Introduction to C

History of C —

C was developed by a System programmer Dennis Ritchie in 1972, at American Telegraph & Telecommunication (AT&T) Bell Laboratories in New Jersey USA.

- C is actually a symbolic instruction code, a set of commands that perform actions on a computer.
- The C language is often referred as middle level lang. bec. we can write HL pr. as well LL programs through C.

(1) fortron → formula translation
(2) COBOL → Common Business Oriented Language
(3) BASIC → Beginners all purpose Symbolic instruction Code.

What is C?

02 JANUARY 2014 THURSDAY

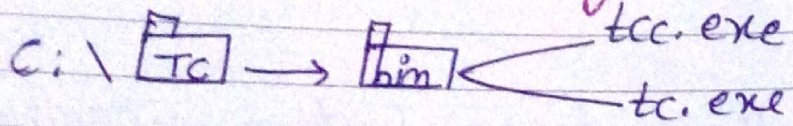
- C is a programming language used to write program.
- Programs are the set of instructions given by a programmer to the computer in high level lang.
- C uses compiler to translate high level program into machine code before executing any instructions.
- The original high level program is called the Source Code (.c) and the resulting language program is called the object program (.obj).

\n → new line character
getch.h → get character → take input from user

Scanned by CamScanner

TATA HITACHI

C- Programming



✡ Steps to develop 'c' programming –

- Open 'C' editor
- Type instructions (Source code)
- file click → new → blue screen will appear
- Press Alt + f9 for compiling
- ctrl + f9 to run the program
- Black window will appear which is called output window.
- To view again output window Alt + F5
- Save extension of C file is .c.
- function is a group of logically related statement.
- Program is a group of instructions.
- Main function is compulsory without main function program will not run.

function

user define function → Statements are made by us

Library functions → which stored in computer

Call → use हिंदी में

Header files which have pre-defined or library function.

Scanned by CamScanner

TATA HITACHI DIARY / FROM 1 BACK

Header files

Math.h — Pow ( ), Sqrt ( )

Stdio.h — Printf ( ), Scanf ( )

Conio.h — Getch ( ), Clrscr ( )

String.h — Strcmp ( ), Strcpy ( )

✡ Structure of Source code —

- Grouping must be done in parenthesis.
- Name of the function written above.
- In parenthesis, statements are written
- In C function / procedure / Sub program / method / Sub routine

07 JANUARY 2014 TUESDAY

function → function is a collection of logically related statements.

- A program have multiple function / function.
- To run a program main ( ) should be declared.
- first statement of main function executed firstly last statement in the end.
- To call a function. e.g. perimeter use typ dynta
- If we want to use library functions then we have to include it in programs.
- for example → # include < Stdio.h >
- Statement → function → program → Software

✡ Merits of C →

1.) C is a general purpose programming language you can generate games, business software, utilities, mathematical models, word processors, spreadsheets and other kinds of S/W.

Scanned by CamScanner

08 JANUARY 2014 WEDNESDAY TATA HITACHI

- C is a structural programming language. It uses structured statements such as while, for loops in place of goto statements which cause bugs (errors) in the program.
- System independence → C does not require any services from the operating system, it runs independently. C can run on any operating system.
- High efficiency - C compilers are generally able to translate source code into efficient machine instructions. C language data and control mechanisms as well match to most small computers and Microcomputers.
- System programing - C is used for System programming i.e. writing operating systems. The Unix operating is also rewritten from C.

Syntax — It is the way of writing the program in a particular programming language. Syntax changes from language to language.

09 JANUARY 2014 THURSDAY

Semantics — It is the logic or planning of the program.
Semantics can be written in any of the following ways: (1) Flowcharts (2) Algorithms (3) Pseudo codes

(1) **Flowchart** →
- It is the symbolic representation of the program logic.
- A flowchart is nothing but diagrammatic representation of various steps involved in solution of a problem.
- Flowchart shows the direction of flow of a process.

(2) **Algorithms** → Once a problem has been properly defined, a detailed, finite, step by step procedure

Scanned by CamScanner

for Solving it must be developed. This procedure is known as algorithm

→ Algorithm to add two numbers :—

→ Read a, b → Set Sum = A+B → write Sum → Exit.

(3) Pseudo code →

- Sometimes it is desirable to translate an algorithm to an intermediate form, between that of a flow chart and the source code.
- Pseudocode is an english approximation of source code that follows the rules, styles, & format of a language but ignores most punctuations.
- for example —

→
```
main ()
{
    integer a,b Sum;
    read in a and b;
    add a & b set it to Sum;
    write Sum;
}
```

11 JANUARY 2014 SATURDAY

→ Values —

Numeric	789, -78234, 7.48, -4.54	int, float
String	"Pravesh", "a", "Rajsinghpura".	char

Variables → variables are reserve space on main memory that hold a single value of a time. We can change the value of the variable.

12 SUNDAY

→ Specifiers

%d (int)

%f

Scanned by CamScanner

13 JANUARY 2014 MONDAY

TATA HITACHI

JAN 2014

S	M	T	W	T	F	S
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	31	

Program for Area of Right angle △

```
#include <stdio.h>
#include <conio.h>
Void main()
{
  float base, height, Area;
   clrscr();
   printf("enter base");
   scanf("%f", &base);
printf("height");
   scanf("%f", &height);
    Area = .5 * base * height;
printf("Area will be %f, area);
   getch();
}
```

use for addressment

14 JANUARY 2014 TUESDAY

```
#include <stdio.h>
#include <conio.h>
void main()
{
int amount, n;
printf("enter amount");
scanf("%d, &amount);
   n = amount / 1000;
   printf("1000 = %d", n); —?
    amount = amount %1000;

n = amount / 500;
printf("500 = %d/n", n)
 amount = amount % 500;
```

Scanned by CamScanner

```
n = amount/100;
printf("100 = %d\n", n);
amount = amount % 100;

n = amount/50;
printf("50 = %d\n", n);
amount = amount % 50;

n = amount/20;
printf("20 = %d\n", n);
amount = amount % 20;
```

✡ Algorithm — for getting greater no. between a, b, c

Step 1 → Read a
Step 2 → Read b
Step 3 → Read c
Step 4 → if a >= b then go to step (5)
Step 5 → m = b, go to (7)
step 6 → if m >= c then print m &
step (7) → go to (9)
Step 8 → print c
Step 9 → Stop

An algorithm is a step by step list of well defined instructions to solve a particular problem. An algorithm start by defining a problem or task & terminate in a define end state.

→ An algorithm has some characterstics —

(1) Input → A first we specified the no. of identifier variables that use to take input to solve a specific problem.

(2) Output → Each & every algorithm must be and after achieving the specific output that we define in the starting of the problem.

Scanned by CamScanner

(3) Definateness — Each steps of an algorithm must be precisely defined; the action to be carried out must be rigorously and unambiguously (without confusion) specified for each case.

(4) Effectiveness

(5) Termination

★ The Efficiency of an algorithm is measured by some criteria of measurement these measurement criterias are time complexity & Space Complexity.

The word complexity specifies the algorithm how much complex in sense of computation

Complexity of an algorithm K is a function $f(n)$ which give the time and storage space requirement of the algorithm in term of the size n of the input data.

The time complexity of a problem is the no. of steps that it take to solve an instance of the problem.

An algorithm not only able to solve a specific problem but also must be able to do so in an efficient manner as much as possible

★ Space Complexity :-

It related to the amount of the memory space needed to run a programme.

The space needed by a programme is the sum of the following components —

Space taken by instructions, space taken by the data, space taken by the stack

Scanned by CamScanner

An algorithm is a series of steps or methodology to solve a problem.

Properties of an Algorithm:—

- It is written in simple english.
- Each step of an algorithm is unique & should is self explanatory.
- An algorithm must have at least one input.
- An algorithm must have at least one output.
- An algorithm has finite numbers of set.

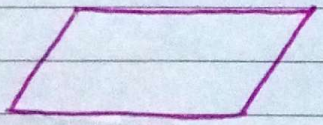
Time Complexity –

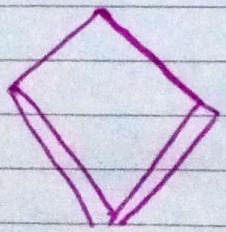
Time Complexity of a programme is the amount of comp. time need to run a programme.
The time taken by a programme is the sum of compile time and run time.
The compile time does not depend on the instance variable characteristics.

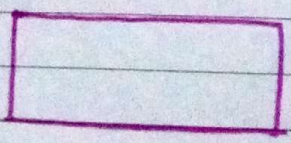
There are three cases for the time complexity of an algorithm —

