11 강

컴퓨터과학 개론

프로그래밍언어(1)

컴퓨터과학과 정광식 교수

학습목차

1 프로그래밍 언어의 개요

2 프로그래밍 언어의 파스 트리

3 프로그래밍 언어의 분석

4 프로그래밍 언어의 공통 개념

01

프로그래밍 언어의 개요

프로그래밍 언어의 개요

개념

- 프로그래밍 언어는 사람의 의도를 컴퓨터에게 전달하여 컴퓨터에게 작업을 수행시키기 위해 만들어짐

- 프로그래밍 언어는 사람의 의도를 추상화하여 압축된 언어로 컴퓨터에 전달되어야 함
- 동시에 컴퓨터에서 실행될 수 있는 이진 코드(binary code)로 번역되어야 함

프로그래밍 언어의 개요

개념

- 만약 일상 언어로 프로그램을 작성할 수 있다면 누구나 자신의 필요에 따라 프로그램을 작성할 수 있음
- 하지만, 일상 언어로 작성된 프로그램은 애매모호한 의미를 가질 수 있으며, 하나의 문장에서 두 개의 의미로 해석이 될 수도 있음

프로그래밍 언어의 개요

개념

- 프로그래밍 언어는 의미적으로 애매모호함이 없고 어떤 경우에도 동일한 의미로 해석되어야 함
- 프로그래밍 언어는 구문론적 측면에서 명확하게 정의되어야 하며, 의미론적 측면에서 언제나 동일하게 해석되어야 함
- 프로그래밍 언어는 논리적으로 설계되어 컴퓨터가 처리할 수 있는 이진코드로의 변환이 명확하고 정확하게 되어야 함

프로그래밍 언어의 전형

종류

- 프로그래밍 언어는 이진 코드에 가까운 어셈블리어와 자연어에 가까운 고급언어로 분류됨
- 어셈블리어와 고급언어사이에 다양한 프로그래밍 언어가 존재함

프로그래밍 언어의 전형

기계어

000000 00001 00010 00110 00000 100000

- 0과 1의 이진수로 구성되는 언어로 컴퓨터 하드웨어를 직접적으로 제어하기 위한 전기 신호의 표현 형태로 전달될 수 있는 수준의 언어임
- 0과 1로 이루어지기 때문에 사람이 의미를 이해하기 어렵고, 프로그램 작성이 매우 어려움
- 하드웨어나 컴퓨터 구조에 따라 기계어의 구성과 명령어 (0과 1의 나열)가 달라지기 때문에 범용성이 떨어짐

프로그래밍 언어의 전형

어셈블리어

mov al, 161h

- 기계어의 0과 1로 이루어진 명령어를 사람의 언어와 유사한 알파벳 심벌 형태로 바꾼 언어임

- 기계어보다는 훨씬 읽기 편하지만 프로그램의 실행 논리를 컴퓨터가 실행하는 논리 순서에 맞추어 생각해야 하기 때문에 이해하기 쉽지 않음

프로그래밍 언어의 전형

```
FOR A = 1 TO 10
PRINT StudentName$; " ";
NEXT A
```

초창기의 고급 프로그래밍 언어

- 연산, 수행 제어, 메모리 접근 등의 프로그램을 사람의 자연어에 유사한 형태로 표현함
- 포트란(FORTRAN, 과학/공학 계산용 언어, 최초의 고급언어)이나 코볼(COBOL, 비지니스용 언어) 등이 초기의 고급 프로그래밍 언어에 해당됨(1950년대 말)
- 1960년대 중반에 등장한 베이직(Basic)도 1980년대 마이크로컴퓨터에서 많이 사용됨

프로그래밍 언어의 전형

함수형 프로그래밍 언어

```
(+ 3 2);
(length (a b c));
(+ 1 (if t 2 3)) ;
```

- 기본적으로 수식(expression)의 연속으로 이루어져 있고 함수들을 사용해 수식을 변환함
- 수식은 사칙 연산 뿐만 아니라 일반적인 의미의 모든 함수를 의미하고, 함수의 결과를 다른 함수의 입력 값으로 사용함
- 리스프(LISP, 1950년대 말)와 같이 심벌의 리스트를 연산의 기본 단위로 하기도 함
- 스킴(Scheme), ML 등의 고급 언어도 함수형 언어에 포함됨

