6장 데이터 타입 (1)

데이터 타입

데이터 타입이란 값들의 모임과 이러한 값들에 대한 미리 정의된 연산들의 집합으로 정의된다.

타입 시스템

타입 시스템이란?

언어의 각 식에 대해서 타입이 어떻게 연관되는지를 정의하고, 타입 호환성과 타입동등을 위한 규칙을 포함

타입 시스템의 용도

- 오류 탐지 (타입 검사)
- 프로그래미 모듈화 지원 (모듈간 인터페이스 검사)
- 프로그램 문서화 (타입 선언)

서술자

서술자는 변수의 속성들의 모임으로 타입 검사와 할당/회수에 활용한다.

기본 데이터 타입

기본 데이터 타입이란?

다른 데이터 타입의 관점에서 정의되지 않는 타입

기본 데이터 타입 종류

- 수치 타입
  - 정수
  - 부동 소수점
  - 십진수
  - 복소수
- 불리안 타입
- 문자 타입
  - 문자 스트링 타입

문자 스트링 타입

문자 스트립 타입이란

문자 스트링 타입은 값들이 일련의 문자들로 구성

스트링 타입 지원

- 작성력 향상
- 배열보다는 기본 타입으로 다루는 것이 자연스럽다.
- 오늘날 언어에서 기본 스트링 타입을 지원하지 않는 것은 정당화하기 어렵다.

문자 스트링 타입 연산

- 배정
- 비교 (=,> 등)
- 접합
- 부분 스트링 참조
- 패턴 매칭

언어에 따른 문자 스트링 타입

- C
  - 기본 타입이 아닌 char 배열로 제공
- C++
  - char 배열
  - string 클래스
- Java
  - string 클래스
- Python
  - string 기본 타입 지원

- JavaScript
  - string 기본 타입 지원

스트링 길이 선택 사항

- 정적 길이 스트링
  - 스트링 생성 시 그 길이가 설정되고 고정
  - Python 의 String, Java 의 String
- 제한된 동적 길이 스트링
  - 스트링 선언 시 고정된 최대 길이까지의 가변적인 길이를 갖는 것을 허용
  - C
- 동적 길이 스트링
  - 최대 길이 제한 없이 가변 길이를 갖는 것을 허용
  - Perl, JavaScript

스트링 타입 구현

- 정적 길이 스트링
  - 컴파일러 시간 서술자
- 제한된 동적 길이 스트링
  - 현재 길이에 대한 실행시간 서술자 필요
- 동적 길이 스트링
  - 실행시간 서술자 필요
  - 할당/회수에 따른 기억공간 관리 필요

기억공간 관리 3 가지

- 연결리스트에 스트링 저장
- 스트링을 힙에 할당되 개개의 문자들을 가르키는 포인터들의 배열로써 저장
- 스트링 전체를 인접한 기억공간 셀들에 저장

사용자 정의 순서 타입

순서 타입은 가능한 값들의 범위가 양의 정수 집합과 연관 가능한 타입

Tip) Java 의 기본 타입 중에서 순서 타입은?

- 정수
- 문자
- Boolean

사용자 정의 순서 타입 종류

- 열거 타입
- 부분 범위 타입

열거 타입

열거 타입은 모든 가능한 값들이 그 정의에서 제공되는, 즉 나열되는 타입

열거 타입 예시

- enum

열거 타입 설계 고려사항 (C++ 기준)

- (X) 한 열거 상수가 한 개 이상의 타입 정의에 올 수 있는가? 또한 그렇다면 상수 참조에 대한 타입 검사가 이루어 지는가?
- (O) 열거 타입이 정수로 강제 변환이 가능한가?
- (X) 다른 타입이 열거 타입으로 강제 변환이 가능한가?

```c
#include <stdio.h>

enum color{red,blue,green,yellow,black};

//열거자 red,blue 의 재정의 불가
//enum color2{red,blue}

int main(void) {
    enum color myColor = blue, yourColor = red;

    myColor++; //가능

    myColor = 4; //가능

    return 0;
}
```

부분 범위 타입

부분 범위 타입이란 갑 범위가 순서 타입의 연속된 부분적 순서 로 정의되는 타입

부분 범위 타입 예시

- subtype

부분 범위 타입 평가

- 판독성 향상 : 특정 범위의 값만을 저장할 수 있다.
- 신뢰성 향상 : 명세된 범위를 벗어나면 오류로 탐지
- 최근 언어에선 지원되지 않는 경향

6장 데이터 타입 (2)

