Character Set in C

The character set are set of words, digits, symbols and operators that are valid in C.

Character Set		
1.	Letters	Uppercase A-Z Lowercase a-z
2.	Digits	All digits 0-9
3.	Special Characters	All Symbols: , . : ; ? ' " ! | \ / ~ _ $ % # & ^ * - + < > ( ) { } [ ]

Tokens in C

Tokens is the smallest individual unit in C language. In fact, every unit that makes a sentence in C is a Token. C has six types of Tokens as given below.

- Keywords
- Identifiers
- Constants
- Strings
- Special Symbols
- Operators

Keywords in C

Keywords are those words who has special meaning for compiler. We can't use kaywords as variable name.

C has 32 Keywords as follows:

Keywords			
auto	double	int	struct
break	else	long	switch
case	enum	register	typedef
char	extern	return	union
const	float	short	unsigned
continue	for	signed	void
default	goto	sizeof	volatile
do	if	static	while

C Programming Identifiers

Identifiers are fundamental building blocks of a program and are used as the general terminology for the names given to different parts of program viz. variables, functions array structures etc.

Rules of naming Identifiers :

- Identifiers can have alphabets, digits and underscore sign characters.
- They must not be a keyword or boolean literal or null literal.
- They must not begin with a digit.
- They can be of any length.
- C is case sensitive i.e., upper-case letters and lower-case letters are treated differently.

Data Types in C

Data types are used to define a variable. Data types represents the type of information present in a variable. Data types are the keywords, which are used for assigning a type to a variable.

Different Data Types available in C

Integer Type : Integer data type are like whole numbers, they also include negative numbers but does not support decimal numbers.

Type	Storage size	Value range
int	2 or 4 bytes	-32,768 to 32,767 or -2,147,483,648 to 2,147,483,647
unsigned int	2 or 4 bytes	0 to 65,535 or 0 to 4,294,967,295
short	2 bytes	-32,768 to 32,767
unsigned short	2 bytes	0 to 65,535
long	4 bytes	-2,147,483,648 to 2,147,483,647
unsigned long	4 bytes	0 to 4,294,967,295

Float-point Type : Float data type allows user to store decimal values in a variable.

Type	Storage size	Value range	Precision
float	4 byte	1.2E-38 to 3.4E+38	6 decimal places
double	8 byte	2.3E-308 to 1.7E+308	15 decimal places
long double	10 byte	3.4E-4932 to 1.1E+4932	19 decimal places

Character Type : Character data type is used to store only one letter, digit, symbol at a time.

Type	Storage size	Value range
char	1 byte	-128 to 127 or 0 to 255
unsigned char	1 byte	0 to 255
signed char	1 byte	-128 to 127

Variable in C

ariables are used to store values. variable name is the name of memory location where value is stored. It must be alphanumeric, olny underscore is allowed in a variable name. It is composed of letters, digits and only underscore. It must begin with alphabet or underscore. It can not be begin with numeric.

Declaration of Variable

Declaration will allocate memory for specified variable with garbage value.

Syntax :

```
Data-Type Variable-name;
```

Examples :

```
int a;
float b;
char c;
```

Initialization of Variable

Initialization means assigning value to declared variable. Every value will overwrite the previous value.

Examples :

```
a = 10;
b = 4.5;
c = 'a';
```

Character value must be enclosed with single quotes.

```
a = 4.5;
```

if we assign decimal value to integer variable, it will accept only integer portion of value. In the above example variable a will accept 4 only.

Operator in C

An operator is a symbol that tells the compiler to perform specific mathematical or logical calculations on operands(variables).

Types of operators available in C

- Assignment operator
- Arithmetic / Mathmetical operator
- Increment Decrement operator
- Relational operator
- Logical operator

- Conditional or Ternary operator
- Binary operator
- Unary operator

Assignment operator:-

Assignment operator is used to copy value from right to left variable.

Suppose we have,

float X = 5, Y = 2;

=	Equal sign	Copy value from right to left.	X = Y, Now both X and Y have 5
+=	Plus Equal to	Plus Equal to operator will return the addition of right operand and left operand.	X += Y is similar to X = X + Y, now X is 7
-=	Minus Equal to	Minus Equal to operator will return the subtraction of right operand from left operand.	X -= Y is similar to X = X - Y, now X is 3
*=	Multiply Equal to	Multiply Equal to operator will return the product of right operand and left operand.	X *= Y is similar to X = X * Y, now X is 10
/=	Division Equal to	Division Equal to operator will divide right operand by left operand and return the quotient.	X /= Y is similar to X = X / Y, now X is 2.5
%=	Modulus or Mod Equal to	Modulus Equal to operator will divide right operand by left operand and return the mod (Remainder).	X %= Y is similar to X = X % Y, now X is 1

Arithmetic operator:-

Arithemtic operators are used for mathematical operations.