프로그래밍 언어의 전형

구조적 프로그래밍 언어

- 우리가 현대의 프로그래밍 언어에서 사용하는 많은 개념들이 구조적 프로그래밍 언어에서 도입됨
- 1950년대 말에 나온 알골 60(Algol 60)은 조건문과 반복문을 사용하여 실행 흐름을 제어하고, 블록(block) 구조, 함수 호출 등 주요 개념을 도입함

프로그래밍 언어의 전형

구조적 프로그래밍 언어

- 이후의 구조적인 프로그래밍 언어인 파스칼(Pascal), C, 모듈라-2(Modula-2) 등에 영향을 줌

```
if (a >= b)   {
     d = a – b;
     printf("%d", b);
  }
else
    printf("%d", a);
```

프로그래밍 언어의 전형

논리형 프로그래밍 언어

```
father(a, b)
father(b, c)
grandfather(X, Z) :- father(X, Y), father(Y, Z)
```

- 형식 논리로 사실(fact)들과 규칙(rule)들로 이루어진 문제 도메인 모델을 정의함
- 원하는 결과를 얻기 위해 문제 도메인에 대한 질의를 주어서 논리적인 추론에 기초한 결과가 나오게 하는 선언형 언어임
- 1970년대에 등장하여 1980년대에 인공지능분야의 인기와 더불어 많은 주목을 받은 프롤로그(Prolog)가 대표적인 예라고 할 수 있음

프로그래밍 언어의 전형

객체지향 프로그래밍 언어

- 객체(object) 개념을 정의하고,
  객체에 대한 연산(메소드)과 성질(멤버 변수)을 정의하여 프로그램을
  작성하는 언어임

- 구조적인 프로그래밍 언어와 달리 객체 중심의 사고의 틀을 제공함

```
public class HelloWorldApp {
  public static void main(String[] args) {
    System.out.println("Hello World!");
  } }
```

프로그래밍 언어의 전형

객체지향 프로그래밍 언어

- 최초로 클래스 개념을 갖춘 Simula 67은 1960년대 말에 등장했지만 C++, 스몰톡-80(Smalltalk 80)등 은 1980년대에 등장

- 1990년대 중반에 등장한 자바(Java)도 현재 가장 대표적인 객체지향 프로그래밍 언어임

프로그래밍 언어의 전형

스크립트 언어

- 유닉스(Unix)와 같은 운영체제의 관리와 자동화를 위해 만들어져 사용되기 시작한 언어임
- 쉘 스크립트(sh, bash, csh 등)와 패턴 처리 스크립트 언어(awk, sed) 등에서 시작됨
- 펄(Perl), 파이선(Python) 등 스크립트 언어들이 웹 기반 서비스에서 많이 사용되고 있음

정리 문제

다음과 같은 프로그램이 속하는 언어 부류는 무엇인가?

```
(+ 3 2);
(length (a b c));
(+ 1 (if t 2 3)) ;
```

