && → AND → is true when both the conditions are true

"1 and 0" is evaluated as false.
"0 and 0" is evaluated as false.
"1 and 1" is evaluated as true.

(ii) || → OR → is true when at least one of the conditions is true. (1 or 0 → 1)(1 or 1 → 1)

(iii) ! → returns true if given false and false if given true

!(3==3) → evaluates to false
!(3>30) → evaluates to true.

As the number of conditions increases, the level of indentation increases. This reduces readability. Logical operators come to rescue in such cases.

else if clause
Instead of using multiple if statements, we can also use else if along with if thus forming an if-else if-else ladder.

```
if {
// Statements;
}
else if {...
}
else {...
}
```

Using if - else if - else reduces indents
The last "else" is optional
Also there can be any number of "else if"

Last else is executed only if all conditions fail.

Operator precedence

Priority	Operator
1st	!
2nd	*, /, %
3rd	+, -
4th	<>, <=, >=
5th	==, !=
6th	&&
7th	||
8th	=

Conditional Operators
A short hand "if - else" can be written using the conditional or ternary operators

Condition ? expression-if-true : expression-if-false

Ternary operators

URBAN EDGE

Switch Case Control Instruction

Switch-Case is used when we have to make a choice between number of alternatives for a given variable.

```
Switch (integer-expression)
{
  Case C1:
      Code;

  Case C2:
      Code;

  Case C3:
      Code;

  default:
      Code;
}
```

C_1, C_2 & C_3 → Constants

Code → Any valid C code.

The value of integer-expression is matched against C_1, C_2, C_3 .... If it matches any of these cases, that case along with all subsequent "case" and "default" statements are executed.

* Quick Quiz: Write a program to find grade of a student given his marks based on below:

→ 90 - 100 → A

→ 80 - 90 → B

→ 70 - 80 → C

→ 60 - 70 → D

→ < 70 → F.

Scanned with CamScanner

URBAN EDGE

Important Notes

1> We can use switch-case statements even by writing cases in any order of our choice (not necessarily ascending)

2> char values are allowed as they can be easily evaluated to an integer

3> A switch can occur within another but in practice this is rarely done.

Scanned with CamScanner

Chapter 4 - Loop Control Instruction

Why Loops

Sometimes we want our programs to execute few set of instructions over and over again. for ex: printing 1 to 100, first 100 even numbers etc.

Hence Loops make it easy for a programmer to tell computer that a given set of instructions must be executed repeatedly.

Types of Loops

Primarily, there are three types of loops in C Language:

1. While loop
2. do - while loop
3. for loop

We will look into these one by one

While loop

```
While (condition is true) {

    // Code
    // Code

    }
```

⇒ The block keeps executing as long as the condition is true.

An example:

```
int i = 0

while (i < 10) {
    printf (" The value of i is %d", i); i++;
}
```

Note: If the condition never becomes false, the while loop keeps getting executed. Such a loop is known as an infinite loop.

Quick Quiz: Write a program to print natural numbers from 10 to 20 when initial loop counter i is initialized to 0.

The loop counter need not be int, it can be float as well.

Increment and decrement operators

i++ → i is increased by 1
i-- → i is decreased by 1

```
printf ("--i = %d", --i);
```

This first decrements i and then prints it

```
printf (" i-- = %d", i--);
```

This first prints i and then decrements it

Scanned with CamScanner

* +++ operator does not exist ⇒ Important
* += is compound assignment operator just like -=, *=, /= & %= ⇒ Also Important

do-While Loop.
The syntax of do-while loop looks like this:

```
do {
    // Code;
    // Code;
} while ( condition)
```

do-while loop works very similar to while loop.
while → checks the condition & then executes the code
do-while → Executes the code & then checks the condition

do-while loop = While loop which executes at least once.

→ Quick Quiz: Write a program to print first n natural numbers using do-while loop.

Input : 4

Output : 1
2
3
4

Scanned with CamScanner

URBAN EDGE

for Loop

The syntax of for loop looks like this:

```
for( initialize ; test ; increment)
  {                          or decrement
    //Code;
    // Code;
    // Code;
  }
```

Initialize → Setting a loop Counter to an initial value
Test → Checking a condition
Increment → Updating the loop Counter

An example:

```
for( i=0 ; i<3 ; i++) {
    printf( "%d", i);
    printf( "\n");
  }
```

Output:

```
0
1
2
```

Quick Quiz: Write a program to print first n natural numbers using for loop

Scanned with CamScanner

A Case of Decrementing for loop

```
for (i=5; i ; i--)
        printf("%d\n", i);
```

This for loop will keep on running until i becomes 0.
The loop runs in following steps:

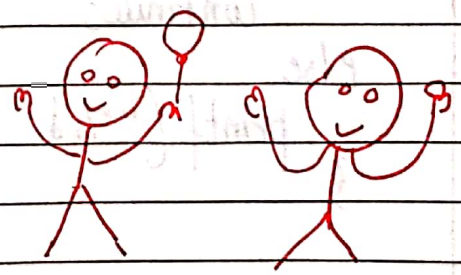
1. i is initialized to 5
2. The condition "i" (0 or nono) is tested
3. The code is executed
4. i is decremented
5. Condition i is checked & code is executed if its not 0.
6. & so on until i is non 0

Quick Quiz: Write a program to print n natural numbers in reverse order.

