튜터링 6주차

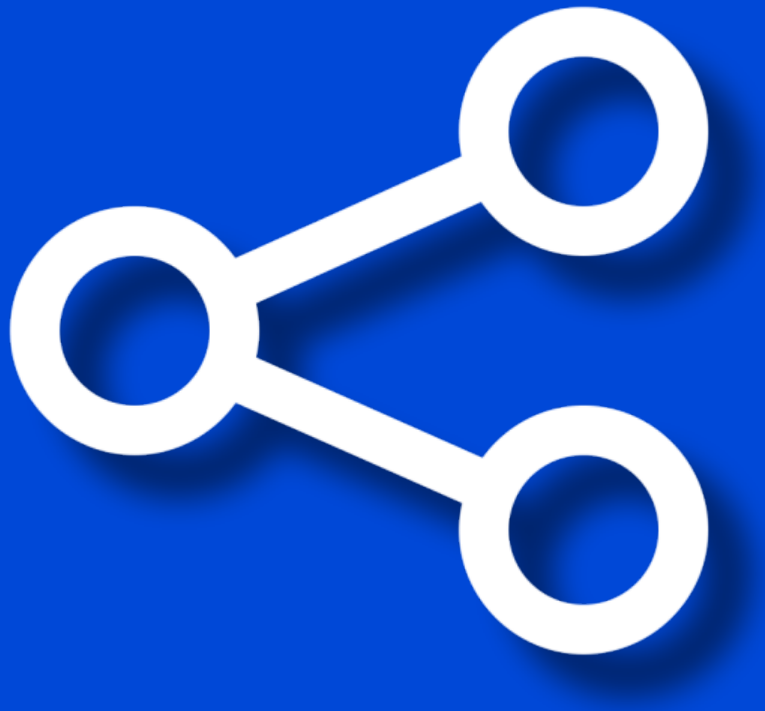
(TUTOR: 성열암)

응용컴퓨터 프로그래밍

TUTORING — https://github.com/developersung13/cbnu-tutoring

공용체의 정의 파악 및 2~5주 차 학습 내용 복습

CONTENTS
INDEX



공용체

공용체(union)는 구조체와 같은 서로 다른 자료형의 집합입니다. 구조체와의 하지만, 차이점으로 공용체는 멤버 중에서 가장 큰 자료형의 크기만큼만 메모리를 할당받습니다.

01

01 공용체 (1/3)

공용체의 정의는 구조체와 동일합니다.
하지만 union이라는 키워드를 사용하며,
공용체명을 이어적습니다.

❑ 공용체 변수 선언

```c
#include <stdio.h>
#include <string.h>

union student {
    int studentID;
    int age;
    char name[20];
};

int main() {
    union student s = { 2020039037 };
    printf("%d, %d, %s", s.studentID, s.age, s.name);
    return 0;
}
```

```
2020039037, 2020039037, }Ygx
```

01 공용체 (1/3)

❑ typedef

```c
#include <stdio.h>
#include <string.h>

union student {
    int studentID;
    int age;
    char name[20];
};

int main() {
    union student s = { 2020039037 };
    printf("%d, %d, %s", s.studentID, s.age, s.name);
    return 0;
}
```

```
2020039037, 2020039037, }Ygx
```

복습

튜터링 2~5주 차 사이에 학습하였던 내용들을 다시 복습하여
C언어의 기본적인 문법을 완벽히 숙지할 수 있도록 합니다.

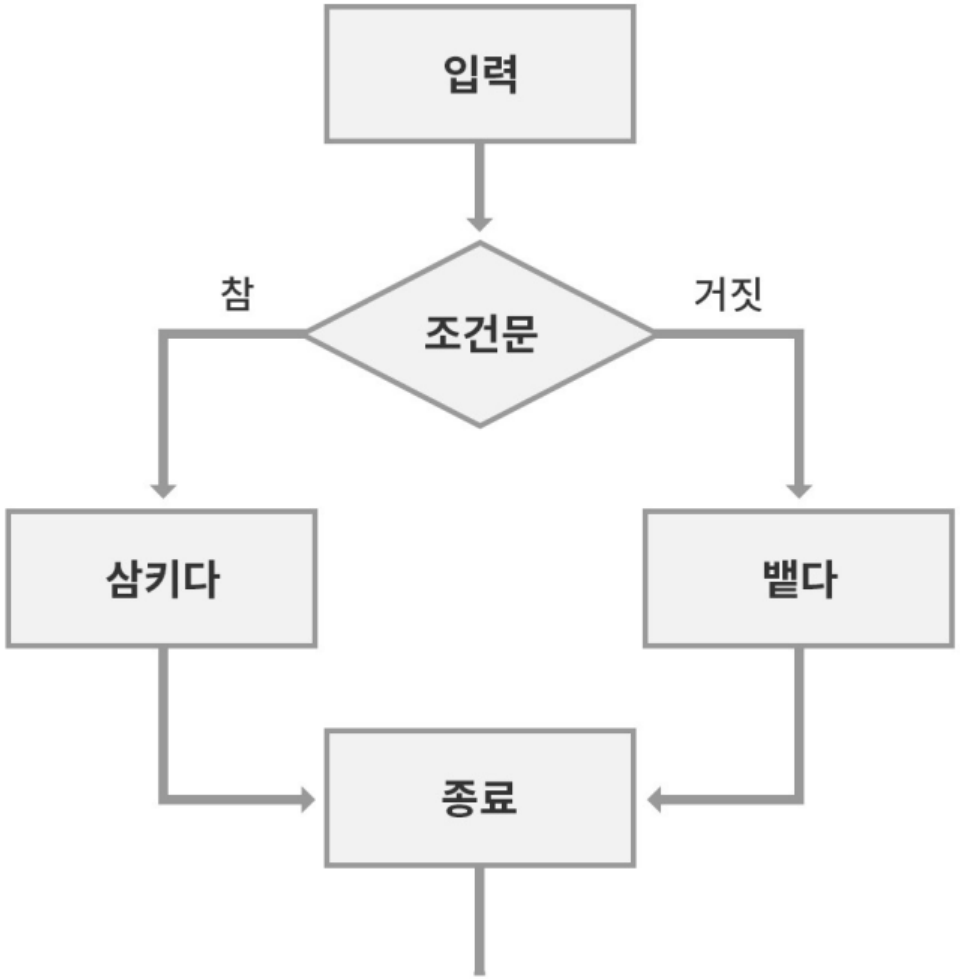
02

조건문

어떤 조건이 주어질 때 특정 동작을 수행하도록 하는,
즉 조건에 따라 실행을 결정할 때 사용하는 문법입니다.