(1) Best case
(2) Average case
(3) Worst case

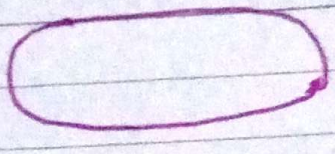
Flow chart :-


→ Input / output

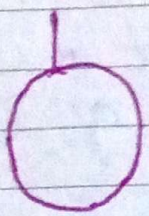

→ Decision taking


→ Processing box

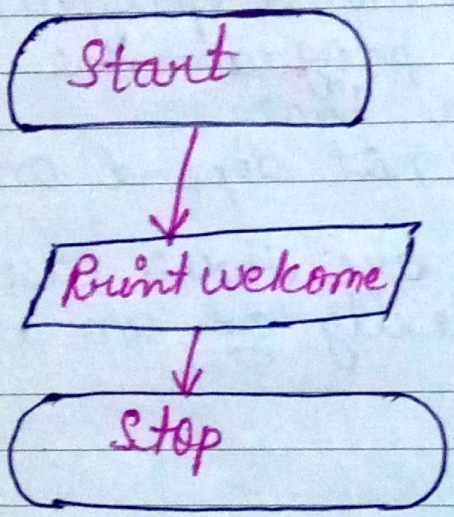
Scanned by CamScanner

 → Start / Stop

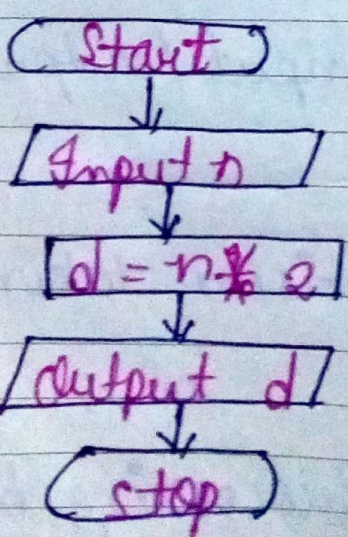
 → Flow lines

 → Connectors

Example → To print welcome

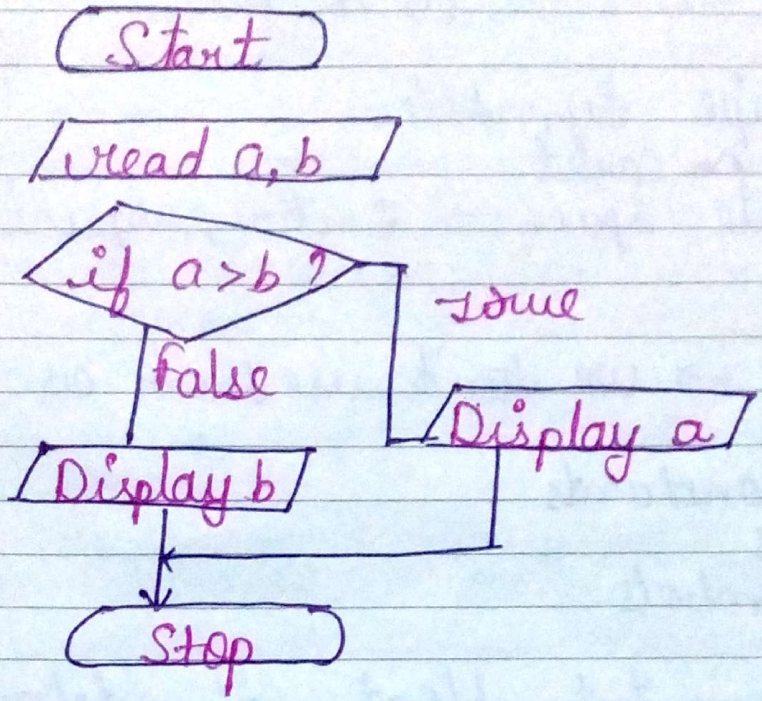


Example →

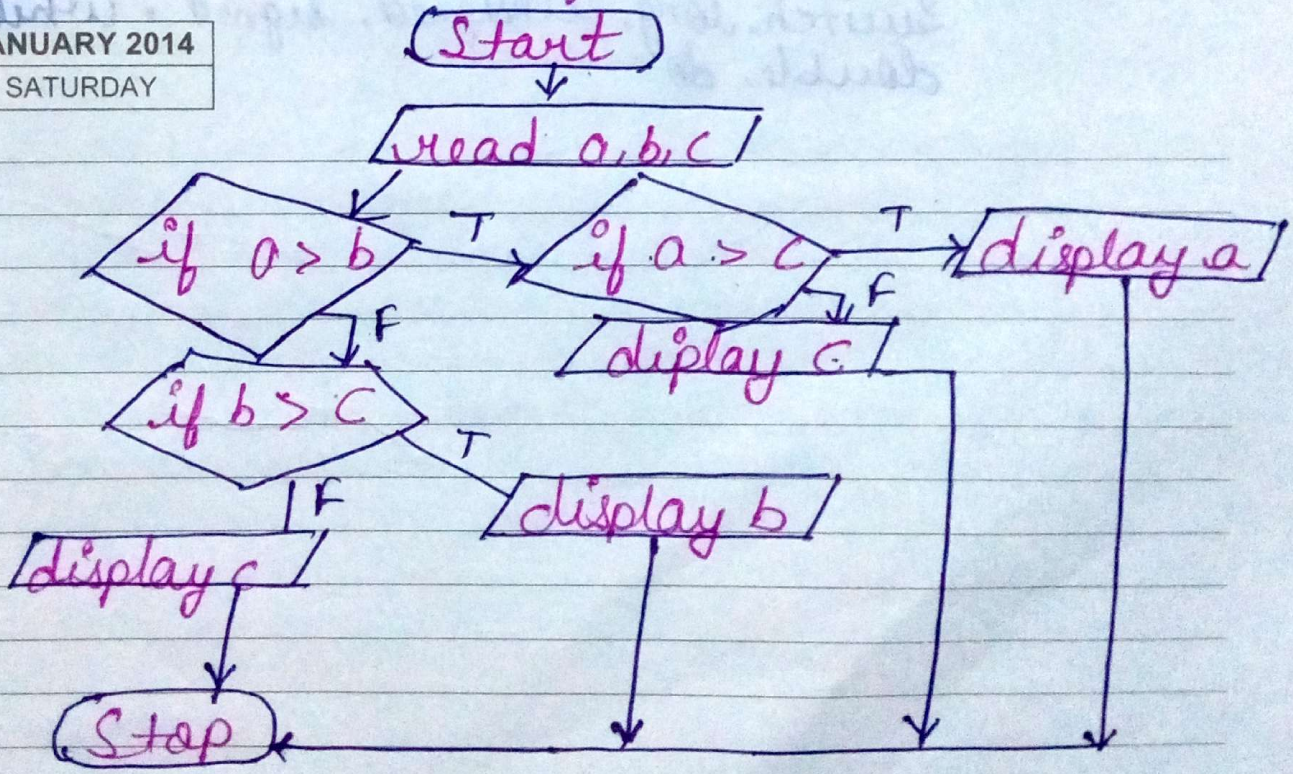


Scanned by CamScanner

Example — Maximum of two numbers



Example — Maximum of 3 number



Scanned by CamScanner

TATA HITACHI

Lowercase letter → A to Z
Uppercase letter → a to z
Symbols —
; pipe Symbols
^ caret
(V) White Space → Enter, Space bar, Backspace

→ C Tokens —
* Keywords → we don't use them as identifiers
* Identifiers
* Literals / Constants
* Operators
* Special Symbols

* Keywords → Int, float, if, default, volatile, union, short, void, struct, auto, switch, long, unsigned, signed, while, else, double, do

Scanned by CamScanner

☆ The # Include Directive →

C programs are divided into functions. Some functions are written by users like us, and many others are stored in the C library.

Library functions are grouped category wise and stored in different files known as header files.

If we want to access the functions stored in the library, it is necessary to tell the compiler about the files to be accessed.

This is achieved by using the processor directive #include as follows:

#include <filename>

Filename is the name of the library file that contains the required function definition. Preprocessor directives are placed at the beginning of a program.

☆ BASIC STRUCTURE OF C PROGRAMS —

- C program can be viewed as a group of building block called functions.
- A function is a subroutine that may include one or more statements designed to perform a specific task.
- To write a c program we first create functions and then put them together. A c program may contain one or more sections.

Scanned by CamScanner

31 JANUARY 2014 FRIDAY TATA HITACHI

Basic Structure of C programme —

(1) Documentation Section → The documentation section consists of a set of comment lines giving the name of the programme, the author and other details, which the programme would like to use later.

(2) Link Section → The link section provides instruction to the compiler to link function from the System library such as using the # include directive.

(3) Definition Section → The definition section defines all symbolic constants such using the # define directive.

(4) Global Declaration → There are some variable that are used in more that one function. Such variables are called global variables and are declared in the global declaration section that is outside of all the function. This section also declares all the user defined function.

FEBRUARY 2014 SATURDAY 01

→ Main function ←

Declaration part — Executable part

SUNDAY 02

(1) Declaration part → It's declare all the variables used in the executable part.

(2) Executable part → There is at least one statement in the executable part. These

Scanned by CamScanner

two parts must appear between the opening & closing braces. The program execution begins at the opening braces and ends at the closing brace. The closing brace of the main function is the logical ends of the program. All statement in the declaration and executable part end with a semicolon.

(6) Subprogram Section → The subprogram section contains all the user define function that are called in the main() function. User-defined functions are generaly placed immediately after the main() function although they may appear in any order.

Example → → D.S.
Program to find area of circle —


```
#include <stdio.h>          — Link Section
#include <conio.h>
#define PI 3.14;            — definition section
float area                  — global declaration
void main()
{
  float r;
printf("enter the radius of the circle r");
  ↳ Executable part
Scanf("%f", &r);
area = PI * r * r;
Printf("Area of the circle = %f", area);
getch();
}
```

Scanned by CamScanner

05 FEBRUARY 2014 WEDNESDAY

TATA HITACHI

✡ Escape Sequence

'\n' → Newline Character
'\a' → beep Sound
'\t' → tab character
'\r' → Carriage character
'\v' → Vertical tab
\" → to print / display double Quotes

Special Symbol
, ; [ ] { } ( )

Operators
+ - * / ≤ ≥

✡ Control Statement

By Default → also known as Sequential Execution

06 FEBRUARY 2014 THURSDAY

Programming construct

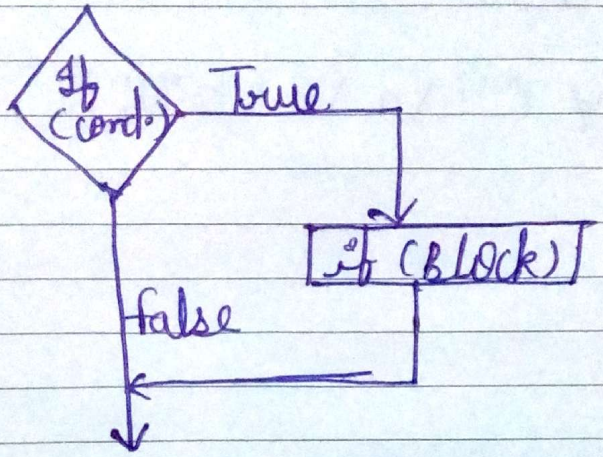
- Decision Making
  - if
  - if else
  - Nested if else
  - else if ladder
  - Switch
- Looping (Iteration / Repetation)
  - While
  - do while
  - for
- Jumping
  - break
  - continue
  - Goto

1. if Statement –

Syntax – if (condition)
{ .... Statement 1;
.... Statement 2.
}

Scanned by CamScanner

Flow chart –



Example 1: To print a message if the no. is greater than 10.

```
#include <stdio.h>
#include <conio.h>
Void main ()
{
    int n;
    clrscr ();
    printf ("Enter number:");
    scanf ("%d", &n);
    if (n > 10)
      {
        printf ("Number is greater than 10");
      }
getch ();
}
```

08 FEBRUARY 2014 SATURDAY

09 SUNDAY

Example 2: To play beep sound if user press character d.

