**Perl Programming**

**Arrays, for loop**

@Lista = (’Otto’, ’Sette’,’Sei’);

for($k = 0; $k < 3; $k = $k + 1) {

print ”@Lista[$k] \n”;

}

**Array operators**

@Lista(1..15);

print ”@Lista\n”;

$length = @Lista;

print ”The size of the array Lista is: $length\n”;

**Array methods for adding/removing elements**

@Tavilo = ('Petros', 'Antonio', 'Jorge');

print "Black list: @Tavilo\n";

push(@Tavilo,'Lopecito'); #adds one element

print "The black list is: @Tavilo\n";

unshift(@Tavilo,'Louis'); #adds one element at the beginning

print "The black list is: @Tavilo\n";

pop(@Tavilo); #removes one element from the end

print "@Tavilo\n";

shift(@Tavilo); #removes one element from the beginning

print "@Tavilo\n";

**Array slicing**

@Lista = ('Petros', 'Antonio', 'Jorge','Louis','Garcia','Altagracia');

@Novo = @Lista[0,1,5];

print "Novo: @Novo\n";

@outro = @Lista[2..5];

print " Outro: @outro\n";

**Replacing array elements**

@Actors = ('Tom', 'Brad', 'Jeffrey','Louis','Norton');

@casting = ('DiCaprio', 'Washington');

print "@Actors";

splice(@Actors, 1, 2, @casting);

print "@Actors\n";

**Converting arrays to strings**

@Actors = ('Tom', 'Brad', 'Jeffrey','Louis','Norton');

$string = join(' <--> ',@Actors);

print "$string\n";

**Converting strings to arrays**

$string = "Behavioral description, RT-level description, cell-level structural description";

@arei = split(' ', $string);

print "The array is: @arei[0..4]\n";

**Sorting and merging arrays**

@Actors = ('Tom', 'Brad', 'Jeffrey','Louis','Norton');

@Arei1 = sort(@Actors);

print "Arei1 is: @Arei1 \n";

@Arei2 = ('Monique', 'Claudia', 'Annete');

@fin = (@Arei2,@Arei1);

print "fin array is: @fin\n";

**Hashes in Perl**

%Actors = (1=>'Tom', 2=>'Brad', 3=>'Jeffrey',4=>'Louis',5=>'Norton');

print "Great actor is: $Actors{2}\n";

**Slicing hashes**

%Stars = (1=>'Tom', 2=>'Brad', 3=>'Jeffrey',4=>'Louis',5=>'Norton');

@Lista = @Stars{2..4};

print "Lista is: @Lista\n";

**Getting keys and values from hashes**

%Stars = (1=>'Tom', 2=>'Brad', 3=>'Jeffrey',4=>'Louis',5=>'Norton');

@rank = keys %Stars;

@Name = values %Stars;

print "Rank is: @rank\n";

print "Name is: @Name\n";

**Check existence of hash element , size of hash**

%Tennis = (1=>'Novak', 2=>'Roger', 3=>'Jeffrey',4=>'Bob',5=>'Valtassar');

if(exists($Tennis {6}))

{

print "$Tennis{5}\n!";

}

else {

print "No such element!\n";

}

@Lista = keys %Tennis;

$size = @Lista;

print "The size of the hash is: $size\n";

**Addition Removal of hash elements**

%Tennis = (1=>'Novak', 2=>'Roger', 3=>'Jeffrey',4=>'Bob',5=>'Valtassar');

@Ara = keys %Tennis;

$size = @Ara;

print "The size of the hash now is: $size\n";

$Tennis{6} = 'O\'neil';

@Ara = keys %Tennis;

$size = @Ara;

print "The size after the addition is: $size\n";

delete $Tennis{5};

@Ara = keys %Tennis;

$size = @Ara;

print "Hash size after deletion: $size\n";

**Unless statement**

$ro= 25;

unless($ro%2==0) {

print "$ro is odd\n";

}

#Conditional ternary operator

$no = 10;