01

01 조건문 (1/6)




```
if (음식이 맛있는가) {
    삼키다;
} else {
    뱉다;
}
```

❑ if 문

```c
if (number > 0)
    printf("양수입니다.\n");
```

if 문에서는 조건을 수식으로 표현하고, 그 수식을 바로 '조건식'이라고 합니다. 따라서 주어진 조건식을 계산 및 판단하여 그 결괏값에 따라 실행을 달리합니다.

❑ else 문

```
if (number > 0)
    printf("양수입니다.\n");
else
    printf("0 혹은 음수입니다.\n");
```

조건식이 거짓인 경우에도
처리를 가능하게 합니다.

❑ else if 문

```
if (number > 0)
    printf("양수입니다.\n");
else if (number == 0)
    printf("0입니다.\n");
else
    printf("음수입니다.\n");
```

다양한 조건식을 정의
하고자 할 때 사용합니다.

❑ switch 문

```c
#include <stdio.h>

int main() {
    int score = 90;
    switch(score / 10) {
        case 10:
        case 9: printf("A"); break;
        case 8: printf("B"); break;
        case 7: printf("C"); break;
        case 6: printf("D"); break;
        default: printf("F");
    } return 0;
}
```

하나의 조건식을 사용하여
발생할 수 있는 경우들을
정의해 선택을 달리합니다.

❑ goto 문

```c
int main() {
    int num = 2;
    if (num == 1) goto ONE;
    else if (num == 2) goto TWO;
    else goto EXIT;

    printf("Entered");

    ONE:
        printf("1입니다.\n");
        goto EXIT;
    TWO:
        printf("2입니다.\n");
        goto EXIT;
    EXIT:
        return 0;
}
```

```
Compiled Successfully. memory: 1632 time: 0 exit code: 0
2입니다.
```

정의된 레이블로 순서를 상관하지 않고 강제로 이동하여 코드를 실행합니다.

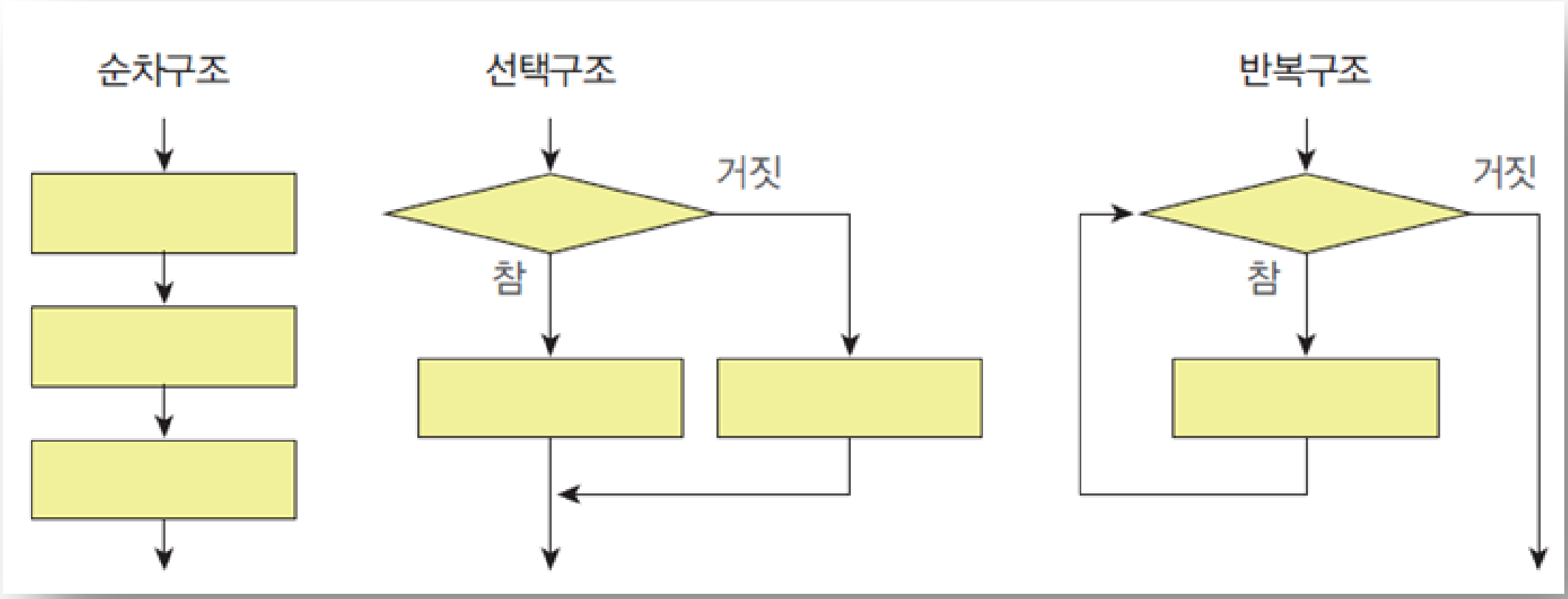
제어구조

프로그램에서 실행되는 문장들의 실행 순서를
제어/변경할 수 있는 문장

02

02 제어구조 (1/2)

