Choose Your Future
by Making Right Choices
in the Present!

Parul® University
NAAC A++ Unit: 1

Page No.:
Date:

History of 'C' language

- It was developed in 1972 By Dennis Ritchie at Bell laboratory.
- Dennis Ritchie is known as founder of 'C' language.
- It was developed to overcome the problems of previous languages such as B/BCPL
- Initially C language was developed to write UNIX operating system.
- It inherits the features of previous languages such as B/BCPL.
- C is a general-purpose, procedural programming language.

Features of C Programming

1. Simple
2. Portable / machine independent
3. Mid level language
4. Structure programming language
5. Rich library
6. Fast / Speed
7. Pointers
8. Money Memory management
9. Recursion
10. Extensible

Parul® University

THE
E YOUR
NY.

Parul® University
NAAC A++

Page No.:
Date:

IDE = Integrated Development Environment

Code block:

- Code block → Google (20.03)
- Online CTDB Compliner
- Visual Studio code (VS)
- Turbo C

```
#include <stdio.h>
    int main()
{
        Printf ("Hellow World");

        return 0;
}
```

include = The include ruler file
stdio = Standard input output Herder file
int = It's an datatype
main = It is an function

Parul® University
NAAC A++

Page No.:
Date:

Project 1 - CBP
CBP for code block Project

Conio - Consal input output Herder file
clrscr - to clear the screen
ctrl F9 - Show the output

Token
↓
keyword
↓
Statement / instruction
↓
Program

Input
↓
Processing
↓
Output

Parul® University

THE
E YOUR
NY.

.in

Parul® University
NAAC A++

Page No.:
Date:

* Program for finding Area of a circle

```
# include <stdio.h>
  void main()
{                                  %d - integer
    int radius                     %f - float
    float area;                    %c - char
    printf("Enter radius");
    scanf("%d", radius);

    area = 3.14 * radius * radius;

printf("Area of circle is = %f", area);

    getch();

- // single line comment

- /* multiline comment */
```

Parul® University
NAAC A++

Page No.:
Date:

\# Programe Development steps

→ Structure of C Program

1. Problem Defination
2. Problem Analysis
3. Algorithm Development
4. Coding and ~~Debugging~~ Documentation
5. Testing and Debugging
6. Maintenance

- The programme develop life cycle is a set of steps or phases that are used to programme in any programming language such a C, C++, Java, etc

1. Problem Defination

Step-1. Define the Problem statement

step-2. Decide the boundaries of Problem

step-3 Need to understand problem statement

Parul® University
NAAC A++

Page No.:
Date:

Step-4. What is our requirement

Step-5. What should be output of problem solution.

(2) Problem Analysis
- Determine the requirement like variable, function etc to solve the problems.

- Gather the require resources to solve the problems define in the problem defination

(3) Algorithm Development
- Develop a step by step process to solve the problem using the specification given in the previous phase.

(4) Coding and Documentation
This phase uses a programming language to write or implement the actual programming instruction for the steps.
Define in the previous phase. The programming language such as C, C++, Java, etc.

Parul® University
NAAC A++

Page No.:
Date:

(5) Testing and Debugging
To check whether the code written in the previous steps is solving the specify problem or not.

- It means we test the Programme whether it is solving the program for various input data values or not.

- We also test that whether it is providing desired output or not.

(6) Maintenance
- The programme is actively used by the users.

- If any enhancement found in the phase all the phases are to be repeated to make the enhancement

- In this phase the soln is used by the end-user.

- If the users encounter any problem or who wants any enhancement that we need to repeat all the phases from the starting.

Parul® University
NAAC A++

Page No.:
Date:

C Charader set:- C language has set of character which include alphabets Digits and special symbols.

- Clanguage support total of 256 character
- Alplabates ← lower case a to z
  Uppercase A to Z

- Digites - 0 to 9

- Special Symbols:- Clanguage support rica set of special symbols that include symbols to proform mathematical operation to check condition, white spaces, back spaces and other special symbols.

Special Characters are #, $, £, ¥, ¢, ¤, @, ¶, *

C - Tokens
- Every C programme is a collection of instruction.
- Every instraction is a collection of some individual units
- Every smallest individual units of the C programe is called token.

