Variable

1. name

2. address -> 배정문의 왼쪽 l-value

3. value -> 배정문의 오른쪽 r-value

aliases 별칭 하나의 메모리 번지를 서로 다른 이름으로 접근이 가능할 때를 말함

5.3 1) union 타입 2) 포인터 변수 값 3) 함수 전달 방식에서 call by reference방식

4. type

5. lifetime

6. scope

5.4

바인딩이라는 것은 entity(주로 변수) <-> attribute 변수에 속성을 결합시키는 것

속성을 결합시키는 타이밍을 binding time

속성이 하나 결합되면 쉽게 바뀌지 않음

5.4.1

바인딩의 종류 2가지

1. static 실행 시 값이 바뀌지 않음

2. dynamic 실행 중에 값이 변함

컴파일할 때 정해짐

5,4,2

프로그램에서 대부분은 static type binding

5.4.2.2.

dynamic type binding 에서는 타입을 미리 정하지 않고 변수를 넣을 때 마다 설정함. 포인터 변수로 구현 해야함 interpreter 방식을 씀

90년대 전에는 다 static type binding 그 이후로는 dynamic을 씀 ex) 파이썬, 루비, 자바스크립트

2가지 단점

컴파일러가 서로 같은 type인지 확인할 수 없기 때문에 less reliable

실행속도가 느림

5,4,3

lifetime 변수가 메모리에 할당 받는 시간부터 할당이 없어지는 순간까지의 시간

static은 static 메모리에 프로그램 내내 고정적으로 할당하고 dynamic은 필요할 때 메모리에 할당하고 필요 없을때 제거함 ex) malloc, delete

5.4.3.1

스태틱 변수는 프로그램 내내 고정적으로 메모리에 할당받음

지역변수는 static을 사용하여 전역변수처럼 쓸 수 있음

지역 번수는 함수 내에서만 필요하기 때문에 함수가 종료되면 메모리도 회수해 감

지역 변수이면서 history sensitive한 값은 static 선언을 이용해야 함 (자바에서는 static이 다름)

5.4.3.2

지역변수는 dynamic 할당함

함수의 activate가 끝나면 공간을 회수함

5.4.3.3

동적 자료구조 explicit heap-dynamic variables

자바에서는 기본형 외 모든 자료구조가 오브젝트다. 자바 오브젝트는 모두 동적 자료구조이다

자바에서는 동적자료구조를 지울 수 없다 그래서 jvm에서 처리해줌

5.5

scope 2가지

1. global : 어디서나 접근 가능

2. local : 그 함수 block안에서만 접근 가능하다

함수를 쓰는 목적

1. 재사용

2. abstraction

3. information hiding

5,5,1 static scope

static 과 dynamic은 구조상 결정됨

ALGOl60 함수의 Nesting 허용 가능 함수안에 함수 가능 C에서는 안됨

sub1() {

int sum;

sub2(){

int sum;

sub1.sum= sub2.sum – 2; // 아무것도 없으면 가까운 것 scope를 정해줄 수 있음

5.5.2 Blocks

코드의 덩어리들 ALGOL60에서는 begin end 사용 C에서는 {} 사용

5.5.3 Declaration Order

C언어에서는 선언문 위치가 자유로움

5.5.4 Global Scope

선언문 위치에 따라 지역변수와 전역변수로 나뉨

5.5.5 Evaluation of Static Scoping

프로그램에서 전역변수를 많이 쓰게 되면 편리하기는 하지만 모듈 간에 coupling(결합)이 생김

전역변수를 최소화 해야함