Data Characteristic Binding Times Operations Sequence Control

#1 Example 1.1

Consider the following **Java** code snippet (which is **inside a function**). Assume integers are 4 bytes.

String name = "Peg Board";

For each of the specified **binding times**, describe what is bound to identifier **name** involving **location, data type**, **structure**, **size**, and **value**.

A. Compile/Load Time

**Data type – String**

**Structure – String, object has its attributes (offset and length which are int and pointer)**

**Size – string object = 16 bytes (since java stores len, offset and string ptr; takes sizeof(len) + sizeof(offset) + sizeof(pointer))**

**String literal – literal = 9\*2 bytes + length = 22 bytes**

**2byte chars + 4**

**Additional overhead for garbage collection and storage management?**

B. Execution Time

**Location – automatic variable heap name, heap –string object, literal pool – string literal**

**Automatic is given memory from stack on function entry and popped off (freed) on exit.**

**Heap memory is allocated when the statement is encountered and isn’t freed until garbage collection**

**String literal placed in literal pool.**

**Value – “Peg Board”**

**Value of variable can change, string cannot.**

**Lengh set to 9, offset to 0.**

C. Language Implementation Time

**Size of pointers**

**Use of UTF-16**

**Format of int values used in objects.**

#1 Example 1.2

Consider the following C code snippet (which is **inside a function**) in answering the questions about binding:

|  |  |
| --- | --- |
| **Line #** | **Code Snippet from within a function** |
| 1 | int arrayM[] = {10, 20, 30, 40, 0}; |
| 2 | static int iHigh = 0; |
| 3 | int i; |
| 4 | for (i = 0; arrayM[i] != 0; i += 1) |
| 5 | { |
| 6 | if (arrayM[i] > iHigh) |
| 7 | iHigh = arrayM[i]; |
| 8 | } |

1. C doesn’t have unbounded arrays
2. Int data type, structure = primitive
3. Variable types – automatic and static/global

Structure - primitive, homogeneous array, record structure; self-referencing; object with attributes x, y, and z

A. For **three** different line numbers, describe the **compile-time** bindings:

Line# Explanation

1 **data type = int, size = 4\*5 = 20 bytes, structure = homogenous array bounded**

**2 data type = int, size = 4, structure = primitive, static location, value = 0;**

**3 data type = int, size = 4, structure = primitive,**

B. For **four** different line numbers, describe the **runtime** bindings:

Line# Explanation

**1 automatic location, assigned the values known at compile time {10, 20, 30, 40, 0}**

**2 n/a**

**3 automatic location**

**4 assign value to i**

**7 assign value to iHigh**

C. One example of **language implementation** binding:

Line# Explanation

**1 representation and size of integer**

**2 representation and size of integer**

**3 representation and size of integer**

#1 Example 1.3

List the **operational characteristics** of the **C logical or** operator (symbol: **||**). Example: x>5 || y==z.

**Syntax = Infix**

**Invariant, produces same data type regardless of what is on each side (T or F int)**

**Precedence needed in C**

**Binary**

**Explicit operator and operands.**

**Only works properly for numerics, pointers, char; doesn’t work for structs.**

**Not compound**

**Short circuit**

C

* data type, structure and *size* are bound at compile time
* arrays can be given a size at runtime (with C99, size can change at runtime)
* static and global variables are bound to location at compile/load time
* automatic variables are bound to locations at runtime (offsets within an activation record)

Java

* datatype and structure are bound at translation time
* size of arrays can be provided at runtime
* parameters receive a reference to an argument which can be an object
* objects contain information which describes what they need:
  + strings have current value location, offset, and length
  + other classes whatever they need (e.g., size)