Parul® University
NAAC A++

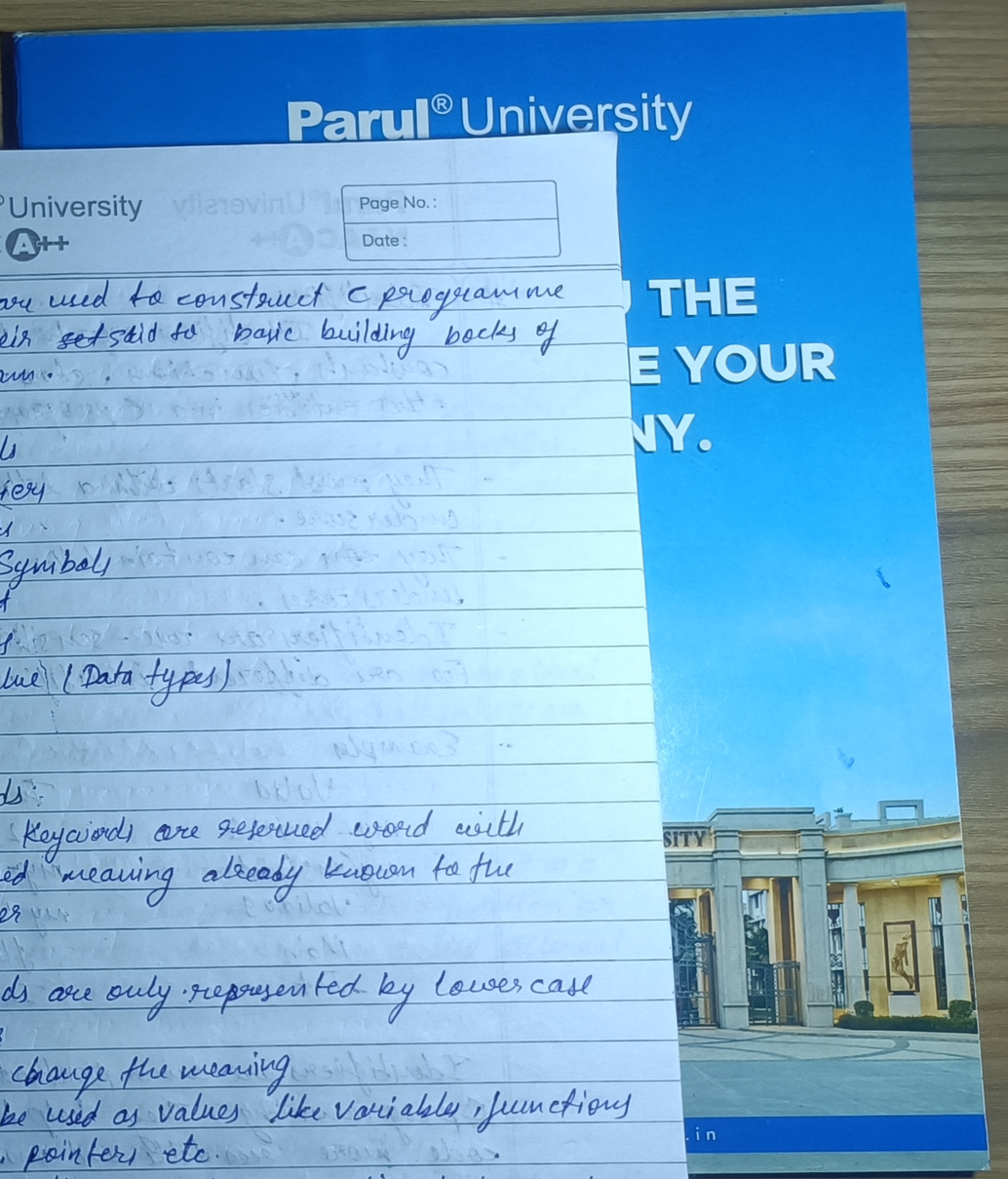
Page No.:
Date:

- Tokens are used to construct C programme and their set said to basic building bocks of C program.

- Keywords
- Fudentifiers
- Operaters
- Special Symbols
- Constant
- Strings
- Data value (Data types)

Keywords:
Keywords are reserved word with predefined meaning already known to the compiler

- Keywords are only represented by lowercase letter
- Cannot change the meaning
- Cannot be used as values like variable, functions array, pointers etc.
- It's specifies some kind of action to be proform by the compiler.



Parul® University
NAAC A++

Page No.:
Date:

Identifier

Identifiers refer to variables, constants, functions, structures, and other entities in a C program.

- They must start with a letter and an under score.
- They can contain letter, digits, and underscores.
- Identifiers are case-sensitive, so foo and Foo are different identifiers.