- 객체지향형 언어
- 구조적 언어
- 논리형 언어
- 함수형 언어

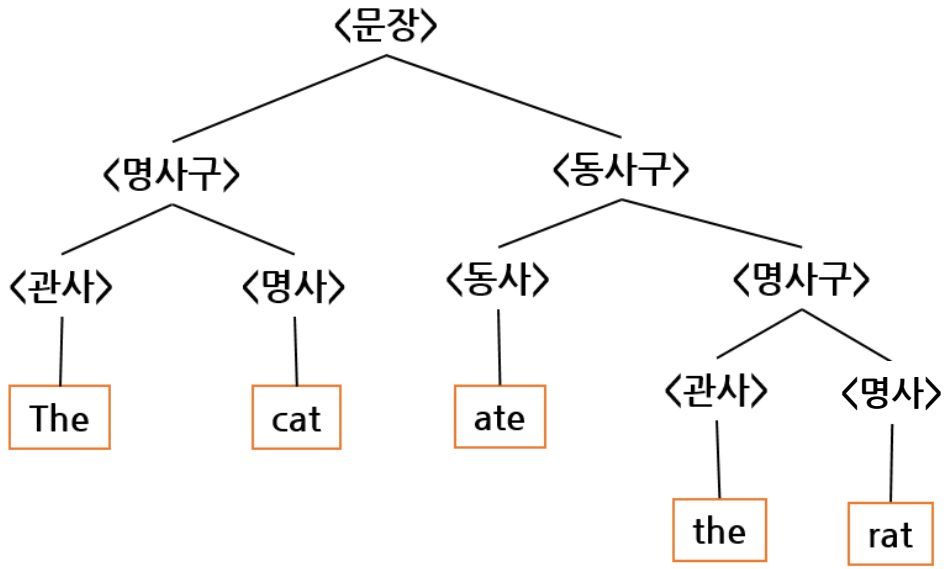
02

프로그래밍 언어의 파스 트리

프로그래밍 언어의 파스 트리

개요

- 모든 언어는 사용될 수 있는 단어들의 집합과 단어들이 나열되어 구조에 대한 규칙(문법)에 따라 문장을 생성하고,
- 각각의 문장은 실세계와 연결되는 의미를 가짐



