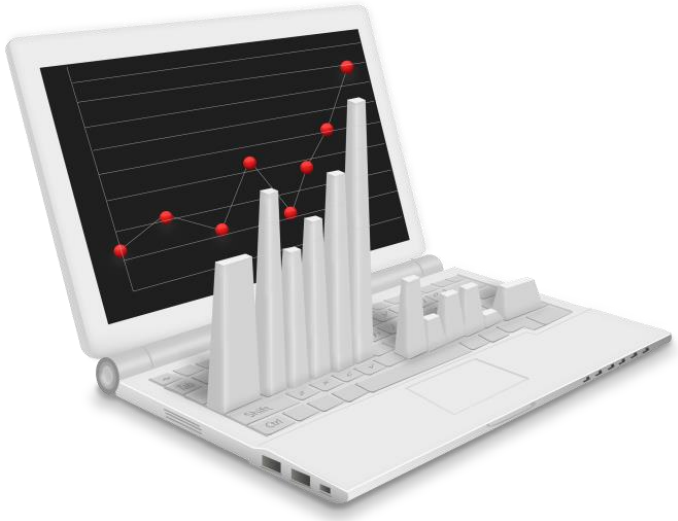
프로그래밍 언어론

포인터의 문제점

컴퓨터공학과

조은선



충남대학교
CHUNGNAM NATIONAL UNIVERSITY

포인터의 문제점

학습목표

- 포인터의 문제점, 유사 개념 등에 대해 알아본다.

학습내용

- 포인터의 문제점과 해결 방안
- 기타 언어의 포인터 유사 개념

목 차

- **들어가기**
- **학습하기**
  - 포인터의 문제
  - Dangling 포인터 문제 해결 노력
  - 분실된 Heap-dynamic 변수 문제 해결
  - Reference (참조) 타입
  - 기타 언어의 포인터
- **평가하기**
- **정리하기**

알고가기

다음 8번 행의 i에는 무슨 값이 assign 될 까?

```
1: int i ;
2: int * p;
3: int * q;
4: p = (int *) malloc (sizeof(int));
5: *p = 3;
6: q = p;
7: free p;
       : …
8: i = *q;
```

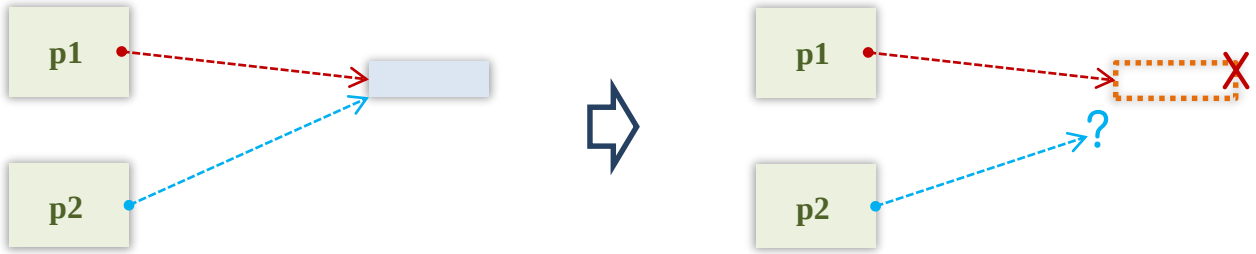
① 3
② p와 동일한 값
③ q와 동일한 값
④ 알 수 없음

포인터의 문제

Dangling(허상) pointer, dangling reference (허상 참조)

- 이미 회수된 heap-dynamic 변수의 주소를 포함하고 있는 포인터
- 이 포인터가 가리키고 있는 위치에 새로운 heap-dynamic 변수가 할당된다면?
  - 더 큰 문제.
- **Danling 포인터의 생성 :**

1. 포인터 p1이 새로운 heap-dynamic 변수를 가리키도록 설정한다.
2. 포인터 p2에 p1의 값을 복사한다.
3. p1이 가리키는 heap-dynamic 변수를 명시적으로 회수