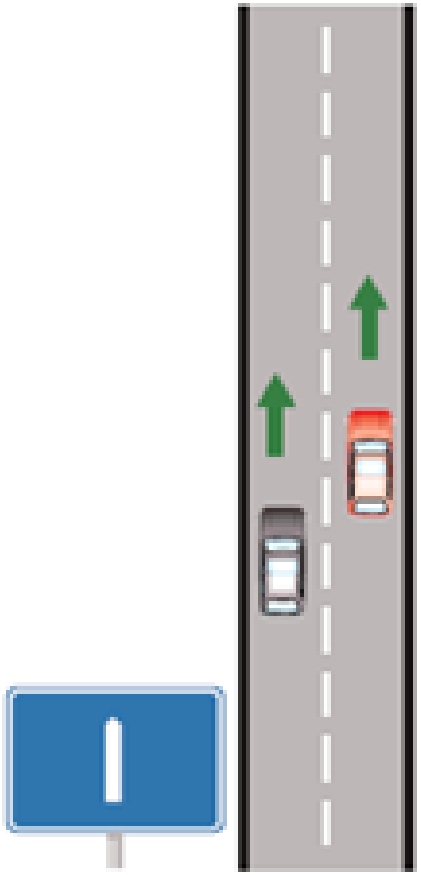
❑ 종류



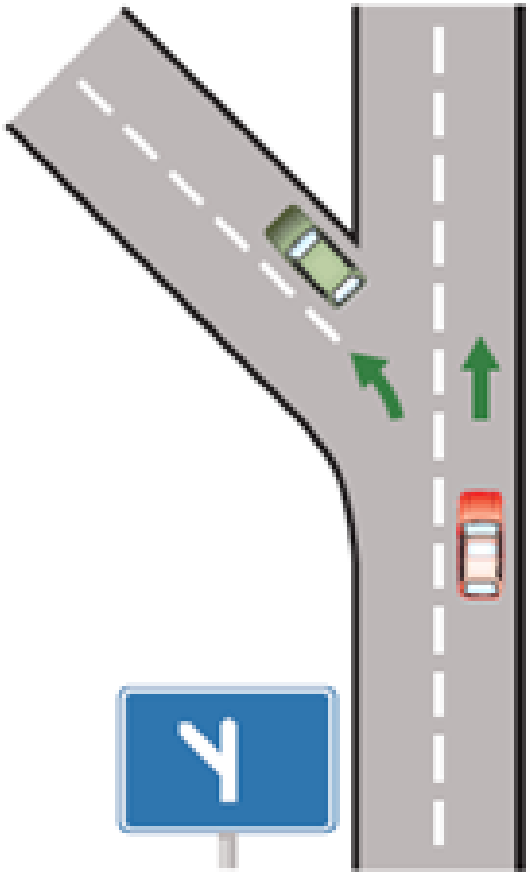
위의 순서도(Flowchart)를 보고
연상되는 문법이 있으신가요?

02 제어구조 (2/2)

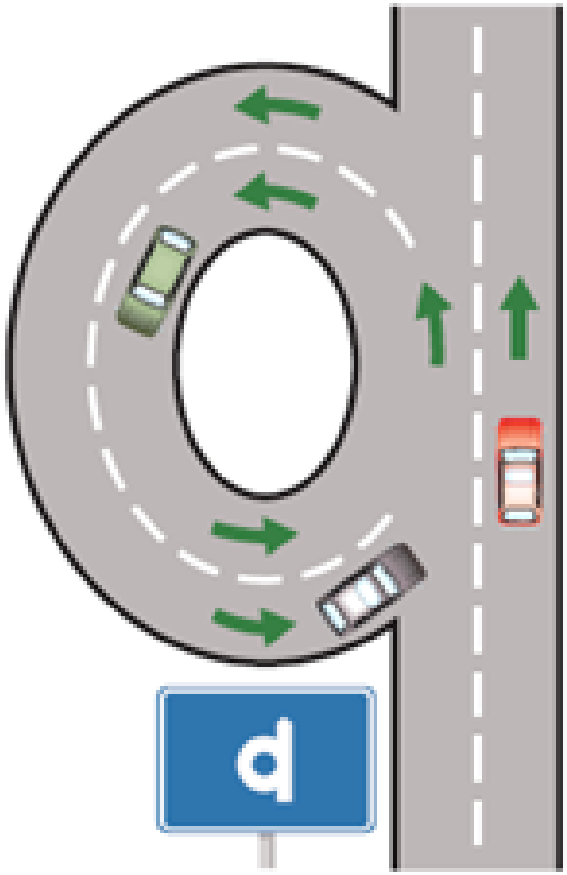
❑ 종류



순차 구조



선택 구조



반복 구조

함수

$$f(x)$$

하나의 특별한 목적의 작업을 수행하기 위해 독립적으로 설계된
프로그램 코드의 집합을 정의하는 문법입니다,

03

❑ 인수(argument)

```c
#include <stdio.h>

int add(int x, int y) {
    return x + y;
}

int main() {
    int result;
    result = add(5, 10);
    printf("%d", result);
}
```

함수를 호출하면서
넘겨주는 값

❑ 매개변수(parameter)

```c
#include <stdio.h>

int add(int x, int y) {
    return x + y;
}

int main() {
    int result;
    result = add(5, 10);
    printf("%d", result);
}
```

호출된 함수가 인수로부터
넘어온 값을 저장한 변수

❑ 함수원형(prototype)

```c
#include <stdio.h>

int add(int, int);

int main() {
    int result;
    result = add(5, 10);
    printf("%d", result);
}

int add(int x, int y) {
    return x + y;
}
```

컴파일러에게 함수에 대한 정보를 미리 알려주는 방법입니다.

❑ 재귀함수(recursive function)

```c
#include <stdio.h>

int fibo(int n) {
    if (n == 0) return 0;
    else if (n == 1) return 1;
    return fibo(n-1) + fibo(n-2);
}

int main() {
    int n;
    scanf("%d", &n);
    for (int k=0; k < n; k++)
        printf("%d ", fibo(k));
    return 0;
}
```

```
Compiled Successfully. memory

 0 1 1 2 3 5 8 13 21 34
```

함수 내에서 함수 스스로를
재호출하는 것입니다.

배열

동일한 자료형의 값들을 하나의 변수에 저장하여
효율적으로 관리하고자 사용하는 자료형입니다.