프로그래밍 언어의 파스 트리

개요

- 대부분의 프로그래밍 언어는 형식 문법을 사용해서 언어의 구조를 기술함

⇒ 문장의 구조는 파스 트리의 형태로 표현하면 이해가 빠르고 문법의 모호성을 파악하기 쉬움

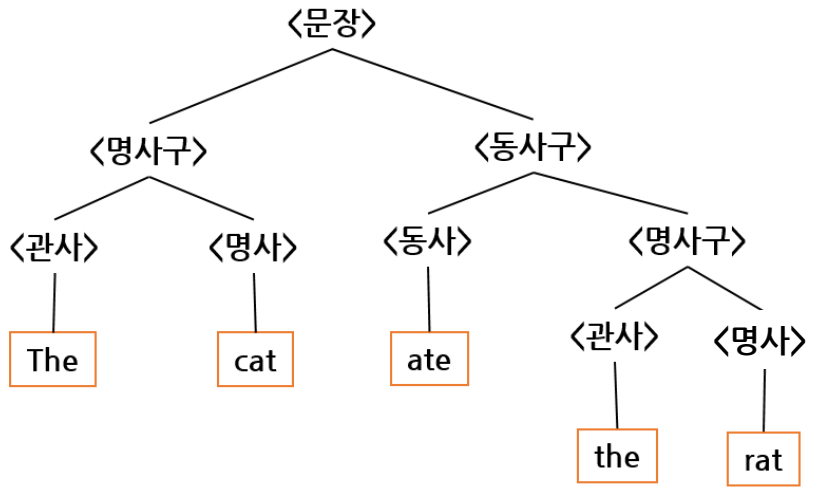
형식문법의 요소

프로그래밍 언어의 파스 트리

단말 심벌

- 문장을 이루는 단어들을 단말 심벌이라 함

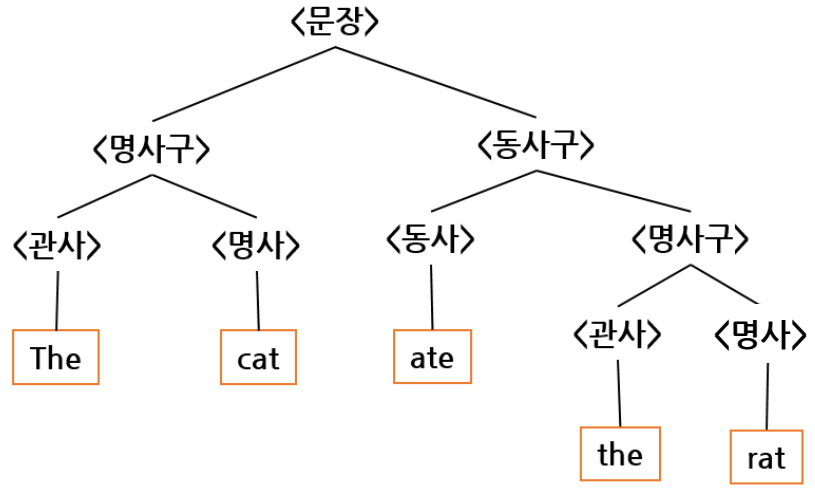
  ⇒ 파스 트리의 단말 노드에 해당됨



- 자연어에서는 사전에 나오는 모든 단어가 단말 심벌이 됨

형식문법의 요소

프로그래밍 언어의 파스 트리

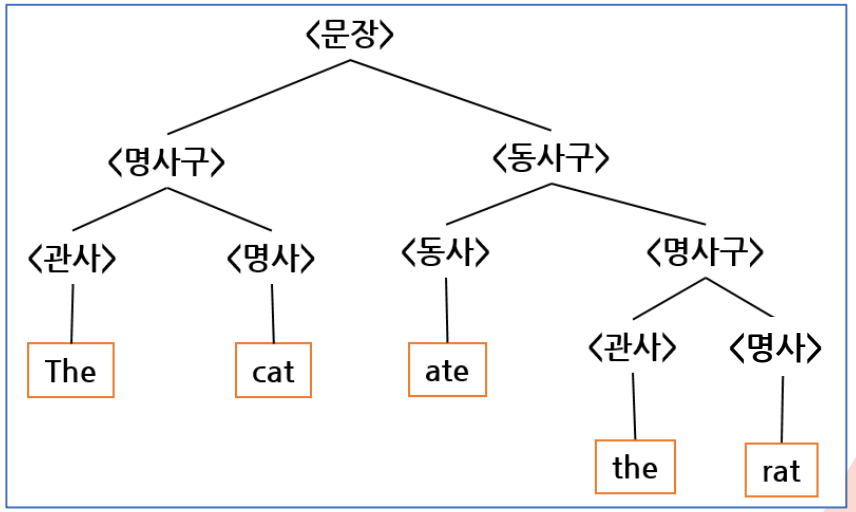


비단말 심벌

- 비단말 심벌은 단말 심벌이 아니면서 복합적으로 나열된 단말 심벌과 비단말 심벌의 조합으로 구성됨
- 파스 트리의 내부 노드에 해당됨
- 자연어에서는 〈명사구〉, 〈동사구〉, 〈문장〉 등이 비단말 심벌임