- Example

Valid	In Valid
my_variable	123
myConstant	my-Variable
Valid09	my Variable!
Main	float
_Int	In Valid

Identifiers are an important part of C programming. They help to make your code more readable and maintainable

Parul® University
NAAC A++

Page No.:
Date:

- Comma and blank space are invalid while naming an identifier.
- Max length of the identifier is 31 character.
- Key words cannot be named as an identifier.

- There are two types of identifier

1. Internal Identifier

Identifiers which are used as a local variable or are not used in external linkage are called internal identifiers.

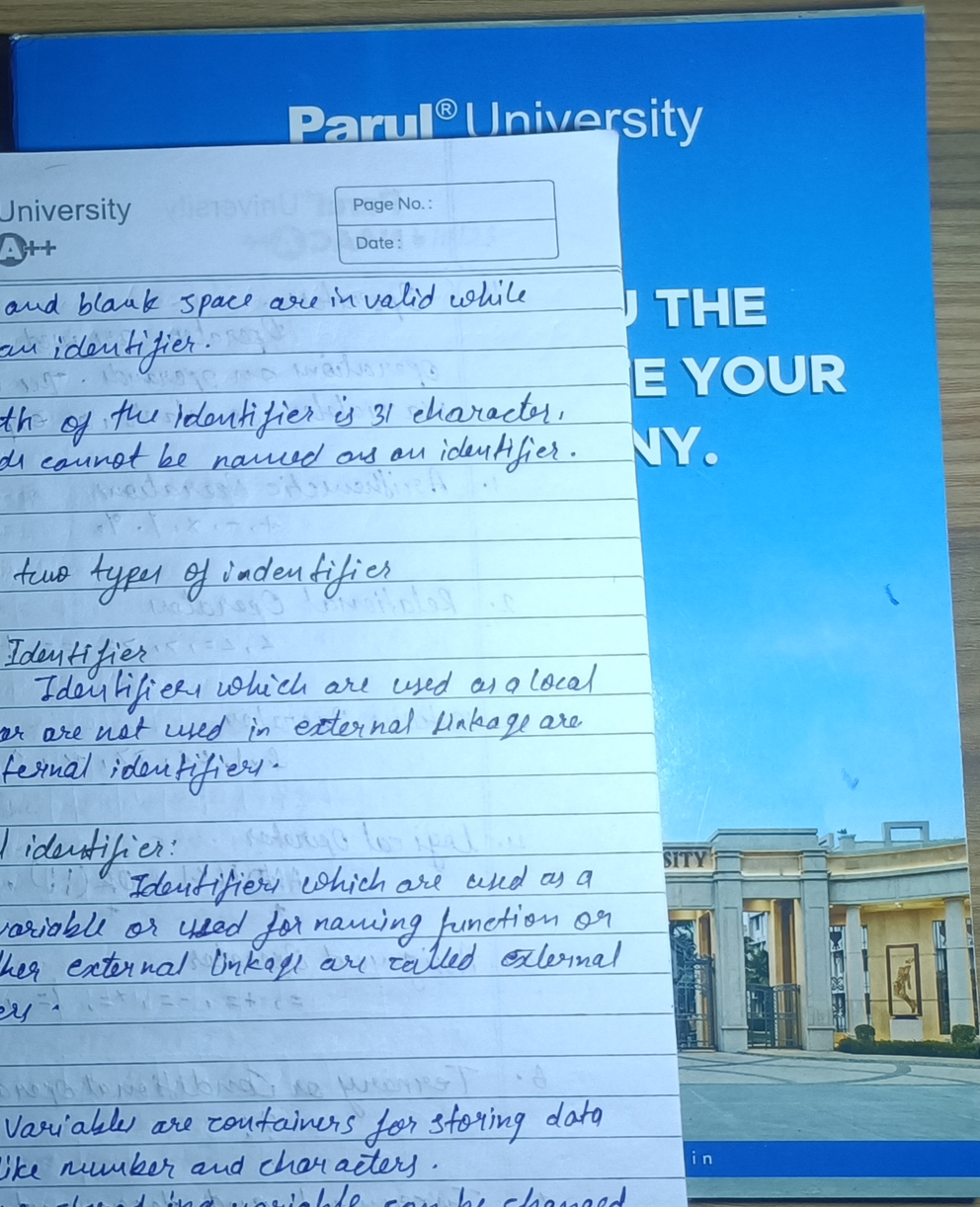
2. External identifier:

Identifiers which are used as a global variable or used for naming function or any other external linkage are called external identifiers.