01

01 배열 (1/2)

❑ 1차원 배열

arr[4]

arr[0]	arr[1]	arr[2]	arr[3]

배열의 첨자는 0부터 시작합니다.

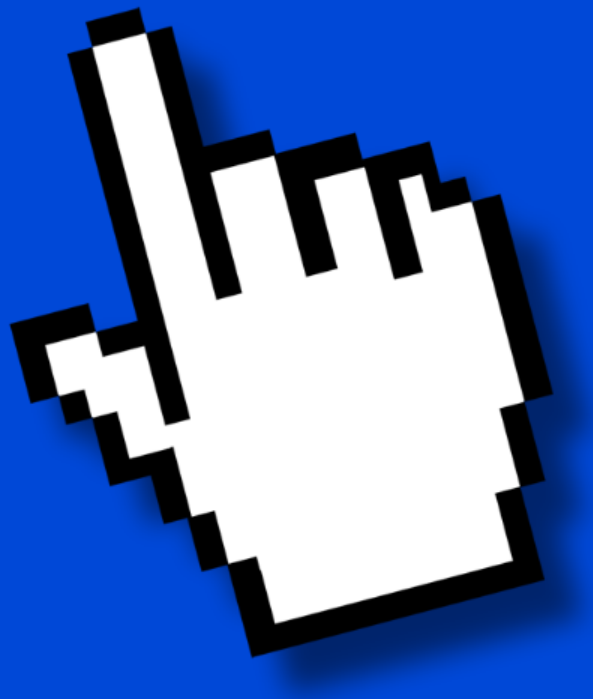
❑ 2차원 배열

arr[2][4]

arr[0][0]	arr[0][1]	arr[0][2]	arr[0][3]
arr[1][0]	arr[1][1]	arr[1][2]	arr[1][3]

2차원 배열은 '**행**' 과 '**열**'로
구성됩니다.

포인터



메모리의 주솟값을 저장하여 해당 주소에 직접적인
참조를 하고자 할 때 사용하는 문법입니다.

02

❑ 포인터

```c
#include <stdio.h>

int main() {
    int num = 10;
    int *pNum = &num;
    *pNum = *pNum + 7;
    printf("%d", num);
    return 0;
}
```

포인터 변수는 다른 변수의
주솟값을 참조합니다.

❑ 2중 포인터

```c
#include <stdio.h>

int main() {
    int num = 10;
    int *pNum = &num;
    int **dpNum = &pNum;
    *pNum = *pNum + 7;
    printf("%d\n", num);
    **dpNum = **dpNum - 5;
    printf("%d", num);
    return 0;
}
```

포인터 변수의 주솟값을
참조하는 포인터 변수입니다.

❑ 3중 포인터

```c
#include <stdio.h>

int main() {
    int num = 10;
    int *pNum = &num;
    int **dpNum = &pNum;
    int ***tpNum = &dpNum;

    *pNum = *pNum + 1;
    printf("%d\n", num);

    **dpNum = **dpNum + 1;
    printf("%d\n", num);

    ***tpNum = ***tpNum + 1;
    printf("%d\n", num);
    return 0;
}
```

포인터 변수의 주솟값을 참조하는 포인터 변수의 주솟값을 참조하는 포인터 변수입니다.

❑ Call-by-value(값에 의한 참조)

```c
#include <stdio.h>

int swap(int x, int y) {
    int temp = x;
    x = y;
    y = temp;
}

int main() {
    int a = 100, b = 200;
    printf("Before: %d, %d\n", a, b);
    swap(a, b);
    printf("After: %d, %d\n", a, b);
    return 0;
}
```

```
Before: 100, 200
After: 100, 200
```

인수값의 복사로
데이터를 처리하는
방법

❑ Call-by-reference(주소에 의한 참조)

```c
#include <stdio.h>

int swap(int* x, int* y) {
    int temp = *x;
    *x = *y;
    *y = temp;
}

int main() {
    int a = 100, b = 200;
    printf("Before: %d, %d\n", a, b);
    swap(&a, &b);
    printf("After: %d, %d\n", a, b);
    return 0;
}
```

```
Before: 100, 200
After: 200, 100
```

인수값이 저장되어 있는 주솟값 자체를 참조하여 데이터를 처리하는방법

❑ 배열과 포인터의 관계

```c
#include <stdio.h>

int main() {
    int a[5] = { 10, 3, 1, 2, 4 };

    printf("a의 address: &a[0] = %p, a = %p \n", &a[0],a);
    printf("a의 value: %d\n", *a);
    for (int k = 0; k < 5; k++)
        printf("\t address: %p, a[%d]: %d, *(a+%d): %d\n", (a+k), k, a[k], k, *(a+k));
}
```

```
a의 address: &a[0] = 0x7fffd7b4c3d0, a = 0x7fffd7b4c3d0
a의 value: 10
        address: 0x7fffd7b4c3d0, a[0]: 10, *(a+0): 10
        address: 0x7fffd7b4c3d4, a[1]: 3, *(a+1): 3
        address: 0x7fffd7b4c3d8, a[2]: 1, *(a+2): 1
        address: 0x7fffd7b4c3dc, a[3]: 2, *(a+3): 2
        address: 0x7fffd7b4c3e0, a[4]: 4, *(a+4): 4
```

hello world

abc

programming language

문자와 문자열

글자와 문장을 변수에 저장하고자
할 때 사용하는 자료형입니다.

01

01 문자와 문자열 (1/3)

❑ 문자

```c
#include <stdio.h>

int main() {
    char c = 'A';
    printf("%c", c);
}
```

```
Compiled Successfully.
A
```

문자형은 1byte 크기의 데이터를 저장할 때 사용하는 자료형입니다.

