1. **What is Tokens? What are types of tokens?**

1. In a passage of text, individual words and punctuation marks are called **Token**.

**OR**

The smallest individual elements or units in a program are called as Tokens.

1. There are six types of tokens in C language. Which are listed below.
2. Keywords.
3. Identifier.
4. Constant.
5. String.
6. Operators.
7. Special symbols.
8. Tree diagram for tokens.

Tokens

Special symbols

S

Operators

String

Constant

Identifier

Keywords

**Fig:-Types of tokens**

1. Above types of tokens are explained below.
2. **Keywords:-**
3. It is also called as **Reserved words**.
4. They have fixed meaning.
5. They may or may not be in lower case.
6. And they also start with underscore
7. They do not used as Identifier (variable, function and pointer).
8. The list of keywords from C89 to C11 is listed below.
9. There are 44 keyword words in C language in C11.

|  |  |  |  |
| --- | --- | --- | --- |
| auto | \_Alignas\*\* | \_Alignasof\*\* | \_Automatic\*\* |
| breaks | \_Bool\* | case | char |
| \_Complex\* | const | continue | default |
| do | double | else | extern |
| enum | float | for | \_Generic |
| goto | if | \_Imaginary\* | inline\* |
| int | long | \_Noreturn\*\* | register |
| return | restrict\* | short | signed |
| static | sizeof | Static\_assert\*\* | struct |
| switch | Thread\_local\*\* | typedef | union |
| unsigned | void | volatile | while |

**Note:-**

1. **All pink color highlighted and no star keywords are in C89 standard.**
2. **All red color highlighted and \* mark keywords introduced in C90 standard.**
3. **All green color highlighted and \*\* mark keywords introduced in C90 standard.**
4. **From C90 standard keywords started from under score symbol followed by capital letter.**
5. **Identifier:-**
6. Name referred (or used) to variable, function and pointer are called as Identifier.
7. Rule for identifier is same as rule for variables.
8. All variables are identifier but all identifier are not variable.
9. Example:-
10. int a;

Here a is identifier.

1. int \*p;

Here p is identifier.

1. void get();

Here get function is identifier.

1. **Constant :-**
2. The value that cannot changed is called **Constant**.
3. If we are trying to change constant value then compiler give error.
4. For defining constant in C language we must use const keyword.
5. We must initialize constant variable at timing of defining variable.
6. Tree diagram for type of constant.

**Constant**

**Non Numeric constant**

**Numeric constant**

**String constant**

**Character constant**

**Float constant**

**Integer constant**

1. There are mainly two type which are given below.
2. Numerical constant.
3. Non Numerical constant.
4. **Numerical constant:-**
5. Numerical constant is that which contain value either it may be integer or float.
6. It is further divided into two type **Integer constant** and **Float constant**.
7. Example:-

const int a=10;//Integer constant.

const float b=10.32;//Float constant.

1. **Non Numerical constant:-**
2. Non Numerical constant is that which do not contain any numeric value.
3. It is also further sub divided into two type **Character constant** and **String constant**.
4. Example:-

const char a=’a’;//Character constant.

cont char b[]={“Chetan”};//String constant.

1. We can also define constant pointer.

int a=10;

const int \*p=&a;

We must not change assign address of

constant pointer.

1. Example 1:-

#include<stdio.h>

#include<conio.h>

main()

{

const int a=10;//Integer constant.

const float b=10.23; ;//Float constant.

const char c=’a’;//Character constant.

const char d[]=”Chetan”;//String constant.

const \*p=&a;//Integer constant pointer.

printf(“Integer constant is=%d”,a);

printf(“\nFloat constant is=%f”,b);

printf(“\nCharacter constant is=%c”,c);

printf(“\nString constant is=%s”,d);

printf(“\nInteger constant pointer is=%u”,p);

getch();

}

1. **String:-**
2. String is group of characters.
3. It is always enclosed in double quotation
4. It is terminated by null character (\0).
5. Example:-

char name[]={“Chetan”}

1. Program:-

#include<stdio.h>

#include<conio.h>

main()

{

char name[]=”Chetan Wawarkar”};

clrscr();

printf(“String is=%s”,name);

getch();

}

1. **Operators:-**
2. Operator is symbol which tell us which type of operation to be performed. Operation may be arithmetic or logical.
3. It evaluate only single value.
4. Operator is unary or binary.
5. The operator which has single operand is called **Unary operator** and the operator which has at least two operand is called **Binary operator**.
6. There are 8 types of operators in C language. Which are listed below.
7. Arithmetic operators.
8. Relational operators.
9. Logical operators.
10. Assignment operator.
11. Increment and Decrement operators.
12. Conditional operators.
13. Bitwise operators.
14. Special operators.