Scanned by CamScanner

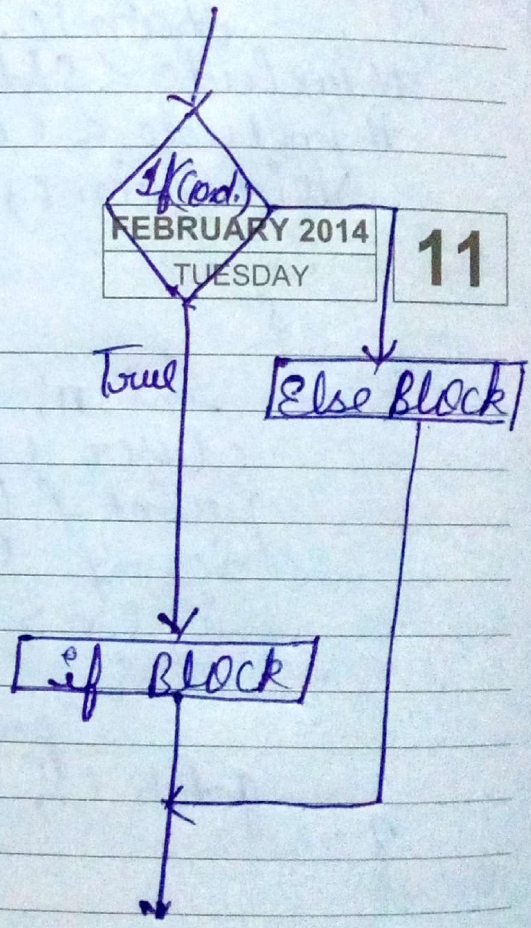
```
printf("Press 'd' for beep");
scanf("%c", &n);
if (n == 'd')
{
    printf("\a\a\a");
}
getch();
}
```

2. If Else Statement

Syntax:

```
if (condition)
{
    =
}
else
{
    =
}
```

Flowchart



Example → To print maximum in two number

```
int first_num, second_num;
clrscr();
```

Scanned by CamScanner

```
printf("Enter two values");
scanf("%d", &first_num & second_num);
if (first_num > second_num)
    {
        printf("%d", first_num);}
else
{
  printf("%d", second_num);
}
```

Ques → To print if the given number is even or odd?

```
int a,b;
clrscr();
printf("enter number");
scanf("%d", &a);
b = a%2;
if (b==1)
  {
    printf("the given no. is odd");
  }
else
  {
   printf("the given no. is even");
  }
```

for the same E/O.

```
if (n%2==1)
```

%(Mod Sign) is used to find remainder.

Scanned by CamScanner

Ques → To print whether no. is negative or positive

```
if (num > 0)
    printf ("The number is positive");
Else
    printf ("The number is negative");
```

(3) Nested if Statement

Syntax

```
if (condition)
{
    =
}
else
{
    if (condition)
    {
        =
    }
    else
    {
        }
}
```


Ex →

```
int n;
printf ("Enter number");
Scanf ("%d", &n);
if (n>0)
    {
    printf ("positive no.");
else
    { if (n<0)
```


Scanned by CamScanner

```
        printf ("No. is Negative");
    else
        printf ("No. is zero");
  }
}
```

Ques. find out maximum of three numbers

Ans.

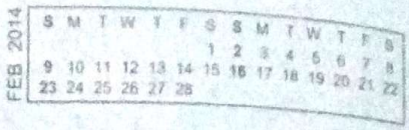
```
if{ a>b && a>c);
  {
    printf ("%d", a);
  }
 else
  {
    if (b>c)
    printf ("%d", b);
  }
```

18 FEBRUARY 2014 TUESDAY

```
    else
    {
      printf ("%d", c);
    }
  }
getch ();
}
```

Scanned by CamScanner

TATA HITACHI



☆ Some important Symbols —

\n → Use for next line

\t → Use for leaves a space and print another message

Example → Hello C Second programme

\r → Due to this "third line is get overwritten on second line. The length of the second line is 11 whereas the length of the third line is 10.

Example → first line
Second line
third line

but after using \r it will be written as follows — first line
third line

Note → But if we use \n and \r togetherly then their is no effect of \r.

\a → Give beep Sound.

→ If we want to use [" "] then we have to use.

Input → printf("""\" Hey! there I am working on comp.\"");

output → " Hey! there I am working on comp."

Scanned by CamScanner

TATA HITACHI

FEBRUARY 2014 | FRIDAY | 21

Data types in C →

- C defines several data types which can be used under different situations.
- There are 3 types of Data types —

(1) Primitive Data type →

- They are already defined by the developer of C or we can say that they are pre-defined data type.

Ex- int, float, char etc.

(2) Derived Data Type →

change in user defined data type and primitive data type. They help in deriving new data type.

Ex → {

```
typdef int fan;
fan a,b,c;
```

}

Use → typedef, array, pointers

22 | FEBRUARY 2014 | SATURDAY

(3) User Defined Data Type →

→ Struct, union

Ex → Student type, teacher, person type etc.

Int → In Int data type we can use it for representing whole number as age of person, roll number etc.

float →

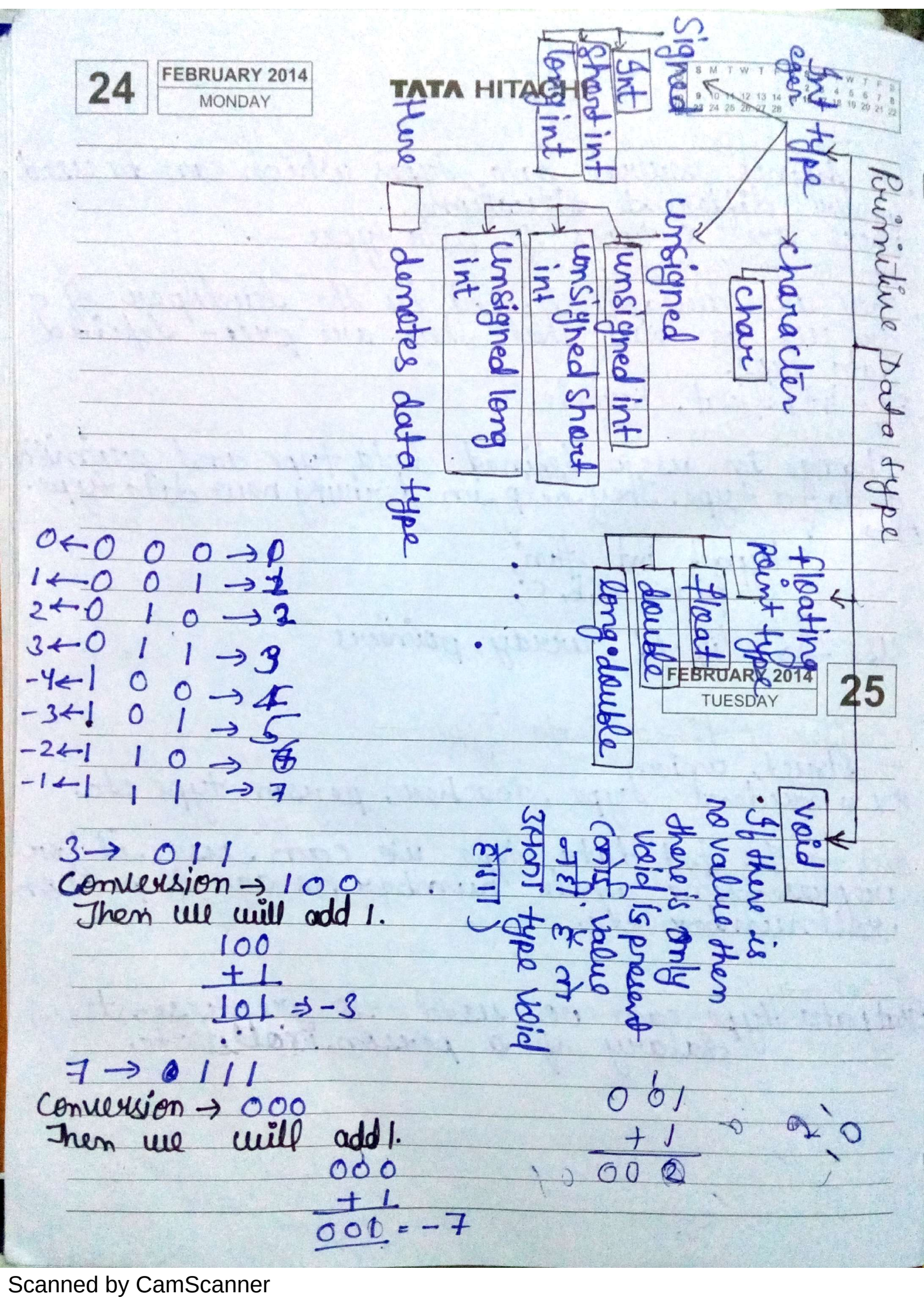
23 SUNDAY

float data type can be used to represent salary of a person, interest etc.

Scanned by CamScanner

24 FEBRUARY 2014 MONDAY

TATA HITACHI



0 ← 0 0 0 → 0
1 ← 0 0 1 → 1
2 ← 0 1 0 → 2
3 ← 0 1 1 → 3
-4 ← 1 0 0 → 4
-3 ← 1 0 1 → 5
-2 ← 1 1 0 → 6
-1 ← 1 1 1 → 7

3 → 0 1 1
Conversion → 1 0 0
Then we will add 1.

100
+ 1
101 ⇒ -3

7 → 0 1 1 1
Conversion → 0 0 0
Then we will add 1.

000
+ 1
001 = -7

0 0 1
+ 1
0 1 0

25 FEBRUARY 2014 TUESDAY

Scanned by CamScanner

Program for printing asky code →

Ex →
```
#include <stdio.h>;
#include <conio.h>
void main()
{
    char n;
    n=89
    printf("%dc",89);
    clrscr();
    getch();
}
```

Ex→
```
#include <stdio.h>
#include <conio.h>
void main()
{
    char
    a;
    clrscr();
    a=N;
    printf("%d",a);
    getch();
}
```


Signed Short int → We can store positive and negative values.

Unsigned int → We can store only +ve value.

