C Program

Introducing C.

C is a general-purpose programing language that has been around for nearly 50 years.

C has been used to write everything from operating systems.

Windows & many other

Complex program like

Python interpreter, Git, oracle database.

Output:-

Most computer programs are designed to produce output.

Ex.- "You've got a new message" notifications.

- "Game over" displayed on the screen when playing video games

Creat.

Ex:- "Welcome to C"

Printf ("welcome to c");

"Hi there"

Printf ("Hi there");

Code Playground.

```
1. # include < stdio.h>
2.
3.  int main(){
4. ....Printf ("welcome to c");
5. .... return 0;
```

★ The Printf keyword is a function, that creates the
The text that we want to output needs to be
inside parentheses:

```
1. # include.< stdio.h>
2.
3. int main(){
4. ....Printf ("Great progress ");
5. .... return 0;
6. }
```

Statements

-> A line of Code is Called a Statement.

-> Each Statement in C needs to end with a Semicol

Outputting Text

Text needs to be enclosed in double quotes.

Program structure

Here is the code to output text. ("Hello");

```
1.  # include <stido.h>
2.
3.  int main(){
4.      Printf ("Hello");
5.      return 0;
6.  }
```

-> The first line of code adds the <stdio.h> header file to the program.

-> # include <stido.h>

-> A header is a file that contains functions and commands.

-> The <stido.h> header contains the Printf function. that we used for outputting

-> # symbol it's necessary

```
int main() {
    Printf ("Hello");
    return 0;
}
```

Each C program has a entry point, or starting point, which is a function called main.

- Curly brackets { } indicate the beginning and end a function, which can also be called the function's body

```
int main() {
    Printf ("Hello");
    return 0;
}
```

- When we run the code, the printf command will be executed, as its inside the main() function.

- The last statement of the main() function is return 0;

- The main() function can contain a lot of statements doing different things.

Ex:- two different printf statements:

```
1. # include <stido.h>
2.
3. int Main () {
4. ....  Printf ("first");
5. ....  Printf ("Second");
6. ....  return 0;
7. }
```

important points:-

→ The #include statement is used to add a header file to the program.

→ To use the printf command, you need to include the <stido.h> header.

→ The starting point of c programs is the function called main, which includes the code that you want to run inside curly braces.

→ The last statement of the main() function is return 0; which indicates the successful completion of the program.

New topic.

A lot of programs work with numbers.

Ex:- a game could output the score of the player.

```
1. #include <stido.h>.
2.
3. int main() {
4. .... Printf (253);
5. .... return 0;
6. }
```

Numbers.

The Printf function supports special format specifiers which allow to output numbers and output formatting.

To output the number 253 from the previous

Ex:-

```
1. #include <stido.h>
2.
3. int main (){
4. ..... Printf ("%d", 253);
5. .... return 0;
6. }
```

-> %d is the format specifier, which denotes that a whole number is expected.

-> The value of the number is given after a comma in Printf.

Output Formatting

There are a number of format specifiers you can use.

%f is used to output floating point numb

```
1. # include <stido.h>
2.
3. int main () {
4. .... Printf ("%f", 3.14);
5. .... return 0;
6. }
```

%c denotes a single character, while %s is used for strings.

```
1. # include <stdio.h>
2.
3. int main () {
4. .... Printf ("%c", 'x');
5. .... return 0;
6. }
```

→ Single character need to be enclosed in single quotes, while string used double quotes.

- A great feature of the formate specifiers is that you can use a combination of multiple specifiers and text in the printf () function.

Ex:-
```
1. # include <stdio.h>
2.
3. int main () {
4. .... Printf ("Score : %f, Level : %d", 42.8, 4);
5. .... return 0;
6. }
```

Ex:- What is the output of this code?

Printf ("%d %d %c", 3, 2, 'z');

→ 32z.

As we have seen in the previous lessons, Pri
does not add a new line after the output.
This means that when you use multiple print
Statements, they will get printed on the same
line.

Ex:-

```
1. # include <stdio.h>
2.
3. int main (){
4. ....Printf ("Name: %S", "James");
5. .... printf ("Age: %d", 42);
6. .... return 0;
7. }
```

New lines :-

The Special character \n is used to create a new line in the Output.

```
1. # include <stdio.h>
2.
3. int main (){
4. .... Printf ("Name: %s\n". "James");
5. .... printf ("Age: %d", 42);
6. .... return 0;
7. }
```

* %d is for whole numbers.

```
1. #include <stdio.h>
2.
3. int main () {
4. .... printf ("%d", 325);
5. .... return 0;
6. }
```

* %f is for floating point numbers.

```
1. # include <stdio.h>
2.
3. int main () {
4. .... printf ("%f", 2.48);
5. .... return 0;
6. }
```

* %c is for single character

```
1. # include <stdio.h>
2.
3. int main() {
4. ...Printf ("%c", 'A');
5. .... return 0;
6. }.
```

* %s is for strings

```
1. # include <stdio.h>
2.
3. int main () {
4. .... Print f ("Hello %s!" %. name);
5. .... return 0;
6. }
```

* Multiple format.

```
1. # include <stdio.h>
2.
3. int main(){
4.     printf("%c", 'Hello');
5.     printf("%f", 2.35);
6.     return 0;
7. }
```

* \n is used special character

```
1. # include <stdio.h>
2.
3. int main (){
4.     ~~printf("Java. %s\n", "Python")~~.
~~5.     printf("Hello World!\n")~~.
4.     printf("Hello, \n World!");
```

Variables :-

- Every program works with values.
- A variable lets you store a value by assigning it to a name.
- The name can be used to refer to the value later in the program.

Ex:- In a game, you would use a variable to store the player's score.

- Every variable has a type, which defines the type of the value it holds.
- A variable can hold a text value, a number, a decimal, etc.

→ int score;

- int is the type that stores whole numbers, or as we call them in programming, integers.

→ In programing terms, the process of creating a variable is called declaration.

- After declaring our variable, we can assign it a value using the assignment = operator:

```
int score
score = 253;
```

Now. score holds the value 253

Now,

```
1. # include <stdio.h>
2.
3. int main() {
4. .... int score;
5. .... score = 253;
6.
```

We used the %d format specifier, as score is a integer, and passed score to the printf function a the value to be inserted.

Ex:- Drag & drop to declare an integer variable calle level, assign the value 5 to it, then output it the format "Level: 5".

```
int level;
level = 5;
Printf ("Level : %d". level);
```

- We can combine the declaration and assignm into one statement, like this:

```
1. # include <stdio.h>
2.
3. int main() {
4. .... int score = 253;
5. .... printf ("%d", score);
6.
7.
8. ....return 0;
```

A variable can change its value during the program multiple times.

Ex:- the Score can change during a game:

```
1. #include <stdio.h>
2.
3. int main() {
4. .... int Score = 253;
5. .... score = 280;
6.
7. .... printf("%d", score);
8.
9. .... return 0;
10. }
```

What is the output of this code?

```
int x = 9;
int y = 8;
x = 3;
Printf("%d%d", y, x);
```

→ 83

Key

* A variable has a name and a type of the value it holds.
* To declare a variable use the type followed by the name of the variable.

- The int type is used to store whole numbe
- You can assign a value to the declared variable using the = Operator.
- A variable can change its value during the Program, by being assigned to a new value

Data Types.

variable need to have their data type, which define the type of the value the store.

We have seen the int data type in the previous example, used to store whole numbers:

```
1.  # include <stdio.h>
2.
3.  int main (){
4.  ....int score =85);
5.  ....
6.  ....printf ("score: % d", score);
7.
8.  ....return 0;
9.  }.
```

Decimals.

To store decimals (or floating point numbers, C provide the flot and double data types.

```
# include <stdio.h>

int main(){
    float height = 61.5;

    Printf("%f", height);

    return 0;
}
```

The format specifier for floats is %f.

→ The double data type is also used to store decimals.

```
# include <stdio.h>

int Main(){
    double temp = 32.64;

    Printf("%f", temp);

    return 0;
}
```

The formate specifier for double is %lf

float VS double

float uses less storage in the memory, but is not as precise as the double type.

This means that calculations that use floats are faster than the ones that use double, however, the results are less occurate in terms of the decimal digits.

- Generally, floats is sufficient for storing 7 decimal digits.
- While double can hold 15 decimal digits.

char.

The char type is used to store a single character. It is similar to declaring a string, but uses single quotes for the value:

```
1. #include <stdio.h>
2.
3. int main(){
4. ....char letter = 'x';
5.
6. ...printf("%c", letter);
7.
8. ...return 0;
9. }
```

- %c is the format specifier for the char type

Fill in the blanks to declare a letter and p variables, then output them using a format specifier

```
char letter = 'c';
int p = 3;
Printf("Letter %c has the position %d", letter, p);
```

Constants.

Variable can be defined as Constants. This means that they cannot be changed.

To define a constant, use the Const keyword. before the data type of the variable.

```
1. # include <stdio.h>
2.
3. int main() {
4. .... const int num = 42;
5. ....
6. .... Printf("%d", num);
7.
8. .... return 0;
9. }
```

Now num is a constant, and its value is read-only.

Open the code and try to reassign num a new value in a separate statement to see the error.

Constants must be assigned a value when declared.

```
1. # include <stdio.h>
2.
3. int main() {
4. .... Const int num;
5. .... num = 42;
6 ... printf("%d", num);
7.
8. ... return 0;
9. }
```

- int is used to hold whole numbers (integ
- float and double store decimals
- float is similar to double, but has less pre, and requires less memory.
- char holds a single character.
- The Const keyword is used to define a const which is a variable that cannot be changed

Doing Math.