The break Statement in C

The break statement is used to exit the loop irrespective of whether the condition is true or false.
Whenever a "break" is encountered inside the loop, the control is sent outside the loop

Let us see this with the help of an Example



Scanned with CamScanner

```
for (i=0; i<1000; i++) {
  printf("%d\n", i);
  if (i==5) {
    break;
  }
}
```

output ⇒

0
1
2
3
4
5

and not 0 to 100 ☺

The continue statement in C

The continue statement is used to immidiately move to the next iteration of the loop.

The control is taken to the next iteration thus skipping everything below "continue" inside the loop for that iteration

Let us look at an example

```
int skip = 5;
int i=0;

while (i<10) {
  if (i != skip)
      continue;
  else
      printf("%d", i);
}
```

output ⇒ 5

and not 0 ... 9

Scanned with CamScanner

Notes :

1. Sometimes, the name of the variable might not indicate the behaviour of the program.
2. break statement completely exits the loop.
3. Continue statement skips the particular iteration of the loop.

Scanned with CamScanner

Chapter 5 - Functions and Recursion

Sometimes our program gets bigger in size and its not possible for a programmer to track which piece of code is doing what.
function is a way to break our code into chunks so that it is possible for a programmer to reuse them.

What is a Function?

A function is a block of code which performs a particular task.
A function can be reused by the programmer in a given program any number of times.

Example and Syntax of a function

```
# include <stdio.h>

Void display();        => Function prototype

int main() {
   int a;
   display();          => Function call
   return0;
   }

Void display() {       => Function definition
   printf("Hi I am display");
}
```

Scanned with CamScanner

Function prototype
Function prototype is a way to tell the compiler about the function we are going to define in the program.
Here void indicates that the function returns nothing.

Function call
Function call is a way to tell the compiler to execute the function body at the time the call is made.
Note that the program execution starts from the main function in the sequence the instructions are written

Function definition
This part contains the exact set of instructions which are executed during the function call. When a function is called from main(), the main function falls asleep and gets temporarily suspended. During this time the control goes to the function being called. When the function body is done executing main() resumes.

Quick Quiz → Write a program with three functions
1. Good morning function which prints "Good Morning"
2. Good afternoon function which prints "Good Afternoon"
3. Good night function which prints "Good night"

main() should call all of these in order 1 → 2 → 3

Scanned with CamScanner

URBAN EDGE

Important Points

→ Execution of a C program starts from main()

→ A C program can have more than one function

→ Every function gets called directly or indirectly from main()

→ There are two types of functions in C. Lets talk about them

Types of Functions

1. Library functions → Commonly required functions grouped together in a library file on disk

2. User defined functions → These are the functions declared and defined by the user.

Why use functions?

1. To avoid rewriting the same logic again and again.

2. To keep track of what we are doing in a program

3. To test and check logic independently.

Scanned with CamScanner

URBAN EDGE

Passing values to functions

We can pass values to a function and can get a value in return from a function.

```
int sum (int a, int b)
```

The above prototype means that sum is a function which takes values a (of type int) and b (of type int) and returns a value of type int

function definition of sum can be:

```
int sum (int a, int b) {
    int c;
    c = a+b;
    return c;
}
```

⇒ a and b are parameters

Now we can call Sum (2, 3); from main to get 5 in return.

↳ Here 2 & 3 are arguments

```
int d = Sum (2, 3);
```

⇒ d becomes 5

Note:

1. Parameters are the values or variable placeholders in the function definition. Ex a & b.

2. Arguments are the actual values passed to the function to make a call. Ex 2 & 3.

Scanned with CamScanner

3> A function can return only one value at a time

4> If the passed variable is changed inside the function, the function call doesnt change the value in the calling function.

```
int change (int a) {
    a = 77;
    return 0;
}
```

