Hello World Program for perl:

```
#!usr/bin/perl

print("Hello World \n\n ");
```

User input on perl

```
#!usr/bin/perl

print("What is Your Name: ");

$name = <STDIN>;

print("Hello ,",$name,"\n\n");
```

To write a data in txt file

```
$fname = "text.txt";

open($fh , ">>",$fname);

print($fh "Ashi\n");
print($fh "Durga\n");

close($fh);
```

To read the content from the file

```
$fname = "text.txt";

open($fh , "<",$fname);

while($r = <$fh>){

chomp($r);
print($r,"\n")

}
close($fh);
```

Here in string operations

Single quoted and double quoted

Here the single quoted string and double quoted string are same as the only string operations differ from the single quotes.

. (Dot) is used to concat the strings .’

And some of the basic operations as follows:

\n - newline
\t = tab space

Number operations in Perl:

In perl the number is treated as the double precision pointer so that it is not considered as the integer and whole number.

```
# number Operations in perl

print(10+2,"\n\n");

print(10*2,"\n\n");

print(10-2,"\n\n");

print(10**2,"\n\n");

print(10%2,"\n\n");

print(10/2,"\n\n");

print(10*2e4,"\n\n");
```

All the arithmetic operations support in the perl.

In this perl language we can only do double quoted in variable to interpolate within the strings whereas in the single quotes we cannot do interpolate.

The example of interpolate in the strings

```
$var1 = "Valorant is Op";
$var2 = "Apex is Op";
$var3 = 100;

#print('$var3 \n $var1\n');

print("$var3 \n $var1\n");
```

Chomp function is used to remove the new line from the string.

```
$s1 = "Mahesh\n";
$s2 = "Pradeep\n";

print "$s1 $s2";

# chomp is used to remove the new line in the string

chomp($s1);
chomp($s2);
```

Decision Making Statements in perl:

Basic if statements

```
# here in this string can be compared with the ascii  value.


if("A" gt "a"){
print('true');
}
elsif("B" ge "A"){
print('true again');
}
else{
print('false');
}
```

```
# looping Statements:
$n = 1;

while($n<=10)
{
$s +=1;
$n++;
print($s,"  ");
}
```

Arrays and List operations as follows:

```
# to create an array

@arr1 = ('mahesh','avinash','gg','op','Hello');

print("@arr1\n");
#print(@arr1)


@arr2 = (10,2,30,12,100);

print("@arr2\n");


# to access the array elements via indices

print("The First Element in arr1: ");

print("$arr1[0]\n");

# in the alternate way we can access the element by assigning to the variable

$arr1_1 = $arr1[1];

print("The Second Element: ");
print("$arr1_1");
```

List operations in perl:

```
# to create a list

@l1 = qw/ Mahesh Nijan 15 10 1998 /;

print("The List Elements: @l1\n");
```

```
# to directly initialize the value

($l1[0],$l1[2]) = ('Pravin',22);


print("$l1[0] $l1[3]\n");


# list assignment

($a,$b) = (10,30);

($n1[0],$n2[1],$n3[2]) = ('Mahesh','Nijan','Pravin');

@names = qw(Mahesh Ashi Durga);


print("@names\n");
print("$n2");
```

Converting arrays into strings:

join operator is used to convert an whole array elements into the single array.

```
#understanding of scalar and list  context

#scalar
$n1 = 90;
$num = 50;
$n2 =  90 + $num;

print("$n2");

# the list
@arr = qw/ Mahesh Nijan Pravin /;

($name1,$name2) = ('Pradeep','Avinash');
print("\n$name1 $name2\n");
```

Subroutines:
Subroutines are similar to functions.

```
# creating subroutines with arguments

#default subroutine
```

```
sub greet(){
print("Hello !");
}

# arguments subroutine
sub name(){
        ($fname,$lname) = @_;
        print("$fname $lname");
}

# parameter subroutine
sub add(){
foreach $val(@_){
print("$val, \n");
}
}

# using return keyword
sub example(){

foreach $name(@_){
if($name eq "Mahesh"){
return $name;
}
}
}

# calling the subroutine to perform the task

#&greet();
#&name("Mahesh","Vaithi");
#&add(1,2,3,4);

#$ex = &example("Nijan","Pradeep","Mahesh");
#print($ex)
```

Subroutines with private variables

Private variable is used to access within a scope of subroutine.

```
#using private access in scope of Subroutine
sub privatenum(){
my($d,$e) = @_;
print("d=$d,e=$e");
}
```

```
# calling the subroutine to perform the task

&privatenum(2,3);
print("The d and e : $d,$e");
```

Standard as similar to the Input syntax:

Diamond Operator with input

```
# using diamond input operator we can read the file name

while(defined($data = <>))
{
print("$data")
}

# to run this script
perl fileread.pl text.txt

# using printf format

printf("Hello I m Mahesh %s","Nice Day\n");

printf("Adding: %g+%g=%g\n",5.5,9.3,(5.5+9.3));

printf("Multiplying: %g*%g=%.2f\n",5.5,9.3,(5.5*9.3));
```

File handling

```
# to read the content from the file
open(FILEHANDLER , "<","text.txt");
while($r = <FILEHANDLER>)
{
print("$r");
}
close(FILEHANDLER);

# to write a content in file
open(FILEHANDLER , ">>","text.txt");
print(FILEHANDLER "Elangovan");
close(FILEHANDLER);
```

Handle fatal errors with die:
It is simply a Try Catch Exception similar to other programming.

```
# error handling in perl

# similar to try catch exception in others.

if(!open(LOG,"<my.txt"))
{
die("Error Occured: $!");
}
```

Hashes in Perl:

```
# create an hash

%names = ("n1" =>"Pravin","n2"=>"Pratap","n3"=>"Nijan");

# to assign hash

$names{"$n4"} = "Harris";

foreach $r (values(%names))
{

print("$r\n");

}


# to accessing hash elements
print($names{"$n2"});


# hash functions in perl


# printing keys and values

print("Keys\n");
foreach $k (keys(%names))
{
print("$k\n");
}
print("Values\n");
foreach $v (values(%names))
{
```

```
print("$v\n");
}
```

```
# to print key and values

while(($k,$v) = each(%names))
{
print("$k => $v\n");
}
```

Regular Expressions in Perl:

```
$_ = "Hello , Good Eve !...";

if(/  \bHello\b  /)
{
print("Match Found");
}
else
{
print("Not Found");
}
```

Character Class in Perl:

```
$_ = "Valorant is founded at year-2020 and becoming best esports game in 2021 won the best award @ USA";

if(/year-[0-9]/)
{
print("Match Found");
}
else
{
print("Not Found");
}
```

Using Match Case m//

```
$_ = "https://maheshvaithi.com/myfavosite";

if(m{^https://})
{
print("Match Found");
}
```

```
else
{
print("Not Found");
}
```

Matching list in the Context using regex:

```
@names = qw/Mahesh mAhEsh MAhesH/;

@res = grep(/Mahesh/gi,@names);

print("Matched Exp: @res")
```

Here the gi is used to remove the case sensitive in the pattern.

Substitution Operator in Perl:

```
# substitution operator

$v2 = "Hello Mahesh Nice to meet You!";

$v2 = s/Mahesh/Nijan/g;


print("\n$v2");
```

Split function in Perl:

```
$names = "Mahesh:Nijan:Pravin:Pradeep";

@arr = split(/:/,$names);

print("@arr\n");

print("After Split\n");
foreach $n(@arr){
print("$n\n");
}
```