형식문법의 요소

프로그래밍 언어의 파스 트리



생성 규칙

- 하나의 비단말 심벌이 다른 단말 심벌이나 비단말 심벌로 대체되는 규칙
- 예를 들어 영어라면

〈문장〉 → 〈명사구〉 + 〈동사구〉,

〈명사구〉 → 〈관사〉 + 〈명사〉,

〈명사〉 → “dog” | “ cat” | “ rat”,

〈동사〉 → “bark” | “ chase” | “ eat” 등이 생성 규칙임

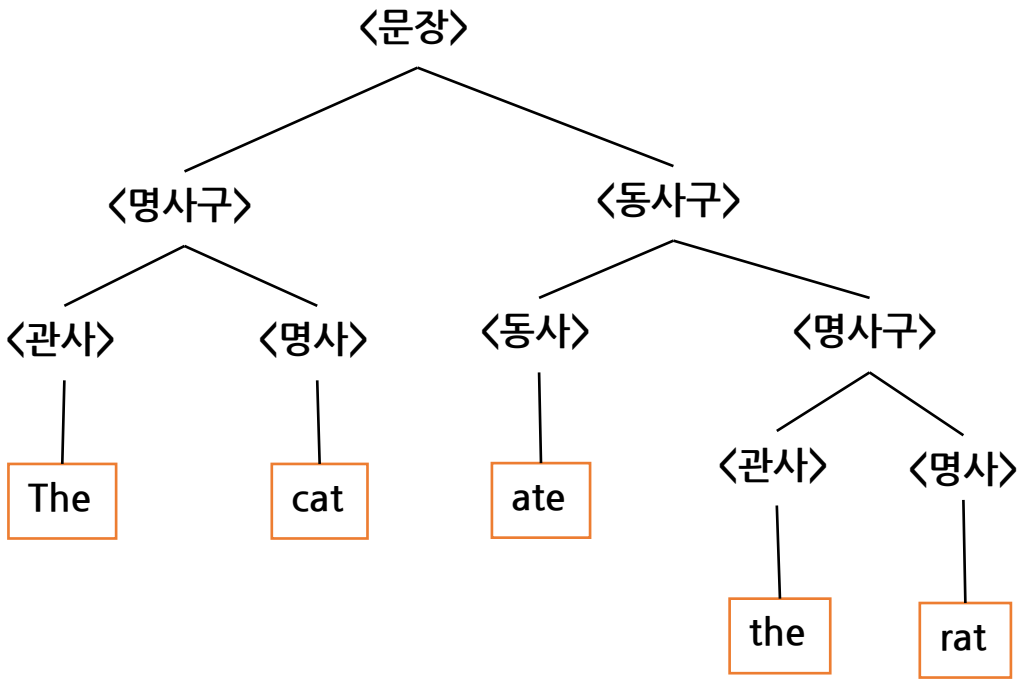
형식문법의 요소

시작 심벌

- 가장 상위 계층의 비단말 심벌로 보통 〈문장〉이 시작 심벌임
- 파스 트리의 루트 노드에 해당됨

문장의 구조와 간단한 문법을 사용한 구문 분석

프로그래밍 언어의 파스 트리



〈문장〉 → 〈명사구〉 〈동사구〉

〈동사구〉 → 〈동사〉 〈명사구〉 | 〈동사〉

〈명사구〉 → 〈관사〉 〈명사〉 | 〈명사〉

〈관사〉 → “a” | “the”

〈동사〉 → “eat” | “ate” | “see” | “sees” | “saw”

〈명사〉 → “cat” | “rat” | “dog”

03

프로그래밍 언어의 분석

프로그래밍에서 실행 가능 코드

실행 가능 코드

- 프로그래밍 언어로 작성된 프로그램 코드는
  사람이 읽고 이해하기가 쉽지만,
  그 자체로는 컴퓨터가 이해할 수 없으며 실행할 수가 없음

- 사람이 작성한 프로그램을 분석하여
  기계어의 이진 코드로 바꾸는 변환 과정을 거치면
  컴퓨터가 이해하고 실행할 수 있는 프로그램이 됨

프로그램 코드의 분석

프로그램 코드의 분석

- 사람이 작성한 프로그램은

  어휘 분석과 구문 분석을 통해 프로그램에 문제가 없음을 확인함

- 어휘 분석과 구문 분석 과정을 통과한 후 최종적으로 코드 생성 단계에서

  실제 실행 가능한 이진 기계어 코드가 생성됨

  : 어휘 분석 → 구문 분석 → 코드 생성

프로그램 코드의 분석

어휘 분석

- 프로그램을 구성하는 문자들의 나열로부터 단어(토큰)를 추출해 내는 과정임
- 빈 칸을 기준으로 단어를 구분하고, 구분안의 각 단어를 이름, 숫자, 수식 기호 등으로 분류하는 것이 어휘 분석임
- 어휘 분석의 결과로 나온 개개의 단어를 토큰이라고 함

프로그램 코드의 분석

구문 분석

- 어휘 분석의 결과로 나온 토큰들의 나열이

  해당 프로그래밍 언어의 문법에 맞는지를 확인하는 과정

- 언어문법의 적합성 확인을 위한 파스 트리의 생성

프로그램 코드의 분석

실행코드의 생성

- 프로그래밍 언어에 대한 구문 분석의 결과로

  변수, 상수, 제어의 흐름 등이 결정되면

  이러한 각각의 명령어를 어셈블리어로 풀어 쓰거나 직접 기계어

  이진 코드가 생성됨

정리 문제

고급언어 프로그램에서 실행 코드까지의 변환 순서로 올바른 것은?