⇒ Misnomer

change is a function which changes a to 77. No if we call it from main like this

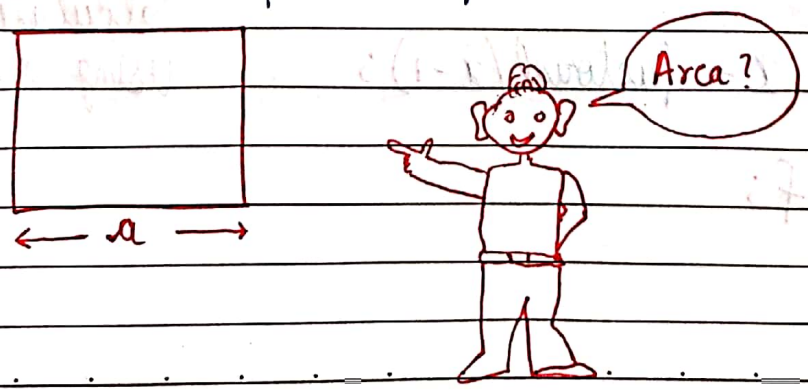
```
int b = 22
change (b);
printf(" b is %d", b);
```

⇒ The value of b remains 22

⇒ prints "b is 22"

This happens because a copy of b is passed to the change function

Quick Quiz → Use the library functions to calculate the area of a square with side a.



Scanned with CamScanner

Recursion

A function defined in C can call itself. This is called recursion.
A function calling itself is also called 'recursive' function.

Example of Recursion

A very good example of recursion is factorial

$$\text{factorial}(n) = 1 \times 2 \times 3 \cdots \times n$$

$$\text{factorial}(n) = \underbrace{1 \times 2 \times 3 \cdots n-1}_{} \times n$$

$$\text{factorial}(n) = \text{factorial}(n-1) \times n$$

Since we can write factorial of a number in terms of itself, we can program it using recursion.

```
int factorial (int x) {
    int f;
    if (x==0 || x==1)
        return 1;
    else
        f = x * factorial(x-1);

    return f;

}
```

⇒ A program to calculate factorial using recursion

Scanned with CamScanner

How does it work?

factorial(5)

5 × factorial(4)

5 × 4 × factorial(3)

5 × 4 × 3 × factorial(2)

5 × 4 × 3 × 2 × factorial(1)

5 × 4 × 3 × 2 × 1

Important Notes:

1. Recursion is sometimes the most direct way to code an algorithm
2. The condition which doesnt call the function any further in a recursive function is called as the base condition.
3. Sometimes, due to a mistake made by the programmer, a recursive function can keep running without returning resulting in a memory error.

Scanned with CamScanner

Chapter 7 - Arrays

An array is a collection of similar elements.
imp

One variable ⇒ capable of storing multiple values

Syntax
The syntax of declaring an Array looks like this:

```
int marks[90];      ⇒ Integer array
char name[20];      ⇒ Character array or String
float percentile[90]; ⇒ float array
```

The values can now be assigned to marks array like this:

```
marks[0] = 33;
marks[1] = 12;
```

Note: It is very important to note that the array index starts with 0!

Marks →

7	6	21	3	91	3			88	89
0	1	2	3	4	5	...		88	89

Total = 90 elements

Scanned with CamScanner

Accessing elements

Elements of an array can be accessed using:

```
Scanf("%d", & marks[0]);
```
⇒ Input first value

```
printf("%d", marks[0]);
```
⇒ output first value of the array

Quick Quiz → Write a program to accept marks of five students in an array and print them to the screen.

Initialization of an Array

There are many other ways in which an array can be initialized.

```
int cgpa[3] = {9, 8, 8}
float marks[] = {33, 40}
```
⇒ Arrays can be initialized while declaration

Arrays in memory

Consider this array:

```
int arr[3] = {1, 2, 3}
```
⇒ 1 integer = 4 bytes

This will reserve 4 x 3 = 12 bytes in memory
4 bytes for each integer.

1	2	3
62302	62306	62310

⇒ arr in memory

Scanned with CamScanner

Pointer Arithmetic

A pointer can be incremented to point to the next memory location of that type.

Consider this example

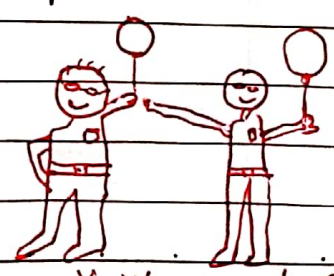
i
32

```
int i = 32;
int *a = &i;   => a = 87994        address -> 87994
a++;           => Now a = 87998

char a = 'A';
char *b = &a;  => b = 87994
b++;           => Now b = 87995

float i = 1.7;
float *a = &i; => Address of i or a = 87994
a++;           => Now a = 87998
```

Following operations can be performed on a pointers:

1. Addition of a number to a pointer
2. Subtraction of a number from a pointer
3. Subtraction of one pointer from another
4. Comparison of two pointer variables

Quick Quiz → Try these operations on another variable by creating pointers in a separate program. Demonstrate all the four operations.



Yay! we understood pointer arithmetic

Scanned with CamScanner

Accessing Arrays using pointers

Consider this array

7	9	2	8	1
0	1	2	3	4

index

ptr

If ptr points to index 0, ptr++ will point to index 1 & so on...