❑ 문자열

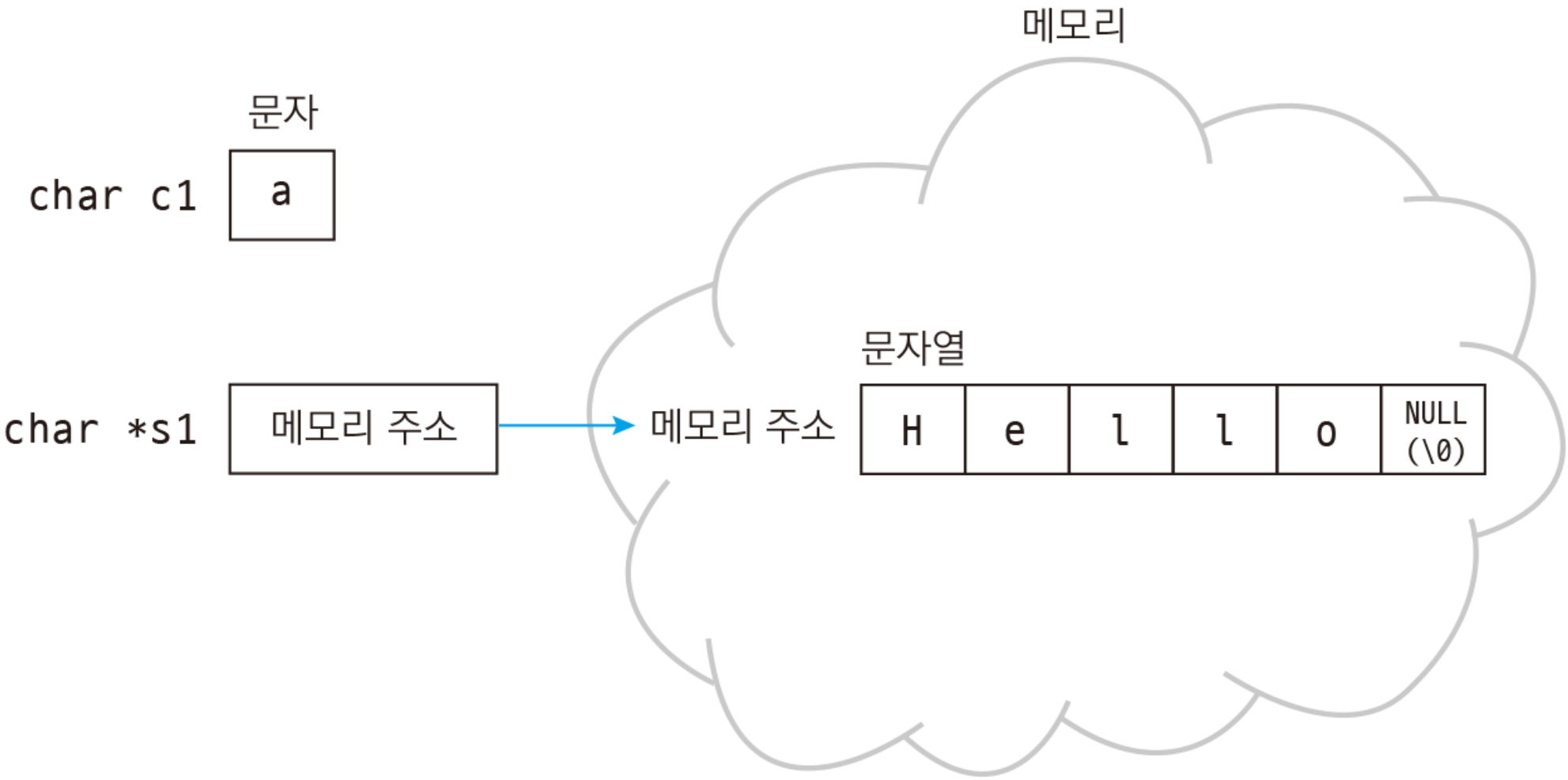
```c
#include <stdio.h>

int main() {
    char c[] = "ABC DEF";
    printf("%s", c);
}
```

```
Compiled Successfully.
ABC DEF
```

1개 이상의 문자들을 하나의 변수에 저장하고자 할 때 사용하는 문자의 배열입니다.

01 문자와 문자열 (3/3)

❑ 구조



배열

동일한 자료형의 값들을 하나의 변수에 저장하여
효율적으로 관리하고자 사용하는 자료형입니다.

02

04 배열 (1/3)

❑ 1차원 배열

arr[4]

arr[0]	arr[1]	arr[2]	arr[3]

배열의 첨자는 0부터 시작합니다.