Complement means conversion of 0 to 1 & 1 to 0.

Scanned by CamScanner

TATA HITACHI

✱ Data Types →

✱ Integer Type

S.N.	Data Type	Size (in bytes)	Range
1.	int or short int %d	2byte 4byte	-32768 to 32767 -2147483648 to +2147483647
2.	long int %ld	4 byte	-2147483648 to +2147483647
3.	Unsigned int %u	2byte 4byte	0 to 65535 0 to +429.....
4.	unsigned short int %u	2byte 4byte	0 to 65535 0 to 429.....
5.	unsigned long int %lu	4byte	0 to 429.....
6.	Char %c	1byte	2^8 → 0 to 255

Example for long int →

```
long int n = 4259
printf("%ld", n);
```


Scanned by CamScanner

float →

S.No.	Data type	Size in byte.	Range
1.	float %f	4 byte.	3.4E-38 to 3.4E+38
2.	Double %lf	8 byte	1.7E-308 to 1.7E+308
3.	Long double %Lf	10 byte	1.1E-4932 to 3.4E+4932

Operators → Decide the operation to be performed on arithmetic values or on logical basis.

Exp → operands + operators

operands ← a + 9 → operators

operators

Ex → a + 9 * 71 / 6

operators

operands can be variable and constant

TRUE - FALSE ~~→~~ Expression → Boolean Expressions

→ Airthmetic expression
a * p , a - b , a + 4 / b

→ Boolean expression
(a * b) > 19
In this answer will come only in True & false

Scanned by CamScanner

05 MARCH 2014 WEDNESDAY TATA HITACHI

① Airthmatic operators →
+, -, *, /, %

② Relational operators →
<, <=, >, >=, ==, !=

③ Logical operators →
&&, ||, !

④ Increment / Decrement operator →
++, --

⑤ Conditional operator →
?, :

⑥ Bitwise operator →
~, &, |, ^, <<, >>

⑦ Assignment Operator →
=, +=, -=, *=, /=, %=, &=, |=, ^=, <<=, >>=

⑧ Size of operator —
, sizeof

06 MARCH 2014 THURSDAY

① Airthmatic operator →
+ → add. op.
- → Sub. op.
* → Mult. op.
/ → division op.
% → Modulus op.

/ → (i) Second operand must be a non-zero value. If we use 0 as second operand then the program will terminate

Scanned by CamScanner

(ii) int/int then the ans. will also be int value.
int / float then the ans. will be float value.

% → Used to generate remainder
(i) Second operand must be a non-zero value.
(ii) It can only used in integer values not in float values.

② Relational operator →
Generate true & false values.
True generate → 1
false generate → 0

Ex → {
int a=14, b= 8, c;
c = a > b
printf ("% d", c);
}
Output → 1 will display which means the condition is true.

< → less than → Small < large

boolean c;
int a = 4, b= 8;
c = a < b;
printf ("% d", c);

 Output → True

Scanned by CamScanner

10 | MARCH 2014 | MONDAY

TATA HITACHI

MAR 2014

S	M	T	W	T	F	S
					1	2
9	10	11	12	13	14	15
23	24	25	26	27	28	29

S	M	T	W	T	F	S
16	17	18	19	20	21	22
30	31					

Conversion into Binary code →

MARCH 2014 | TUESDAY | 11

Scanned by CamScanner

Ques → Program for Selection of Candidate

→
```
#include <stdio.h>
#include <conio.h>
Void main()
{
  int age, percentage;
  if (age>20)
  {
    if (P>75%)
    print f ("Selected");
  }
  else
  {
    print f ("Rejected");
  }
getch();
}
```


* Logical operators —

→ && → logical AND
→ || → Logical OR
→ ! → Logical NOT
→ .

. Truth table for logical AND (&&) operator —

Exp	Exp	Result
T	F	F
F	T	F
T	T	T
F	F	F

Scanned by CamScanner

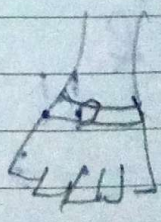
TATA HITACHI

Example →

```
#Include <Stdio.h>
#include <conio.h>
Void main()
{
  int age, p;
clrscr();  Scanf("%d%d",&age,&p);
  if (age>=20.&& p>=7.5)
{
  printf ("Selected");
}
 else
{
  printf ("Rejected");
}
getch();
}
```


Example →

```
#include <stdio.h>
#include <conio.h>
Void main()
{
  Int a=10, b=20;
  clrscr();
  if ( a>=10 && b==20);
  {
    printf ("Selected");
  }
else
{
  printf ("Rejected");
}
getch();
}
```



Scanned by CamScanner

Logical AND operators are work with two operands which may be any expression, variable, constant or function,
- It checks both of its operands and returns true value or false value.
- If either of its operands is false, a false value is returned (i.e. a decimal 0).
- Otherwise returns true value (i.e. a decimal 1)

(ii) Logical OR (||) —
- The operator with two operands which may be any expression, variable, constant or function.
- It checks ~~may~~ any of its operands and return e value or false value.
- If both of its operands are false, a false value is returned (i.e. a decimal 0) otherwise returns true value (i.e. a decimal 1).

Truth Table of OR

operand 1	operand 2	Returned value
false	false	false
True	false	True
false	True	True
True	True	True

Example →

```
#include <stdio.h>
#include <conio.h>
void main()
{
int age, p;
printf("enter age and percentage");
scanf("%d %d", &age, &p);
```

Scanned by CamScanner

if (age >= 20 || p > 7.5)
{
printf ("selected");
}
else
{
printf ("Rejected");
}
getch ();
}

Example →

```
#include <stdio.h>
#include <conio.h>
void main()
{
int a=100, b=120;
clrscr();
if (a>=100 || b<0)
{
printf("returned value");
}
getch();
}
```


(iii) Logical NOT (!)

The operator convert a true value into false and vice versa. Again the operand may be any expression, constant, variable or function.

Scanned by CamScanner

Truth Table of NOT

operand	Returned value
false	True
True	false

Example →

```
#include <stdio.h>
#include <conio.h>
void main()
{
int p;
printf("enter age and p");
scanf("%d %d", &age, &p);
if(!age<20)
{ printf("selected");
}
else
{
printf("Rejected");
}
getch();
}
```


Increment operator (++) and Decrement operator (--);

- The operators ++ is known as increment operator and the operator -- is known as decrement operator.
- They are only used in variables not in constants.
  Ex → a+4 is right but 17++ is not.

Scanned by CamScanner

Example →

```
#include <stdio.h>
#include <conio.h>
Void main()
{
  int a=9;
  --a;
printf("%d", a);
 getch();
}
```

```
#include <stdio.h>
#include <conio.h>
Void main()
{
  int a=9;
    a--;
printf("%d", a);
getch();
}
```


--a → Pre decrement
a-- → Post decrement

Example →

```
#include <stdio.h>
#include <conio.h>
Void main()
{
  Int  a=9, b;
    b= --a;
 printf("%d", a); getch();
```

Scanned by CamScanner

Example → Demo of ++ operator —

```
# include <stdio.h>
# include <conio.h>
void main
{
  int x, y;
  clrscr ();
  x=10;
  y= x++;
  printf ("x= %d\n", x);
  printf ("y=%d\n", y);
  getch();
}
```

Output →

x=11
y=10

In this condition at first x will be assigned to y and x will be incremented by 1 i.e. y= x++ is equivalent to the following two statements.

```
y=x;
x = x+1;
```


Example →

```
# include <stdio.h>
#include <conio.h>
void main()
{
int x, y;
clrscr();
x=10;
y= ++x;
printf("%d", x);
printf("%d", y);
getch();
}
```

Scanned by CamScanner

Output —

x = 11
y = 11

Due to pre first x will be increase by 1 & the same increment value will be assigned to y; y = ++x is equivalent to the following two statements:

```
x = x + +;
y = x;
```

NOTE → THERE IS ONLY ++ OPERATOR WHICH IS USED TO INCREMENT THE VALUE OF THE OPERAND BY ONE. THERE IS NO SUCH OPERATOR AS ++ OR +++ OR +.+ + WHICH INCEMENT THE VALUE BY 2 OR 3.
SIMILAR TO DECREMENT OPERATOR WE HAVE DECREMENT OPERATOR (--).
WHICH IS AGAIN MAY Be OF two Types -
PRE-DECREMENT OR POST-DECREMENT

Example → Demo of ++ and -- operator —

```
#include <stdio.h>
#include <conio.h>
void main()
{
  int x, y;
clrscr();
 x=10;
 y= ++x;
   --x;
    y--;
  x=y++;
```

x=y++;

Scanned by CamScanner

```
    y= --x;
    x= y++;
printf("x= %d\n", x);
printf("y = %d\n", y);
getch();
}
```

Output →

x=9

y=10

Explaination →

Statement	value of x	value of y
y = ++x	11	11
--x	10	11
y--	10	10
x = y++	10	11
y = --x	09	09
x = y++	09	10

01 APRIL 2014 TUESDAY

✡ Explaination →

Statment	value of x	Explaination
x--	10	value of x will be printed then decrement
x++	09	value of x will be printed then increment
x	10	Value of x will be printed
++x	11	value of x will be incremented then printed
--x	10	Value of x be decrement then printed

Scanned by CamScanner

Example →

```
#include <stdio.h>
#include <conio.h>
void main()
{
   int x;
   clrscr();
    x= 10;
printf("%d %d %d %d %d", --x, ++x,
                x, x++, x--);
   getch();
}
```

Output →

10 11 10 9 10

✱ BASIC OPERATORS —

- They are called so because they operate on bits.
- C provides total 6 types of bitwise operators. They are follows:

Bitwise Operators

S.NO.	Operator	Meaning used for
1.	&	Bitwise AND
2.	|	Bitwise OR
3.	^	Bitwise X-OR
4.	~	One's complement
5.	>>	Right shift
6.	<<	Left shift

a) Bitwise AND (&) —

Scanned by CamScanner

First Bit	Second Bit	Result
0	0	0
0	1	0
1	0	0
1	1	1

Example → Demo of bitwise operator & (AND)

Decision making

```
#include <stdio.h>
#include <conio.h>
Void main()
{
   int a=2, b=3;
int c=;
   clrscr();
   c=a&b;
   Printf("C=%d\n", C);
   getch();
}
```


C=2

Binary values of a=2 is 0010 and b=3 is 0011.

