Date :
Page :

C Tokens:

C Tokens:
- Keywords
- constants
- Identifiers / variables
- Strings
- special symbols
- operators

1) In C-programming any symbol, individual word, characters etc are called tokens.

2) Tokens are basic building blocks of C-programming.

a) Keywords:-

Keywords are the words whose meaning are already defined by compiler. We can't use keywords as variable.
Eg: void, char, int, float, break, switch.

b) Constant:-

Constant are expression with fixed value. It is identifier that have same value throughout the program when we declare constant it is somewhat like a variable declaration but it is declared alone with keyword. It's key word is 'const'. There are two type of constant:- literal and symbolic constant.

i) Literal constant: A literal constant is value type directly into computer

ii) Symbolic constant: A symbolic constant is the name just like a variable initialized with const.

Date :
Page :

→ Two ways to declared symbolic constant

Syntax 1:

<const><data type><identifier><const value>

Eg:-

```
const int a=10;
```

or,

```
int const a =10;
```

Syntax 2:

By using pre processor

define <indentifier><value>

Eg:-

```
# define Pi 3.14
```

c) Variable:-

It is a container to store the data. It is a name given to the memory location where the actual data is stored. Variable names may have different data types to identify the type of value stored. Eg:- int a; this means that 'a' can store only one integer.

→ Types of variable:

i) Local variable:

- Local variable is variable having local scope.
- local variable is accessible only from function or block in which it is declared.
- Local variable have higher priority than the Global variable.

Date :
Page :

Eg:

```
#include <stdio.h>
void main()
{
 int var 1 = 10;
{
 int var 2 = 20;
 printf ("%d %d", var 1, var 2);
}
}
```

ii) Global variables:-

→ Global variable is a variable that is globally accessible.

→ Scope of Global variable is throughout the program.

Eg:

```
#include <stdio.h>
 int var = 10;
 void message ();
 void main()
 {
  int var = 20;
  {
  int var = 30;
   printf ("%d", var);
  }
   printf ("%d", var );
   message ();
  }
   void message()
   {
   Print ("%d", var);
  }
```

iii) Static variable:

A static variable can be either Global or local variable. Both are created by preceding the variable declaration with the keyword 'static'.

a) Local static:

A local static variable is a variable what can maintain its value from one function call to another and it will exists until the program ends. When a local static variable is created its should be assigned with initial value. If it is not then, the value will be 0 by default.

b) Global static:

A Global static variable is one that can be excess only in the file where it is created.

Rules for constructing variable names:-

1) We need to use only allowed characters. Following are the allowed characters.

a) Underscore (_)
b) Capital Letters (A-Z)
c) Small Letters (a-z)
d) Digits (0-9)

2) Blanks and commas are not allowed.
3) No special symbols other that underscore (_) are allowed.
4) First character should be alphabet or underscore.
5) Variable name should not be reserved word.

Escape sequence / Backslash character:-

① They are non-printable characters.
② They consist of two characters, combination of two characters represent single characters.
③ Each escape sequence has unique ASCII value.
④ Each and every escape sequence starts with backslash(\)

Escape Sequence	Meaning
\a	Audible bell
\b	Backspace
\f	Form feed
\n	New line
\r	Carriage return
\t	Horizontal tab
\v	Vertical tab
\'	Single quote
\"	Double quote
\?	Question mark
\\	Backslash
\0 - zero	Null

Data Types:

1) A data type is the type of data use in the program.
2) We need to declare the type of each every variable or constant that we are going to use in our program.
3) Data type is the storage representation and machine instruction to handle constant and variable.
4) There are four kind of data types:
They are:- empty, primary, user defined & derived.

a) Primary data type:

Primary data type is such data type which doesn't use any modifiers. Primary data type can be characterized as follow:-

Integers

Signed	Unsigned
int	unsigned int
short	unsigned short int
long int	unsigned long int

eg:- int a = 5

In signed the number can be positive or negative in result. Whereas, in unsigned the number will be positive result.

