학번 : 20171646

이름 : 박태윤

**1. Some programming languages are typeless. What are the obvious advantages and disadvantages of having no types in a language?**

Typeless language는 프로그래밍을 할 때 데이터 타입을 따로 선언할 필요가 없는 프로그래밍 언어를 의미한다. 이러한 프로그래밍 언어의 장점으로는 데이터 타입을 반드시 선언해야 하는 언어에 비해 데이터 타입 및 구조체에 대한 완벽한 이해가 없어도 프로그래밍을 할 수 있기에 배우기 쉽다는 장점이 존재하며 코드를 수정하는데 간편하다는 장점이 존재한다. 또한 해당 언어를 사용하면 유연한 프로그래밍을 할 수 있다는 장점이 존재한다. 하지만 type checking관점에서 프로그래머가 짠 코드 내에서 정확한 타입 표현 및 연산을 하고 있는지를 확인하는 것이 전적으로 프로그래머의 책임이 된다는 단점이 존재한다. 또한 팀 단위 협력 프로젝트 내에서 코드를 유심히 살펴보지 않으면 해당 함수가 어떤 타입을 return하고 특정 변수가 어떤 타입을 가지고 있는지 알기 어렵다는 단점이 존재한다.

**2. Consider the following JavaScript program: List all the variables, along with the program units where they are declared, that are visible in the bodies of sub1, sub2, and sub3, assuming static scoping is used**

텍스트이(가) 표시된 사진

자동 생성된 설명

현재 코드에서 확인할 수 있는 서브루틴 sub1, sub2, sub3가 모두 main내에서 선언이 된 서브루틴이라고 가정을 하면 static scoping 기법에서는

|  |  |  |  |
| --- | --- | --- | --- |
| **subroutine** | **variable** | **visible** | **define location** |
| sub1() | var a | o | sub1() |
| var b | x | - |
| var x | o | main() |
| var y | o | sub1() |
| var z | o | sub1() |
| var w | x | - |
| sub2() | var a | o | sub2() |
| var b | o | sub2() |
| var x | o | main() |
| var y | o | sub1() |
| var z | o | sub2() |
| var w | x | - |
| sub3() | var a | o | sub3() |
| var b | x | - |
| var x | o | sub3() |
| var y | o | main() |
| var z | o | main() |
| var w | o | sub3() |

**3. Consider the following Python program. List all the variables, along with the program units where they are declared, that are visible in the bodies of sub1, sub2, and sub3, assuming static scoping is used.**

**텍스트이(가) 표시된 사진

자동 생성된 설명**

현재 코드에서 확인할 수 있는 서브루틴 sub1, sub2, sub3가 모두 main내에서 선언이 된 서브루틴이라고 가정을 하면 static scoping 기법에서는

|  |  |  |  |
| --- | --- | --- | --- |
| **subroutine** | **variable** | **visible** | **define location** |
| sub1() | var a = 7 | o | sub1() |
| var b | x | - |
| var x = 1 | o | main() |
| var y = 9 | o | sub1() |
| var z = 11 | o | sub1() |
| var w | x | - |
| sub2() | var a = 13 | o | sub2() |
| var b | x | - |
| var x = 15 | o | main() |
| var y = 3 | o | main() |
| var z = 5 | o | main() |
| var w = 17 | o | sub2() |
| sub3() | var a = 19 | o | sub2() |
| var b = 21 | o | sub3() |
| var x = 15 | o | main() |
| var y = 3 | o | main() |
| var z = 23 | o | sub3() |
| var w = 17 | o | sub2() |

**4. Consider the following skeletal C program. Given the following calling sequences and assuming that dynamic scoping is used, what variables are visible during execution of the last function called? Include with each visible variable the name of the function in which it was defined.**

**a. main calls fun1; fun1 calls fun2; fun2 calls fun3.**

**b. main calls fun1; fun1 calls fun3.**

**c. main calls fun2; fun2 calls fun3; fun3 calls fun1.**

**d. main calls fun3; fun3 calls fun1.**

**e. main calls fun1; fun1 calls fun3; fun3 calls fun2.**

**f. main calls fun3; fun3 calls fun2; fun2 calls fun1.**

**텍스트이(가) 표시된 사진

자동 생성된 설명**

dynamic scoping기법이 사용되었다. 그러므로 마지막으로 호출된 함수에서 각 변수의 visibility는 다음과 같다.

a. main calls func1; fun1 calls fun2; fun2 calls fun3.

|  |  |  |  |
| --- | --- | --- | --- |
| **last call function** | **variable** | **visible** | **define location** |
| fun3() | int a | o | main() |
| int b | o | fun1() |
| int c | o | fun2() |
| int d | o | fun3() |
| int e | o | fun3() |
| int f | o | fun3() |

b. main calls fun1; fun1 calls fun3.

