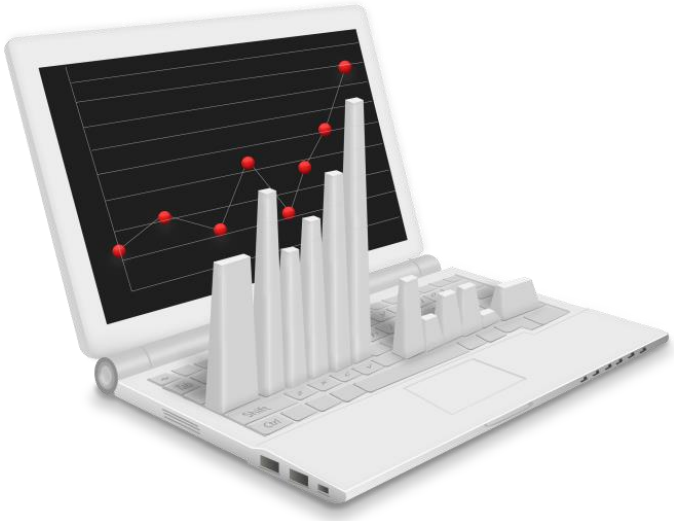
프로그래밍 언어론

포 인 터 개요

컴퓨터공학과

조은선



충남대학교
CHUNGNAM NATIONAL UNIVERSITY

포인터 개요

학습목표

- 포인터의 필요성, 종류 등에 대해 알아본다.

학습내용

- 포인터의 개념 및 종류
- C/C++ 의 포인터

목 차

- **들어가기**
- **학습하기**
  - 포인터의 개념
  - 포인터 사용
  - 포인터
  - C와 C++의 포인터
- **평가하기**
- **정리하기**

알고가기

번호 3 번행부터 시작해서 차례로 Letter 필드의 글자를 읽고, 같은 행의 Next 필드가 지시하는 다음 행을 찾아가 다시 Letter 필드의 글자를 읽기를 반복하시오. 방문한 모든 글자를 순서대로 연결하여 단어를 만들면 무엇인가? 단, Next 필드에 '*' 가 나오면 읽기를 멈춘다.

Number	Letter	Next
1	'M'	9
2	'E'	7
시작 → 3	'C'	5
4	'U'	8
5	'O'	1
6	'T'	2
7	'R'	*
8	'A'	1
9	'P'	10
10	'U'	6