Floating point:

float, Double float, long double

Eg: float b = 5.6

Character:

char

Eg: Char a = 'i'

char a[5] = "Nepal"

Data Type	Memory (bytes)
short int	2
unsigned short int	2
unsigned int	4
int	4
long int	4
unsigned long int	4
long long int	8
unsigned long long int	8
signed char	1
unsigned char	1
float	

Date : .....
Page: .....

Data Type	Memory (bytes)	Range	Format specifier
short int	2	-32,768 to 32,767	%hd
unsigned short int	2	0 to 65,535	%hu
unsigned int	4	0 to 4,294,967,295	%u
int	4	-2,147,483,648 to 2,147,483,647	%d
long int	4	-2,147,483,648 to 2,147,483,647	%ld
unsigned long int	4	0 to 4,294,967,295	%lu
long long int	8	-(2^63) to (2^63)-1	%lld
unsigned long long int	8	0 to 18,446,744,073,709,551,615	%llu
signed char	1	-128 to 127	%c
unsigned char	1	0 to 255	%c
float	4		%f

double	8		%lf
long double	16		%Lf

b) User defined data type:

The user defined data types can be later used to declare variable. User defined data types define using type data and ~~user~~ enum.

As an example let us consider how we can define a new type named "letter". This letter, can then be used in the same way as the predetermined data type of C.

- Syntax:- <type def><data type><new type>;<new type> <variable>;

Eg: type def int letter;
letter a;
a = 5;

Here, a new type letter is created whose data type is integer. Later this letter is used as data type to declare the variable name & address.

c) Derived data type:

Different user defined data type can be created using fundamental data types. They are array, union, structure, function are derived. data types.

Statements

a) Statements

to do certa
of statemen

b) Expressions
A
that repres
own value
Eg: sum

c) comments
Co
in the pro
the instru
comments

1) Single li
Slash "//"

2) Multi-lin
It
It treats

Statements and comments.

a) Statements:

A statements is a command given to the computer to do certain task. A computer program is made up of collection of statements. Eg:- printf ("Hello");

b) Expressions:

An expression is the legal combination of symbols that represent a value. Each programming language have their own values for what is legal and illegal expressions.
Eg: sum = a+b;

c) comments:

Comments are the set of lines used to tell something in the programs. The compiler doesnot treat the comments as the instructions of the program. There are two types of comments in C-programming:-

1) Single line:-

It is represented by ~~single forward~~ double forward slash "//". It treats only one line as comment.

2) Multi-line:-

It is represented by single forward slash and star (/*). It treats more than one line as a comment.

Input and Output:

a) Printf ( ):-

The C language comprises of general library function that carry out various commonly used operations or calculations For eg: There are library functions that carry out standard input/output operations functions that operate on strings and functions that carry out various mathematical calculations.

Functionally similar library functions are grouped together inseperate library files. In order to use a library function it may be necessary to include certain specific information within the main portion of the program. This is accomplished with the following pre-processor directive statement.

The I/O statement can be categorized as formatted and unformatted I/O. The unformatted I/O statement doesn't specify few the input and output are carried out by the formatted I/O. Statement determine the format in which the input & output are executed. The function such as getc(), putc(), getchar(), putchar(), are considered as unformatted I/O because they don't contain any information about the formats specifier. They simply take variable as parameters. Formatted I/O generate output under the control of a format string which consists of literal characters to be print & it also can contain format specifiers which request the other arguments be fetch, formatted and inserted into the variable.

Date :
Page :

Conversion specification :-

Format Specifier	Meaning
%c	Data item is displayed by string characters.
%d	Data item is displayed as single integer.
%u	Data item is displayed as unsigned integer.
%f	Data item is displayed as floating point values.
%e	Data item is displayed ~~as an octal integer~~ as floating point value with exponent.
%o	Data item is displayed as an octal integer
%x	Data item is displayed as hexadecimal integer
%s	Data item is displayed as string.

b) Scanf( ):-

Scanf( ) can be used to get input from users. scanf( ) is a c-library function that can be used to take input. This function can be used to enter any combination of numerical values single character & strings.

Syntax:-

Scanf("controlstring", arg1, arg2, ...);

There control string consist at any format conversion specifiers &arg1, arg2..... are arguments control string and argument are seperated by comma.