배열 타입

배열이란

배열은 동질적인(동일한 타입을 갖는) 데이터 원소들의 집단체(Aggreate)이고, 개개의 원소는 그 집단체의 첫번째 원소에 상대적인 위치로 식별

배열 설계시 고려사항

- 첨자 범위 언제 바인딩되는가
    - 동적
    - 정적
- 배열 할당은 언제 일어나는가
    - 동적
    - 정적

첨자 바인딩 및 배열 유형

- 배열 변수의 첨자 타입의 바인딩은 보통 정적이지만, 그 첨자 값 범위는 때때로 동적으로 바인딩
- 첨자 범위 바인딩 시기, 기억공간 바인딩 시기 및 그 할당 위치에 따라 배열을 5가지로 유형화
    - 정적
    - 고정 스택-동적
    - 스택-동적
    - 고정 힙-동적
    - 힙-동적

정적 배열

- 첨자 범위가 정적으로 바인딩
- 기억 공간 할당이 정적
- 장점 : 효율적
- 예시 : c 의 static 배열

고정 스택-동적 배열

- 첨자 범위가 정적으로 바인딩
- 기억 공간 할당은 동적 (선언문 세련화 시간에 할당)
- 장점 : 기억공간의 효율적
- 예시 : c 의 함수 내부에 선언된 배열

스택-동적 배열

- 첨자 범위가 동적으로 바인딩 (세련화 시간)
- 기억 공간 할당이 동적 (세련화 시간)
- 변수 존속기간 동안 고정
- 장점 : 유연성
- 예시 : Ada 의 배열

고정 힙-동적 배열

- 첨자 범위가 동적으로 바인딩 (사용자 프로그램 요청시)
- 기억공간 할당이 동적 (사용자 프로그램 요청시)
- 힙으로부터 할당
- 바인딩 이후 고정
- 장점 : 유연성
- 단점 : 할당 시간이 길다
- 예시 : c 의 동적 배열
- 예시 : Java 의 배열

힙-동적 배열

- 첨자 범위가 동적으로 바인딩
- 기억공간 할당이 동적
- 힙으로부터 할당
- 바인딩 이후 변경 가능
- 장점 : 유연성
- 단점 : 할당 및 회수 부담
- 예시 : Java 의 ArrayList
- 예시 : JavaScript 배열

배열 연산

배열 연산은 배열 단위의 연산을 수행

공통 배열 연산

- 배정
- 접합
- 동등/비동등 비교
- 슬라이스

배열의 생김새

- 직사각형 배열
  - 모든 행이 동일한 개수의 원소
  - 모든 열이 동일한 개수의 원소
  - 즉 정확히 직사각형 테이블을 모델링
- 톱니형 배열
  - 행들의 크기가 동일할 필요가 없는 다차원 배열 (행우선 배열)
  - 다차원 배열에서, 배열의 원소가 배열인 경우

슬라이스

슬라이스는 배열의 부분적 구조로 배열 일부분에 대한 참조 메커니즘을 구현

배열 구현

1차원 배열의 원소 접근 함수

주소(list[k]) = 주소(list[하한])+(k-하한)*원소 크기

다차원 배열의 저장 순서

- 행-우선 순서 (Row Majgor Order) -> 대부분의 언어
- 열-우선 순서 (Column Major Order) -> Fortran

다차원 배열에서 접근 함수

주소(a[i,j]) = 주소(a[0,0]) + ((i번째 행보다 앞선 행의 개수)*(열의 개수)) + (j번째 열의 왼쪽에 위치한 원소 개수)*원소_크기

연상 배열

연상 배열은 원소들간에 순서가 없으며, 키 값을 통해서 원소에 접근하는 배열

레코드 타입

레코드 타입은 이질적 가능한 데이터 원소들의 집단체

레코드 타입 예시와 반대 개념

- 예시 : Struct
- 반대 개념 : 배열

레코드 필드 참조 방식

- 완전 자격 참조
  - 가장 큰 포괄적인 레코드부터 특정 필드에 이르기까지 모든 중간 레코드 이름들이 그 참조에 포함
- 생략 참조
  - 레코드 이름 일부가 생략되는 것을 허용

레코드와 배열 차이

- 배열 원소 접근 -> 동적
- 레코드 필드 접근 -> 정적

6장 데이터 타입 (3)

튜플 타입

튜플은 레코드와 유사하나 원소들이 명명되지 않는 데이터 타입

리스트 타입

- Python 의 배열로서도 역할
- 리스트는 변경 가능
- 튜플처럼 사용 가능 : 인덱싱,접합,슬라이싱
- 리스트 함축 (List Comphrehension) 지원 : 집합 표기법에 기반