Bitwise AND of these two values is performed as follows:

```
  0010
  0011
C=0010 = 2
```


(ii) Bitwise OR (|) →

Scanned by CamScanner

It takes two bit as operand and returns the value 1 if at least one is 1. If both are 0 only the result is 0 else it is 1.

Truth Table for Bitwise OR

first Bit	Second Bit	Result
0	0	0
0	1	1
1	0	1
1	1	1

Example → Demo of bitwise operator |(OR)–

```
#include <stdio.h>
#include <conio.h>
void main()
{
  int a=12,b=7;
  int c;
  clrscr();
  c=a|b;
  printf("c=%d\n",c);
  getch();
}
```


Output ⟹ c=15

Explaination → Binary values of a=12 is 1100 and b=7 is 0111.
OR of these two value is performed as follows–

```
  1 1 0 0
| 0 1 1 1
---------
  1 1 1 1   (Output is c will be 15 in decimal)
```

Scanned by CamScanner

(iii) Bitwise XOR (^) —

This operator takes at least two bits (may be more than two). If number of 1's are odd then result is 1 else result is 0.

Truth Table of Bitwise XOR

First Bit	Second Bit	Result
0	0	0
0	1	1
1	0	1
1	1	0

same bit → 0
different bit → 1

Example → Demo of bitwise operator ^ (XOR) —


```
#include <stdio.h>
#include <conio.h>
void main()
{
 int a=5, b=6;
  int c;
  clrscr();
  c= a ^ b;
 printf("c= %d\n",c);
  getch();
}
```

output →

c= 3

Explaination → Binary value of 5 is 0101 and binary value of 6 is 0110. XOR of these two values is performed as follows—

Scanned by CamScanner

```
  0 1 0 1
  0 1 1 0
  0 0 1 1   (output of c is 3)
```

(iv) 1's Complement (~) —

The Symbol (~) denotes one's complement. It is unary operator and complements the bits in its operand i.e. 1 is converted to 0 and 0 is converted to 1.

0 → 1
1 → 0

Example → Demo of Bitwise operator ~

```
#include <stdio.h>
#include <conio.h>
void main()
{
  int a=5,b;
  b=~a;
  clrscr();
  printf("b=%d\n",b);
  getch();
}
```

Output —

b=10

Explanation —

Binary value of 5 is 0101

Then, it get converted into —

b=10=1010

☆ Left Shift operator (<<) —

Scanned by CamScanner

The operator is used to shift the bits of its operand. It is written as x << num, which means shifting the bits of x towards left by num number of times. A new zero entered in the least significant bit (LSB) position.

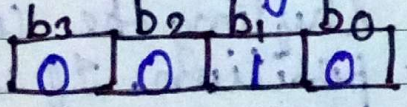
Example → Demo of bitwise operator <<

```
#include <stdio.h>
#include <conio.h>
void main()
{
  int a=2, b;
  b = a<<1;
  clrscr();
  printf("b= %d \n", b);
  getch();
}
```

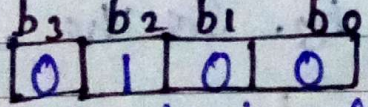
Output →

b=4

Explanation → a<<1 means shifting the contents of a towards left by 1 position. If we represent the number 2 in binary as:



b0 from first left side is least Significant Bit (LSB)
b3 is called Most Significant Bit (MSB)
Shifting left by 1 bit position results in b3 losing its value and taking from b2, b2 getting from b1 and b1 from b0, a new zero is inserted at b0. So the resultant bit pattern will be.



which is 4 in decimal.

Scanned by CamScanner

(vi) Right shift operator (>>)

The operator is used to shift the bits of operand. It is written as x >> num; which means shifting the bits of x towards right by num number of times. A new zero is entered in the most significant Bit (MSB) position.

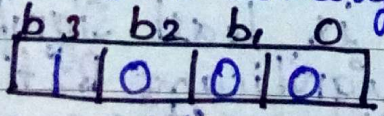
Example → Demo of bitwise operator >>.

```
#include <stdio.h>
#include <conio.h>
void main()
{
  int a=8, b;
  b=a>>1;
  clrscr();
  printf("b=%d\n", b);
  getch();
}
```

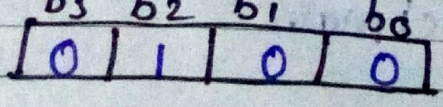

Output →

b=4

Explaination → a>>1 means shifting the contents of a towards right by 1 bit position. If we represent the number 8 in binary as:



Shifting by right by 1 bit position results in bo losing its value and taking from b1, b1 getting from b2 and b2 from b3, a new zero is inserted in b3. So the resultant bit pattern will be.



Scanned by CamScanner

conditional operator → (?:)

⇒ Syntax : expression ? true part : false part

Example →

```
#include <stdio.h>
#include <conio.h>
Void main()
{
  float p;
  printf("enter p");
  Scanf("%f", &p);
  p>=50? printf("passed"); printf
        ("failed");
  getch();
}
```

Example →

19 APRIL 2014 SATURDAY

```
#include <stdio.h>
#include <conio.h>
Void main()
{
float p,t,r,i;
printf("enter principal & time");
  Scanf("%f %f", &p, &t);
   r= t>=5 ? 9: 6;
    i= p x t x r/100;
    printf("%f", i);
getch();
}
```

20 SUNDAY

Assignment operator =>

The = operator is called assignment operator.

Scanned by CamScanner

=, +=, -=, *=, /=, %=, &=, |=,
<<=, >>=

Example → a = 8;
a = a + 8;
or
a = +6 → it means the value of a after adding the value of a is again store in a.

Ex → K = 6;
K = K % 6;
or
K = %6;

☆ Special operator →

(i) Comma operator (,) →

- This is most important operator in C yet overlooked by most of the programmers. But understanding how the operator works is most in C.
- The operator works from left to right and in expressions seperated by commas rightmost. expression becomes the operand for this operator.
- Let us, see some examples related to comma operator.

Example 1 →

```
#include <stdio.h>
#include <conio.h>
void main()
{
  int a=2, b=3, c=4, x=2;
  clrscr();
```

Scanned by CamScanner

```
x=(a,b,c);
printf("x=%d\n",x);
getch();
}
```

output
x=4

Explaination → In this comma operator is from left to right and the rightmost value is given to x i.e., value of c in above programm

Note → when, d=(a,b,c);
in this condition value of a first store in d then value of b, then value of c and finaly output will be the value of c which get stored in d.

Whereas, d=a,b,c

In this condition value of d first Store in d then value of b, then value of a and finally output will be the value of a which get stored in d.

(ii) Size of operator →
- This operator is used to find size of in bytes a particular variable of particular type or a constant takes up in memory. This is the only operator which also acts as a function.
- Let's take an example to understand this operator—

Example →

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

Scanned by CamScanner

```
char c;
double d;
clrscr();
printf("Size of int variable a = %d \n", sizeof(a));
printf("Size of float variable b = %d \n", sizeof(b));
printf("Size of char variable c = %d \n", sizeof(c));
printf("Size of double variable d = %d \n", sizeof(d));
getch();
}
```

Output →
Size of int variable a=2
Size of float variable b=4
Size of char variable c=1
Size of double variable d=8

Explaination →
The program gives the amount of memory in bytes taken by variable of different data types. Sizeof int variable in turbo c is 2 byte whereas in windows it is of 4 bytes.

☆ Unary operators →
• which works on singal value. These type of operators are → ++, --, !, ~, sizeof

☆ Binary operators →
• Requisite two number of operands.
+, -, *, %, /, >, <, >=, <=, ==, !, &&, ||, &, |, ^, <<, >>, =, +=, -=, *=, /=, %=, >>=, <<=, &=, |=, ^=, !=

☆ Terniary operators →
Require three number of operands. These type of operators are ?:

Scanned by CamScanner

☆ Operators precedence Rules and Associativity

Priority	operators	Associativity
1	+, -, ++, --, !, &, ~, sizeof	Right to left
2	*, /, %	Left to R
3	+, -	Left to Right
4	<<, >>	Left to Right
5	<, <=, >=, >	Left to Right
6	==, !=	Left to Right
7	&	Left to Right
8	^	Left to Right
9	|	Left to Right
10	&&	Left to Right
11	||	Right to Left
12	?:	Right to Left
13	=	Right to Left
14	,	

Precedence → tells in an expression which operation should be performed first depending upon priority of operators.

Associativity → means when two or more operators have same priority then from left to right we operate.

Example → Let's take an example to understand it —

```
#include <stdio.h>
#include <conio.h>
void main()
{
  int a=2, b= 3; c=4, d;
    clrscr();
  d= a * 4 /b+ c -10 %3;
  printf("d=%d",d);
    getch();
}
```

Scanned by CamScanner

Note → + and - are binary operators

30 | APRIL 2014 | WEDNESDAY

TATA HITACHI

MAY 2014

S	M	T	W	T	F	S
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	31	

Output → d = 5

Explaination →

- Operator with higher priority will be executed at first.

When all go down priority will be low.