You can use arithmetic operators to perform calc-tions.

Let's declare two variables and output their su

```
1. #include <stdio.h>
2.
3. int main () {
4. .... int x = 8;
5. .... int y = 4;
6. .... Printf ("%d", x+y);
7.
8. .... retur 0;
9. }
```

The result of a calculation can be assigned to another variable.

```
1. # include <stdio.h>
2.
3. int Main () {
4. .... int won = 8;
5. .... int lost = 3;
6. .... int score = won - lost;
```

```
7.  printf ("%d", score);
8.
9.  return 0;
10. }
```

```
int a = 2;
int b = 3;
int c = 1;
int x = a + b - c + a;
        2 + 3 - 1 + 2
        x = 6
```

Multiplication.

Multiplication is done using the * operation.

Ex:

```
1. #include <stdio.h>
2.
3. int main () {
4.     int width = 52;
5.     int height = 14;
6.
7.     int area = width * height;
8.     printf ("%d", area);
9.
10.    return 0;
11. }.
```

Division

The division operator performs differently depending on the data types of the operands. When both operands are integers, integer division, (also called truncated division), removes any remainder to result in an integer.

Ex : convert the given kilobytes value to megabyte

```
1. # include <stdio.h>
2.
3. int main () {
4.     int kb = 35800;
5.     int mb = kb/1024;
6.     printf ("%d", mb);
7.
8.     return 0;
9. }
```

- the result will be an integer.

Division.

In case we want a more precise result we can use floats in our division!

```
1. #include <stdio.h>
2.
3. int main() {
4.     float kb = 35800;
5.     float mb = kb/1024;
6.     printf ("%f", mb);
7.
8.     return 0;
9. }
```

what is the output of this code

```
int x = 9;
int y = 4;
printf (" %d", x/4);
```

→ 2

Remainder

The % remainder operator is used to find the remainder of a division. It also called Modulo.

Let's find out how many of 100 items will be left over if we divide them into boxes of 8:

```
1. # include <stdio.h>
2.
3. int main (){
4.        int items = 100;
5.        int per_box = 8;
6.        int left_over = items % per_box;
7.        Printf ("%d", left_over);
8.
9.        return 0;
10. }
```

Guess the output of this code:

```
int x = 9;
int y = 5;
Printf ("%d", x % y);
```

= 4

- + is addition
- - is subtraction
- / is division
- Dividing integers results in an integer, while dividing floats results in a float.
- % finds the remainder of a division.

Comments

A single-line comment starts with two forward slashes and continues until it reaches the end of the line.

Ex:

```
#include <stdio.h>

int main() {

    int age=42;
    Printf ("%d", age

    return 0;

}
```

Fill in the required symbols to make a comment

// this is some text.

Multi-line Comments:

If you need to comment out multiple lines, or write a long, multi-line comment, you can use the /* */ symbol like this:

/* and */

* // starts a single line comment.
* /* */ is used for multi-line comment

" C Programming "

Computer fundamentals:

* History & Evolution of Computer. ✓

1. Abbacus.
2. Pascalein
3. Mark - I
4. Napier's Bones
5. UNIVAC - I

Characteristics of Computer:- ✓

i) High processing Speed
ii) Accuracy
iii) Reliability
iv) versatility
v) Diligence.

Computer is electronic device - that is used for information processing it accepts Data & instructions, store its an memory; process and gives the resu to the user. The term Computer is derive from the lattric word compute which means to calcute or to manncular late.

i). High processing Speed: A Computer is an Extremly fast information processing device. It carris out all ~~Star~~ sorts of Computation within fruction of Second & Exicutes millions of instruction per Second

ii) Accuracy :- It gives accurant results for correct input Data, here Accuracy means the correctness of processed Data. if the input Data is Erroneous the output will not Correct.

iii) Reliability :- It gives Consitant results Every through it runs Electric Connection Eg Electronic Secur. Sercuits which are often prone to Errors.

iv) versatility :- Computers are use Everywhere be it Scaintific calculation bussiness processing. They are used in teatching training, Similation media Eg Entertainment Extc.

v) Diligence :- it doesn't feel tierd it is used for hours it can also used for number day & or ones non stop it will work Scatisfactrilly without fatigue.

Generations of computers; ✓

1. Non-Electronic generation -> zero generation Computer.
2. Electronic. -> i) First generation computers.
   ii) Second "
   iii) Third "
   iv) Fourth "
   v) Fifth "

Generation of computer : this Generation based on a technology used in Architucture developing Computers Gnrn of computers.

1. This are also reffered as the zero Generation Computer they were develop before the Semiconductor. took placed in 1946. They were made up of woodn't & mechanical Companis.

Ex:- Abbacus, Napier's, differential Engiens Extra.

2. Electronic Generation:-

i) I^st Generation Computer [1946-59] That were build with magnitudes. Their Speed is 10^{-3}.

ii) [1957-64]: mainly build in diagnozi & constriction. Their Speed 10^{-6}.

iii) [1965-70] were build integroted Secuits (ICS) Their Speed is 10-100 transistors on single Siliconchip.

iv) [1970-90] build with large Scale integration (LSI) -100-1000 transistors in single chip.

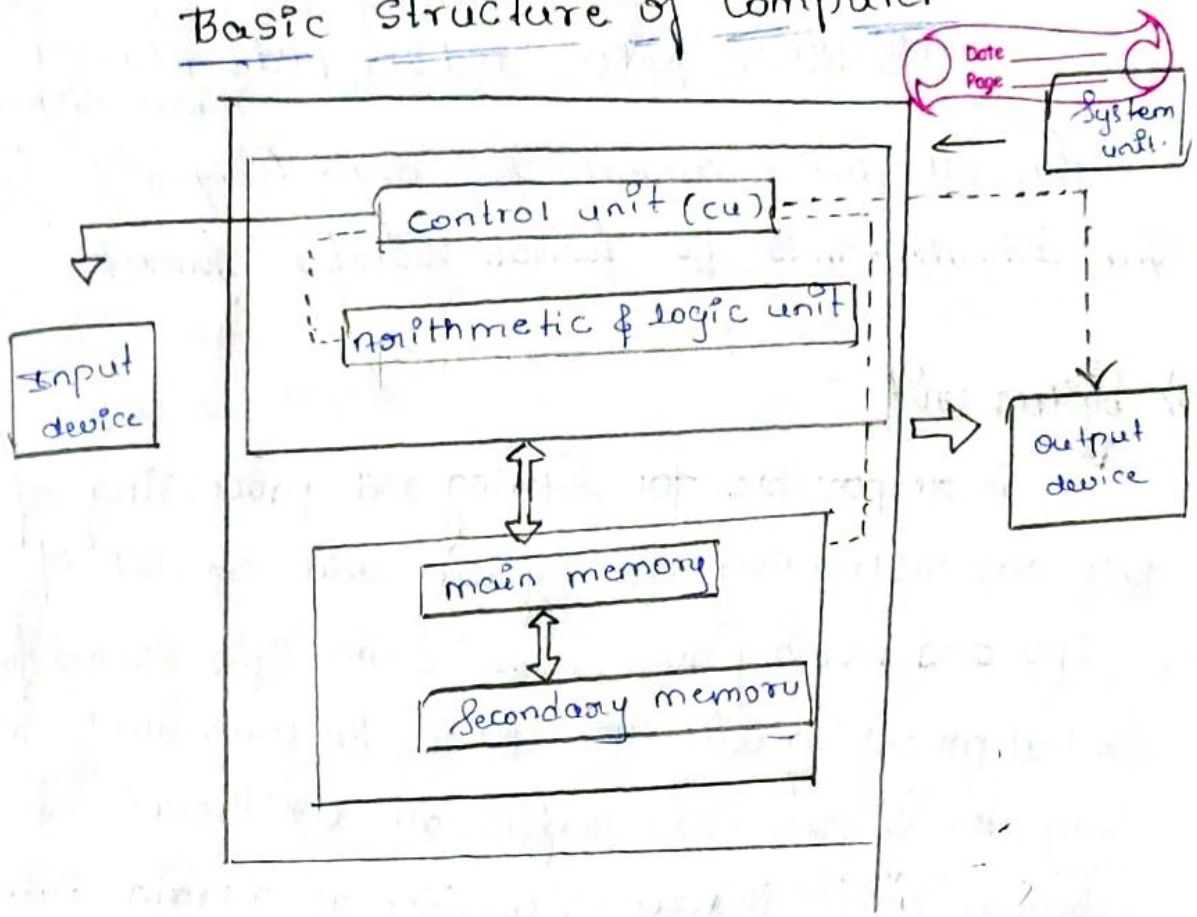
v) That are develop super large Skin.

Generation of Computer's chat:

Generation of Computer	Time Period	Evolved Hardware	Key characteristics.
I st generation	1946-59	Vaccum tubes	large scale, high power consumption limited memory.
II nd generation	1957-64	Transistors	Small size, increased reliability, reduced heat gnrn.
III rd generation	1965-70	Integrated circuits (ICS)	Further size reduction incoresed speed, impr-ed effecinus.
IV th generation	1970-90	LSI [large scale integra-tion]	Personal computer. increased processing power user friendly interferess
V th generation	present & beyond	AI hardware, neural network	Machine learning capabilites natural language processing

Von- neumon

Basic Structure of Computer.



- - - - - > Control

⟹ Data flow.