Suppose we have,

int X = 5, Y = 2;

Operator	Name	Description	Example
+	Plus	Return the addition of left and right operands.	(X + Y) will return 7
-	Minus	Return the difference b/w right operand from left operand.	(X - Y) will return 3
*	Multiply	Return the product of left and right operands.	(X * Y) will return 10
/	Division	Return the Quetiont from left operand by right operand.	(X / Y) will return 2(both are int, int doesn't support decimal)
%	Modulus or Mod	Return the Modulus (Remainder) from left operand by right operand.	(X % Y) will return 1

Relational Operators

Relational operators are used for checking conditions whether the given condition is true or false. If the condition is true, it will return non-zero value, if the condition is false, it will return 0.

Suppose we have,

int X = 5, Y = 2;

Operator	Name	Description	Example
>	Greater then	Check whether the left operand is greater then right operand or not.	(X > Y) will return true
<	Smaller then	Check whether the left operand is smaller then right operand or not.	(X < Y) will return false
>=	Greater then or Equal to	Check whether the left operand is greater or equal to right operand or not.	(X >= Y) will return true
<=	Smaller then or Equal to	Check whether the left operand is smaller or equal to right operand or not.	(X <= Y) will return false
==	Equal to	Check whether the both operands are equal or not.	(X == Y) will return false
!=	Not Equal to	Check whether the both operands are equal or not.	(X != Y) will return true

Logical Operators:-

Logical operators are used in situation when we have more then one condition in a single if statement.

Suppose we have,

int X = 5, Y = 2;

Operator	Name	Description	Example
&&	AND	Return true if all conditions are true, return false if any of the condition is false.	if(X > Y && Y < X) will return true
||	OR	Return false if all conditions are false, return true if any of the condition is true.	if(X > Y || X < Y) will return true
!	NOT	Return true if condition if false, return false if condition is true.	if(!(X>y)) will return false

Conditional Operator ?:

The conditional operator is also known as ternary operator. It is called ternary operator because it takes three arguments. First is condition, second and third is value. The conditional operator check the condition, if condition is true, it will return second value, if condition is false, it will return third value.

Syntax :

```
val = condition ? val1 : val2;
```

Example :

```
void main()
        {
    int X=5,Y=2,lrg;

    lrg = (X>Y) ? X : Y;

    printf("\nLargest number is : %d",lrg);
}
```

Output :

```
    Largest number is : 5
```

Unary operator

Unary operators are those operators that works with singal operands such as (Increment or Decrement operators) ++ and --.

if Statement:-

if statement takes condition in parenthesis and a block of statements within braces. If condition is true, it will return non-zero value, and statements given in if block will get execute.

Syntax of simple if statement :

```
if (condition)
{
    ----------
    ----------
}
```

if-else Statement:-

if statement takes condition in parenthesis and a block of statements within braces. If condition is true, it will return non-zero value, and statements given in if block will get execute. If condition is false, it will returns zero, and statements given in else block will get execute.

Syntax of simple if-else statement

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

Example of simple if-else statement

```
#include<stdio.h>

void main()
{
    int num;

    printf("Enter any number : ");
    scanf("%d",&num);

    if (num>0)
        printf("%d is Positive.");
     else
        printf("%d is Negative.");
}

Output :

    Enter any number : 24
    24 is Positive
```

if-else Ladder:-

if-else ladder is used for checking multiple conditions, if the first condition will not satisfy, compiler will jump to else block and check the next condition, whether it is true or not and so on.

Syntax of if-else ladder statement

```
    if (condition)
    {
        - - - - - - - - - -
        - - - - - - - - - -
```

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

Example of if-else ladder statement

```
        #include<stdio.h>

    void main()
    {
                int amt,discount;

        printf("Enter amount : ");
        scanf("%d",&amt);

        if (amt>20000)
                discount=15;

        else if (amt>15000)
```

```
            discount=10;

        else if (amt>10000)
            discount=5;

        else
            discount=0;

        printf("You will get %d% discount.",discount);
    }
```

Output :

```
        Enter amount : 14000
        You will get 5% discount.
```

Nested if-else Statement:-

In nested if-else, one if-else statement contains another if-else statement.