As priority of Lay opr 1 (*, /, %) is higher than Lay opr 2 (+, -), so expressions involving opr 1 are evaluated at first. Again at the same level the associativity of opr 1 is from left to right, so evaluation will proceed as;

$d = {}^{13}2 \overset{2}{*} 4 /_{2} 3 +_{3} 4 -_{3} 10 \%_{2} 3;$

$d = {}^{13} 8 /_{2} 3 +_{3} 4 -_{3} 10 \%_{2} 3$

$d = {}^{13} 2 +_{3} 4 -_{3} 10 \%_{2} 3$

$d = {}^{13} 2 +_{3} 4 -_{2} 1$

$d = {}^{13} 6 -_{2} 1 \rightarrow d = {}^{15} 5$

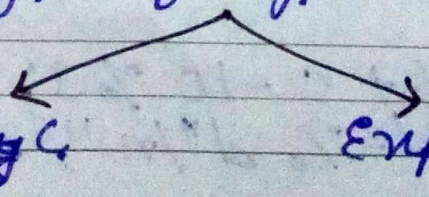
$d = {}^{13} 5$ ←

MAY 2014 | THURSDAY | 01

✱ Type Casting / Conversion →

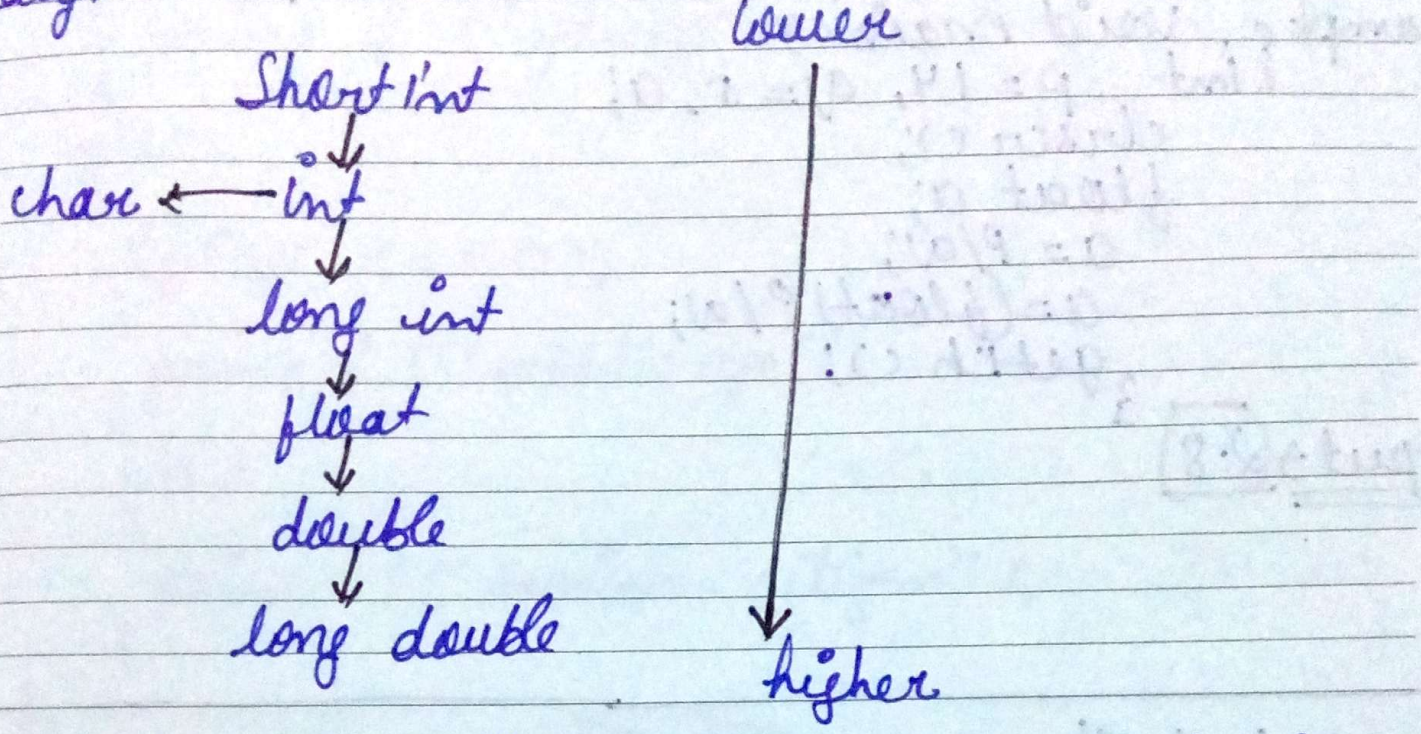
- Some time in expression we require to convert the data type of a variable or a constant ∮ Lay from float to int or int to float etc.
- When we write data type in brackets then they become type cast.
- There will be no type casting if variables are of same.

Two type of type casting



Scanned by CamScanner

Type different then it will execute automatically.



- When any operation to be performed the low data automatically get converted into higher data type

03 MAY 2014 SATURDAY

Example →

a = (14) / (5.0);

14 → Integer type value

5.0 → double type value

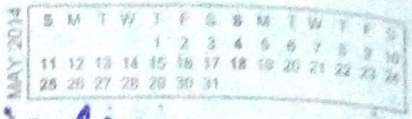
- In above expression we have '14' integer type value where '5.0' is double type value.
- In this case 14 will be automatically get converted into 14.0 i.e., in double data type.

04 SUNDAY This is called as implicit data type. This is also called as winding conversion.

Explicit → Explicit type casting is done by using data type cast operator i.e. (float), (int), (char) the data types are put into round brackets. It is done forcefully.

Scanned by CamScanner

TATA HITACHI



float a; #include <stdio.h>
a = 14/5; #include <conio.h>

Example { void main()
{ int p=14, q=5, a;
clrscr();
float a;
a = p/q;
a = (float) p/q;
getch();
}

Output → 2.8

✱ Nested If →


```
#include <stdio.h>}
#include <conio.h>
void main()
{
int age;
printf ("enter age");
Scanf ("%d", &age);
if(age &<=12)
{
printf ("child");
}
else
{
if (age <=19);
{
printf ("teenage");
}
else
{
```

Scanned by CamScanner

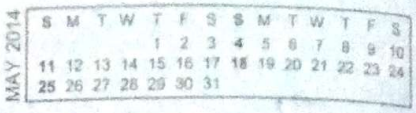
```
{ if (age <=40)
  { printf ("young");
  }
  else
  {
    if (age <=60);
  {
    printf ("middle age");
  }
    else
    {
      printf ("senior citizen");
    }
}
getch();
}
```


Else if ladders or stairs;

Syntax →

```
if (condition)
{
}
else if (condition)
{
}
  else if (condition)
  {
  }
    else if (condition)
    {
    }
```

Scanned by CamScanner




```
#include<stdio.h>
#include<conio.h>
void main()
{
 int age;
printf("enter age");
 scanf("%d",&age);
  if(age<=12)
  {
    printf("child");
  }
  else if(age.<=19)}
  {
   printf("Teenage");
  }
  else if(age<=40)
  {printf("young");
  }
  else if(age<=60)
   {
    printf("middle age");
  }
   else
   {
    printf("Senior citizen");
   }
   getch();
}
```


Scanned by CamScanner

* Switch case (multiway) —
four keywords used
Switch, case, Default, Break

Syntax → Switch (int exp)
{
Case Value 1: ==
Break;
Case Value 2; ==
break;
Case Value 3: ==
break;
Case Value 4: ==
break;
Default: ==
break; → not compulsory
}

* Example →

```
#include <stdio.h>
#include <conio.h>
Void main()
{
  int day;
printf ("enter between 0-6");
 Scanf ("%d", & day);
```

Scanned by CamScanner

```
Switch (day)
{
 Case 5; Printf ("friday");
          break;
 Case 3: Printf ("wednesday");
          break;
 Case 1: Printf ("Monday");
          break;
 Case 0: printf ("Sunday");
          break;
  Default : Printf ("Out of range");
          break;
  Case 6;  Printf ("Saturday");
          break;
  Case 2: printf ("Tuesday");
          break;
   Case 4: Printf ("Thursday");
  }
   getch();
}
```


Example →

```
#include <stdio.h>
#include <conio.h>
Void main()
{
  int n;
 printf ("enter a number");
  Scanf ("%d", &n);
   Switch (n%2)
 {
  Case 1: Printf ("odd");
          break;
  Case 2; print f ("even");
 }getch();
}
```

Scanned by CamScanner

Looping is a process in which set of statements are executed repeatedly for a finite or infinite number of times.

TATA HITACHI

MAY 2014 FRIDAY 16

The while loop →

The syntax of the while loop is simple →

```
while (condition)
{
    statements:
    (loop body)
}
```

The statements inside { } is called body of the while loop.

To work with any type of loop three things have to be performed:

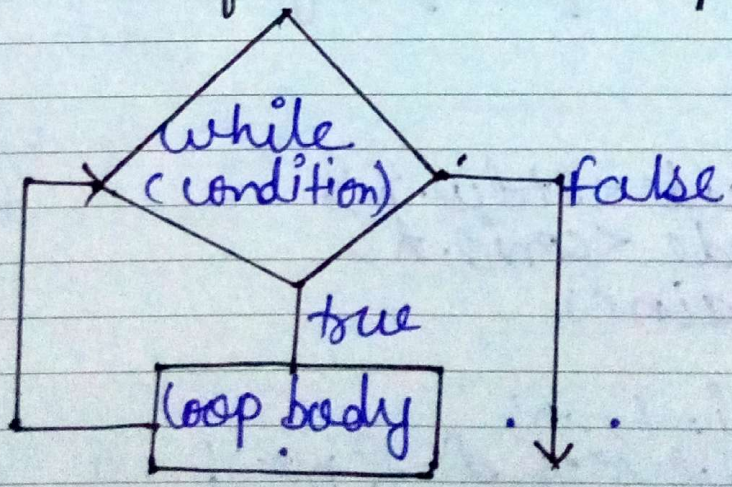
(1) Loop control variable & its initialization.

17 MAY 2014 SATURDAY

(2) condition for controlling the loop.

(3) Incorrect / decrement of control variable.

flow chart for (while loop) —



18 SUNDAY

Scanned by CamScanner

Example →

```
#include <stdio.h>
#include <conio.h>
Void main()
{
  int i=1;
  while (i<=5)
  {
    printf("Hello");
    i=i+1;
  }
getch();
}
```

output →

i = 1
i = 2
i = 3
i = 4
i = 5

* Ctrl + Break → Use to close the programme forcefully.

Example →

```
#include<stdio.h>
#include<conio.h>
void main()
{
  int i=1, n;
  Scanf("%d", &n);
while (i<=n);
```

Scanned by CamScanner

```
    {
        printf ("Hello ");
          i=i+1;
    }
  getch();
}
```

Output →

n = 5
1
2
3
4
5

Example →

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



```
Void main()
{
  int i=1, n;
  printf ("enter last number");
    ("%d", &n);
      (i<=n);
  {
      printf ("%d", i);
        i=i+1
  }
  getch();
}
```

Example → Sum of n natural numbers.