1) Input devices:- This is used to feed data and instruction into the Computer. It is conected to

Their are several types of input devices used for inputing data. the keyboard and mouse are the most well known input devices for entering data and navigating the System.

Ex: mouse, microphones, Scanners, web cames, joistics, discs, and touch panal.

2) Output devices:- Onece's the data & instruction's are process the user can chose to display such results on the output devices. the monitor is the primary output devices display user interprys, application & multimedia Content.

Ex:- Printers, Speaker, lcd's, PPd, (Plasma Panel display)

This all devices convert the digital signal's from the Computer into the human readable formate

3) System unit :-

Is responsible for Storing and processing of data and instruction the System unit consist of cpu and memory device the down Cpu Stand for Central processing unit the cpu is the main unit in Computer System It perform all arithmatic & logical unit It may be consider as a brain System unit.

4) Control unit :-

The control unit It controls and coordinates the activities of all the unit's of Computer System It ~~send~~ sends the control and timing signal ~~an~~ to variable units of computer to coordinates for &

the function of control unit fecting data and instruction of ~~from~~ the memory, interpreting the instruction and controling input & output devices, controling the transfer of data and instruction to and from the main memory

5) ALU (Arithmatic logical unit).

Performs all arithmetic operations such as addition, Substraction, multiplication, devision and modulus operation. It also performs the logical operation such as ~~&~~, ~~or~~, ~~not~~. And, OR, NO the logical operation results in either true or false value

6) memory devices:- are use to store the data and instruction fed by the user the computer memory is measured in terms of bits, bytes and words. bit is binary digit either 0 or 1. A bit is the basic unit of memory and is defined as sequence of 8 bits. A word can be defined as sequence of 16 bits or 2 bytes

The computer memory classified into,

- cpu resistor
- main memory
- Secondary memory
- cache memory

-> The cpu resistor hold a limited amount of membory during.

-> The main the is the primary memory which hold data and instruction supplyed by the user.

-> The Secondary memory is the ~~temprory~~ perminate memory which holds data and instruction as long as user can do.

- The Cache memory high Speed memory and Comes in between the CPU and the main memory.

=> Storage devices:-

- The Storage device it is very deficult to Enter the data instruction for large application again again.

∴ it nessory to Store such data and instruction permenetly for ~~data~~ ~~usage~~ Storage devices

~~Comparison~~ :-

memory devices	Storage devices.
- They Specifise the main memory of computer System.	- Secondary memory of a Computer System
- This are temporary memory; Data is lost when the Computer is lost	- Data is lost when the Computer is lost. - This are permanet memory Data is not lost after shut down.
- The memory is limited here	- The memory is unlimited here
- Faster excessing of data and instruction	- Slower Excessing of data instruction
- This are design on a base of Semiconductor technology	- This are design on base on magnetic and optical technology
Ex: Ram and ROM	Ex: magnetic disc floppy disc, hard

disk and optical disk like series and dvd's

Date
Page

memory devices

RAM. (Random access memory)

It is a read and write memory the data and instruction entered by the user are a fiel store in this memory. The ram chip is made up of metal oxide semiconductor. (MOS). The Ram is temporary memory because ~~poo~~ during power failure the ~~information~~ ~~store~~ ~~thing~~ will be deleted ∴ it is called volatile.

The capacity of Ram place and important role in in faster execution of program. the more Ram a computer has the more it can do and faster it executes specific tasks the computer memory is measured in terms of bits, kilo bits. (KB) mega bities (mb), gegha bits (GB), Tera bits (TB).

RDM. (Read only memory).

The Rom stores data and instructions permenty the information stored in it will not be lost even if the power is turn out. It stores manly the monitor programs and bios programs. The information stored in it can only be read but can't be modified.

PROM (Programable Read Only memory.

The contains of this memory are disided by the user. It is initialy Emty latter the user can Store program, data or any other information permanent in this memory. How ever this programs can't be erase one's they are return on to it.

EPROM (Erasseable programble read only memory)

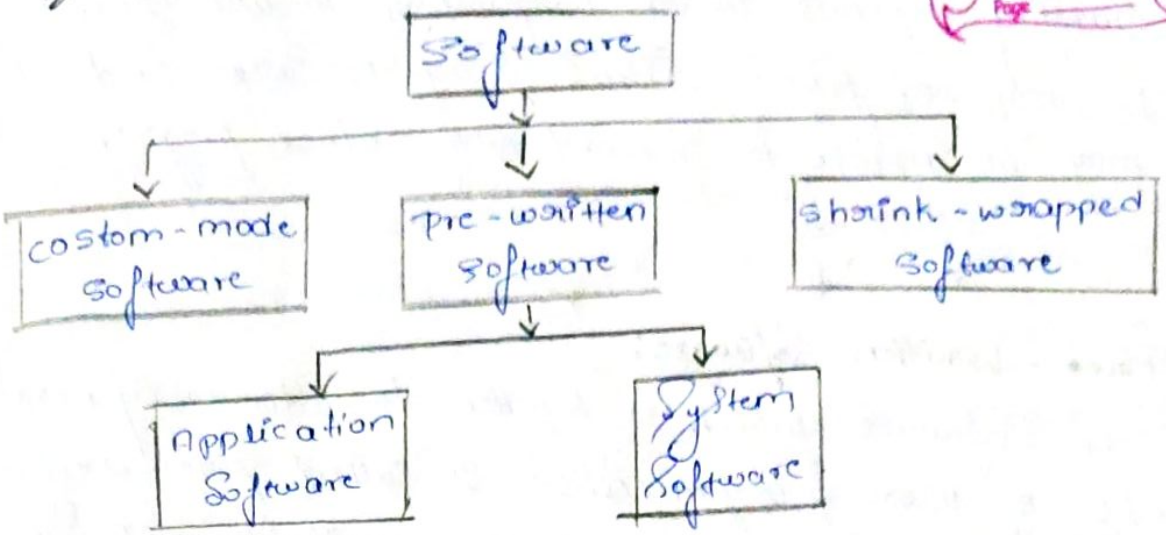
It is modification of PROM

The contain store in this can be Erased by Exposing it to and ultra violet light source for about 10-20 minites. then it is reprogramed that is a new information can be stored in it is most popular Prom in modern computer.

EEPROM (Electricaly Errosable programeble read only memory).

The contains store in this can be errosed Electrica latter new information stored in it.

Types of Software :-



Costam - mode Software :-

This Software which is written to address the Specific needs of an Enterprise or an individual is called a costom mode Software.

The Costam Software are much different from the Software which are already available in the market (Self Software also called). The main perpose of the Costam Software is to Solve the problems the clients is Encountering that the Exiting cannot fix it. This Software as used in ERP (Enter price resource planing) and the SAP (System application program)

Ex:- Airline reservation System for Specific airline Services. Hotel booking System for hotel change.

Advantages :-

* Completely meets the unique needs of an organization or an individual.
* high flexible to make changing market.
* less time to learn and use it effectively

Disadvantages:-

* Quality depends on the Company or an individual.
* It may not follow Standard photo calls. and it may be unable to interact with other program Effectively.

Pree - written Software:

A Software which is written to adress general needs of user & organization is called Pre-written Software this are readymade and available for Sale in the market. This are developed by the manufacturers.

There are 2 types of pre-written Softwares.

1. Application Software
2. System Software

1. Application Software:- This is general purpose Program collection of program written by the user to Solve a Specific problem. They help the user to Complete the given task quickly and efficiently this are also refferred to as the application Packages. These Software are user friendly and learnt easily

2. System Software:- This is a collection of a program written for Computer System management the System Software are developed by the Computer manufacturers and independent Software developers The System Software are supervisory program and helps in Executing the user's program Effectively.

There are 2 types of System Software.

1). Operating System 2). System utilities.

Operating System:- An operating System is an integrated collections of programs which make the Computer operational and help in exicuting user programs. It acts as an interface between its user an the Computer System. It manager the Computer System resources such as memory, processor, input, output, devices and files.

Ex:- windows xp, windows 7, unix, LINUX, etc.

Translating Program:- The programs which are used to translate the programs written in a high level ~~language~~ ~~to~~ program language to machine readable from are called as translating program. This Software are also called as the language processors.

1. **Compiler:-** This is a program which takes the entire high level language program as input and it translates it into machine language at a time.
   - All the errors that appear in the source program are leasted and displayed.
   - The process of converting high level program into machine language using the compiler is called compiler.

Ex:- Turbo C++, Java compiler, GNU Compiler.

2. **Interpreter:-** This is a program which translates one statement of high level language program into machine language, at a time and exicutes it. Here both translation & exication process hapens simeltinies. There is any error in the program it must be correct

Other it doesn't take the next Statement for translation & execution.

- The process of converting a high level language is called interpretation.

Ex:- basic c language interpeter, Java language interpreter.

3) Assembler :- This is a program which translates assembly level language program into machine level language

- The process of converting assembly level language is called assembling

Ex:- Turbo assembler (TASM)
micro assembler (MASM).

* System utilities :-

The programs that assist in loding & linking the object program (object code) are called System utilities the program in machine readable form is know as object program

Ex:- loders. linkers.

4) loders :-

A loder is a System program which lode the object program into the main memory for exicution. the operating System uses the loder to lode the programs from the Secondary Storage into the main memory & preporse for exicution.

2) linker:- linker is a program which links the specified sub programs or functions referenced in the main program for the execution of the program.

It takes one or more object modules that as generated by a compiler and combines then into a single executable code.

A linker program is also reper to as link editer.