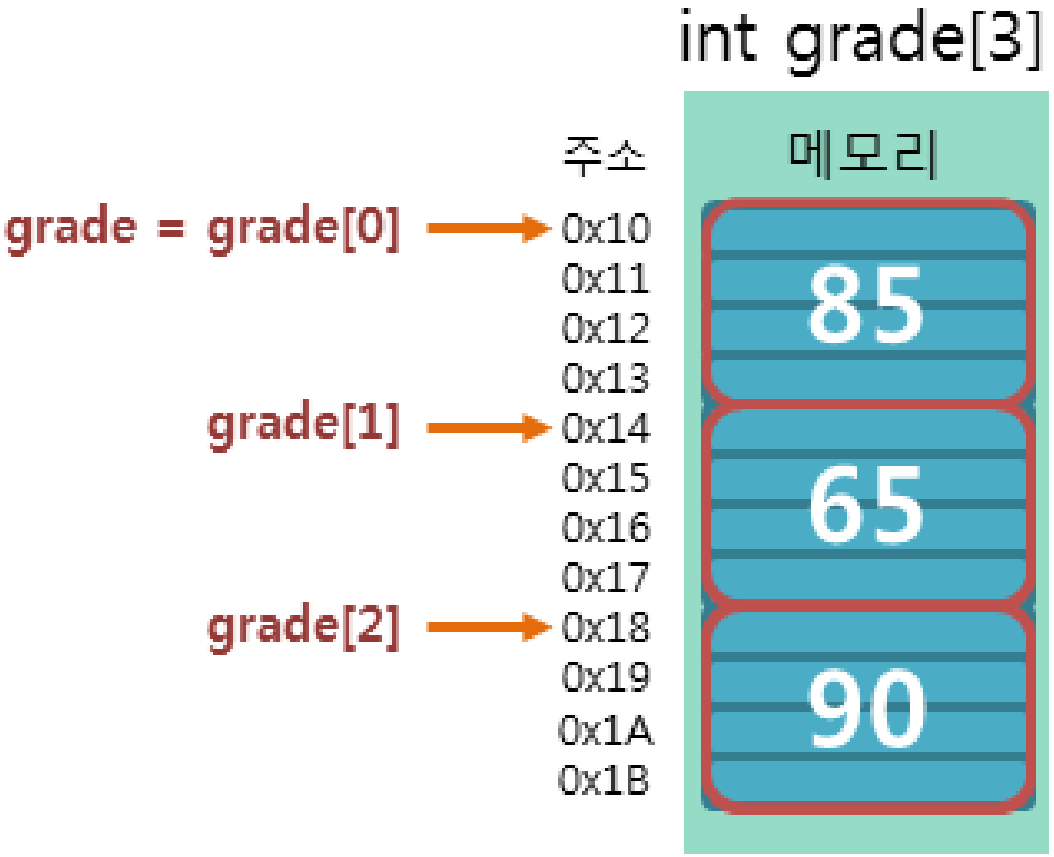
❑ 2차원 배열

arr[2][4]

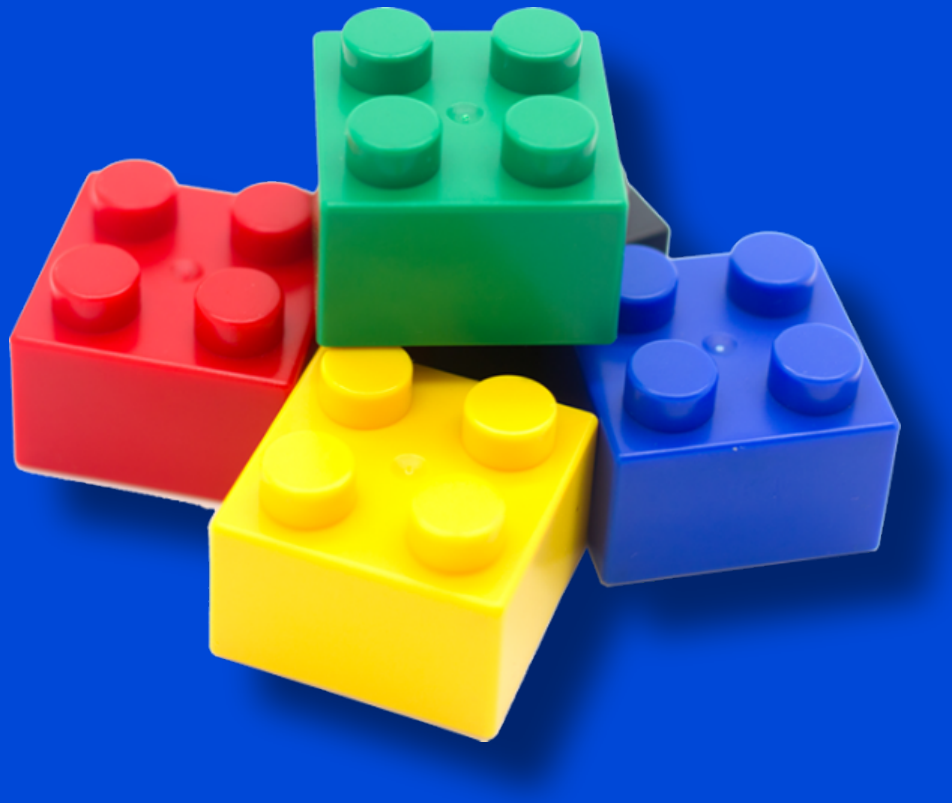
arr[0][0]	arr[0][1]	arr[0][2]	arr[0][3]
arr[1][0]	arr[1][1]	arr[1][2]	arr[1][3]

2차원 배열은 '**행**' 과 '**열**'로
구성됩니다.

❑ 메모리 상 배열의 구조



배열명은 배열의 첫 번째 요소가 저장되어 있는 그 주솟값을 가리킵니다.