Variable

Variables are containers for storing data values. like number and characters.

- The value stored in a variable can be changed during program execution.



Parul® University
NAAC A++

Page No.:
Date:

Operators:

Operators are used to perform operations on operands. The operands can be variables, constants, or expressions.

1. Arithmetic Operations
   +, -, *, /, %

2. Relational Operator
   <, <=, >, >=, ==, !=

3. Bitwise Operator
   &, |, <<, >>, ~, ^

4. Logical Operator
   AND (&&), OR (||), NOT (complement)
   * * !

5. Assignment Operator
   =, +=, -=, *=, /=, %=

6. Ternary or Conditional Operator (if-else)
   ?:

7. Increment and decrement Operators
   ++, --

Parul® University
NAAC A++

Page No.:
Date: 6/10/23

8. Size of (other operators)

(5) Assignment

=	a=b	~~a=a+5 = 5+5 = 0~~
+=	a+=5	a=a+5 = b+5 = 11
-=	a-=5	a=a-5 = b-5 = 1
*=	a*=5	a=a*b = 30
/=	a/=5	a=a/5 = 1
%=	a%=5	a=a%5 = 1

(7) Increment and decrement Operators

a=2

Pre Increment ++a, 1+2

Post Increment a++, 2+1

Pre Decrement --a, -1

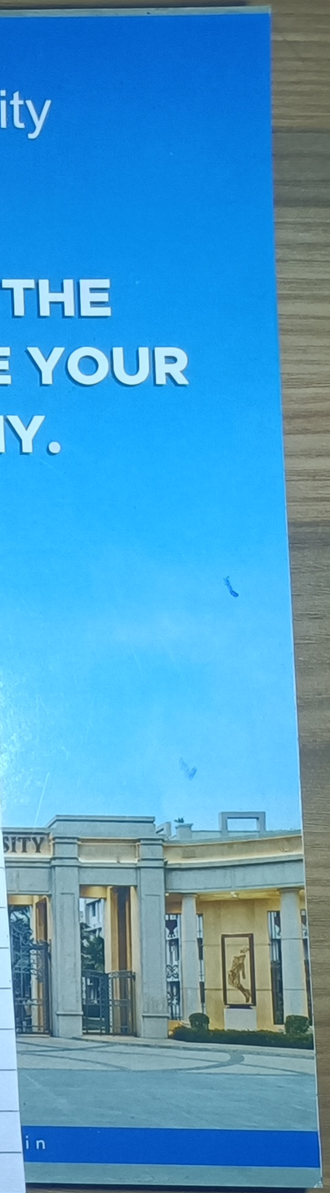
Post Decrement a--, 2-1

(6) Ternary / Conditional / if-else

a=2, a=3

exp1? exp2: exp3;

(a>b)? a: b;



Parul® University
NAAC A++

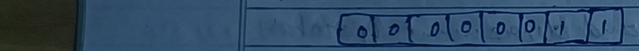
Page No.:
Date:

Bitwise Operators

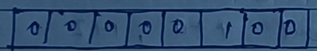
Symbols	Meaning
&	Bitwise AND
|	Bitwise OR
^	Bitwise XOR
~	" Comp
<<	" Left Shift
>>	" Right Shift

Bitwise AND

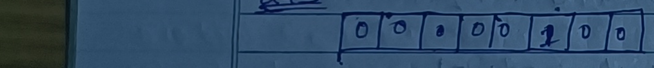
a = 3



a = 4

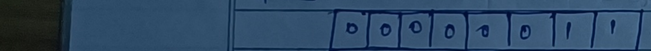


a & b

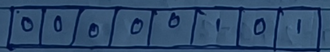


2	3	1
	1	1

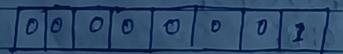
a = 3



a = 5



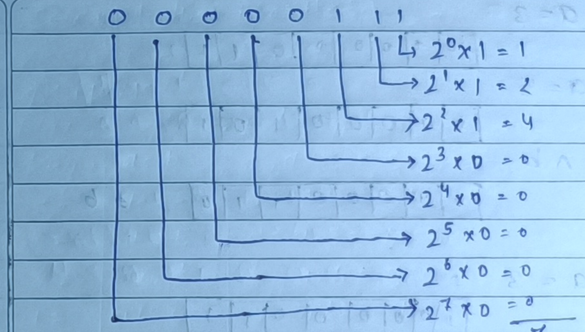
a & b



2	5	1
2	2	0
2	1	

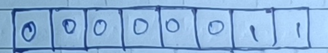
Parul® University
NAAC A++

Page No.:
Date:

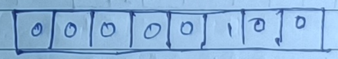


Bitwise OR

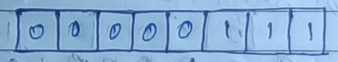
a = 3



a = 4



a | b = 7



Bitwise XOR

a = 0 , b = 1 $a \oplus b$, $a \wedge b$

$a\bar{b} + b\bar{a}$

$0|0 + 1|1$

$0 + 1 = 1$

Parul® University
NAAC A++

Page No.:
Date:

a = 3

0	0	0	0	0	0	1	1

a = 5

0	0	0	0	0	1	0	1

a & b

0	0	0	0	0	1	0	0

= 6

a = 3

0	0	0	0	0	0	1	1

a = 4

0	0	0	0	0	1	0	0

a & b

0	0	0	0	0	1	1	1

= 7

Bitwise Left Shift x2

a = 6

0	0	0	0	0	1	1	0

a << 1 : 0 0 0 0 0 1 1 0 0

x2

0	0	0	0	0	1	1	0	0

= 12

Bitwise Right Shift /2

a = 12

0	0	0	0	0	0	1	0

a >> 2 : 0 0 0 0 0 0 1 0

>> 0 0 0 0 0 0 0 0 = 0

a = 10 : 0 0 0 0 1 0 1 1

a >> 7 : 0 0 0 0 0 0 1 1

Parul® University
NAAC A++

Page No.:
Date:

Bitwise Comp (~)

a = 2

~a = -3

0	0	0	0	0	0	1	0

1's 1 1 1 1 1 1 0 1

2's 0 0 0 0 0 0 1 1

+ 1

0 0 1 1 = 3

a = 10

~a

= -11

0 0 0 0 1 0 1 0

1's 1 1 1 1 0 1 0 1

2's 1 1 1 1 0 1 1 0

+ 1

1 1 1 1 0 1 1 1 = -1

a = 5

~a = -4

0	0	0	0	0	1	0	1

1 1 1 1 1 0 1 0

+ 1

1 1 1 1 1 0 1 1

Parul® University
NAAC A++

Page No.:
Date:

2 | 10
2 | 2 - 0
1

a = -5
~a = 4

| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |

1's 1 1 1 1 1 0 1 0

2's 1 1 1 1 1 0 1 1

is 0 0 0 0 0 1 0 0 = 4

2 | 5 | 1
2 | 2 | 0
1

2 | 4 | 0
2 | 2 | 0
1

2 | 0 | 0
2 | 3 | 1
1 | 1

a = -6 0 0 0 0 0 1 1 0

1's 1 1 1 1 1 0 0 1

2's 1 1 1 1 1 0 1 0

is 0 0 0 0 0 1 0 1 = 5

Parul® University
NAAC A++

Page No.:
Date:

Functions in stdio header file

printf() fopen()
scanf() fclose()
getc() remove()
putc() fflush()
gets()
puts()

→ Operator precedence.

Precedence Operators	Associativity
(), [], ->, -, ++ --	left to right
++ --, +, -, !, ~, (type)	
*, &, sizeof	right to left
* / %	left to right
+ -	left to right
<< >>	left to right
< <=	left to right
> >=	left to right
== !=	left to right
&, ^, |	left to right
&&	left to right
||	left to right
? : Ternary Operator	right to left
=	right to left
+= -=	right to left
*= /=	
%= &=	
^= |=	
<<= >>=	
,	left to right

Parul® University
NAAC A++

Page No.:
Date:

```
int a,b,c,d,e;
a = 20      e = (a+b) x c /d;
b = 10        = 30 x 15/5 = 90
c = 15
d = 5       e = ((a+b) * c)/d;
              = 90

e = (a+b) * (c/d);      e = a+(b*c)/d;
  = 90                    = 50
```

Parul® University
NAAC A++ Unit = 3

Page No.:
Date: 17/10/23

Conditional flow Statement
Iterative, Jumping and Pointers

Conditional flow

```
int age;
    age = 25;
if(a>=18)
{
    // Town
    Printf("Eligible");
else {
{
    Printf("Not eligible");
}
int mark
  mark=35;
if(mark >= 35)
{
  Printf("Pass");
}
else
{
    Printf("Fail");
}
```