3) Shrink wrapped Software:-

It is a free packaged software i.e., developed for selling it to general users and bussinesse. Shrink wrapped software comes with licens agriment (terms & condition) for use. If you by a shrinke warpped micro soft office with a single users you can only use it, install it on your dekstop or laptop but you can't share it.

And it is less expensive as compare to castome -r mad software.

and those are normally stored in CD ROMS - DVDS.

i) write an algorithm to compute area of circle?

→ Step 1:- Start.

Step 2:- Declare variables

Step 3:- Read the radkes a circle = 3.142 * r * r

Step 4:-

2) Algorithm for area of triangle.

Step 1 :- Start

Step 2 :- Read the three sides of triangle.

Step 3 :- Comput the a traingle Sqrt (

[S * (s-a) * (s-b) * (s-c)]

Step 4 :- Print area of traingle.

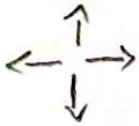
Step 5 :- Stop

4). Step 1 :- Start

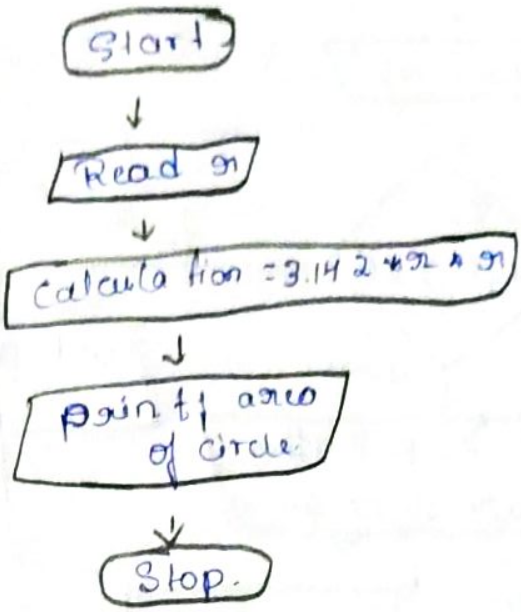
Step 2 :- Read the three largest number

Step 3 :- checking for condition.

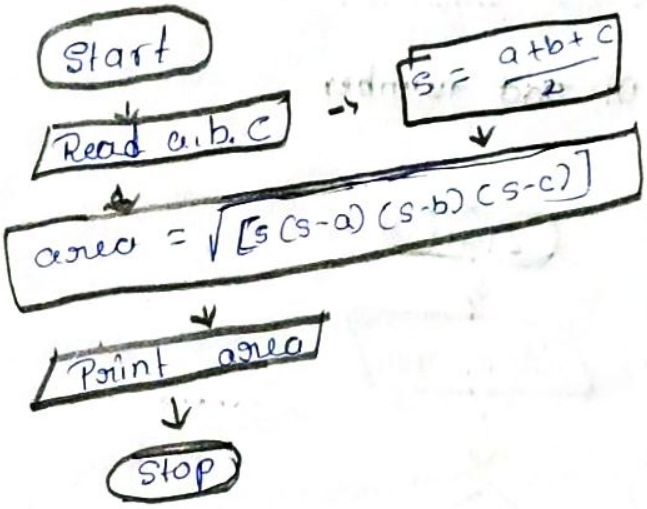
Flow chart ✓

Geometrical figure	Name	function
	Oval	Start and Sto
	Parallelo gram	Input and outpu
	Rectangle	Processing
	Diamond	Decision making
	Arrows	Connections
	Small circle	Continuation
	Hexagon	Repetition / looping

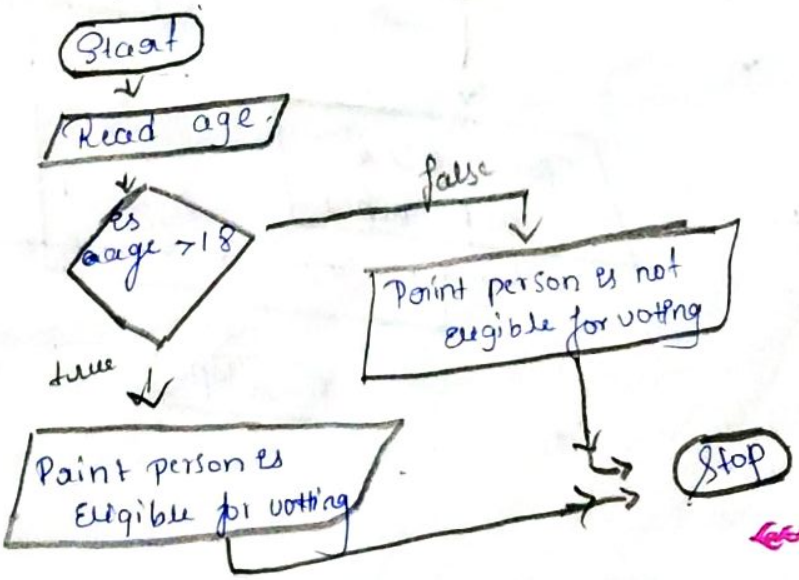
Area of circle.



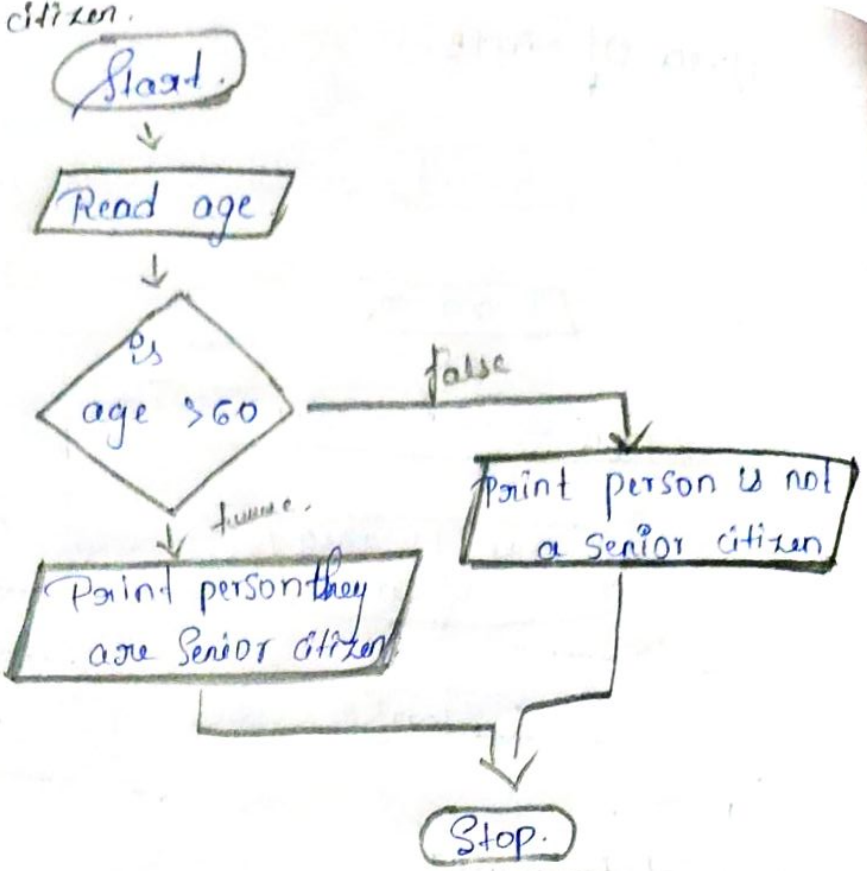
Area of triangle:-



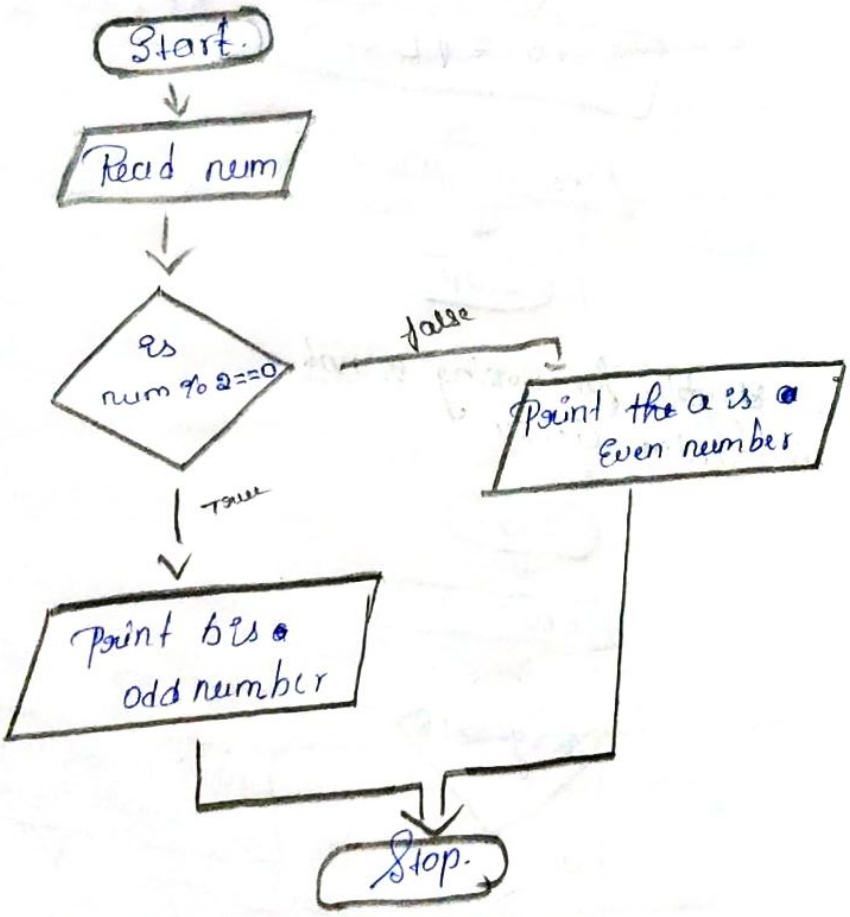
1) Eligible for voting or not
2) Senior citizen.