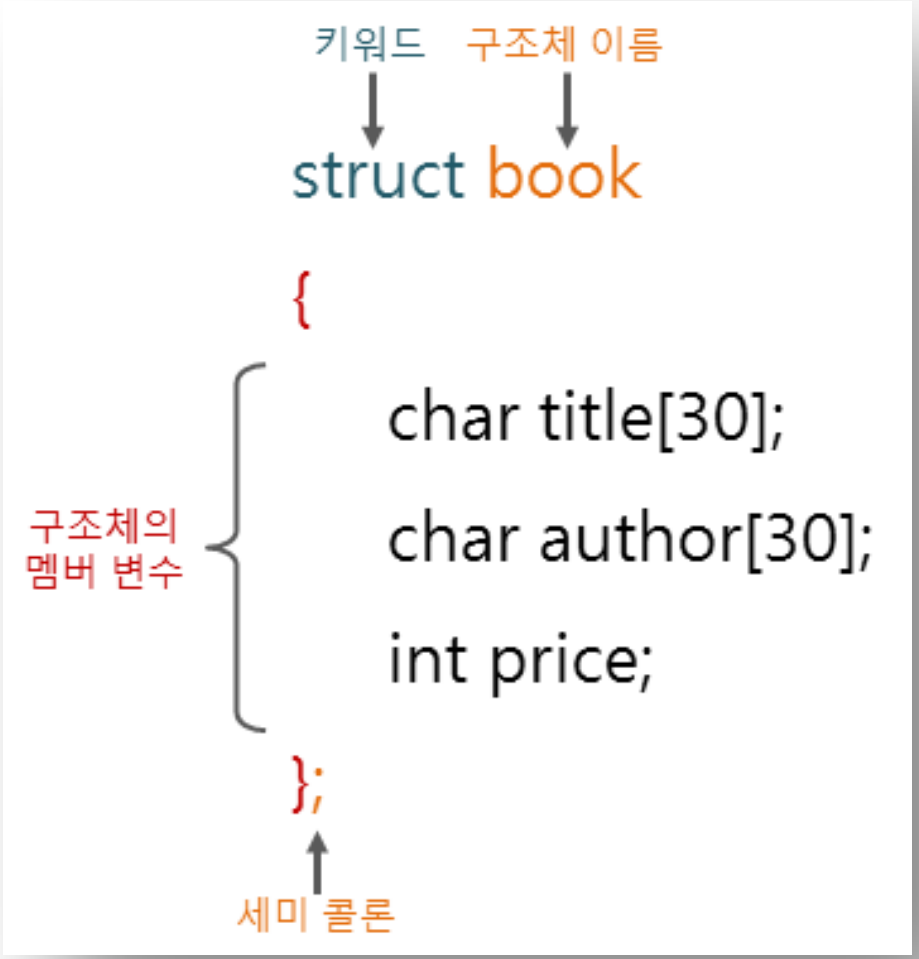
구조체

C언어에서 사용되는 기본 타입을 가지고 새롭게 정의할 수 있는 사용자 정의 타입입니다. 구조체는 기본 타입만으로는 나타낼 수 없는 복잡한 데이터를 표현할 수 있습니다.

01

01 구조체 (1/3)

❑ 구조체



배열이 같은 타입의 변수 집합이라고 한다면, 구조체는 다양한 타입의 변수 집합을 하나의 타입으로 나타낸 것입니다.

❑ 구조체 변수 선언

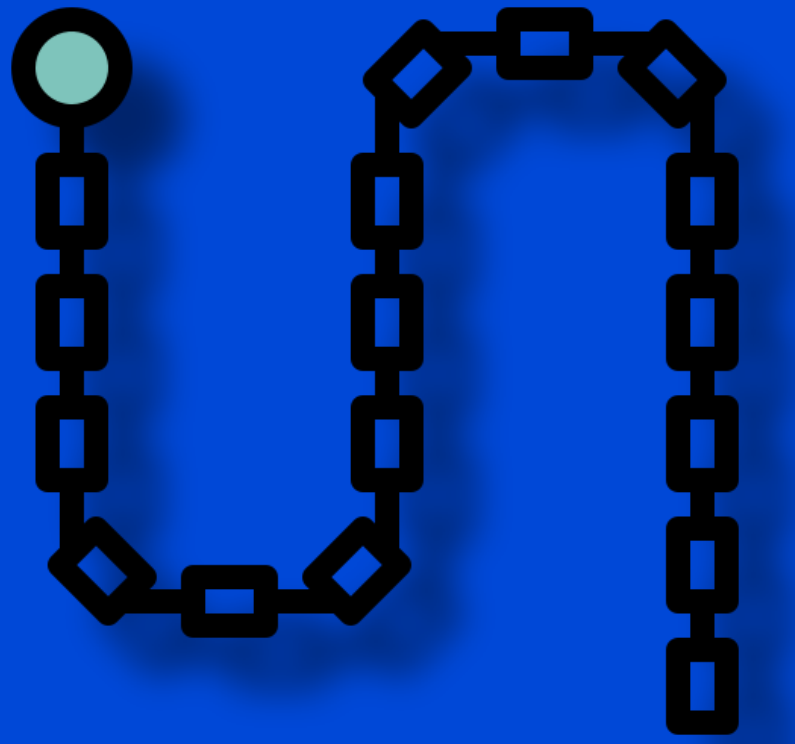
```c
#include <stdio.h>

struct book {
    char title[30];
    char author[30];
    int price;
};

int main() {
    struct book myBook = { "제목", "작가", 5000 };
    printf("%s\n%s\n%d", myBook.title, myBook.author, myBook.price);
}
```

01 구조체 (1/3)

❑ typedef

```c
#include <stdio.h>

typedef struct {
  int age;
  char phone_number[14];
} Student;

int main(){
  Student goorm;

  printf("나이 : ");
  scanf("%d", &goorm.age);
  printf("번호 : ");
  scanf("%s", goorm.phone_number);

  printf("----\n나이 : %d\n번호 : %s\n----", goorm.age, goorm.phone_number);

  return 0;
}

```

연결 리스트



물리적으로 흩어져 있는 자료들을 서로 연결하여 하나로 묶는 방법을 연결리스트(linked list)라고 한다.

02

02 연결 리스트 (1/5)

❑ 장점

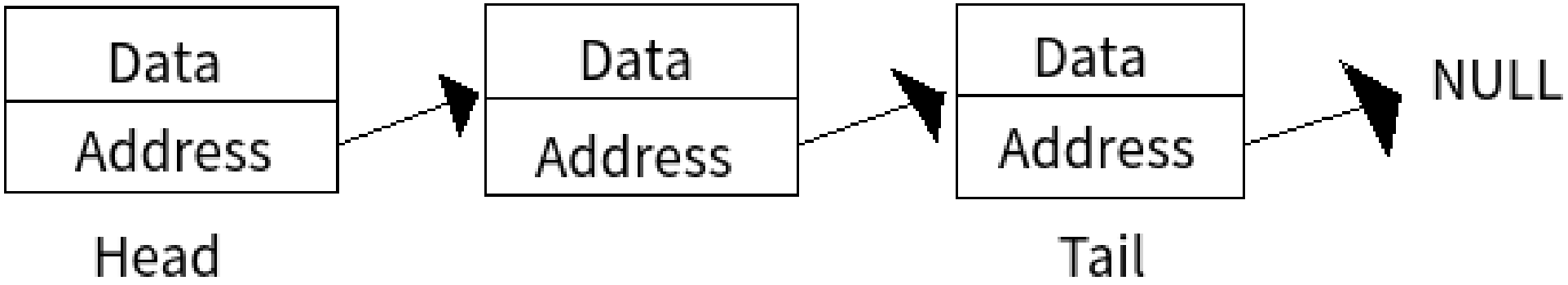
데이터를 저장할 공간이 필요할 때마다 **동적**으로
공간을 만들어서 쉽게 추가할 수 있다는 것.
이것은 순차적인 표현 방법은 배열에 비하여
상당한 장점.