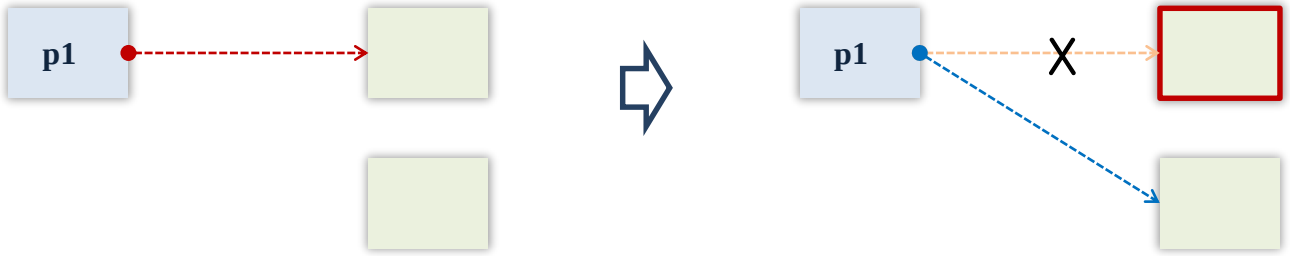


포인터의 문제

분실된 (lost) heap-dynamic 변수

- **garbage(쓰레기)** : 사용자 프로그램에서 더 이상 접근될 수 없는 할당된 heap-dynamic 변수
  - 이러한 heap-dynamic 변수는 분실됨
- heap-dynamic 변수의 명시적 회수가 요구될 경우에 발생: **메모리 누수(memory leakage) 초래**
- **분실된 heap-dynamic 변수의 생성**

1. 포인터 p1이 새롭게 생성된 heap-dynamic 변수를 가리키도록 설정
2. 나중에 p1이 다른 heap-dynamic 변수를 가리키도록 설정

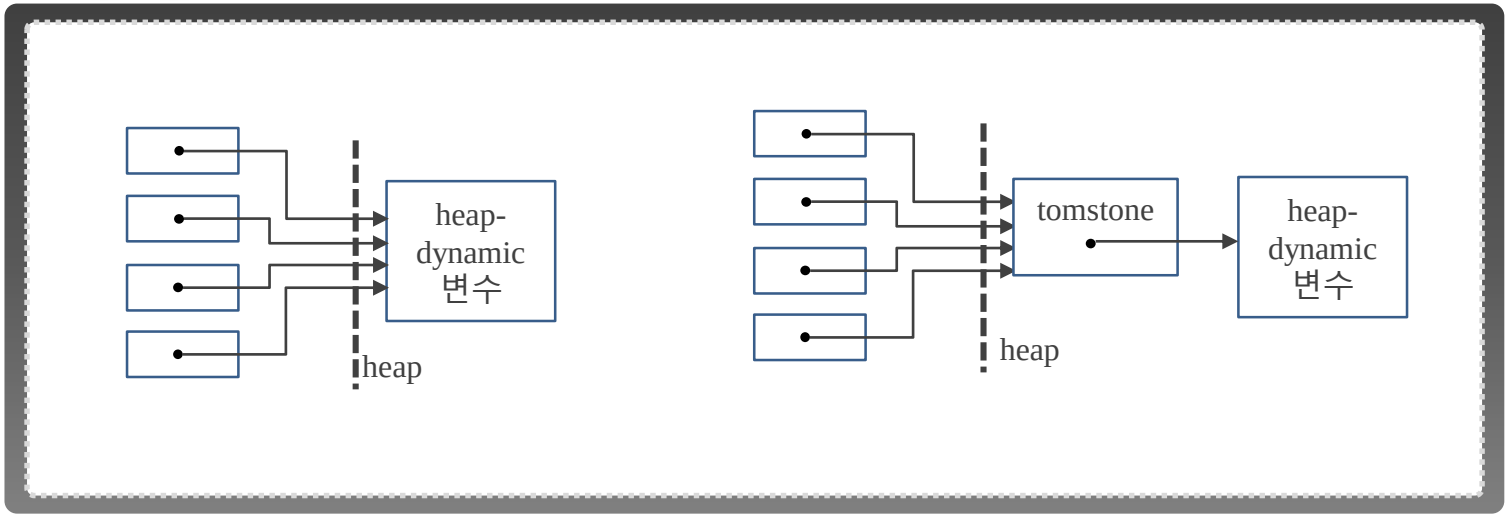


포인터의 문제

Dangling 포인터 문제 해결노력

비석 (tombstone) 방법

- **비석(tombstone)** : heap-dynamic 변수를 가리키는 메모리 셀
- 실제 포인터는 직접 heap-dynamic 변수를 가리키지 않고, 비석을 가리키도록 구현
- heap-dynamic 변수가 반환되면 비석 값은 nil로 바뀜 (없어지지는 않음)
- **Dangling 포인터 해결**