공용체 타입

공용체란

공용체 (Union) 은 그 변수가 프로그램 실행 중에 다른 시기에 다른 타입의 값이 저장할 수 있는 타입

공용체 유형

- 자유 공용체 (Free Union)
  - 타입 검사를 지원하지 않음
  - 안전하지 않음
- 판별 공용체 (Discriminated Union)
  - 타입 검사 지원
  - 판별자 또는 태그라는 타입 지시자를 포함
  - 안전

6장 데이터 타입 (4)

포인터와 참조 타입

포인터 타입

포인터 타입은 메모리 주소와 특정 값, nil 로 구성

포인터 타입의 용도

- 간접 주소 지정

- 동적 메모리 관리
  - 포인터를 사용하여 동적으로 할당되는 기억공간 접근
  - 동적 할당되는 기억공간을 힙 이라 한다.
  - 힙으로부터 할당되는 변수는 힙-동적 변수
  - 이러한 변수는 이름이 없는 무명 변수

포인터는 타입 연산자를 사용하여 정의

- 포인터는 참조 타입
- 스칼라 변수는 값 타입

포인터 연산

- 배정
  - 배정은 포인터 변수에 어떤 유용한 주소로 설정
- 역참조 (Dereferencing)
  - 포인터가 가리키는 메모리 위치에 저장되어 있는 값을 참조

포인터의 문제

- 힙-동적 변수와 다른 변수들도 참조 가능
- 유연성이 높지만 프로그래밍 오류 초래 가능성을 높임
- 포인터의 두가지 문제 : 허상 포인터와 분실된 힙-동적 변수
- 최근의 언어는 포인터를 참조 타입으로 대체하는 경향
- 허상 포인터(Dangling Pointer) 또는 허상 참조(Dangling Reference)는 이미 회수된 힙-동적 변수를 여전히 카리키는 포인터
  - 허상 포인터의 원인은 힙 동적 변수의 명시적 회수
- 분실된 힙-동적 변수(Lost Heap-Dyanamic Variables)는 사용자 프로그램에서 더 이상 불가능한 할당된 힙-동적 변수
  - 힙-동적 변수 분실을 메모리 누수(Memory Leakage)라 합니다.

포인터 예시

- c/c++
  - 제한 없는 포인터 허용으로 극도의 유연성 제공 -> 주의 깊게 사용 필요
  - 포인터는 함수를 가리킬 수 있다.

참조 타입 (Reference Type)

포인터와 유사하나 포인터는 주소를 참조하는 것에 반해, 참조 변수는 객체나 메모리의 값을 참조하는 근본적인 차이

- 참조타입은 제한된 연산을 갖는 포인터

포인터 / 참조 타입 평가

- 포인터는 장치 드라이버 작성에 필수적 (특정 절대 주소 접근 필요)
- 참조 변수는 위험성 없이 포인터의 어느 정도의 유연성과 능력 제공
- 포인터나 참조 변수는 동적 자료 구조 구성에 필수

6장 데이터 타입 (5)

타입 검사

- 타입 검사는 연산자에 대한 피연산자들의 타입이 호환가능하지를 확인하는 행위
- 호환가능 타입는 연산자에 대해서 적법하거나, 또는 적법한 타입으로 컴파일러에 의해서 묵시적으로 변환하는 것이 언어 규칙에서 허용되는 타입
- 타입의 묵시적 자동변환을 타입 강제 변환이라 한다.
- 타입 오류는 연산자를 부적절한 타입의 피연산자를 적용할 때 발생

강 타입

프로그래밍 언어에서 강타입은 타입 오류가 항상 탐지되는 것을 의미한다. * 실제로는 존재 하지 않는다. * 타입 강제 변환은 강 타입을 약화 시킴

타입 동등

두 변수의 타입이 동등하면 한 변수가 다른 변수에 할당될 수 있다.

타입 동등을 정의하는 2가지 접근 방법

- 이름 타입 동등
- 구조 타입 동등

이름 타입 동등

아래 기준 중 하나를 만족하면 이름 타입 동등 * 두 변수가 동일한 선언문에 속함 * 같은 타입 이름을 선언

이름 타입 동등 장/단점

- 장점 : 구현하기 쉬움
- 단점 : 매우 제약적

구조 타입 동등

*두 변수 타입이 동일한 구조를 가지면 구조 타입 동등 이라 한다.

구조 타입 동등 장/단점

- 장점 : 매우 유연
- 단점 : 구현하기 어려움

타입 동등 예시

- C
    - 이름 타입 동등 적용
    - struct / union
    - enum
    - 구조 타입 동등
    - 비 스칼라 타입