Eg:-

```
#include <stdio.h>
#include <conio.h>
void main()
{
int a;
char b;
float c;
printf("Enter int, character and float");
scanf("%d %c %f", &a, &b, &c);
printf("int = %d, char = %c, float = %f", a, b, c);
getch();
}
```

Date :
Page :

c> getchar() and putchar() function:-

Single character input can be entered into the computer using the C library function getchar(). The function doesn't require any argument in a pair at empty ~~para~~ that follow getchar().

Syntax:

```
char ch;
ch = getchar();
```

Single character output can be displayed using the (library function) putchar(). The character being transmitted will normally be represented as a character type variable. It must be expressed as an argument to the function.

Syntax:-

```
putchar(character variable);
```

Eg:-

```
char ch;
putchar(ch);
```

WAP to take a single character as input and display it.

⇒
```
#include <stdio.h>
#include <conio.h>
  void main( )
  {
   char ch;
   clscr( );
   printf("\n Enter a character : \t");
   ch = getchar( );
   printf("\n This character is ");
   putchar(ch);
   getchar( );
  }
```

*. The gets( ) and puts( ) function:-
→ C provides the function gets & puts for string input/output. Though the operation can be done using scanf( ) and printf( ) function with (%s) conversion character there is a limitation with scanf( ). i.e. scanf( ) can accept a string after space the function gets( ) can accept the whole string along with space.

Syntax: gets (variable);
Example:- char name[50];
gets(name);
Syntax: puts (variable); puts("string");
charname [50);
puts(name);
puts ("same text");

WAP to take string as input

→
```
#include <stdio.h>
#include <conio.h>
  void main( )
  {
   char str [50];
   clrscr( );
   printf ("\n Enter string
   gets(str);
   puts ("\n Your string
   puts (str);
   getch( );
  }
```

WAP to take string as input & display it :-

```
#include <stdio.h>
#include <conio.h>
void main()
{
 char str [50];
 clrscr();
 printf ("\n Enter string : ");
  gets(str);
   puts ("\n Your string is : \t ");
    puts (str);
    getch();
}
```

Operators & Expressions (V.VIP)

An operator is a symbol that helps to perform certain mathematical calculations and logical manipulation. Eg: +, -, *, ==, =, >, <. The data item that operator acts upon is called operands on the basis of number of operands they take, operators can be classified into:-

a) Unary operator
b) Binary operator
c) Ternary operator

a) Unary operator:

The operator which use only one operand for operation is called unary operand. Eg:- Increment and decrement i.e a++, --a, ++a etc.

b) Binary operator.

The operator which use two operand for operation is called binary operator. Eg:- +, -, =, <, > i.e. 2+2, 2-2

c) Ternary operation:

It is also known as conditional operator. It requires three operands. It evaluates the condition and then assigned the required expression.

Syntax:

Condition ? Expression 1 : Expression 2;

If condition is true then expression 1 is evaluated and this becomes the value of the conditional expression and if condition is false, then expression 2 is evaluated.

Date : .....
Page: .....

xample :

Find the greatest among three numbers

```
#include <stdio.h>
#include <conio.h>
void main()
{
    int a, b, c, large;
    clrscr();
    printf("\n Enter three numbers : ");
    scanf("%d %d %d", &a, &b, &c);
    large = a>b ? (a>c ? a:c) : (b>c ? b:c);
    printf("\n Largest number is %d"; large);
    getch();
}
```

Type of operators (on the basis of operation)

1) Arithmetic operator
2) Relational operator
3) Logical operator or Boolean operator
4) Assignment operator
5) Increment & Decrement operator
6) Conditional operator (ternary operator)
7) Comma operator
8) Size of operator

1) Arithmatic operator:

It is used for arithmatic operation like addition, substraction, multiplication & division.

Operator	Meaning	Example
+	Addition operator	a+b
-	Substraction operator	a-b
*	Multiplication operator	a*b
/	Division operator	a/b
%	Modulus operator	a%b

2) Relational operator:

Operators that are used for comparing relation between operands. In C-programming we can compare the variables using the relational operators. It is also called comparison operator as it is used to compare any two expression.