$str = ($no%2==0) ? "The number is even.\n" : "The number is odd.\n";

print "$str\n";

**Until statement**

$x = 10;

until ($x> 12) { #false condition

print ”$x\n”;

}

**Foreach statement**

foreach $iterator (@iterable) { # iterable could be a list as @tennis =(’Dan’, ’Bob’, ’Roger’);

#statements

}

do {

//statements

}

while (condition)

for ($x=6; $x<12; $x = $x+1)

{

If ($x==11)

{

next; #equivalent to continue

}

print ”$x\n”;

}

Last statement equivalent to break statement

**Redo statement**

While ($a <3)

{

print ”$a\n”;

$a =$a+1;

if($a ==3)

{

redo;

}

}

**The exponential operator \*\***

$y = 2;

$k= 5;

print „”$y\*\*$k\n”;

**Comparison operators for strings**

$c =”zoinder”;

$v=”waarschui”;

if ($c lt $v)

{

print ”$c is less than $v\n”;

}

if ($c gt $v)

{

print ”$c is greater than $v\n”;

}

if ($c le $v)

{

print ”$c is less than or equal to $v\n”;

}

if ($c ge $v)

{

print ”$c is greater than or equal to $v\n”;

}

if ($c eq $v)

{

print ”$c and $v are the same\n”;

}

if ($c ne $v)

{

print ”$c and $v are not the same\n”;

}

Assignmnet operators

+= add and assign \*= multiply and assign

-= subtract and assign /= divide and assign

%= assign remainder after division

\*\*= exponentiate and assign

**Bitwise operators**

$x = 10;

$w = 70;

printf ”x is: %\n”, $x;

printf ”y is: %\n”, $y;

$and\_op=$x&$w;

printf ”Result of binary and is: %b\n”, $and\_op;

$or\_op=$x|$w;

printf ”Result of bitwise oris: %b\n”, $or\_op;

$xor\_op=$x^$w;

printf ”Result of bitwise XOR: %b\n”, $xor\_op;

$comp=~$x;

printf ”1 ’s complement of x: %b\n”, $comp;

**Logical operators**

and, or, not

E.g. if ($x > 0 or $x % 2==0)

{

...

}

**Miscelaneous operators**

$string = ” This is a set of characters.”;

print($string x 20); #Repeats the string

@list = (1..10); #The range operator

print ”\n@list\n”;

$vari= 10;

print ++$vari . ”\n”; #Preincrement

print --$vari . ”\n”; #Predecrement

print ”\n”. $vari++ . ”\n”; #Postincrement

print ”$vari\n”;

print ”\n”. $vari-- . ”\n”; #Postincrement

print ”$vari\n”;

Date and time values

@Monaten = (’Jan’, ’Feb’, ’Mar’, ’Apr’, ’Mai’, ’Jun’);

@Woche=(’Son’, ’Mon’, ’Dien’,’Mit’, ’Don’, ’Fri’, ’Sam’);

@thisDatetime=localtime();

Print @thisDatetime;

$seconds= @thisDatetime[0];

$minutes= @thisDatetime[1];

$hours = @thisDatetime[2];

$day\_of\_month= @thisDatetime[3];

$month= @months[@thisDatetime[4]];

@year= 1900 + @thisDatetime[5];

@week\_day= @week[@thisDatetime[6]];

print ”\nCurrent Time: $hours: $minutes: $seconds\n”;

print ”Current Date: $week\_day, $month-$day\_of\_month-$year\n”;

**Formatting time**

@thisDatetime= localtime();

$seconds = @thisDatetime[0];

$minutes = @thisDatetime[1];

$hours= @thisDatetime[2];

if ($hours<12)

{

$digital\_hours= $hours;

$str=”AM”;

}

elsif($hours==12)

{

$digital\_hours=$hours;

$str”PM”;

}

else

{

$digital\_hours=$hours-12;

$str=”PM”;

}

print ”Current Time in Digital Format: $hours : $minutes : $seconds\n”;

**Subroutines in Perl**

sub Bericht() {

print ”Bericht versendet\n”;

}

Bericht();

sub rectangle {

$area = @\_[0]\*@\_[1]; #@\_[0] and @\_[1] are arguments of the subroutine.