|  |  |  |  |
| --- | --- | --- | --- |
| **last call function** | **variable** | **visible** | **define location** |
| fun3() | int a | o | main() |
| int b | o | fun1() |
| int c | o | fun1() |
| int d | o | fun3() |
| int e | o | fun3() |
| int f | o | fun3() |

c. main calls fun2; fun2 calls fun3; fun3 calls fun1.

|  |  |  |  |
| --- | --- | --- | --- |
| **last call function** | **variable** | **visible** | **define location** |
| fun1() | int a | o | main() |
| int b | o | fun1() |
| int c | o | fun1() |
| int d | o | fun1() |
| int e | o | fun3() |
| int f | o | fun3() |

d. main calls fun3; fun3 calls fun1.

|  |  |  |  |
| --- | --- | --- | --- |
| **last call function** | **variable** | **visible** | **define location** |
| fun1() | int a | o | main() |
| int b | o | fun1() |
| int c | o | fun1() |
| int d | o | fun1() |
| int e | o | fun3() |
| int f | o | fun3() |

e. main calls fun1; fun1 calls fun3; fun3 calls fun2.

|  |  |  |  |
| --- | --- | --- | --- |
| **last call function** | **variable** | **visible** | **define location** |
| fun2() | int a | o | main() |
| int b | o | fun1() |
| int c | o | fun2() |
| int d | o | fun2() |
| int e | o | fun2() |
| int f | o | fun3() |

f. main calls fun3; fun3 calls fun2; fun2 calls fun1.

|  |  |  |  |
| --- | --- | --- | --- |
| **last call function** | **variable** | **visible** | **define location** |
| fun1() | int a | o | main() |
| int b | o | fun1() |
| int c | o | fun1() |
| int d | o | fun1() |
| int e | o | fun2() |
| int f | o | fun3() |

**5. Consider the following program, written in JavaScript-like syntax. Given the following calling sequences and assuming that dynamic scoping is used, what variables are visible during execution of the last subprogram activated? Include with each visible variable the name of the unit where it is declared.**

**a. main calls sub1; sub1 calls sub2; sub2 calls sub3.**

**b. main calls sub1; sub1 calls sub3.**

**c. main calls sub2; sub2 calls sub3; sub3 calls sub1.**

**d. main calls sub3; sub3 calls sub1.**

**e. main calls sub1; sub1 calls sub3; sub3 calls sub2.**

**f. main calls sub3; sub3 calls sub2; sub2 calls sub1.**

**텍스트이(가) 표시된 사진

자동 생성된 설명**

dynamic scoping기법이 사용되었다. 그러므로 마지막으로 호출된 함수에서 각 변수의 visibility는 다음과 같다.

a. main calls sub1; sub1 calls sub2; sub2 calls sub3.

|  |  |  |  |
| --- | --- | --- | --- |
| **last call function** | **variable** | **visible** | **define location** |
| sub3() | int a | o | sub3() |
| int b | o | sub2() |
| int x | o | sub3() |
| int y | o | sub1() |
| int z | o | sub2() |
| int w | o | sub3() |

b. main calls sub1; sub1 calls sub3.

|  |  |  |  |
| --- | --- | --- | --- |
| **last call function** | **variable** | **visible** | **define location** |
| sub3() | int a | o | sub3() |
| int b | x | - |
| int x | o | sub3() |
| int y | o | sub1() |
| int z | o | sub1() |
| int w | o | sub3() |

c. main calls sub2; sub2 calls sub3; sub3 calls sub1.

|  |  |  |  |
| --- | --- | --- | --- |
| **last call function** | **variable** | **visible** | **define location** |
| sub1() | int a | o | sub1() |
| int b | o | sub2() |
| int x | o | sub3() |
| int y | o | sub1() |
| int z | o | sub1() |
| int w | o | sub3() |

d. main calls sub3; sub3 calls sub1.

|  |  |  |  |
| --- | --- | --- | --- |
| **last call function** | **variable** | **visible** | **define location** |
| sub1() | int a | o | sub1() |
| int b | x | - |
| int x | o | sub3() |
| int y | o | sub1() |
| int z | o | sub1() |
| int w | o | sub3() |

e. main calls sub1; sub1 calls sub3; sub3 calls sub2.

|  |  |  |  |
| --- | --- | --- | --- |
| **last call function** | **variable** | **visible** | **define location** |
| sub2() | int a | o | sub2() |
| int b | o | sub2() |
| int x | o | sub3() |
| int y | o | sub1() |
| int z | o | sub2() |
| int w | o | sub3() |

f. main calls sub3; sub3 calls sub2; sub2 calls sub1.

|  |  |  |  |
| --- | --- | --- | --- |
| **last call function** | **variable** | **visible** | **define location** |
| sub1() | int a | o | sub1() |
| int b | o | sub2() |
| int x | o | sub3() |
| int y | o | sub1() |
| int z | o | sub1() |
| int w | o | sub3() |