- 구문 분석, 코드 생성, 어휘 분석
- 코드 생성, 구문 분석, 어휘 분석
- 어휘 분석, 구문 분석, 코드 생성
- 어휘 분석, 코드 생성, 구문 분석

04

프로그래밍 언어의 공통 개념

대입문

개요

- 대입문(할당문)은 변수나 기억장치 주소에 값을 저장하는 역할을 함
- 대입문은 명령형 언어의 가장 주요한 기능 중의 하나임
- 대입문은 보통 다음과 같은 일반적인 형태를 가짐

〈수식1〉 = 〈수식2〉

대입문

개요

〈수식1〉 = 〈수식2〉

x = x + 1

- 〈수식1〉은 '왼쪽 값'(l-value)이라고 하고
  값이 저장될 위치(기억장치의 주소)를 가리킴

- 〈수식2〉는 '오른쪽 값'(r-value)이라고 하고
  〈수식 1〉이 가리키고 있는 곳(기억장치의 주소)에 저장될 값
  (정수값, 실수값, 문자열 등)을 의미함

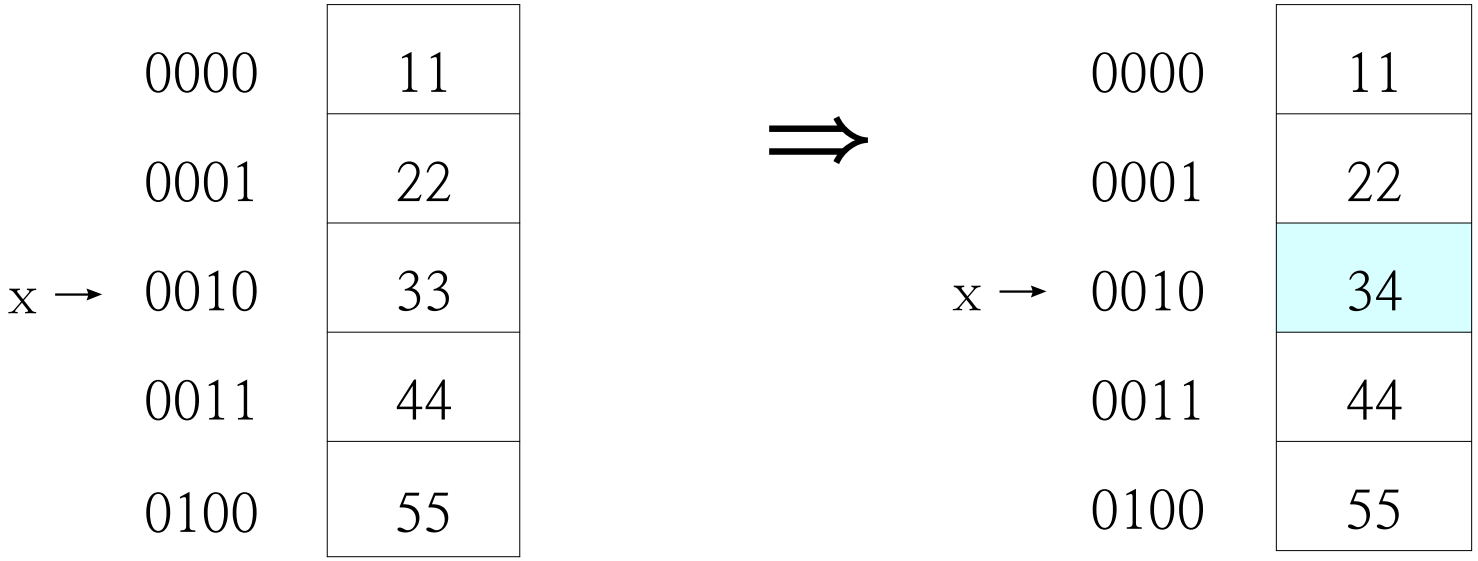
대입문

개요

〈수식1〉 = 〈수식2〉

- 대입문은 〈수식 2〉의 '값'을 〈수식 1〉의 '주소가 가리키는 기억장치의 저장 장소'에 '저장' 함

x = x + 1

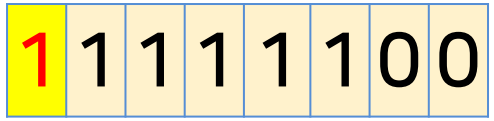


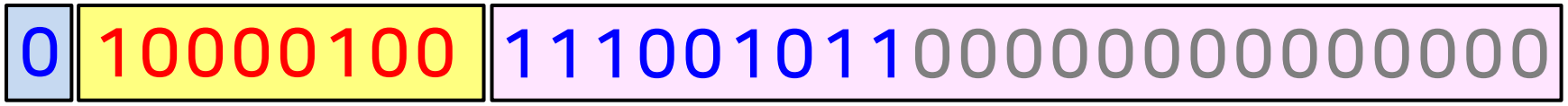
변수형 검사

개요

- 변수형은 연산에 사용되는 상수(constant)나 변수(variable)의 종류를 지정해서 연산 수행 시에 호환성이 없는 변수형간의 연산을 막아서 연산의 결과로 얻게 되는 정보의 손실을 최소화하기 위해 사용됨

  예) 정수를 문자열로 나누거나 복잡한 구조체(struct) 타입으로 곱하는 것은 연산의 의미가 없음





변수형 검사

개요

- 정수와 실수의 덧셈에서 정수형으로 연산을 수행하면 실수값의 일부분을 잃게 됨

  예) 3 + 2.5 ⇒ ?

- 변수형이 호환되지 않는 연산을 찾아내는 것을 형 검사라고 함

변수형 검사

개요

- 형 검사는 컴파일 과정에서 이루어지는 정적(static) 형 검사 방식과 프로그램의 실행(run-time) 중에 이루어지는 동적(dynamic) 형 검사 방식이 있음