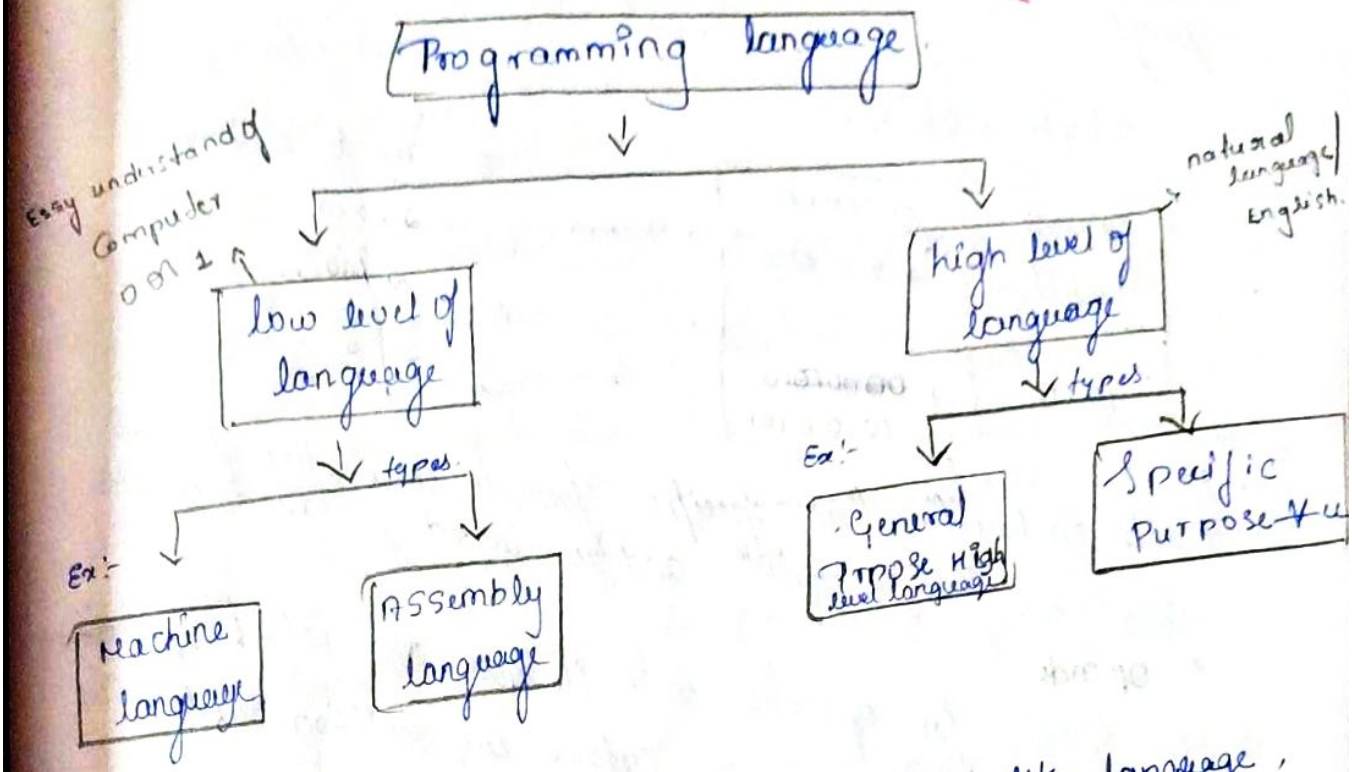
2). Senior citizen.



3) Even or odd number.



Computer languages:-



Computer does not understand English like language, It understands and process anything that in a form of 0's and 1's. The 0 and 1 are the alphabets in Computer readable languages.

Computer languages are classified into:-

1. low level programming language:-
2. high level programming language:-

- The low level languages are those languages which are easily understand by the machine.
- the high level language are those which are nearer to the natural languages such as English.

machine language:-

As a name of self implise programing is done at the machine level the vocabulary of a computer consiste only two word 0, 1. that is the digites of the binary number system. A zero indicates lower voltage level and

1 indicates high low

The machine level language is consider as the first generation language.

1 byte = 8 bits.

OP code	address
→ 4 bits ←	→ 8 bits ←
1010	~~00101010~~ 10100101

0 - low
1 - high

Binary to Decimal

2 → 10

2	10	
2	5	- 0
2	2	- 1
	1	- 0

Each instruction has specific formate consisting of two field the first is opcode and the second field is address

op code

It It stands for operation code it indicates what operation is to be perform the operation code be addition subtraction multiplication, store or halt.

address.

It → memory location where in the data is refrenced

→ the length of each instruction is 12 bites → 4 bits for opcode 8 bits → address

advantages :- ✓

→ Can be directly typed and exicuted

→ No translater program is required

→ disadvantages :- ✓

→ difficult to remember machine instruction

→ difficult to understant, modified and debug errors.

→ difficult to remember address of data instruction.

→ lack of flexibility → A program that runs on one machine does not run on other

Write a C-program to print sum of first 50 natural numbers.

```
# include <stdio.h>
int main ()
{
    int num, sum = 0;
    for (num = 1; num <= 50 ; num ++)
        sum = sum + num;
    Printf ("sum = %d\n", sum);
}
```

Write a c-program to print following patern

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

```
# include <stdio.h>
int main ()
{
    int i, j;
    char ch = '*';
    for (i=0 ; i<=5 ; i++)   outer for loop.
    {
        for (j=0; j<=i; j++)   inner for loop
        {
            Printf ("%c", ch);
        }
    } Printf ("\n");
}
```

15

```
1
12
123
1234
12345
```

```c
# include<stdio.h>

int main()
{
    int i, j;
    int rows = 5;
    for (i = 1; i <= rows; i++)
    {
      for (j = 1; j <= i; j++)
      {
        Printf ("%d", j);
      }
      Printf ("\n");
    }
    return 0;
}
```

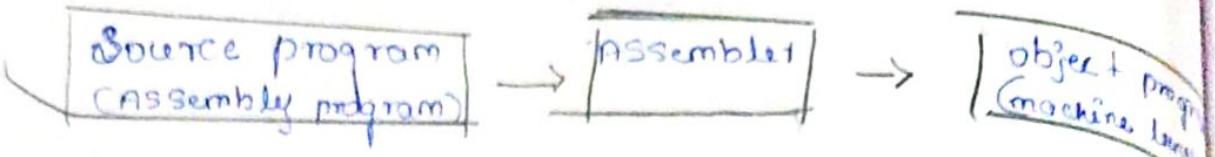
2)

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

```
# include <stdio.h>

int main()
{
    int i,j;
    int rows = 5;
    for (i = rows; i >= 1; i--)
    {
      for (j = 1; j <= i; j++)
      {
        Printf ("*");
      }
      Printf ("\n");
    }
    return 0;
}
```

2) Assembly language :- (symbolic in



To over come the draw backs of the machine level language Computer Enginions developed new programming language it's use the symbolic instruction.

This symbolic instruction orientent programing language is called assembly level language and this languges considered as the second generation programming language.

Symbolic words used in this languges are referred in mnemonics. The meaning of mnemonic memory seek and storage.

ADD :- for addition.

SUB :- for Substraction

MUL :- for Multiplication

STA :- for store at accumulator

HALT :- for Halt

JMP :- for Jump

INR :- for Increment and so on,

Ex :- ADD x, y STZ.

The above figure shows translation process of the assembler. The assembly ~~pr~~ language program that is given as input to an assembler is called the source program and output of the assembler called

advantages:-

1) Easy to remember the operation codes because they are mnemonic

2) Easy to understand the program.

3) Easy to write program, modified and debugg has compared to machin language.

4) Need not remember address of operands and instruction locations.

Disadvantages:-

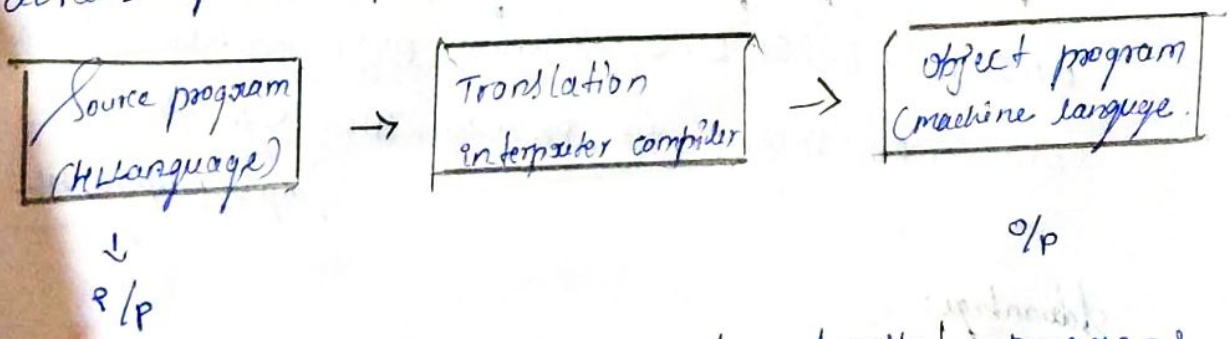
1) The mnemonics are machine dependent. They defer from one computer to another.

