Deniz Yüksel
21600880
Section 1

CS 315

PROGRAMMING

LANGUAGES

HOMEWORK 1

Comparison of Design Issues Regarding Arrays in C, Javascript, Python, Perl and PHP

In this report, I will attempt to point out the following design issues language by language:

1. What types are legal for subscripts?

2. Are subscripting expressions in element references range checked?

3. When are subscript ranges bound?

4. When does allocation take place?

5. Are ragged or rectangular multidimensional arrays allowed, or both?

6. What is the maximum number of subscripts?

7. Can array objects be initialized?

8. Are any kind of slices supported?

C: In the C language, the array subscripts are integers ranging from [0, INT_MAX]. Also, variables that have integer values can be used for subscripts. Float is not allowed and is reported as an error in compile time. Moreover C does not have objects therefore it is not a matter of question that object types can/cannot be legal for subscripting. The below code segment compiles and executes. However the negative subscript gives a logical runtime error. Negative subscripting is allowed in the compile time, but C language does not treat negative subscripts like Perl or Python. In C, array variables are pointers and accessing array elements is achieved by pointer arithmetic. Also, the program crashes at line 18 because I attempted to access an array element at location 11 while the array was initialized with 9 elements. It gives the following error when tried to access an array element that is out of bounds:

```
*** stack smashing detected ***: <unknown> terminated
Aborted (core dumped)
```

Therefore, subscripts in C are range checked so the program crashes because the array is out of bounds.

In C, subscript ranges are bounded for arrays that are created in the stack in the compile time, whereas for a heap allocated array with malloc, ranges are bound in run time. There is also calloc, and calloc furthermore initializes the array elements to zeros when array is created in the heap. The C language also supports jagged arrays, not rectangular. In C, slices are not supported since arrays are not dynamic, meaning that they cannot grow or shrink by automatically. The maximum number of subscripts is predefined for only FORTRAN, but infinite for C, and the programmer specifies it.

```
#include <stdio.h>
#include <stdlib.h>

int main () {
  int i,j;
  int myArray[9]; // Subscript ranges of an array in the stack is bound in compile time.
  int* ptr = malloc(sizeof(int) * 5);
  int* ptr2 = calloc(5, sizeof(int));
  printf("Malloc [%d] = %d\n", 0, ptr[0]);
  printf("Calloc initializes to zero [%d] = %d\n", 0, ptr2[0]);
  int myInitArray[] = {1,2,3}; // an array can be initialized.
  int **multArray = malloc(sizeof(int)*3);
  for(i = 0; i < 3; i++){
        multArray[i] = malloc(sizeof(int)*(i + 1)); // This is a jagged array. Every row has number
has their row no. of elements.
  }


  myInitArray[2] = 5;
  printf("Element[%d] = %d\n", 2, myInitArray[2]); //accessing the modified element...

  /* initialize elements of array n to 0 */
  for (i = 0; i < 8; i++ ) {
    myArray[i] = i + 20; /* set element at location i to i + 100 */
  }
  myArray[9] = -27;
  /* output each array element's value */
  for (j = 0; j < 8; j++ ) {
    printf("Element[%d] = %d\n", j, myArray[j] );
  }
  printf("LastElement[%d] = %d\n", 9, myArray[-5]);
  myArray[11] = 5;


  return 0;
}
```

Python: In python, positive and negative integers including zero is legal for subscripting. Strings are not legal for subscripting. For dictionaries and tuples strings are also accepted. Regarding the code below, we can see that negative subscripts are used to reach to the end of the list. In Python, the arrays we know are lists. Array is another python module that can be imported an used. In this report, I will give examples of a list below. Python is not a compiled language, it's an interpreted language. Therefore when an element in an array which is out of bounds is attempted to be reached, the checking happens in run time. Also, arrays are allocated dynamically in the heap, in the very line where the code segment is executed. However in Python, an element that is out of the range cannot be reached. This is valid for both negative and positive subscripts. The program crashes in the runtime and gives an error for the following program, and it cannot reach the next line where there is another out of bounds error as well. Likewise C the maximum number of subscripts is predefined for only FORTRAN, but infinite for Python, meaning that the programmer specifies it. Lists in Python can be initialized directly and with multiple type of elements inside. Python supports slicing for lists.

```
myList = [3, 4, 5, 10, -7, 21]
print( myList[2])
print( myList[-2])
print( myList[-8])
print( myList[10])
```

This code segments the lines below (continued in the next page):

```
5
```

```
-7
Traceback (most recent call last):
  File "yuksel_deniz_python.py", line 4, in <module>
    print( myList[-8])
IndexError: list index out of range
```

The below code is without errors version with additional features:

```
# coding=utf-8
myList = [3, 4, 5, 10, -7, 21]
print(myList[2])
print(myList[-2])
#print(myList[3.5])
myMixedList = [1, 2, 3, "kedi", "köpek", 4, 5]
print(myMixedList[3] + " MİYAVLAR " + myMixedList[4] + " HAVLAR!")
print(myList[1:4])

myJaggedArray = [[1,2,3], [4,5], [6]]
print(myJaggedArray[1][0])
```

This code segment prints and runs without any errors.

```
5
-7
kedi MİYAVLAR köpek HAVLAR!
[4, 5, 10]
4
```

