Complex Data Structure

Palani Karthikeyan

Reference

- A reference variable can point to any data structure.
- references are scalars, you can store them in arrays and hashes, and thus build arrays of arrays, arrays of hashes, hashes of arrays, arrays of hashes and functions, and so on.
- **Creating References**
- create a reference to any named variable with a backslash. (****)

In C Language

- A pointer is simply a variable that contains the location of some other piece of data.
- **In C language :-**
- int a=10; a|10|0x100
- int *p; p|GB|0x104
- p=&a; p|0x100|0x104
- **printf("%p\n",p); # 0x104**
- print("%d\n",*p); # 10

  ***p (or) *(&a) ← De-reference a pointer**

SCALAR-Reference

- **$a=10;**
- **$r1=\$a;** # creating a reference (like in c p=&a)
- # **$r1** is holding a address(reference) of $a variable
- **print "$r1\n"; # SCALAR(0x100)**
- $r1 is holding scalar ($) type data so prefix to add $ to dereference $r1.
- **print "$$r1\n"; # 10 (like in c: *p)**

ARRAY-Reference

- **@Array=(data1,data2);**
- $r1=****@Array; **# creating a array reference**
- print “$r1\n"; # ARRAY(0x100)
- # De-reference a array reference
- $r1 is an array reference to de-reference prefix use **@**
- print **“@$r1\n**”

HASH-Reference

- %Hash=(key1=>”value1”,key2=>”value2”);
- $r1=\%Hash; # creating hash reference
- print “$r1\n”; # display HASH(0x100)
- $r1 is holding a hash reference
- To de-reference $r1 (hash reference) prefix use **%**
- **%H=%$r1;**
- foreach(keys(%H)){

  print “$_\t $H{$_}\n”;

  }

- **@os=(unix,linux,aix,winx,minix);**
- $os[1]; # Single data ($) from @os array (@)
- **$r1=\$os[1];** # $r1 is holding a scalar reference
- print "**$r1**\n"; # SCALAR(0x100)
- print "**$$r1**\n"; # De-reference a scalar –linux

- **%Hash=(k1=>"value1",k2=>"value2");**
- $Hash{k1}; # Single data ($) from %Hash , hash variable (%)
- $r1=\$Hash{k1}; # $r1 is holding a scalar reference
- Print "$r1\n"; SCALAR(0x200)
- Print "$$r1\n"; # De-reference a scalar –Value1

Data References

- The ability to dynamically allocate data structures without having to associate them with variable names.
- We refer to these as "anonymous" data structures.
- The ability to point to any data structure, independent of whether it is allocated dynamically or statically.

$reference =[array elements]; anonymous array

$reference={key=>”values”} ; anonymous hash

- use Data::Dumper;
- $ra=["Data1","Data2","Data3"];
- @array=("D1","D2","D3","D4");
- $rv=\@array;
- print "$ra\n";
- $rh={"K1","V1","K2","V2"};
- print "$rh\n";
- print Dumper($ra);
- print Dumper(@array);
- print Dumper($rv);
- print Dumper($rh);

- @os=(unix,linux,qnx,winx);
- $ar=\@os;
- print "A.$ar\n";
- print "B.@$ar\n";

- $AR=["SH","BASH","CSH","TCSH"]; # *p=10;
- print "C.$AR\n";
- print "D.@$AR\n";

- **%Hash=(Key1=>"value1",Key2=>"Value2");**
- **$hr=\%Hash;**
- print "A.$hr\n";

- **%H1=%$hr; # De-reference**
- foreach (keys(%H1)){
- print "$_ value is $H1{$_}\n";
- }
- print "\n-----------------------------------\n";
- **$rv={K1=>"V1",K2=>"v2",K3=>"v3"};**
- print "B.$rv\n";
- **%H2=%$rv; # De-reference**
- foreach (keys(**%H2**)){
- print "$_ value is $H2{$_}\n";
- }