2) Not Standerdize.

3) less efficient than machine language.

High level language:-

The high level language is the computer engineers though of developing third generation programming language which are more related to problems to the machine. Programing language that are more concern with the problem specification and not oriented towards the details of a computer are called as high level languages



The Compiler and interpriter are two translator programs that are used to translate the high level language program into machine language.

Near the program written in high level language is called Source program. The corresponding program after translation either by Compiler or interpreter is called Object program.

*** Different between Compiler Interpreter

Compilers	Interpreter
It takes the entery high level language program as an input and translate it into machine language.	It takes one statements of a high level language program as an input and translate it into machine language and executes it.
All errors that occur in the program are listed and displayed.	Errors that occur only in the statement being taken for translation are displayed.
debugging is faster	debugging lower
Compiler require more memory	interpritor require low memory
Compiler costly exe	interpreter cheaper

Classifications of high level languages:-

- BASIC, PASCAL, C -> General purpose HLL
- COBOL, FORTRON, C++, PROLOG, JAVA, C# -> Specific purpose HLL

Advantage:

- Easy to read and
- Easy to write and modified
- They machine independent language.

- Standardize langugs.
- Portability {run on different machines with little and modification}

disadvantage:-
- less efficient and slower to the low level languges

* * * difference between machine level langugs high level lang

machin level langugs	high level langugs.
It uses only two symbols 0 and 1.	It uses alphabets, digits functuations and some special ~~langugs~~ symbols
Each instruction is a sequance of 0 and 1's	Each instruction is a English like statement
MLL can be directly exciguted	HLL can be not directly exciguted
MLL machin dependent	HLL independent.
deficult to understand modified, right and dibugg	Easy to understand and modified, right and dibugg.
High efficient (fast)	low efficient (low)
not standerdise	standerdise
machine level called first generation langugs	high level called third generation langugs

*** Difference between assembly & compilation.

assembly	compilation.
mnemonic or Symbolic Instruction.	English like Statement.
machin dependent	machine Independent
It is not directly executed first it should be translated into machine language by the process called assembly.	It is also not directly executed it should be translated into machine language by the process called compiler
Assembler used in translation Process.	interpreter and compiler are used to translation process.
This is converted to machine readable for in one step.	It is first converted into machine assembly language and then into machine language. Hence it takes two steps.

Psudo code:-

Psudo code is neither an algorithm nor a program it is an obstract form of a program in a psudo code the program is represented in interms of words and precess, but the syntax of program is not strictly followed.

- It is easy to read, easy to easy to modify

Ex: write a psudo code to perform basic arithmatic operations

read n_1, n_2
sum = $n_1 + n_2$
diff = $n_1 - n_2$
multi = $n_1 * n_2$
div = n_1 / n_2
Print sum, diff, multi, div. End
End.

Important Codes for 1st unit.

2M.
→ 1) what is Software mention the types. -5M.
2) list the characteristics of computer
3) differentiat machine language and high level language
4) define compiler and loder.

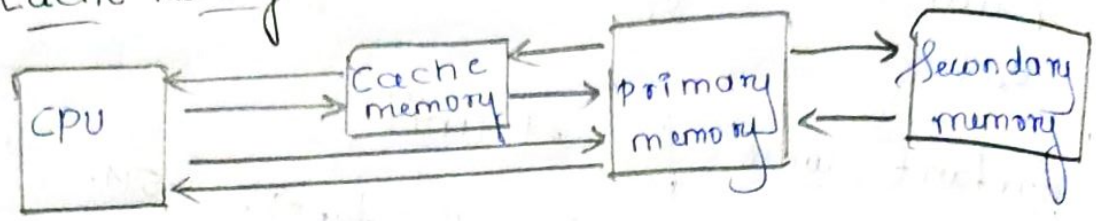
5M
1) Explain Scanf and printf with
2) Explain the generation of computer

10M
1) Draw & explain the block diagram of computer.

Extra.
1) Explain the Storage devices and input devices
2) what is ram? what are the feature of using RAM.
3) Explain the following ROM, PROM, EPROM, EEPROM and chace memory.
4) Explain the types of computer languages.
5) what is an algorithm explain the characteristics of algorithm with ex.
6) ~~write~~ draw a flow charat for largest of three numbers
7) what is Psudo code

8) differentiat between compaiser and interpreator.
9) differentiat between machine level and high level
10) differentiat between assembly level and high level lang
11) Explain the cache memory.

Cache memory.



Cache memory is small fast memory storge with in computer it holds dublicates of data from commnly excess location in the main memory

- the most important use of cache memory is that it is used to reduce the average time the excess ~~time~~ data from the main memory.

How cache memory works.

1) cheaks cache first - by the cpu needs data or and ~~cach~~ instruction it first check the cache.

2) cache his:- If the data is found in the cache the cpu can excess it very quickly by including performance

3) cache miss:- If the data is not inter cache the cpu features it from the main memory & a copy is than placed in the cache for feauture use

Characteristics:-

1) high speed. cache memory is much faster than main memory (RAM)

2) Small capacity :- It has a limited storage capacity as compared to main memory

> Volatility:- Cache memory is a volatile it loses it when the power is turned out.

> The primary purpose of cache memory is to acts as a buffer to sincronise the high speed cpu with the slower main memory.

Unit II :-

Introduction to C.

C develop dennis ritche and brain kernighan. C is high level language it is both the general purpose and the specific language program. (Bell laboratory, telephone 1972)
application program.

characteristics of C.

-> C is general purpose programing language
-> Structured programming language.
-> Hence in the development of System software.
-> It has a rich set of operators.
-> It has the pointer arithmatic and pointer manufulatio
-> portability any C-program can be run on different machines with little or none modification.
-> No rigidi formate, any number of statement can be typed in single line.

Application of C.

System Software (operating system, interpreter, compailers, assembler, editors, loders, linkers

Application Software.

- Data base manganment System. (storing data)
- Grafic pacakges
- Spread Sheet
- chat/cham application
- world process
- Office automation tooles
- Signtific and Engiriaring applications

features of C:

1) Simplicity and efficiency
2) Portability.
3) Structure programming.
4). Modularity.
5) Data types and operators.
6) Memory Mangement.

Basic structure of C program.

PreProcessor statements. -> Header files #
global declarations: -> bubble main()
main ()
{
declarations;
statements;
}
user defined function.

1) Preprocessor statement
This statements begin with # symobe and also called Pre-processor directive
include -> This is used to include contains of stdio.h> it includes the standard input, output head file.
Ex:- # include
define :- It is macrose that can be used through out the court Ex:- # define PI = 3.142.

2) Global declaration.
1) "Global variables" -> variable declared outside of any function they are accessable from any problem.

2) Global constant.
Constant declar out side any function similar to ~~any function~~ global variable but immutable.

3) The main

As name itself indicates this is main function execution of C-program starts with main. The function main should be written in lower case letters and should not be terminated by a semicolon.

4) brees.

C-program uses pay of curly bress ({})

4) Declaration

The declation is the part of C-program where all the variable arrays, functions etc. This all used in C-program are declared and may be initialized are

5) Statements:-

This are the instruction to the computer to perform for specific. They may be input output statement arithmatic statements. Control statement and other statements they also includes comments.

6) User defined functions:-

The user defined functions contain set of statements to perfor specific task this are the return by the user. hence the name user define the function.

Ex:-
```
# include<stdio.h>
int main ()
{ Printf ("Hello word");
  return 0;
}
```

Character set of C:-

1) Alphabets
2) Digits
3) Special character +, -, *, #, @
4) Escape sequences \n, \t
5) Comments [/* ... */]

C - Tokens:-

1) Keywords -> return, void, break, continue, static.
2) Identifiers -> variables.
3) Constants -> const. PI = 3.142.
4) Strings
5) Operators -> arithmatic, logical, comprasion.
6) special Symbol -> #, @.

define
C-Tokens is the basic and the smallest units of C-program are called C-tokens. There are 6-types of tokens in a C-keywords, Identifiers, Constant, strings, operators, Special Symbol.

1) Keywords

Every word in a C-program is a keyword or a identifire all keywords are basicaly the sequence of character that have one or more fixed meaning. all C-keywords must be written in lower case letters.

Ex:- break, char, const, continue, double, float, for, return, static, switch, struct, union, while, void.

2) Identifiers.

Identifiers are the names given to the program elements such as variables, arrays and functions. Identifire are the sequence of alphabets and digits.