$perimeter= 2\*(@\_[0] + @\_[1]);

print ”The area is : $area sq.units\n”;

print ”The perimeter is : $perimeterunits\n”;

}

rectangle(3, 20);

sub CharacterOrdering {

foreach $c(@\_) {

if( ord($c) >=65 and ord($c)<= 90) {

push($upper, $c)

}

elsif( ord($c) >=97 and ord($c)<= 122) {

push($lower, $c)

}

elsif( ord($c) >=48 and ord($c)<= 57) {

push($digits, $c)

}

else {

push($special, $c)

}

}

print ”Upper case alphabet: @upper\n”;

print ”Lower case alphabet: @lower\n”;

print ”Digits case alphabet: @digits\n”;

print ”Special case alphabet: @special\n”;

}

CharacterOrdering(’a’,’b’,’c’, ’$’, ’#’, ’&’, ’\*’, ’S’, ’X’, ’V’,8, 3, 0, 4, 7);

**Count the number of arguments in subroutines**

Sub countArgs {

$num = @\_; #This is the special list;

print ”No of args passed: $num\n”;

}

countArgs(5, 2, ”Text transmision 1”, 10);

**Passing lists to subroutines**

sub random {

@hobbies=@\_;

print ”Hobbies: @hobbies\n”;

}

@favs(Playing the guitar”, ”playing tennis”, ”riding bike”);

random(@favs);

sub circle {

$radius= @\_[0];

Return 3.14\*$radius\*$radius;

}

$area= circle(10);

print ”The area of the circle is $area sq. units\n”;

**Private variables in subroutines**

$var = 500;

sub test {

my $var = 20; # the my keyword is used

pirnt ”Val of var inside the subroutine: $var\n”;

}

test();

print ”Val of var outside the subroutine: $var\n”; #Here the value of 500 will appear since the 20 value is #dereferenced from the local variable $var in test’s body #definition.

**Reading/Writing text from a file**

open(NEW, ”C:/Users/Prisma/Documents/Dogs.txt”);

while(<NEW>) {

print ”$\_”;

}

close(NEW);

open(NEW, ”C:/Users/Prisma/Documents/Dogs.txt”);

$text = ”I like Perl programming!”;

print NEW $text; #pushes the string $text in the file.

close(NEW);

**Receiving input**

print ”Enter a number:\n”;

$num=<STDIN>;

$square=$num\*\*2;

print ”The square of $num is: $square\n”;

**How to copy files**

open(SOURCE, ”C:/Users/Prisma/Documents/Dogs.txt”);

open(DESTINATION, ”C:/Users/Prisma/Desktop/File.txt”);

while(<SOURCE>) {

Pirnt DESTINATION $\_;

}

close(SOURCE);

close(DESTINATION);

**Renaming files**

#Attempt to rename the file

if (rename(”C:/Users/Prisma/Desktop/File.txt”, ”C:/Users/Prisma/Desktop/Archif.txt”)) {

print ”File renamed successfully\n”;

}

else {

print „”Unable to rename the file. \n”;

}

**Listing the files in a folder**

$source= ”C:/Users/Mahmoud/Documents/PerlScripts/\*.pl”; #The path with the folder

@list = glob($source); # Create a list with the names of files with the glop() function.

$size = @list;

print ”Total number of files in the directory: $size\n”;

foreach(@list) {

print substr($\_, 38). ”\n”; #substr() function displays after the 38th character of the path

# above. It will show the name of the file and discard the name of #the full path.

}

**Creating directories**

$folder=”C:/Users/Prisma/Documents/PerlScripts/AnotherFolder”,

If(mkdir($folder)) {

print ”Directory created successfully.\n”;

}

else {

print ”Unable to create directory.\n”;

}