Syntax of nested if-else statement

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

Example of nested if-else statement

```
#include<stdio.h>

void main()
{
    int a,b,c;

    printf("\nEnter value of A : ");
    scanf("%d",&a);

    printf("\nEnter value of B : ");
    scanf("%d",&b);

    printf("\nEnter value of C : ");
    scanf("%d",&c);

    if (a>b)
     {
          if (a>c)
            printf("\n\nA is Greatest");
           else
             printf("\n\nC is Greatest");
       }
    else
       {
           if (b>c)
              printf("\n\nB is Greatest");
            else
              printf("\n\nC is Greatest");
        }
}
```

```
Output :

        Enter value of A : 45
        Enter value of B : 89
        Enter value of C : 78

        B is Greatest
```

Loops in C:

Loops are used, when we need to execute one or more statements multiple times until some conditions are satisfied.

Types of Loop

- while loop
- do...while loop
- for loop

While Loop:-

While loop is also called entry control loop because, in while loop, compiler will 1st check the condition, whether it is true or false, if condition is true then execute the statements.

Syntax of While Loop

```
initialization;

while (condition)
{
    - - - - - - - - - -
    - - - - - - - - - -

     inc/dec;
}
```

Example of While Loop

```
#include<stdio.h>

void main()
{
    int a=1,num;

    printf("Enter any number : ");
    scanf("%d",&num);

    while (a<=num)
     {
        printf("\nHello...!!");
        a++;
      }
}
```

Output :

```
Enter any number : 5
Hello...!!
Hello...!!
Hello...!!
Hello...!!
Hello...!!
```

Do-While Loop:-

The do-while loop is also called exit control loop because, in do-while loop, compiler will 1st execute the statements, then check the condition, whether it is true or false.

Syntax of Do-While Loop

```
initialization;
```

```
do
{
    - - - - - - - - - -
    - - - - - - - - - -

     inc/dec;
} while (condition);
```

Example of Do-While Loop

```
#include<stdio.h>

void main()
{
    int a=1,num;

    printf("Enter any number : ");
    scanf("%d",&num);

    do
     {
        printf("\nHello...!!");
        a++;
      } while (a<=num);
}
```

Output :

```
Enter any number : 5
Hello...!!
Hello...!!
```

```
Hello...!!
Hello...!!
Hello...!!
```

Difference b/w while loop and do-while loop	
while loop	**do-while loop**
It is entry control loop.	It is exit control loop.
In this loop condition is checked before loop execution.	In this loop condition is checked at the end of loop.
It will never execute loop if condition is false.	It will executes loop at least once when the initial condition is false.
There is no semicolon at the end of while statement	There is semicolon at the end of while statement.

For Loop:-

In for loop we put initialization, contidion and increment/decrement all together. Initialization will be done once at the beginning of loop. Then, the condition is checked by the compiler. If the condition is false, for loop is terminated. But, if condition is true then, the statements are executed until condition is false.

Syntax of For Loop

```
for (initialization;condition;inc/dec)
{
    ----------
    ----------
}
```

Example of For Loop

```
#include<stdio.h>

void main()
{
    int a,num;

    printf("Enter any number : ");
    scanf("%d",&num);

    for (a=1;a<=num;a++)
        printf("\nHello...!!");
}
```

Output :

```
Enter any number : 24
Hello...!!
Hello...!!
Hello...!!
Hello...!!
Hello...!!
```

Jump Statements in C:-

ump statements are used to interrupt the normal flow of program.

Types of Jump Statements

- Break
- Continue
- GoTo

Break Statement:-

The break statement is used inside loop or switch statement. When compiler finds the break statement inside a loop, compiler will abort the loop and continue to execute statements followed by loop.

Example of break statement

```
#include<stdio.h>

void main()
{
    int a=1;

    while(a<=10)
     {
        if(a==5)
          break;

         printf("\nStatement %d.",a);
        a++;
      }
         printf("\nEnd of Program.");
}
```

Output :

```
Statement 1.
Statement 2.
Statement 3.
Statement 4.
End of Program.
```

Continue Statement:-

The continue statement is also used inside loop. When compiler finds the break statement inside a loop, compiler will skip all the followling statements in the loop and resume the loop.