변수형 검사

개요

- 프로그래밍 언어에 따라 변수형을 다루는 방법과 변수형을 제한하는 정도에도 많은 차이가 있음

예) 숫자 10과 문자열 '33'을 더하는 것을 허용하는 프로그래밍 언어도 있고 허용하지 않는 프로그래밍 언어(pearl)도 있음

: x = 10 + '33';

print x;

⇒ 출력값 : 43

변수형 검사

C

- int x = 10 + “30”;
  printf(“%d”,x);

- 결과는 컴파일러에 따라 다를 수 있음
- gcc 컴파일러의 경우 출력값 : 134513854

변수형 검사

C 언어

- 컴파일 시 경고가 발생함 :

  warning: initialization makes integer from pointer without a cast

- C 와 같이, 모든 변수형을 변수의 선언에서 지정해주고

  모든 연산에서 변수형을 검사하는 언어에서도

  특정 경우에 다른 변수형 간의 연산이 필요할 때가 있음

변수형 검사

C 언어

- 올바른 연산을 위해 한 변수나 데이터의 형을 수동으로 전환시켜주는 것이 형 변환(type casting)임

- 컴파일러 경고 중에 “cast”란 단어가 바로 이 캐스팅을 의미함

정리하기

1. 프로그래밍 언어의 파스 트리

- 단말 심벌 : 문장을 이루는 단어들
- 비단말 심벌 : 단말 심벌이 아니면서 복합적으로 나열된 단말 심벌과 비단말 심벌의 조합

2. 프로그램에서 실행 가능 코드로의 변환

- 어휘 분석 : 프로그램을 구성하는 문자들의 나열로부터 단어를 추출해 내는 과정
- 구문 분석 : 어휘 분석의 결과로 나온 토큰들의 나열이 해당 프로그램이 언어의 문법에 맞는지를 확인하는 과정
- 코드 생성 : 구문 분석의 결과로 변수, 상수, 제어의 흐름 등이 결정되면 이러한 각각의 명령어를 어셈블리어로 풀어쓰거나 직접 기계어 이진 코드가 생성

3. 프로그래밍 언어의 기본 공통 개념

- 대입문(할당문) : 변수나 기억장치 주소에 값을 저장하는 역할

12 강

다음시간 안내

프로그래밍 언어(2)