This way we can have an integer pointer pointing to first element of the array like this:

```
int *ptr = & arr[0];  → or simply arr
ptr++;
 *ptr  => will have 9 as its value
```

Passing arrays to functions

Arrays can be passed to the functions like this

```
printArray(arr,n);          => function call

Void printArray(int *i, int n);   => function prototype
        or
Void printArray(int i[], int n);
```

Scanned with CamScanner

Multidimensional Arrays

An array can be of 2 dimension / 3 dimension / n dimensions

A 2 dimensional array can be defined as:

```
int arr[3][2] = { { 1, 4 }
                  { 7, 9 }
                  { 11, 22 } };
```

We can access the elements of this array as arr[0][0], arr[0][1] & so on...

arr[0][0] ⇓ Value = 1

arr[0][1] ⇓ Value = 4

2-D arrays in Memory

A 2d array like a 1-d array is stored in contiguous memory blocks like this:

arr[0][0]	arr[0][1]	...			
1	4	7	9	11	22
87224	87228	..			

Quick Quiz: Create a 2-d array by taking input from the user. Write a display function to print the content of this 2-d array on the screen.

Scanned with CamScanner

Chapter 8 - Strings

A string is a 1-D character array terminated by a null ('\0')
↳ This is null character

null character is used to denote string termination
characters are stored in contiguous memory locations

Initializing Strings

Since string is an array of characters, it can be initialized as follows:

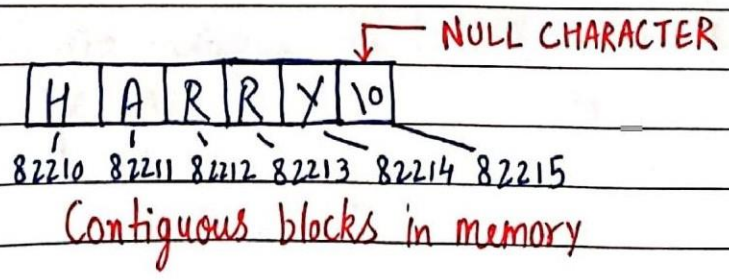
```
char S[] = {'H', 'A', 'R', 'R', 'Y', '\0'};
```

There is another shortcut for initializing strings in C language:

```
char S[] = "HARRY";
```
⇒ In this case C adds a null charater automatically.

Strings In Memory

A string is stored just like an array in the memory as shown below



Quick Quiz → Create a string using " " and print its content using a loop.

Printing Strings
A string can be printed character by character using printf and %c
But there is another convenient way to print strings in C.

```
Char St[ ] = " HARRY";
printf(" %s", St);
```
⟹ prints the entire string.

Taking string input from the user
We can use %s with scanf to take string input from the user:

```
Char St[50];
scanf(" %s", &St);
```

Scanf automatically adds the null character when the the enter key is pressed.

Note:
1. The string should be short enough to fit into the array
2. Scanf cannot be used to input multi-word strings with spaces.

gets() and puts()
gets() is a function which can be used to receive a multi-word string.

```
Char st[30];
gets(st);
```
⇒ The entered string is stored in st!

Multiple gets() calls will be needed for multiple strings

Likewise, puts can be used to output a string.

```
puts(st);
```
⇒ prints the string
places the cursor on the next line

Declaring a string using pointers
We can declare strings using pointers

```
char *ptr = "Harry";
```

This tells the compiler to store the string in memory and assigned address is stored in a char pointer

Note:
1. Once a string is defined using char st[] = "Harry", it cannot be reinitialized to something else.
2. A string defined using pointers can be reinitialized.
```
ptr = "Rohan";
```

Standard library functions for Strings
C provides a set of standard library functions for string manipulation.

Some of the most commonly used string functions are:

Strlen()
This function is used to count the number of characters in the string excluding the null ('\0') character.

```
int length = Strlen(st);
```

These functions are declared under <string.h> header file

Strcpy()
This function is used to copy the content of second string into first string passed to it.

```
Char source[] = "Harry";
char target[30];
Strcpy(target, source);
```
⇒ target now contains "Harry"

Target string should have enough capacity to store the source string.

URBAN EDGE

Strcat()
This function is used to concatenate two strings

```
char S1[] = "Hello";
char S2[] = "Harry";

Strcat(S1, S2);
```

⇒ S1 now contains "Hello Harry"
<No space in between>

Strcmp()
This function is used to compare two strings.
It returns: 0 if strings are equal
Negative value if first string's mismatching character's ASCII value is not greater than second string's corresponding mismatching character. It returns positive values otherwise.

```
Strcmp("Far", "Joke");
Strcmp("Joke", "Far");
```

→ Positive value
→ Negative value