Above operators is described below.

1. **Arithmetic operators:-**
2. It is used to perform arithmetic operation.
3. There are five arithmetic operators in C language. Which are listed below.

|  |  |  |
| --- | --- | --- |
| **Sr.no** | **Operators** | **Meaning** |
|  | + | Addition |
|  | - | Subtraction |
|  | \* | Multiplication |
|  | / | Division |
|  | % | Mode |

1. Remember that there is difference between division and mode operator.
2. Mode operator is used to find reminder of division whereas division operator is used to find division of given number.
3. **Note:-Mode operator is not applied on float and**

**double data type or value in C language.**

**But in Java mode operator is applied on float**

**and double data type.**

1. Program for Arithmetic operators.

#include<stdio.h>

#include<conio.h>

main()

{

int a,b,c,d,e,f,g;

clrscr();

printf("Enter the two numbers");

scanf("%d%d",&a,&b);

c=a+b;

printf("Addition is=%d",c);

d=a-b;

printf("\nSubtraction is=%d",d);

e=a\*b;

printf("\nMultiplication is=%d",e);

f=a/b;

printf("\nDivision is=%d",f);

g=a%b;

printf("\nMode is=%d",g);

getch();

}

1. **Relational operators:-**
2. It is also called as comparison operators.
3. It is used comparison.
4. These operator give only true (1) and false (0) value.
5. There are 6 relation operators in C language. Which are listed below.

|  |  |  |
| --- | --- | --- |
| **Sr.no** | **Operators** | **Meaning** |
|  | < | Less than |
|  | > | Greater than |
|  | <= | Less than equal to |
|  | > | Greater than equal to |
|  | == | Equal to |
|  | != | Not equal to |

1. We can store the result of relation operators into integer, float and \_Bool data type.
2. Program for relation operators is:-

#include<stdio.h>

#include<conio.h>

main()

{

int a,b,c,d,e,f,g,h;

clrscr();

printf("Enter the two numbers");

scanf("%d%d",&a,&b);

c=a<b;

printf("Less than is=%d",c);

d=a>b;

printf("\nGreater than is=%d",d);

e=a<=b;

printf("\nLess than equal to is=%d",e);

f=a>=b;

printf("\nGreater than equal to is=%d",f);

g=a==b;

printf("\nEqual to is=%d",g);

h=a!=b;

printf("\nNot equal to is=%d",h);

getch();

}

1. **Logical operators:-**
2. It is used to combine two or more conditions.
3. It is most used with relational operator expression.
4. It gives either true (1) or false (0) after evaluating expression.
5. There are 3 logical operators. Which are listed below.

|  |  |  |
| --- | --- | --- |
| **Sr.no** | **Operators** | **Meaning** |
|  | & | AND |
|  | || | OR |
|  | ! | NOT |

1. AND operator will give true if all condition must true.
2. OR operator will give true if any one condition must is true.
3. NOT operator inverts (reverses) result that mean it convert true into false and false into true.
4. Simply we can understand AND perform multiplication and OR perform addition and NOT perform invert operation.
5. Let us understand point no. 7 by using table.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Condition**  **1** | **Condition**  **2** | **Condition**  **3** | **Condition**  **4** | **AND** | **OR** |
| 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 0 | 1 | 1 | 0 | 1 |
| 0 | 0 | 0 | 0 | 0 | 0 |

1. Program for Logical operator:-

#include<stdio.h>

#include<conio.h>

main()

{

int a,b,c,d,e;

printf("Enter the two numbers");

scanf("%d%d",&a,&b);

c=a<b&&a>b;

printf("AND operation is=%d",c);

d=a<b||a>b;

printf("\nOR operation is=%d",d);

d=!(a<b);

printf("\nOR operation is=%d",d);

getch();

}

1. **Assignment operator:-**
2. It is used assign value into the variable.
3. It is also used to assign address of variable into the pointer.
4. It works form right to left.
5. Program for assignment operator:-

#include<stdio.h>

#include<conio.h>

main()

{

int a,\*p;

clrscr();

a=10;

p=&a;

printf(“Assignment operator demon is=%d”,a);

printf(“Assigning address to pointer variable=%u”,p);

getch();

}

1. **Increment and Decrement operators:-**
2. Increment operator further categories into following.

