Quiz #4 - Programming Language, 2017, Term 2

Name ( ) St. Id ( )

NB) 배점동일, 부분점수 인정

1. 다음 물음에 답하시오.

A. 변수와 기본적으로 연관된 것들 5가지를 나열하시오.

이름(Name)

주소(Address)

타입(Type)

값(Value)

수명(Lifetime)

B. 유효 범위 (Scope)에 대해 설명하시오.

유효범위 : 그 이름을 통해 연결된 대상을 접근할 수 있는 문장들의 집합

정적 유효범위 지정(static scoping) : 이름의 유효범위가 소스 프로그램 내의 이름의 위치에 의하여 정해지는 것

대부분의 현대의 언어들은 정적(구문적) 유효범위 지정을 사용한다.

문맥자유 문법 Context-free grammar (BNF)

C. 기호표 (Symbol Table)에 대해 설명하시오.

기호표(Symbol Table)는 번역기에 의하여 유지되는 자료 구조로서 모든 선언된 이름들과 그 연결 정보를 관리하는 역할을 한다.

본 절에서는 모든 선언된 이름이 그 지역적인(local) 유효범위 안에서 중복되지 않는다고 가정한다.

지역 유효범위를 위한 자료 구조로는 임의의 사전(directory)의 구현을 모두 사용할 수 있으며, 이 경우 이름이 키 값으로 사용된다.

D. 가시성 (Visibility)에 대해 설명하시오.

이름 선언의 참조 환경이 어떤 이름에 대한 한 참조를 포함하고, 안쪽의 유효범위에서 그 이름을 재선언 하지 않았을 경우, 이름이 참조에 대하여 가시적(visible)이라고 한다.

안쪽의 유효범위에서 이름이 다시 선언되면 바깥쪽의 선언을 가리게(hide) 된다.

2. 다음 물음에 답하시오.

A. C와 C++는 선언과 정의를 구별한다. 차이점은 무엇인가? 각각의 예를 제시하시오.

A definition refers to the place where a variable is created or assigned storage. A declaration refers to places where the nature of a variable is stated but no storage is allocated. Declarations commonly occur in header files, while definitions do not.

B. C와 C++의 헤더 파일의 사용을 설명하시오. 왜 Java는 헤더파일을 사용하지 않는가?

External declarations of variables, types and functions are commonly collected in a separate header file, which is included at the beginning of any source file that references these. Historically, this has occurred because compiled source files do not contain type information.

In contrast, Java stores type information in its `class` files, so that the need for a separate header file does not exist.

C. C 언어에서 우변값이 좌변값으로 사용될 수 없는 세가지 예를 제시하시오. 좌변값의 예를 세가지 더 제시하시오. 우변값이 될 수 없는 좌변값이 있는가? 설명하시오.

R-values that cannot be L-values: literals (0, 'c', etc.); function calls; an expression, such as `x+1`. L-values: variable names (e.g., `a`), subscripted names (e.g., `x[i]`), pointer references (e.g., `*p`).

All L-values can be R-values since an address can be replaced by the value stored at that address.

3. 다음 물음에 답하시오.

A. Big-endian과 little-endian 컴퓨터의 차이점은 무엇인가? 각 구분에 대하여 최소한 한 개의 컴퓨터 구조를 인용하여 제시하시오.

A *big-endian* machine, such as the Motorola M680x0 series, numbers both bits and bytes from left to right; i.e., from the most significant byte to the least.. So the bytes for a 32-bit integer are numbered: 0 1 2 3.

In contrast, a *little-endian* machine, such as the DEC Vax, numbered both bits and bytes from right to left; i.e., from least significant to most. So the bytes for a 32-bit integer are numbered: 3 2 1 0.

Consider the value 0x01234567. On a big endian machine this value would be treated as: 01 23 45 67, whereas on a little endian machine the value would be treated as: 67 45 23 01.

The data given in Table 5.2 assumes a big-endian machine.

B. 다음의 C 프로그램 일부를 참고하시오

```
char a[20];

char b[20];
```

저장문 a = b는 컴파일 오류를 발생시킨다. 저장문에 대한 두 가지 가능한 해석을 제시하고 각각의 경우에 대하여 해당 해석의 요구에 맞도록 선언문이나 저장문을 수정하시오.

The assignment statement `a = b` generates a compile error. One possible interpretation is that the statement copies the string `b` to the variable `a`. A second interpretation is that the statement assigns the value of the pointer `b` to the pointer `a`. This interpretation would be allowed if both variables had been declared `int *`.

<End of the Quiz>