Example of continue statement

```
#include<stdio.h>

void main()
{
    int a=0;

    while(a<10)
     {

        a++;

        if(a==5)
          continue;

         printf("\nStatement %d.",a);

      }
         printf("\nEnd of Program.");
}
```

Output :

```
Statement 1.
Statement 2.
Statemnet 3.
Statement 4.
Statement 6.
```

```
Statement 7.
Statement 8.
Statement 9.
Statement 10.
End of Program.
```

Goto Statement:-

The goto statement is a jump statement which jumps from one point to another point within a function.

Syntax of goto statement

```
goto label;
    - - - - - - - - - -
    - - - - - - - - - -
label:
    - - - - - - - - - -
    - - - - - - - - - -
```

In the above syntax, label is an identifier. When, the control of program reaches to goto statement, the control of the program will jump to the label: and executes the code after it.

Example of goto statement

```
#include<stdio.h>

void main()
{
        printf("\nStatement 1.");
        printf("\nStatement 2.");
        printf("\nStatement 3.");

        goto last;
```

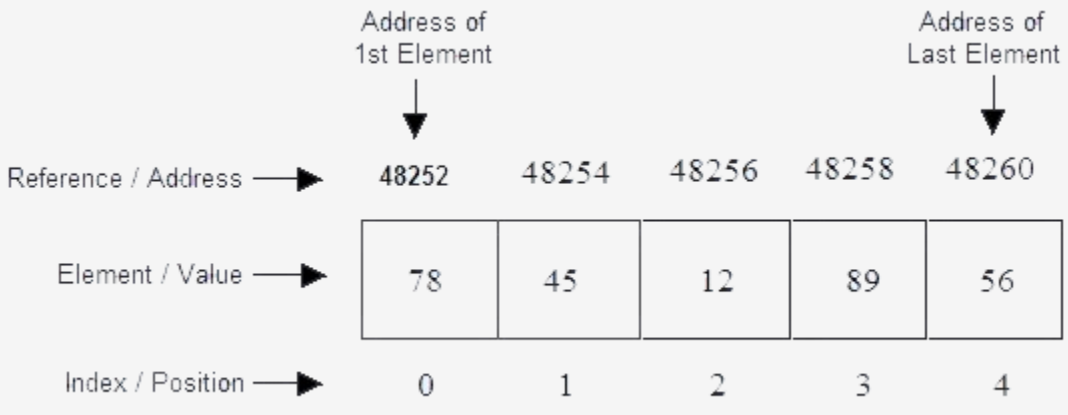
```
        printf("\nStatement 4.");
        printf("\nStatement 5.");

        last:

        printf("\nEnd of Program.");
}
```

Output :

```
Statement 1.
Statement 2.
Statement 3.
End of Program.
```

Array in C:-

Array is a collection of similar data type. A single variable can hold only one value at a time, If we want a variable to store more than one value of same type we use array. Array is linear data structure. It can store fixed number of values.



Address of first element is random, address of next element depend upon the type of array. Here, the type is integer and integer takes two bytes in memory, therefore every next address will increment by two.

Index of array will always starts with zero.

Declaration of Array:-

Declaration of array means creating sequential bolcks of memory to hold fixed number of values.

Syntax of array declaration :

```
Data-type Array-name[ size of Array ];
```

Example of array declaration :

```
int Array [5];                    //Statement 1

float Array [10];                 //Statement 2
```

In the above example, statement 1 will allocate memory for an integer array which will hold five values and statement 2 will allocate memory for float point array which will hold ten values.

Initialization of Array:-

Initialization means assigning value to declared Array.

Examples 1 :

```
int Array [ ] = { 78, 45, 12, 89, 56 };
```

In the above example we are declaring and initializing an array at same time. When we declare and initialize an array at same time, giving the size of array is optional.

Examples 2 :

```
#include<stdio.h>

void main()
```

```
{
    int array [5];
    int i;

    for(i=0;i<5;i++)            //Input arrray from user.
    {
      printf("\nEnter any number : ");
      scanf("%d",&array [i]);
    }

    for(i=0;i<5;i++)            //Output arrray to console.
    printf("%d, ",array [i]);

}
```

Output :

```
    Enter any number : 78
    Enter any number : 45
    Enter any number : 12
    Enter any number : 89
    Enter any number : 56

    78, 45, 12, 89, 56,
```