1. Pre increment operator.

2. Post increment operator.

1. **Pre increment operator:-**
2. It is unary operator.
3. It is placed before operand.
4. It increments by 1.
5. It work before all another operation to be performed.
6. It has highest priority than all another operator in operation.
7. Program for pre increment operator.-

#include<stdio.h>

#include<conio.h>

main()

{

int a=10;

clrscr();

printf(“Pre increment operator

demo is=%d”,++a);

getch();

}

1. **Note:-**
2. **This operator never applied on any value.**
3. **It is only applied on variables only. Let us understand following example.**

**int a=10;**

**++a//valid statement.**

**++10//Not valid statement in C**

**Language, C++, Java.**

1. **Post increment operator:-**
2. It is unary operator.
3. It is placed after operand.
4. It increments by 1.
5. It has lowest priority than all another operator in operation.
6. Program for assignment operator:

#include<stdio.h>

#include<conio.h>

main()

{

int a=10;

clrscr();

printf(“Post increment operator

demo is=%d”,a++);

getch();

}

1. **Note:-**
2. **Its priority never changed even if it placed in bracket.**
3. **It evaluate first if it does involve any operation like this.**

**int a=10;**

**a++;//does not have any operation**

**other than post increment.**

1. **Note:-**
2. **This operator never applied on any**

**value.**

1. **It is only applied on variables only. Let us understand following example.**

**int a=10;**

**a++//valid statement.**

**10++//Not valid statement in C**

**Language, C++, Java.**

1. Decrement operator further categories into following.
2. Pre decrement operator.
3. Post decrement operator.
4. **Pre decrement operator:-**
5. It is unary operator.
6. It is placed before operand.
7. It is decrement by 1.
8. It work before all another operation to be performed.
9. It has highest priority than all another operator in operation.
10. Program for pre increment operator.-

#include<stdio.h>

#include<conio.h>

main()

{

int a=10;

clrscr();

printf(“Pre decrement operator

demo is=%d”,--a);

getch();

}

1. **Note:-**
2. **This operator never applied on**

**any value.**

1. **It is only applied on variables only.**

**Let us understand following example.**

**int a=10;**

**--a//valid statement.**

**--10//Not valid statement in C**

**Language, C++, Java.**

1. **Post decrement operator :-**
2. It is unary operator.
3. It is placed after the operand.
4. It is also decrement by 1.
5. It has lowest priority than all another operator in operation.
6. Program for post decrement operator:-

#include<stdio.h>

#include<conio.h>

main()

{

int a=10;

clrscr();

printf(“Post decrement operator

demo is=%d”,a--);

getch();

}

1. **Note:-**

**Its priority never changed even if it placed in bracket.**

**It evaluate first if it does involve any operation like this.**

**int a=10;**

**a--;//does not have any operation**

**other than post decrement.**

1. **It is only applied on variables only.**

**Let us understand following example.**

**int a=10;**

**a--//valid statement.**

**10--//Not valid statement in C**

**Language, C++, Java.**

1. **Conditional operators:-**
2. It is also called as ternary operator.
3. Here ? and : symbol is used.
4. Let us take general example

Expression1? Expression2: Expression3

**Explaination**

1. If the first expression is true then part after ? mark is executed that is expression2 is executed.
2. If the condition is false then after : part is executed that is expression 2 is executed.
3. Let us takes example to understand operator.

#include<stdio.h>

#include<conio.h>

main()

{

int a=10,b=20,c;

clrscr();

c=a<=b?a:b;

printf(“Greatest number is=%d”,c);

getch();

}

Explaination:-

1. Here a is less equals to b than value of a is assigns to c.
2. If we do reverse condition (a>=b) then it assigns to c.
3. **Bitwise operators:-**
4. They are used to work with bit level.
5. There are six bitwise operators in C language which are listed below.

|  |  |  |
| --- | --- | --- |
| **Sr.no** | **Operators** | **Meaning** |
|  | & | Bitwise AND |
|  | | | Bitwise OR |
|  | ^ | Ex-OR |
|  | << | Left shift |
|  | >> | Right shift |
|  | ~ | One complement |

1. Let us understand above listed operator in details.
2. **Bitwise AND:-**
3. Its perform anding between bits. That simply means that it perform multiplication between bits.
4. Its output is similar to AND gate.
5. Following table understand Bits operations.

|  |  |  |
| --- | --- | --- |
| **A** | **B** | **RESULT** |
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

1. It gives 1 only when if all bits are 1.
2. Program for Bitwise AND operator:-

#include<stdio.h>

#include<conio.h>

main()

{

int a=5,b=6,c;

clrscr();

c=a&b;

printf(“Result of Bitwise AND is=%d”,c);

getch();

}

Output:-

Result of Bitwise AND is=4

Let us Understand Output

Here

a=5;

b=6;

5 0101

IN BINARY

6 0110

IN BINARY

Now performing bitwise AND operation

0 1 0 1

0 1 1 0

0 1 0 0 4

For understanding above result please

refer above table.