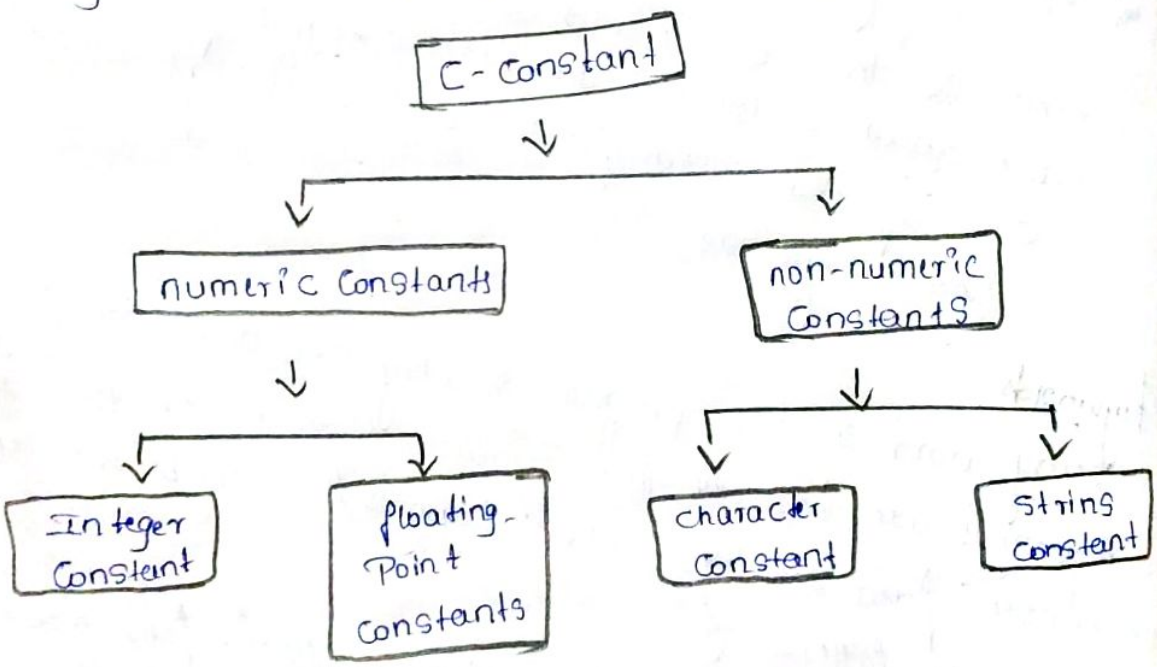
3) Constants :-

Rules of naming Identifiers.

* The first character must be an alphabet or underscore
* All subsidic character ~~are~~ must be either letters or digit
* Keywords should not be used as a identifiers
* No special character are punctuation symbols are allow except the underscore ['_']
* No two successive underscores are not allowed.

3) Constant:-

Constants are fixed values that can't be changed during the execution of program



The quantity which does not change during a execution of a program is known as program.

There are two types of constants in C 1) numeric Constants
2) non-numeric constants

1) numeric Constants.

It has a two types there is

a) integer constant

b) floating point Constant

a). Integer Constant is a whole number it is a Sequence of digits without a decimal point. It has prefixed with + or -

The general for of integer Constant is

sign	digits

sign -> + plus for +ve numbers & minus for -ve numbers.

digits -> a sequence of digits.

for Ex :- +342 , -769 , 999.

b) Floating Constant it a number with decimal Point and it is defined as Sequence of digits presided and followed by decimal point.

The general for of floating point Constant is

sign	integer part	decimal Point	fractional Part

Sign -> optional + or -

integer -> Sequence of digit before a decimal point.

decimal point -> Period Symbol.

fractional Part -> Sequence of digits after a decimal point.

for Ex :- -246.01 , 0.0 , 0.0005 , -0.159.

2) Non - numeric Constants:

a) Character Constant:-
It is single character enclosed within a pair of apostrophes
Ex :- 'a', '9', '#', ':'

b) String Constant:
It is a sequence of character enclosed within a pair of double codes
Ex :- "Hi", "x+5", "Hello".

* The quantity that change during the execution of the program called as variable * the variables are the Names given to identity for the specific
* → Variable are also called identifiers.
→ the variable represent a particular memory location where data ~~store~~ can be store they are use to describ, constant, functions, arrays, failes of structures etc.

Rules for Programming variables.
1) first character of a variable name
2) must be an alphabet or alphabet score.
3) all subsiding character consider as latter & digits
3) keywords should not be used as a variable
4) special character are not variables
5) variables name should be in both upper case and lower case

→ Declaration of variables.

All the variables must be declarade before they are used. ~~The before declara~~ in c-program ~~part~~ ~~part~~ ~~part~~ ~~part~~ The declaration made in declaration part specified the variable declare outside the main function consider as a top globle declaration where as within a main function is ~~decla~~ consider as local declaration.

→ General form of declaration.

Datatype	variablelist	Semicolon

↓ is a basic data types such as int, flot, double char.

→ One or more variables of ~~d~~ data types and this variable must be spegrated by a variable

→ of delimiter of this declaration.

Ex :- int a,b;
float x,y,z;
char c;
int p,q,r,s;

→ assigning values to variables.

The variable represent sum memory location where the data is stored each variable associated with one or more values, the process of giving values to variables is called the assignment of values.

The assignement operator "=" is use to a value to a variable.

Variable_name	=	value	;

Ex :- int x=10; float a=39.49;
char ch = "xyz"

Data types:-

1. Built-in data type
2) Derived data type
3) user defined data type.

1) Built-in data type
- a) Integer [2 bytes]
- b) floating point or real data type [4 bytes]
- c) double data type [8 bytes]
- d) character. [1 byte] -> 8 bits.

Define

Data type indecates, the type of data that variable can hold the data may be numeric or non-numeric in nature

1) a) integer:- This is sequence of digits without a decimal point an the keyword int is used. The range of int depends on the word length of the computer Ex:- 1.2

b) float:- The floating point numbers as the same as real numbers. and the keyword use it float.

ex:- 1.2, 3.4

c) character:- char is the keyword for this use to indicate the character type data. the data may be character constant or string constant.

Ex:- x, y, z.

- The character type data takes one bite size (8 bits) for storage.
- Each character has integer value this value given by the ASCII [American Standard code for Information interchange]

'A' - 65 Z = 99.

0 - 97.

'b' - 66 z = 122

* The character constant may be signed or unsigned the signed character have values in
-> range -128 to +127
-> unsigned -> 0 to 255

* double data type:- The keyword use to indicat
- e a double precision floating point number.
The precision associate with the accuracy data.
a double value is normal float value the
numbe of significant digits, data stron
-d the after the decimal point double point value
Ex:- 24.56.1056 00 000

The blak slash constant is the combination of
two characters in ~~which~~ wich the first chard
-cter is always the balack slash.
* And the characte can be
'\n' - New line (line feed)
'\t' - Horizontal tab (fixed set of space)
'\v' - vertical tab
'\0' - Null character (end of strings)
'\b' - back space

Symbolic Constants:- is a name that substitutes the
numeric constants, character constant, string constants.
Ex:- area = 3.142 * radius * radius
In above formula 3.142 replaced by PI
variable
area = PI * radius * radius.
# define Name = "Anu"
# defined PI = 3.142

Type Modifier	size (bytes)	Range of Values
int	2	-32768 to +32767
signed int	2	-32768 to +32767
unsigned int	2	0 to 65535
short int	2	-32768 to +32767
float	4	-3.4E +48 to -2147483648
double	8	-1.7E +308 to +1.7E-308
char	1	-128 to 127
signed	1	-128 to +127
unsigned	1	0 to 255

Formated input and formated output

Input and output function.

1) Scanf()
2) Printf()
3) getchar()
4) Put char()
5) gets.
6) puts
7) getch()
8) getche()

two types of input-output (I/O) functions.

1) formated I/O function.
2) unformated I/O function.

Type.	Input function(s)	output function(s)
Formatted	Scanf()	Printf()
unformatted	getch()	Putchar()
	gets()	Puts()

1) Formated I/O function (Syntax).

Scanf ("%d", &n);

↓ Controlstring ↘ Address list

Scanf ("control String", address list);

C provides standared liberary of a functions for inputt output operations and it's denoted by <stdio.h>. which also called as the Header file.

Formated Input:-

C provides function called as scanf to read the values of variables using a keyword. It's used to enter numberic, character & string type of data. It's Included in <stdio.h> Header file.

Ex:- sf ("%d", &n);

sf ("%d %f %c", &n, &4, &2);

sf ("%g", &n);

Control string

It's a sequence of one or more character group. Each group is a combination of % symbol and string. specifise the type of the value which are supplied to the variable.

Address list.

It's specific the address of the memory location where the values of input variables should be store.

Rules.

1) The control String must be enclosed with an a double code.

2) Each character should be beging with %, follo... d conversion character.

3) Address list contain address of input variable.

4) and the address list cannot enclosed within a double codes.

5) Each input variable in the address list must be presided by an ampersand (&).

Character groups.

%c -> read a single character.

%d -> read a decimal integer

%f -> read a floating point number

%s -> read a string

%p -> read a pointer

%o -> read an octal number

C provides Printf () function to display data on the standard output device. and printf () included in the <stdio.h> header file

Printf ("control String ", varlist);

Printf ("%d", a[i]);

Rules:-

* The control string must be enclosed within an the double code.
* character string.
* The variable is does not contain double code.
*

Formatted I/O.

Scanf()

* integer Input
* floating point i/p
  %f
* character input
  %c

unformatted functions:

1. Input functions
2. Output functions.

unformated input function:-

These functions are chosen with reading the character type data from the keyboard.

getchar & get function are used in

1. Getchar():- These function reads a single character from the standard input devices there is no parameter in the parenthesis {}

Sintex:- ch_var = getchar();

Ex:- int main()

{ ch letter;

letter getchar();

}

2) Gets ( ) :- This function reads in everything you Enter from the keyboard until the Enter key or return is rest it will except sequence of characte of type char string.

Syntax :- gets (string).

unformated output function :-

Putchar( ) function ->

Puts ( ) function ->

These functions are mainly ~~consen~~ chosen with displaying are printing character type data on a pointer.

1) Putchar ( ) :- These function prints a single charac-er on a screen on the display is of type.

Syntax :-

Putchar (ch_var).