Date :
Page :

Operator	Meaning	Example
<	is less than	a<b
<=	less than equal to	a<=b
>	greater than	a>b
>=	greater than equal to	a>=b
==	equal to	a==b
!=	Not equal to	a!=b

3) Logical operator or Boolean operator:

Logical operators are used for combining logical expressions which are finally evaluated as true or false according to operator use. Logical operators are used for logical decision making in C-programming.

Operator	Meaning	Example
&&	logical AND	(a>b) && (a>c)
|| or |	logical OR	(a>b) || (a>c)
!	Logical NOT	!(a==b)

4) Assignment operator:

It is used for assigning a value or a result of an expression to an identifier. It is the combination of arithmetic operator & assignment operator & performs the task. They are also known as short hand operator.

Operator	Meaning	Exo

→

Date :
Page :

Operators	Statement with short hand operator	Statement with simple assignment
+=	a+=1	a = a+1
-=	a-=1	a = a-1
*=	a*=1	a = a*1
/=	a/=1	a = a/2
%=	a%=b	a = a%b

Here, Arithmetic operator performs it's operation first and their assignment operators assigns the value.

5) Increment / Decrement operators :-

These operators are also known as unary operator which performs operation upon one operand. There are two types of numeric operator.

a) Prefix :- ++variables; --variables;
b) Postfix : variables ++; variable --;

i) Eg : m = 5
y = ++m;
The value of y & m would be 6.

ii) Eg : m = 5;
y = m++;
The value of y is 5 & m is 6.

a) Prefix operator :
It first adds 1 to be operand & then result is assigned to the variable on the left.

b) Post

left

6) Cond

It e
exp

If cond
this be
if con

7) Com
togeth
and +
combi

Eg :

8) Size

data
memor

b) Postfix operator:

It assigns the value to the variable on the left first and then increases or decreases the operand.

6) Conditional operator (Ternary) :- (?:)

Conditional operator requires three operands. It evaluates the condition & then assigns the required expressions.

Condition-1? Expression 1 : Expression 2

If condition 1 is true then expression 1 is evaluated and this becomes the value of the conditional expressions & if condition 1 is false, then expression 2 is evaluated.

Eg: a=5;
b=9;
x=(a>b)?a:b

7) Comma operator : It is used to link related expression together. It evaluates the expression from left to right and the value of right post expression is the value of combined expression.

res = (num1 , num2);

Seperator : In the functional call declaring variable etc.

Eg:- int num1 = 10, num2 = 20;

8) Size of operator( ):-

The size of operator gives the size of the data type of variable in terms of bytes occupied in memory. Eg: int k;
n=size of (x);

char n;
size of (n);

Expression:

An operator expression is the combination of variables, constant & operators. In C every expression results same value that can be assigned to the variable.

→ Precedence of arithmetic operators:

High priority: *, /, %

Low priority: ~, +, -

Rules for evaluation of expression:-

1) Paranthized sub expression from left to right are evaluated.
2) If paranthesis are nested the evaluation begins with the inner nested sub expression.
3) The precedence rule is applied in determining the order of application of operators is evaluating sub expression.
4) The associativity rate is applied when two or more operators of the same precedence level appeared in a sub expression.
5) Arithmetic expression are evaluated from left to right using the rule of precedence.

Precedence & Associativity:

Operator precedence determines the grouping of term in an expression. This affects how an expression is evaluate. Certain operators have higher precedence. For example: multiplication operator has higher precedence then addition operator. A list of operator with their precedences is given below:

Date: .....
Page: .....

Category	Operator	Associativity
postfix	( ) [ ] ->	Left to Right
Unary	++ - -	Right to Left
Multiplicative	* / %	left to Right
Additive	+ -	Left to Right
Relational	< <= > >=	Left to Right.
Equality	= = ! =	Left to Right
Logical AND	&&	Left to Right
Logical OR	||	Left to Right
conditional	?:	Right to Left
Assignment	= +, = -, = *, = /, =	Right to Left
comma	,	Left to Right

Example:

1. x = 7 + 3 * 2
   = 7 + 6
   = 13

2. y = 5 % 2 * 4
   = 1 * 4
   = 4

3. y = 5 * 4 % 2
   = 0 //