① COMPUTE

② COMPUTER

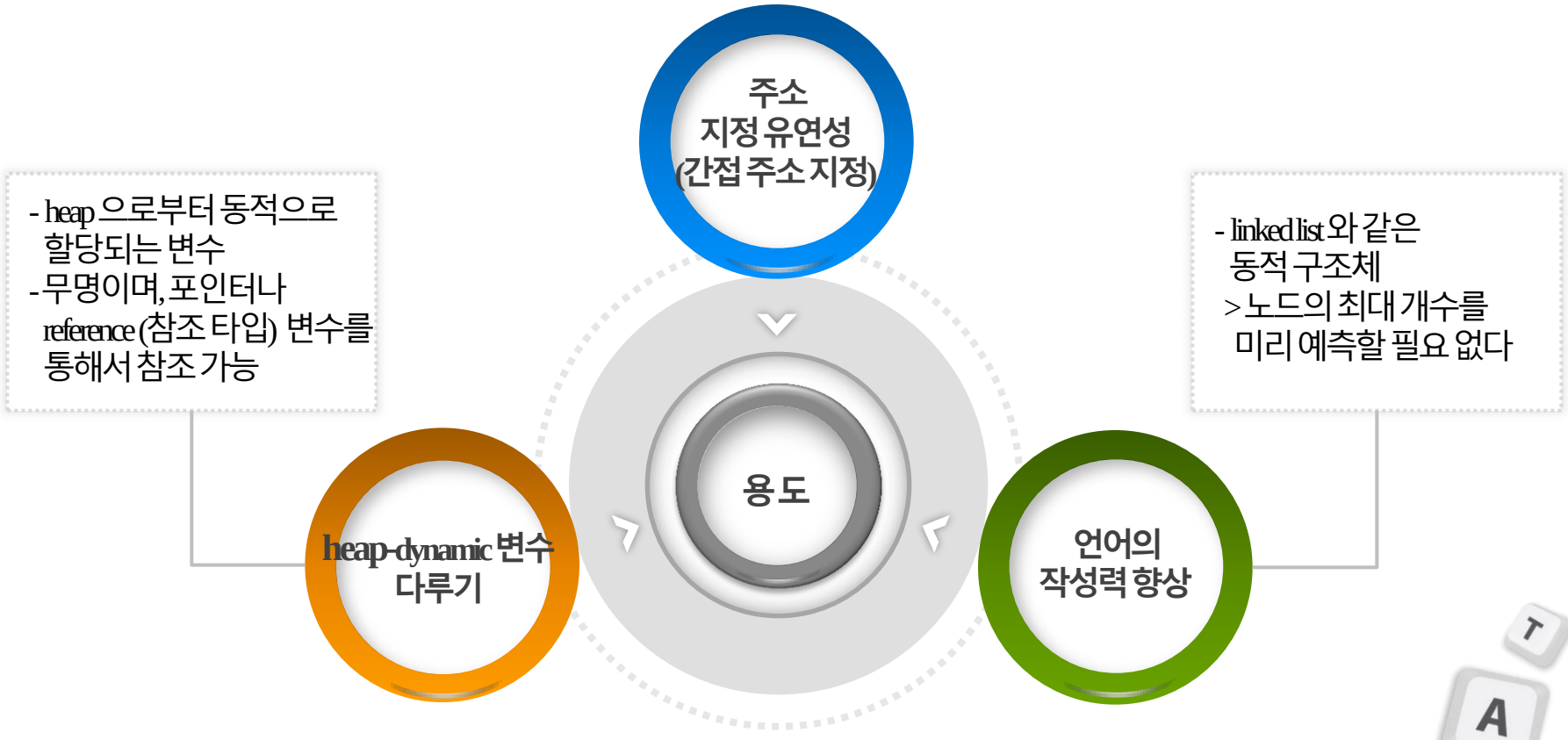
③ COMPUTING

④ COMP

Ⅰ 포인터의 개념 및 종류

포인터 (Pointers)의 개념

포인터 메모리 주소와 특수 값 nil로 구성되는 값의 범위를 갖는 타입



Ⅰ 포인터의 개념 및 종류

포인터 (Pointers)에 대한 평가와 논점

평가 : "일종의 필요악"

- 프로그램은 복잡해짐
- 메모리 접근 유연성은 향상

논점

- 포인터 변수의 영역과 존속기간?
- Heap-dynamic (힙-동적) 변수의 존속기간은?
- 포인터가 가리킬 수 있는 값의 타입에 제한이 있는가?
- 포인터의 용도는? (동적 메모리 관리 또는 간접 주소지정)
- 언어가 포인터 타입, reference 타입 혹은 둘 다를 지원하는가?

포인터의 개념 및 종류

포인터 사용

포인터 변수는 값으로 메모리 주소를 가짐

- 메모리 주소를 값처럼 얻어내는 방법
  - 일반변수 앞에 '&'붙이기, 또는 `malloc /new` 등으로 가져옴

일반변수 주소

Heap-dynamic 변수 주소

Dereferencing(역참조하기)

- 포인터 변수가 내용으로 가지고 있는 주소를 취하여 그 주소에 저장된 값을 가져오기
- *를 사용

Heap 관리를 위한 연산 제공

```
int *ptr, j; // in C

…..

ptr = (int *) malloc(sizeof(int));

j = *ptr;
```

포인터의 개념 및 종류

포인터 연산

- 포인터 변수가 구조체를 가리킬 때, 그 필드 참조는?

```
struct student {        char * name;
                        int score;
                        char grade;
                    };
…
struct student *s;
….
(*s).score = 90;
s->grade = 'A';         // dereference와 필드 참조를 통합
```

- 메모리 관리를 위해서 포인터를 제공하는 경우, 메모리 allocation 연산을 제공한다.

내장 부프로그램	할당 연산자 제공
C의 malloc	C++, Java의 new

C와 C++의 포인터

C와 C++의 포인터

- 포인터 산술 연산이 가능하다.
- 메모리에 있는 거의 모든 변수를 가리킬 수 있다.

  (Pascal/Ada-83은 단지 힢 기억장소만을 가리킬 수 있다)

```
int *ptr;
int count, init;
…
ptr = &init;


count = *ptr;
```

```
int list[10];
int *ptr;
...
ptr = list; // 배열이름은 상수
...
*(ptr+1)…
*(ptr+index)…
ptr[index]…
```

C와 C++의 포인터

C와 C++의 void 포인터

void * 타입의 포인터

- 어떠한 타입의 값도 가리킬 수 있다.
- dereference 될 수 없다.
- 메모리 관련 함수의 parameter 로 사용

```
genfun(void * v, int i)
{ switch (i) {
  case 1: printf("%d\n", *(int *)v); break;
  case 2: printf("%c\n", *(char *)v); break;
  }
}

int *p; char *c;
  …
p = (int *) malloc(sizeof(int));
c = (char *) malloc(sizeof(char));
genfun(p, 1); genfun(c, 2);
```

// malloc()의 return 타입은 void*

평가하기

마지막으로 내가 얼마나 이해했는지를 한번 확인해 볼까요?

총 2문제가 있습니다.

START

평가하기 1

1. 포인터에 대한 설명으로 거리가 먼 것은?

① 포인터 변수는 값으로 주소 값을 가진다.

② dereference (역참조) 란 포인터 변수가 내용으로 가지고 있는 주소를 취하여 그 주소에 저장된 값을 가져오는 것을 말한다.

③ 포인터의 목적은 오류를 사전에 알려주는 것이다.

④ Heap dynamic 변수란 heap 으로부터 동적으로 할당되는 변수이다.

확인

평가하기 2

2. C 함수 f 가 인자는 없고, void*를 리턴 타입으로 가질 때 다음 중 타입 오류가 발생하지 않는 경우는?

① `char * c = (char *) f();`

② `float d = (float) f();`

③ `char * c = (void *) f();`

④ `float d = (float *) f();`

확인

정리하기

포인터의 개념

포인터는 heap, stack의 메모리 주소 값을 프로그램에서 직접 다루도록 허용하는 고급 개념이며 많이 사용되는 언어인 C, C++ 에서 지원한다.

C의 void *는 어떠한 타입의 값도 가리킬 수 있으며, dereference 될 수 없지만, 메모리 관련 함수의 parameter 로 사용 된다.