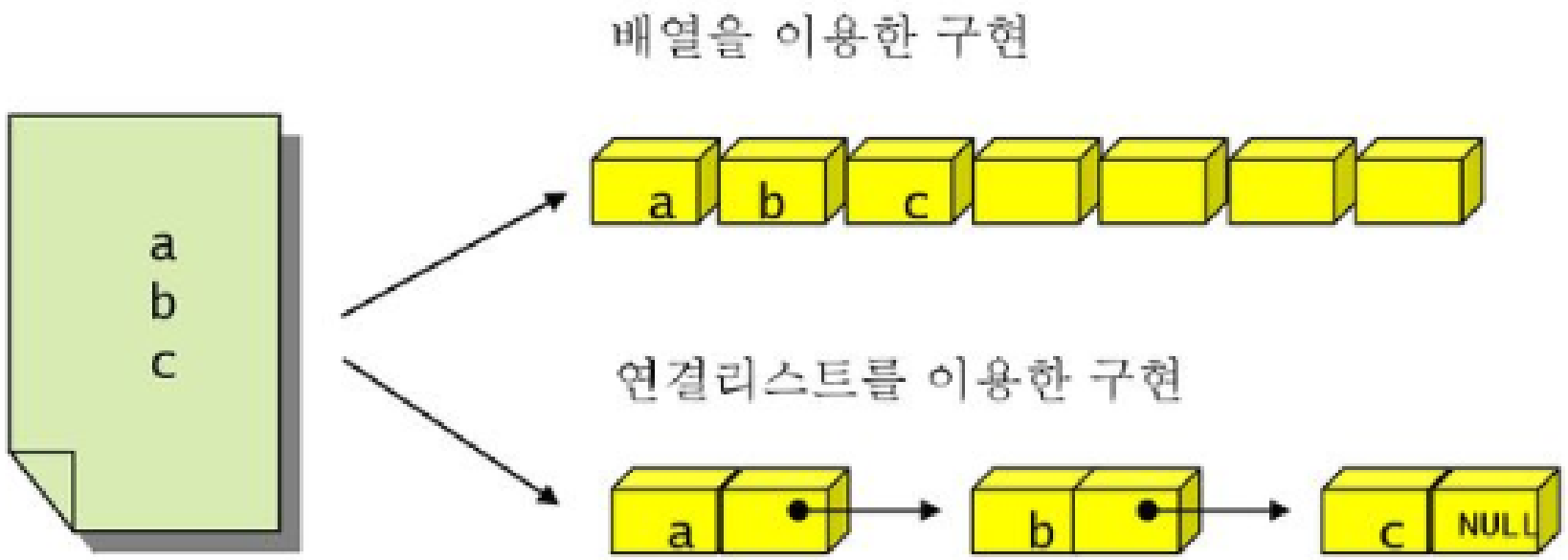
❑ 단점

배열에 비하여 상대적으로 구현이 어렵고 오류가 발생하기쉬움 또한 데이터 뿐만 아니라 포인터도 저장해야 하므로 메모리 공간을 많이 사용. 또 i 번째 데이터를 찾으려면 앞에서부터 순차적으로 접근해야함.

02 연결 리스트 (3/5)

❑ 구조

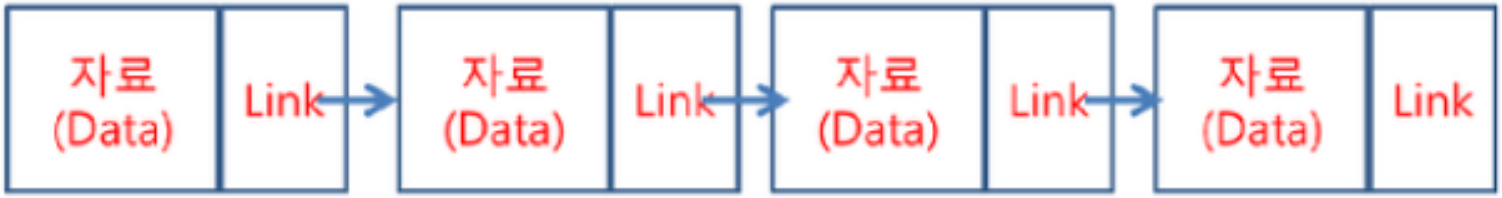




02 연결 리스트 (4/5)

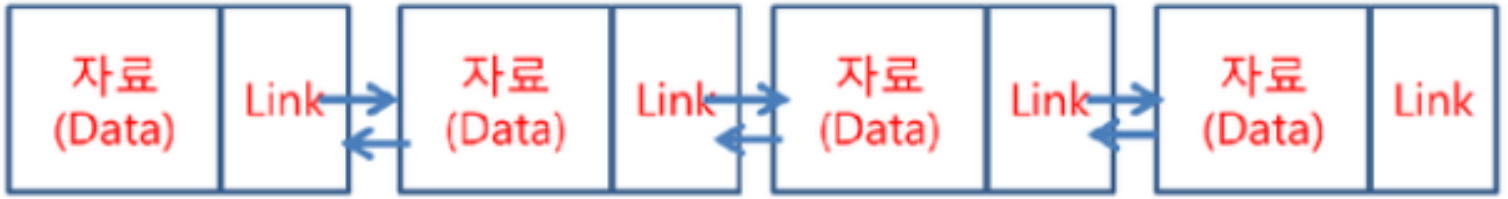
❑ 종류

단순 연결 리스트



원형 연결 리스트

이중 연결 리스트



❑ 적용 코드

shorturl.at/bGJLS

```
INSERT [10]
INSERT [30]
INSERT [20]
INSERT [50]
HEAD >  10 30 20 50 END.
DELETE [30]
DELETE [10]
HEAD >  20 50 END.
DELETE [15]
Can't find the key!
```

질의응답

금일 튜터링을 진행하며 이해가 어려운 부분이 있었거나,
교과목과 관련하여 궁금한 내용을 질문하고 답변드리는
시간입니다.

04

THANKYOU

TUTORING

https://github.com/developersung13/cbnu-tutoring