Javascript: In the Javascript the language allows subscript types of Strings, any int, object, float. However only positive integer subscripts are not undefined. There is no range check in Javascript as well. However in the console, inappropriate ranges output undefined. Subscript ranges are bound during run time because Javascript does not have a compile time. In Javascript, jagged arrays are supported. Theoretically rectangular arrays can be formed by initializing a jagged array with equal number of elements in all rows. But I don't think we consider that as a rectangular array. Allocation for arrays take place in the runtime in the heap. Arrays are managed dynamically in Javascript language. They are allocated in the heap. Arrays can be initialized or can be declared using keyword "var". Javascript supports slices with its method that can be applied through the console on the table.

```
<!DOCTYPE html>
<html>
	<body>
		<p>PL HW1 Javascript fun</p>
	<script>
		//array initialization...
		var myJSArray = ["word", "WELCOME", "zero", 4, 55];
		var intArray = [ 1, 3, 4, 19, 20, -2, -36];
		alert( myJSArray[1]);		// Greets the grader.
		var initArr = [];
		initArr.push(0);
```

```
                initArr.push(1);
                initArr.push(2);
                initArr.push(4);
                initArr.push(10);
                initArr.push(20);

                //range checking
                console.log(initArray[-50]);   // prints undefined to the console.
                console.log(initArray[-100]);// prints undefined to the console.


                console.log(myJSArray[0]);   // prints word to console.
                console.log(initArr[2]);     // prints 2 to console.
                console.log(initArr[5.5]);   // prints undefined to console.
                console.table(myJSArray.slice(2));   //prints to the console the elements after 1st
index.
                console.table(myJSArray.slice(1, 3));
                //prints to the console the elements between index 1 and 3. 1 included.
                myJaggedArray = [[0, 5, 10, 20], [1, 22, 5], [2, 3], [5]];      // This is the jagged array.
                console.log(myJaggedArray[2][0]);  // prints 2 to the console.
                console.log(myJaggedArray[0][3]);  // prints 20 to the console.

        </script>
        </body>
</html>
```

Perl: Like Python, Perl is also an interpreted language. So it does not have a compile time. Perl supports positive and negative subscripting in range. Perl also allows string subscripting but prints the element at 0th index when a string is provided as a subscript. Perl does not give an error if subscript range is out of bounds. If tried to print, Perl will print empty. Therefore does not generate an out of bounds error. Arrays are allocated in the run time and in the heap, they are managed dynamically. One can initialize an array with given elements and then attempt to insert an element to a location which is out of bounds. This is allowed and size will jump up to that new subscript number if it's greater than the previous size. However the elements in between will not be reached, if printed they will be space characters. Arrays can be initialized with prior elements in Perl. Also, slicing is supported. The code is included below:

```
#!/usr/bin/perl

@ints = (10,20,30,40,50, 60, 70);
print "$ints[-10]\n";
print "$ints[10]\n";
print "$ints[1]\n";
print "$ints[2]\n";

print scalar "@ints\n";          # Prints the elements.
print "##################\n";
@ints[20] = 200;
print "##################\n";
print scalar "@ints\n";          # Prints the elements after an insertion of an element out of bounds.
# In between, elements are displayed as spaces
```

```
print "##################\n";

@slice = @ints[0,1,2,6];
print "##################\n";

print "@slice \n";

print "##################\n";

@myMixArray = (1, 2, 3, 4, 5, 6, "word", "yes", "this is perl yeah", 5, 10, 2);
print "##################\n";

print "@myMixArray \n";

print "##################\n";

@jagged = ([0, 1], [3], [5, 6, 7, 8], [1]);

print "$raggedarray[2][2] \n"; #Prints 7.
```

PHP: PHP language is interpreted and therefore does not have compile time errors. Integers, strings, float are legal for subscripting. However float values are truncated and their integer values are taken. For arrays, there is a range check in the runtime and the output will be a PHP error and therefore it will not be executed. PHP supports ragged arrays but rectangular arrays can also be formed by using ragged arrays. PHP supports slicing for arrays. In PHP, arrays are allocated during runtime and in the heap. Arrays can be initialized with prior elements. The below code runs and executes but generates the following PHP Notices:

```
<!DOCTYPE html>
<html>
<body>

<?php
$myArray = array(1, 3, 4, 5, 100, 40, "emre", "deniz", "ege", 190, "sevgi");

/*Echoing the elements...*/
echo $myArray[0];
echo "\n";
echo $myArray[3];
echo "\n";
echo $myArray[4];
echo "\n";
echo $myArray[7];
echo "\n";
echo $myArray[9];
echo "\n";
echo $myArray[110];
echo "\n";
echo $myArray[-4];
echo "\n";
echo $myArray[-50];
echo "\n";

$jagged = array( array(1, 3, 4), array (0, 20, 35, 50), array(100000));
echo $jagged[2][2]; /*prints 35*/

$mySliceArray = array_slice($myArray,1,7);
echo $mySliceArray[0];
echo "\n";
echo $mySliceArray[3];
echo "\n";
echo $mySliceArray[4];
echo "\n";
echo $mySliceArray[7];
echo "\n";
echo $mySliceArray[9];

?>

</body>
</html>
```

```
PHP Notice:  Undefined offset: 110 in source_file.php on line 19
PHP Notice:  Undefined offset: -4 in source_file.php on line 21
PHP Notice:  Undefined offset: -50 in source_file.php on line 23
PHP Notice:  Undefined offset: 2 in source_file.php on line 27
PHP Notice:  Undefined offset: 7 in source_file.php on line 36
PHP Notice:  Undefined offset: 9 in source_file.php on line 38

<!DOCTYPE html>
<html>
<body>

1
5
100
deniz
190



3
100
40


</body>
</html>
```