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

Scanned by CamScanner

```
int i=1, sum=0;
printf("enter number:");
scanf("%d", &no);
while(i<=n)
{
  sum = sum+i;
    i = i+1;
}
getch();
}
```

Output

n = 4

sum = 0 + 1 = 1
sum = 1 + 2 = 3
sum = 3 + 3 = 6
sum = 6 + 4 = 10

Ex → generation of table of any given number →


```
#include <stdio.h>
#include<conio.h>
void main()
{
  int n, t=1, value;
  clrscr();
  printf("enter any +ve number");
  scanf("%d", &n);
  while(t<=10)
  {
    value = n*t;
    printf("%d*%d=%d", n, t, value);
    t++;
  }
getch();
}
```


Scanned by CamScanner

Output →

Enter any +ve value = 6

6 x 1 = 6
6 x 2 = 12
6 x 3 = 18
6 x 4 = 24
6 x 5 = 30
6 x 6 = 36
6 x 7 = 42
6 x 8 = 48
6 x 9 = 54
6 x 10 = 60

Example → Printing number between 1 and 100 which are exactly divisible by 3 and 5.

```
#include<stdio.h>
#include<conio.h>
void main()
{
  int t=1;
  clrscr();
  while(t<=100)
  {
    Printf("enter number=");
    if(t%5==0 & t%3==0)
    t=t+1;
  }
  getch();
}
```

Output →

15 30 45 60 75 90

Scanned by CamScanner

If remainder, divisible by 3 is zero then number is completely divisible by 3. Same is true for 5 so we have combined the two condition by && operator so if the remainder in both the case is zero then && condition is true and that number will be printed.

Example → Sum of numbers →

```
#include <stdio.h>
#include <conio.h>
Void main()
{
 int n, a, t = 0;
  printf ("enter number = ");
  scanf ("%d", &n);
  while (n > 0)
  {
    a = n % 10;
    t = t + a;
    n = n/10;
  }
printf ("Total is %d", t);
 getch();
}
```


Output →

Enter the number → 322

~~Sum of~~ total is → 7

Explaination → finding Sum of digits of a given number

a	n	t
0	322	0
2	32	2
2	3	4
3	0	7

Scanned by CamScanner

8. Pallandream →

```
#include <stdio.h>
#include <conio.h>
void main()
{
  int n, a, t = 0;
  or = n;
printf ("enter number=");
Scanf ("%d", &n);
while (n>0);
  char  r=n ;
  closer ();
{
   a = n % 10;
   t = t * 10 + a;
   n = n/10;
}
if r == t
{
   printf ("yes");
}
else
{
 printf ("no");
}
getch ();
}
```


Output →

Scanned by CamScanner

Ex → Reverse of a number →

```
#include <stdio.h>
#include <conio.h>
void main ()
{
  int n, a, R = 0;
  printf ("enter number=");
  scanf ("%d," & n);
  while (n > 0)
  {
    a = n % 10;
    R = R * 10 + a;
    n = n / 10;
  }
  printf ("Reverse is %d," R);
getch ();
}
```


Scanned by CamScanner

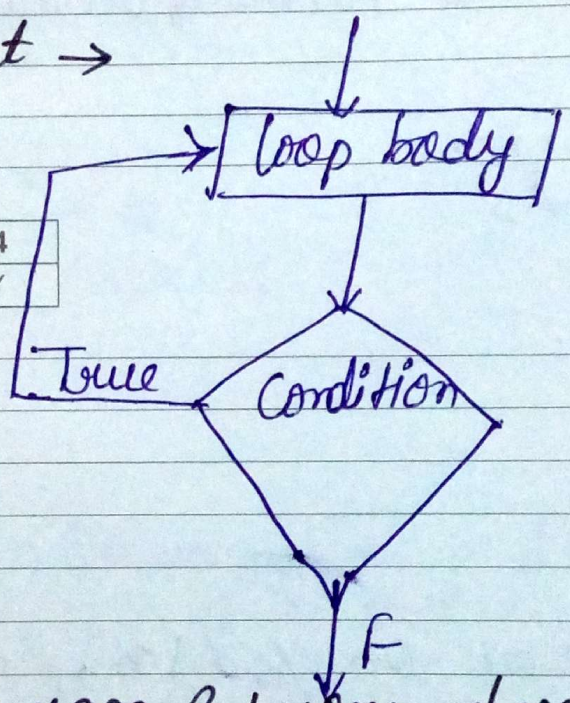
Do while →

It is a exist control statement

Syntax →

```
do
{
   ==
   loop body
   ==
} while(condition);
```

flow chart →



Example → Square of two numbers

```
#include <stdio.h>
#include <conio.h>
void main()
{
   int n, r;
   char ch;
   do
   { clrscr();
     printf("enter no. = ");
     scanf("%d", &n);
     r = n * n;
     printf("Result is %d", r);
     printf("press y for continues");
     fflush(stdin);
     scanf("%c", &ch);
   } while(ch == y)
```

Scanned by CamScanner

06 JUNE 2014 FRIDAY

TATA HITACHI

JUN 2014 S M T W T F S S M T W T F S 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30

Pointers →

Pointer is most powerful and interesting feature of C language. They are simple to use provided us understand the concept better.

Each memory location has got some address represented in numeric or as a whole number. Pointers are special variables which store these addresses; hence we can say that a pointer is a valid address which is stored in a pointer type variable.

Syntax →

Type * Name of Variable

```
#include <Stdio.h>
#include <Conio.h>
Void main()
{
  int a;
  float b;
  clrscr();
  a=10;
  b=20.34;
  printf("address of a = %d\n",&a);
  printf("Value of a = %d\n",a);
  printf("address of b = %f\n",&b);
  printf("Value of b = %f\n",b);
  getch();
}
```

JUNE 2014 SATURDAY 07

SUNDAY 08

Output →

Scanned by CamScanner

address of a = 65432
value of a = 10
value of b = 65434
address of b = 20.3400

Explaination → We have simply output the value and the address. for address we have used & operator which prints address of the variable.

Example 2 →

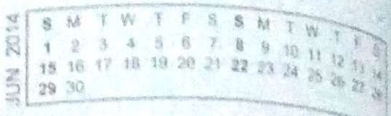
```
# include <stdio.h>
# include <conio.h>
void main ()
{
   int a, * Pa;
   float b, * Pb;
```



```
   clrscr ();
   Pa = & a;
   Pb = & b;
   * Pa = 10;
   * Pb = 20.34;
   printf ("Address of a = %d \n", & a );
printf ("Address of a = %d \n", Pa);
printf ("Address of a = %d \n", * Pa );
printf ("value of a = %d \n", a );
printf ("value of a = %d \n", * Pa);
printf ("value of a = %d \n", * (& a) );
printf ("address of b = %d \n", & b);
printf ("address of b = %d \n", * (& Pb) );
printf ("value of b = %f \n", b);
printf ("value of b = %f \n", * Pb);
printf ("value of b = %d \n", (& b) );
getch ();
}
```

Scanned by CamScanner

TATA HITACHI



Output →

```
Address of a = 65230
Address of a = 65230
Address of a = 65230
Value of   a = 10
Value of   a = 10
Value of   a = 10
Address of b = 65232
Address of b = 65232
Value of   b = 20.340000
Value of   b = 20.340000
Value of   b = 20.340000
```


Scanned by CamScanner

for while

- The for loop is most frequently used by programmers just because of its simplicity

Syntax →

```
for (initialization ; condition ; increment /decrement)
{
    Statements;
    - - - - - ;
    - - - - - ;
}
```

Example → printing numbers 1 to 10.

```
#include <stdio.h>;
#include <conio.h>;
void main()
{
    int t;
    clrscr();
    for (t=1, t<=10; t++);
      printf ("t = %d\n", t);
    getch();
}
```

Output →

```
t = 1
t = 2
t = 3
t = 4
t = 5
t = 6
t = 7
t = 8
t = 9
t = 10
```


Scanned by CamScanner

Example → Demo of for loop, add numbers between 30 and 3.

```
#include <stdio.h>
#include<conio.h>
Void main()
{
  int t;
  clrscr();
  for (t=30; t>=1; t=t-3)
 getch();
}
```

Output; 30 27 24 21 18 15 12 9 6 3

Explaination →
In this program we are printing number from 30 to 3 with decrementing t in every iteration by 3.

Scanned by CamScanner

— Arrays —

Array →

- An array is a data structure which can structure which can store multiple elements of same type under one name.
- In another way array is a collection of variables of same type all of which referred by a common name. The declaration of array is done as —

Syntax →

Type arrayname a[ ]

int a[5];

* Important points about array →

1. Array is a homogeneous data structure which stores elements of ~~different~~ similar type

5 elements → int type

10 elements → float type

20 elements → char type

All the elements share the same name and each can be modified individually for e.g.

int a[5]; a[0]=30; a[2]=56;

Now changing value in a[0] does not affect value of all other elements.

The array name gives the base address of the ~~array~~ array. Once its base address we can ~~refer to address~~ we can reach to address of any element and find out its value.

The amount of storage required by an array depends upon its type and its size.

19 JUNE 2014 THURSDAY

Scanned by CamScanner

```
#include <stdio.h>
#include <conio.h>
Veide main()
{
  clrscr();
  int a[5];
  int i;
  for (i=0;
```

Example →

```
#include <stdio.h>
#include <conio.h>
void main()
{
  int a[5],k;
  for (k=0; k<5; k++)
  {
    printf ("age");
    scanf ("%d", &a[k]);
  }
  for (k=0; k< 5; k++)
  {
    a[k] = a[k]+10;
  }
  for (k=0; k<5; k++)
  {
    printf ("%d", a[k]);
  }
  getch();
}
```


Scanned by CamScanner

Loading time allocation of variables called Static allocation. Array is a collection of similar type of values.
It is homogenous in nature

Ex→
```
#include <stdio.h>
#include <conio.h>
void main()
{
  int a[100], k, n;
  clrscr();
  for (k=0; k<n; k++)
  {
    printf("age");
    scanf("%d", &a[k]);
  }
  for (k=0; k<n; k++)
  {
    a[k]=a[k]+10;
  }
  for (k=0; k<n; k++)
  {
    printf("%d", a[k]);
  }
  getch();
}
```


(1) Declaration of 1-D array -
Syntax - type array name[size];
eg - int a[10];

(2) Memory allocation of 1-D array -
int a[5] 110 byte [1 array - 2 bytes]

a[0]	a[1]	a[2]	a[3]	a[4]