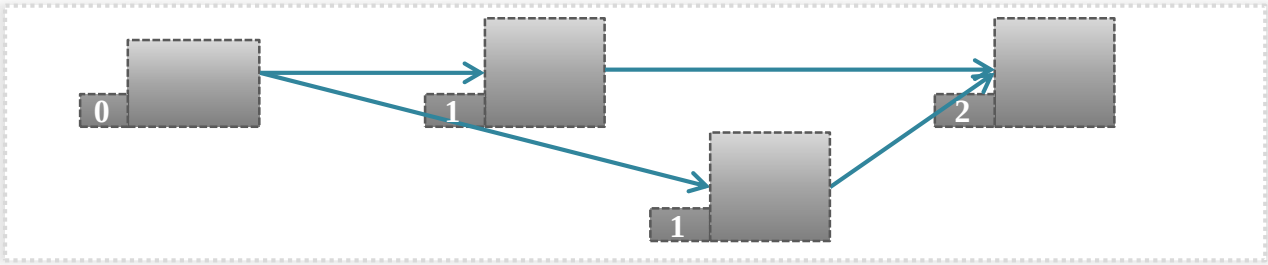


포인터의 문제

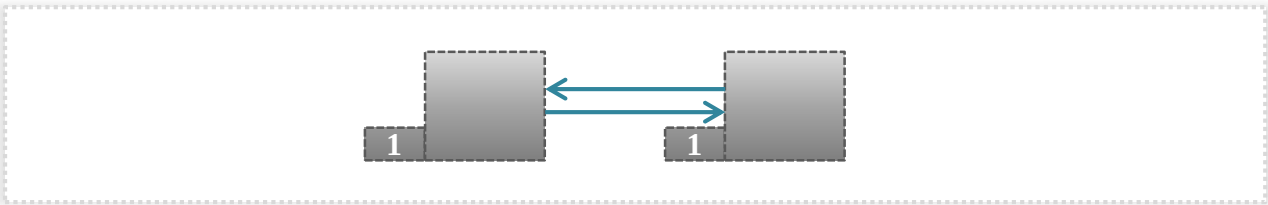
분실된 heap-dynamic 변수 문제 해결 노력

Regerence Count (참조 계수)

+ 각 셀이 자신을 참조하는 포인터 개수를 기록
+ 0이 되면 제거
+ memory leak에 대해 reference 즉시 제거 조치를 취하는 것임 **(eager approach)**



+ 단점: 추가 셀/시간 필요, 순환 구조에서 복잡

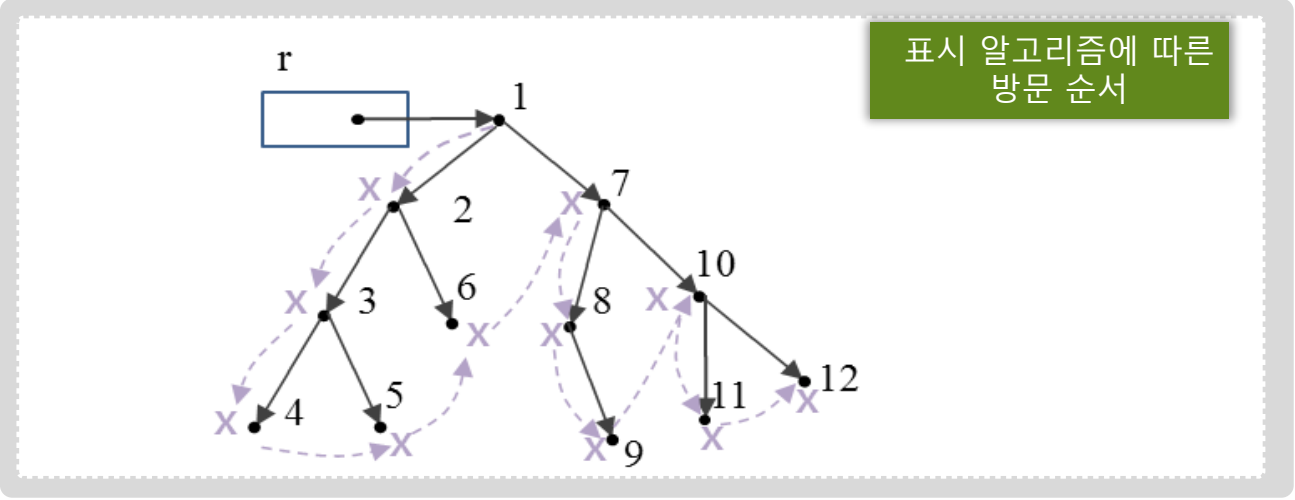


포인터의 문제

분실된 heap-dynamic 변수 문제 해결 노력

Garbage collection(쓰레기 수집)