Array of Array

```
@Aoa=(
        ["Unix", "Programming Perl",  "Web2.0",  "XML“  ],
        [ 456,645,657,700 ],
        ["Mr.A","Mr.B","Mr.C","Mr.D"],
        [ "Vol1","Vol2","Vol3","Vol4“ ]
);
print $Aoa[0],"\t", $Aoa[0][0],"\n";  # same as $Aoa[0]->[0];

          #  ARRAY Reference            # De-reference
=begin
foreach $rv (@Aoa){
      print "$rv\n";              # Array reference
              foreach (@$rv){ # De-reference
               print "$_\n";
           }
print "---------------------------------\n";
}
=cut
```

Array of Array

```
@AoA=(  [
                    ["D1",D2,D3,D4],   [D5,D6,D7],        [D8,D9,D10]
                  ],
                  [
                      [Unix,Linux,Aix,minix],   [SH,BASH,CSH,TCSH]
                  ]
);
print "@AoA\n";
print "$AoA[0]\t$AoA[1]\n";
print "$AoA[0][0]\t $AoA[1][0]\n";
print "$AoA[0][0][0]\t $AoA[0][0][1]\n";
```

Note: using -> operator try the above print statement.

Array of Hash

- @AoH=(

  { key1=>"Value1", key2=>"value2", key3=>"Value3" },

  { key4=>"Value4", key5=>"value5" } ,

  { key6=>"value6", key7=>"value7"}

); # Array of Hash structure

- print "$AoH[0]\n"; # print HASH Reference
- print "$AoH[0]{key2}\n"; # display value2
- Using **->** operator
- print “$AoH[0]**->**{key2}\n”; # display value2

Array of Hash

- @AoH=(
  {
  key1=>"Value1", key2=>"value2“ ,key3=>"Value3“
  },
  { key4=>"Value4",key5=>"value5"},
  {key6=>"value6",key7=>"value7“}
  );

```
# Array of Hash structure
print "$AoH [0]{key2}\n";

#  using -> operator
print “$AoH[0]->{key2}\n”;
```

Hash of Hash

```
%HoH=(
        K1=>{  key1=>"Value1", key2=>"Value2", key3=>"value3"},
        K2=>{  key4=>"value4", key5=>"value5"},
        K3=>{  key6=>"value6}
);
# Hash of Hash

# %H1=(K1=>"value");   $H1{K1} ---> value   - Single data structure

print "$HoH{K1}\n";    # display HASH Reference
print "$HoH{K1}{key1}\n"; # display Value1
print "$HoH{K3}{key6}\n";  # display value6
# using -> operator de-reference
print "$HoH{K1}->{key1}\n"; # display Value1
print "$HoH{K3}->{key6}\n";  # display value6
```

Hash of Hash

```
use Data::Dumper;
%Emp=(
    name=>{
                Fname=>"John",
                Lname=>"paul”
        },
    Dept=>{
                Dcode=>"E123",
                Dname=>"Sales”
    },
    Place=>{
                street=>"5th A Main",
                city=>"Bangalore"
        }
);
#
print "$Emp{name}{Fname}\t $Emp{Place}{city}\n";
```

Hash of Array

- %HoA=(

```
        name=>["Ram","Kumar","Ashok","BaBu","Xerox"],
        dept=>["Sales","HR","Prod","FI","Admin"],
        place=>["Pune","Chennai","Bangalore","Hyderabad"]
);
print "$HoA{name}\n";
$AR=$HoA{name};
print "@$AR\n";
print "$HoA{name}[0]\t $HoA{dept}[0]\t $HoA{place}[0]\n";
print Dumper(\%HoA);
```

Hash of Array

```
%HoA = (
stones => [ "fred", "barney" ],
sons => [ "george", "jane", "elroy“]
 );
```

To add another array to the hash, you can simply say:

```
$HoA{key} = [ “data1”, “data2”,”data3”,”data4”,”data5”];
```

Thank you