1. **Bitwise OR:-**
2. It perform oring between bits. That simply means it performs addition between bits.
3. Its output is similar to OR gate.
4. Following table understand Bits operations.

|  |  |  |
| --- | --- | --- |
| **A** | **B** | **RESULT** |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

1. It gives 1 only when if any one bits are 1.
2. Program for Bitwise OR operator:-

#include<stdio.h>

#include<conio.h>

main()

{

int a=5,b=6,c;

clrscr();

c=a|b;

printf(“Result of Bitwise OR is=%d”,c);

getch();

}

Output:-

Result of Bitwise OR is=7

Let us Understand Output

Here

a=5;

b=6;

5 0101

IN BINARY

6 0110

IN BINARY

Now performing bitwise OR operation

0 1 0 1

0 1 1 0

0 1 1 1 7

For understanding above result please

refer above table.

1. **Bitwise Ex OR:-**
2. It performs Ex Oring between bits.
3. Its output is similar to Ex OR gates.
4. Following table understand Bits operations.

|  |  |  |
| --- | --- | --- |
| **A** | **B** | **RESULT** |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

1. It gives 1 if pair of input is odd.
2. Program for Bitwise EX OR operator:-

#include<stdio.h>

#include<conio.h>

main()

{

int a=5,b=6,c;

clrscr();

c=a^b;

printf(“Result of Bitwise EX-OR

is=%d”,c);

getch();

}

Output:-

Result of Bitwise EX-OR is=3

Let us Understand Output

Here

a=5;

b=6;

5 0101

IN BINARY

6 0110

IN BINARY

Now performing bitwise EX OR

operation

0 1 0 1

0 1 1 0

0 1 1 1 3

For understanding above result please

refer above table.

1. **Bitwise left shift operator:-**
2. It is used to shift bits at left side.
3. In general (mathematical) way to understand this operator explaination are given in next points.
4. As we know the base of binary number is 2.
5. Hence we take power of 2 of shifting number.

Let us understand above point.

int a=10,c;

c=a<<4; /\*Equation 1 operation\*/

Here we take power of 4 this is because binary number is 2. Above operation became like below.

c=a<<42;

**Remember that in point 4 we understand bitwise left shift operation in mathematical way. Actual operation of bitwise left shift operator is given in point 5.**

1. See what happen actual in equation 1 operation.

c=a<<4;

c=a<<42

128 64 32 16 8 4 2 1 For Binary

Conversion

numbering

1 0 1 0 Original

number

shifted at left

side by 4.

1. 0 1 0
2. Program for bitwise left shift operator are given below:-

Program for Bitwise left operator:-

#include<stdio.h>

#include<conio.h>

main()

{

int a=10,b;

clrscr();

b=a<<4;

printf(“Result of Bitwise left shift

is=%d”,b);

getch();

}

Output:-

Result of Bitwise left shift=140

1. **Note:-Discarding of bits is not**

**possible in bitwise left shift**

**operator.**

1. **Bitwise right shift operator:-**
2. It is to shift bits at right side.
3. It must possibility of discarding bits. If the shifting bits is greater than 0.
4. In general (mathematical) way to understand this operator explaination are given in next points.
5. As we know the base of binary number is 2.
6. Here we take power of 2. This is because base of binary number is 2. And then follow along with example in mathematical way.

int a=10

Here a=10.

a>>2

10>>2

10>>2

10>>4

10>>4

10/4

1

1. Above is explaination in mathematical way but explain in actual way. Let us see

8 4 2 1 0 0

1 0 1 0

1. 0 1 0 1 0

2 in Decimal

1. Program for bitwise left shift operator are given below:-
2. Program for Bitwise right shift operator:-

#include<stdio.h>

#include<conio.h>

main()

{

int a=10,b;

clrscr();

b=a>>2;

printf(“Result of Bitwise right shift

is=%d”,b);

getch();

}

Output:-

Result of Bitwise right shift=2

1. **One compliment operator:-**
2. This is unary operator.
3. It takes 2’s compliment of number.
4. It firstly convert number into 1s compliment then convert it into two compliment.
5. 2’s compliment is negative representation of any number in binary. Hence it gives negative answer of any positive decimal number.
6. And it gives positive answer of negative decimal number.
7. Let us understand this above mention point in mathematical way.
8. General formula for positive number for one compliment operator is –(N+1) where N is number.
9. General formula for positive number for one compliment operator is –(-N-1) where N is number.
10. Let us understand this term in programmatically.

Here

a=9;

1001 Binary form

let us take 1s compliment of number. For doing that we covert 0 into 1 and 1 into 0

1 0 0 1

0 1 1 0 1s complement of 9

After take 2s complement by adding 1 into 1s compliment.

0 1 1 0 1s complement of 9

1

1. **Special operators:-**
2. **Special symbols:-**