- **지연된 조치 (lazy approch)** : 메모리가 모자라면 시작
- 각 셀마다 1bit씩 더 필요 (garbage 또는 non-garbage 표현)
  1) 처음에는 모두 garbage로 초기화하고,
  2) heap을 순회하며 도달가능한 셀을 non-garbage 표시로 바꿈
  3) 남은 셀은 모두 garbage 므로 회수



Reference 타입 : 포인터 유사 개념

Reference(참조) 타입

- 포인터와 비슷한 메모리 주소 접근 기능으로서
- 판독성/안전성 등의 목적으로 포인터의 성격에 제약을 가한 것

➡ **임의의 주소 접근 불가, 산술 연산 불가, dangling reference 없음**

C++ 의 reference

- **항상 묵시적으로 dereference 되는 상수 포인터**
  - 정의시에 어떤 변수의 주소값으로 초기화되고,
  - 초기화된 후에 다른 변수를 참조하도록 변경될 수 없음

```
int result = 0;
int & ref_result = result;
…
ref_result = 10;
```

```
swap(int& i, int& j) {
        int t = i;
        i = j; j = t;
}
…;
swap(r,s);
```

Reference 타입 예 (Java, C#)

Java

- C의 포인터 변수를 제거, reference 로 대체
- C 포인터 변수와의 비교
  - + 포인터 변수는 메모리 주소 참조, reference 타입 변수는 객체 단위 참조
  - + reference 변수에 대한 산술 연산 불허
  - + 다른 객체를 참조하도록 배정될 수 있다.
  - + 객체는 묵시적으로 회수 (danling reference 발생 안됨)

```
String s;     //s는 null로 설정
…
s = "Hello, Java";  // s는 스트링 객체 참조
```

C#은 Java의 reference 개념과 C의 포인터를 모두 제공

- C/C++ 코드와의 호환성 제공
- 단, 포인터 사용시 `unsafe` 사용이 요구됨

Reference 타입 예 (Fortran90)

FORTRAN 90 포인터

- heap과 stack을 모두 가리킬 수 있는 것은 C와 비슷
- dereferencing 는 암묵적으로 실행
- TARGET 속성이 있다고 선언된 변수만을 포인트 가능

기타 언어의 포인터

Pascal

- 동적인 메모리 관리만을 위한 참조 제공
  - 임의의 주소를 접근할 수는 없음
- reference 뒤에 ^를 붙여 값 접근 (dereference)
- dangling reference 발생 가능 (dispose)

평가하기

마지막으로 내가 얼마나 이해했는지를 한번 확인해 볼까요?

총 2문제가 있습니다.

START

평가하기 1

1. 이미 회수된 heap-dynamic 변수의 주소를 가지는 포인터가 있다. 이에 대한 설명으로 맞는 것은?

① 분실된 heap-dynamic 변수 라고 한다.

② 메모리 누수(leak)가 초래된다.

③ 사용자 프로그램에서 더 이상 접근될 수 없지만 해당 프로그램에 할당된 heap-dynamic 변수가 발생 된다.

④ 포인터대신 reference (참조) 를 사용하면 발생하지 않는다.

확인

평가하기 2

2. 다음 중 reference(참조) 타입과 관련된 설명으로 가장 가까운 것은?

① reference 타입 변수는 포인터보다 안전하지 못하다.

② reference 타입 변수는 임의의 주소를 가질 수 있다.

③ reference 타입 변수에 대한 산술 연산이 불가능하다.

④ reference 타입을 사용하더라도 dangling reference 가 발생할 수 가 있다.

(.

확인

정리하기

포인터의 문제 및 해결 노력

> 포인터에서 발생할 수 있는 문제는 dangling pointer, lost heap dynamic variables 등이 있고, 해결 방안으로는 비석 (tombstone)방법, reference count 사용, garbage collection 등이 있다.

> Reference 타입 변수는 포인터와 유사하며 할수 있는 기능은 덜하지만 좀 더 안전한 방법으로서, Java, C++, C#, Fortran90 등에서 사용된다.