Scanned by CamScanner

25 JUNE 2014 WEDNESDAY TATA HITACHI

(3) Initialization of 1-D array

(a) with declaration

int a[5] = {1, 3, 9, 4, 8, 12} (a_0, a_1, a_2, a_3, a_4 - stored)

or int a[5] = {13, 9}; (a_0, a_1 stored)

int a[] = {13, 9, 4, 8} (size of array is 4)

Char a[10] = {a, b, c, d, e, f, g, h, i, j};

int a[5] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11}

(remaining are not reserved hence risky for the programme but at same time no error occurs)

(6) int a[5];

a[0] = 13;

a[2] = 14;

a[3] = 12; ---- So on

During run time through keyboard

int a[5];

int k;

JUNE 2014 THURSDAY 26

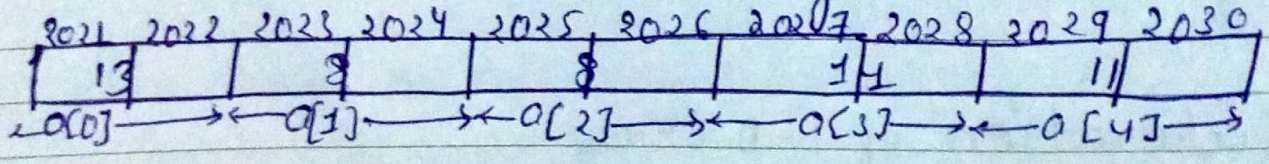
for (k=0; k<5; k++)

{

scanf("%d", &a[k]);

}

How initialization actually done:



Base address + (Size * index number)

2021 + (2 * 0)

=> 2021

Scanned by CamScanner

```
#include <stdio.h>
#include <conio.h>
{
  int a[5] = {13, 9, 8, 14, 17};
   int k;
    for (k=0; k<10; k++)
     {
       printf ("%d", a[k]);
     }
   getch();
}
```

→ Total and average of array

```
# include <stdio.h>
# include <conio.h>
void main()
  {
    int a[100], n, i, t=0;
     float avg;
   printf ("How many values");
   scanf ("%d", &n);
   for (i=0; i<n; i++)
  {
   printf ("Enter %d value", i+1);
   scanf ("%d", &a[i]);
  }
for (i=0; i<n; i++)
   {
     t = t + a[i];
   }
  avg = float (t)/n;
  printf ("Total is %d \n", t);
  printf (" Average is % \n", avg);
  getch();
}
```

28 JUNE 2014 SATURDAY

29 SUNDAY

Scanned by CamScanner

30 JUNE 2014 MONDAY

TATA HITACHI

JUL 2014

S	M	T	W	T	F	S	S	M	T	W	T	F	S
		1	2	3	4	5	6	7	8	9	10	11	12
13	14	15	16	17	18	19	20	21	22	23	24	25	26
27	28	29	30	31									

* Maximum and minimum of array -

```
{
 int a[100], n, i, max, min;
 printf ("How many values");
 scanf ("%d", &n);
 for (i=0; i<n; i++)
  {
   printf ("Enter %d value", i+1);
   scanf ("%d", &a[i]);
  }
max = a[0];
min = a[0]
for (i=1, i<n; i++)
 if (a[i] > max)
  {
   max = a[i]
  }
 if (a[i] < min)
  {
   min = a[i]
  }
printf ("maximum is %d /n", max);
printf ("minimum is %d /n", min);
 getch();
}
```

JULY 2014 TUESDAY 01

* Print only even number of array →

```
#include <stdio.h>
#include <conio.h>
Void main ()
 {
  clrscr();
  int a[100], n, i;
  print f ("How many values");
```

Scanned by CamScanner

```
scanf("%d", &n);
for (i=0, i<n, i++)
  {
    printf ("Enter %d value", i+1);
     scanf ("%d", &a[i]);
  }
for (i = 0; i<n; i++)
  if (a[i] % 2 == 0)
  {
  printf ("%d", a[i]);
  }
else
  {
    printf (" no. it is not even");
  }
getch();
}
```

03 JULY 2014 THURSDAY

```
/* Count even no ——
{
int a[100], n, i, k=0;
  :
  k=k+1;
```

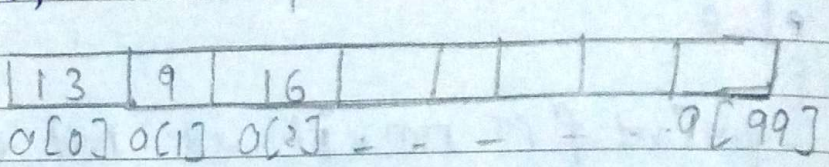
— Searching programme —

```
#include <stdio.h>
#include <conio.h>
void main()
{
  clrscr();
int a[100], n, i, x; f=0;
 printf ("How many values");
   scanf(" %d", &n);
for (i=0; i<n, i++)
  {
```

Scanned by CamScanner

```
printf("enter %d value", i+1);
scanf("%d", &a[i]);
}
printf("Enter the value to be searched");
scanf("%d", &x);
for(i=0, i<n, i++)
{
    if(a[i]==x)
    {  f=1;
       break;
    }
}
if(f==1)
{
    printf("yes present at %d position",
           i+1);
}
else{ printf("Not present");
}
getch();
}
```

n = 7	7	7
i = 0	1	2
x = 16	16	16
f = 0	0	1



* Binary Method (when all no. are in a order)

```
#include<stdio.h>
#include<conio.h>
void main()
{
    int a[100], n, i, x, f, l, m;
    printf("How many values");
    scanf("%d", &n);
    for(i=0; i<n; i++)
{
    printf("Enter %d value", i+1);
```


Scanned by CamScanner

```
scanf ("%d", &a[i]);
}
printf ("Enter the value to be search");
scanf ("%d", &x);
f = 0;
l = n-1;
while (f<=l)
m = (f+l)/2
{
if (a[m]>x)
l = m-1;
if (a[m]<x)
f = m+1;
if (a[m]==x)
break;
if (f<=0)
printf ("yes at %d position", m+1);
}
else
{
printf ("not present");
}
getch();
}
```

n = 7
f = 7
x = 56
f = 0 #, 5
l = n-1 = 6, 4
m = 3, 4, 5

n = 4
x = 48
f = 0, 4
l = 6, 4
m = 3, 5, 4

Two Dimensional (2-D) Array –

A two dimension array can be thought of a matrix with rows and columns.
for example if we declare 2-D array –

```
int arr [3][4];
```

2D array arr of 3 rows and 4 columns,

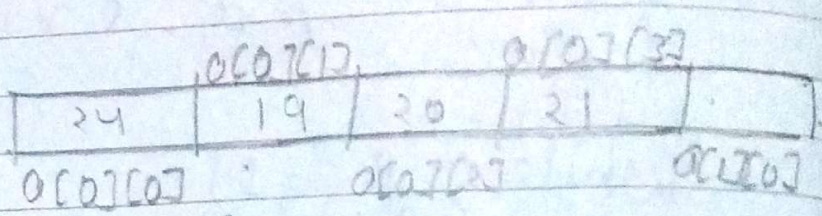
Scanned by CamScanner

09 | JULY 2014 | WEDNESDAY

TATA HITACHI

Syntax → type array name [row][column]

Ex →




```
#include <stdio.h>
#include <conio.h>
void main()
{
int a[5][4], i, j;
clrscr();
for (i=0; i<5; i++)
  {
    for (j=0, j<4, j++)
    {Scanf ("%d", &a[i][j]);
    } Printf(" Enter value");
}
print f ("your values are \n");
for (i=0; i<5; i++)
 {
  for (j=0; j<4; j++)
   {
    print f ("%d", a[i][j]);
   }
 print f ("\n");
  getch();
}
```

JULY 2014 | THURSDAY | 10

i = 0, 1
j = 0, 1

Output

Enter value = 2
Enter value = 19

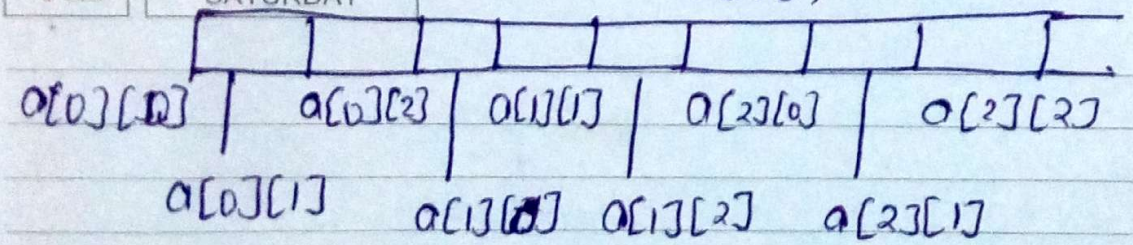
Scanned by CamScanner

	I	II	III	IV
0	1	21	19	20	21
1	2	17	16	13	12
2	3	10	30	30	29
3	4	25	26	28	24
4	5	18	19	16	14

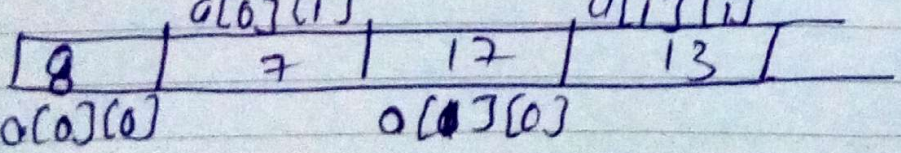
we can also write one thing —
printf (" enter value of %d row and %d column", i+1, j+1);

* Declaration of 2-D array —
Syntax → type array name [row][column]
e.g. → int a[4][3];
float fee [10][10];

Memory allocation of 2-D array →
int a[4][3];



(3) Initialization of 2-D array —
(i) with declaration —
(a) int a[4][3] = { {4,2,6}, {3,1,8}, {8,10,2} }
(b) int a[][3] = { {2,9,3}, {3,8,4}, {7} };
(c) int a[5][4] = { {6, 2}, {3,4}, {8} };
13 SUNDAY (d) int a[3][2] = { {8,7,19,6}, {7,17,13} };



Scanned by CamScanner

After declaration —

```
int a[3][4];
a[0][0] = 6;
a[0][1] = 19;
a[2][3] = 18;
```

* During run time through keyboard —

```
int a[5][4];
Scanf("%d", a[0][0]);
Scanf
{int a[5][4];
  for ( )
```


Scanned by CamScanner