Ex :-

```
{ ~~cho~~
  char letter;
  letter getchar ();
  Putchar (letter);
```

These functions prints a string of character of a screen.

The new line character that signles the end of the string will not be.

Syntax :- puts (string)

Ex :- {char msg [20]

Pf("Enter the message"),

gets (msg);

puts (msg);

Operators

Binary Operators.

- Arithmetic operator
- Logical operators
- Relational operator
- Bitwise operators

1) Arithmetic Operators:-

Arithmetic operators are used to perform basic arithmatic operations. Such as (+), (-), (*) (/).

Operations	operator	Procedence	Associativity
Addition	+	2	L to R.
Substraction	-	2	L to R
Multiplication	*	1	L to R
Division	/	1	L to R
Modulus	%	1	L to R.

Here lowest number indicates the highes priority and the highest number indicates the lowest priority associivity indicate the order in which the terms in the Expression are Evaluet that is left to right right to left.

Ex:- int x,y ; x = 15 , y = 19 ..

then x+y = ?	x-y	x*y	x/y
= 15+19	= 15-19	= 15*19	15/19
= 34	= 4	= 285	= 0.71

An Expression involving arithmatic operation called arithmatic operators. It takes one or more operents and connects them by an arithmatic operators.

Mathematical Expression	Equivalent expressions
$a+b$	$a+b$
$\frac{x+y}{2}$	$(x+y)/2$
$\frac{(a+b)^2}{(a-b)^2}$	$((a+b)*(a+b))/((a-b)*(a-b))$ or $pow((a+b),2)/pow((a-b),2)$
$\sqrt{a^2+b^2}$	$sqrt(a*a+b*b)$

Rules for ~~arithmatic~~ Evaluation of

1) ~~If the given expression~~ Evolves paranthesis, The expression must be evaluated. The priority apply to the unparenthis.

1) If the given Expression Evolves paranthisis. the expression as applied to the an paran thase. paranthesis must be evaluated first. The priority as same

2). If a unary minuse is a present in the Expression, then the term associated with the unary minus. must be evaluated before the other Expressions.

1) write a c-program evaluate basic arithmatic operations.

```
# include<stdio.h>
int main()
{
    int a,b, sum, sub, multi, divition
    Printf("Enter ~~adding~~ two ~~numbers~~ values");
    Scanf("%d", &a, &b).
    sum = a+b
    sub = a-b
    multi = a*b
    divis = a/b
```

```
Printf (" Display the values");
return 0;
}
```

output :-

adding two values = a+b
= 20+30
= 50

Substracting two values = a-b
= 20-30
= -10

multiplying " " = a*b
= 20*30
= 600

division " " = a/b
= 20/30
= 1.5

Increment operator.

P.

That is the value of intvar that the value intvar must be increment before it is used (increment and use).

intvar ++

it indicates post increment that is use the value of intvariable first and then increment it (use and increment)

Decrement operator:-

This operator is use to reduce the value of an ~~increm~~ integer by one. This is represented by --.

This symbol can be place either before or after an integer variable.

Ex:- A = 6 b = 8

--A = 5

b-- = 8

x = --a a = a-1 a = b	a = b-- a = b b = b-1

decrement the value of int variable first use it

int var -- indicates post decrement.

use the value of decrement and then decrement it.

Ex a = 5 a-- = 4

b = 6 b--b = 6

4 = 5 - 5 -

a = 5 - 6

a = b - 1

6 = --5

6 = 6 - 5

= 1

Relational operators (= <, >, !)

This are used to compare two operants given in an expression. They define the relationship that exists between two constants and variable. There are in either true or false.

There are 6 oper relational operators in C.

Operators	meaning	procedure	Associativity L to R
<	lesser than	1	1
<=	lesser than equal to	1	1
>	greater than	1	1
>=	greater than equal to	1	1
==	Equal to	2	1
!=	not equal to	2	1

Ex:- * (a > b)
* (x == y)
* (number != 100)
* (salary == 18000.00)
* (age < 25)

Relation expression can be defined as combination of operants and relation operators. The operants may be constant or variables.

~~by evaluating~~ relation between the first four operator priority is 1 and their other priority are two. (2)

logical operators:-

This logical operators as use to make desitions
there are three such operatore
they are and, or, not
The first two operators are Binary operators.
(and last one is unary operators.
The results of this operaturer es. either true or false

Operators	meaning	Precedence	Associativity
&& (double amperesent)	logical AND	2	L to R.
|| (double Pipeline)	logical OR	3	L to R.
! (Exclamanta-tion)	logical Not	1	L to R

logical AND.

operand 1	operand 2	operand 1 && operand 2
false	false	false
false	true	false
true	false	false
true	true	true.

logical Not.

operand not 1	Not 2
true	false
false	true

Evaluation of logical expressions:-

Expression involving the logical operators are called logical expression.

The result of logical expression is either true or false.

Ex: ~~a&&b||c&&~~

a&&b||c&&(!b) → by sub value = 2&&4||3&&(
= 2&&4||3&&
= 1||3&&0
= 1||0
= 1

~~int a~~

C-program to perform logical Anding and oring operation.

```
{
  int a,b,c.
  a=5, b=6;
  c=a&&b.
  b=a||b||c;
  a=a&&b||c;
  Printf("%d,%d,%d", a,b,c);
}
```

o/p: 1, 1, 1.

We can use the logical operators with the relational operations above ex is use of logical operators used with relational expression.

Bitwise Operators:

Operator	Symbole name	Meaning
&	Ampersand	Bitwise AND
|	Pipeline	Bitwise OR.
^	caret	Exclusive - OR (XOR)
~	Tilde	1's Compliment
<<	Double less Then	left shifting bits.
>>	Double greater then	Right shifting bits.

1) AND

a -> 4 = 0000 0100
b -> 3 = 0000 0011
a & b = 0000 0000

2) OR.

a = 0000 0100
b = 0000 0011
a | b = 0000 0111

3) XOR

a = 0000 0100
b = 0000 0011
a ^ b = 0000 0111

4) Compliment:-

a = 0000 0110
~a = 1111 1001

1's Complement

b = ~a
= ~(1010)
= 0101

Bitwise operators in this operators in this all data items are stored in the Computer's memory as a sequence of bits (0's and 1's) manufulation of indisual bits is carried out in a machine language or assembly level language. To perform the bitwise operators ~~ther are 6 operators~~ C- provide 6 - bitwise operators this operators work with int and char type of data. The bitwise operator and", OR, and XOR are binary operators and produce the result depending on the comparison of

corresponding bit in each of the operantes

* The result of bitwise Anding is a one when the both the bits are one. Other wise it is zero.

* The result of bitwise oring is one if one of the bits is one and the other is zero. other wise it is zero.

* The result of bitwise xoring is one if the bits are different (1 and 0) otherwise it is zero.

* The bitwise Complement operator is an unary operators that reverses the state of each bit within an integer or a character.

* The left Shift operators Shifts the bits towards the left and the right Shift operator Shift the bits towards the right.

Ex:- For 8 bits.

Bitwise left shift:-

Before shift

1 0 0 0 0 1 0 0

After shift 0 0 0 0 1 0 0 0

Bitwise Right Shift

Before Shift

1 0 0 0 0 1 0 0

After Shift. 0 1 0 0 0 0 1 0

Write a C program to perform bitwise operators:

```
{
    unsigned int x,y;
      x = 128;
      y = 32;
      x = x >> 1;
      Printf ("After right shifting, by 1, x = %d", x);
      y = y << 2;
      Printf ("After left shifting by 2, y = %d", y);

}
```

Conditional operators:-

<Expression> ? <value 1> : <value 2>;

Expression -> relational expression.

value1 -> value to assigned when the result of expression is true.

value 2 -> value to be assigned when the result of expression is false.

Ex:- c = a<b ? a : b;

This is used to test the relationship between two variables. It is used in conjuction with the relational Expressions. The Conditional operators takes three operends the Symbol ? denotes as conditional operations

C-program to perform conditional operator:

```
{ int a=4, b=5, res1, res2 ;
  res1 = a>b ? a : b;
  Printf ("the result = %d\n", res 1);
  result2 = a<b? a : b;
  Printf ("the res 2 = %d\n", res 2);
}
```

res 2 : a<b ? a : b

: 4 < 5 ? 4 : 5

res 2 = 4

Special operators of C:-

1) comma operator
2) size of ( ) operator
3) Address operator
4) Dereferencing operator
5) Dot operator
6) Arrow operator.

1) Comma operator

The comma operator is basically associated with the for Statement. It is also used to link two or more related Expression together in such cases. the expressions are evaluated in left to right fasion.

Ex: C program to perform comma operator.

```
{
    int a, b, temp;
    temp = (a=20, a+10);
    Printf("Enter two numbers);
    Scanf("%d%d", &a, &b);
```

size of operator:-

The size of operator return's the size (number of bits) of an operend. It is size of operators it is function and therefor has to be return in lower case letters. It's must precied in ~~TI~~ The operand may be constant or variable or a data type. It is also used for dynamic memory alcastion.

The syntax of size of operator is

size of (operand);

C-program to perform size of operator:-

```
{ int x;
  float y;
  char ch = 'y';
    x = 10;
    y = 100.0;
  Printf ("size of x = %d \n", size of (x));
  Printf ("size of y = %d \n", size of (y));
  Printf ("size of ch = %d \n", size of (ch));
  Printf ("size of double = %d \n", size of (double));

}
```

output:-

x = 2
y = 4
ch = 1
double = 8.