

3. C 프로그래밍 복습 1

한국외국어대학교
고 석 훈

목차

3.1 C 프로그래밍 도구

3.2 배열(Array)

3.3 포인터(Pointer)

3.4 구조체(Structure)

3.5 동적 메모리(dynamic memory)

3.6 연결 리스트(Linked List)

3.7 트리(Tree) 자료구조

3.1 C 프로그래밍 도구

- Visual Studio IDE
 - Microsoft에서 제공하는 가장 진보된 통합 개발 환경
 - 소스코드 편집기, 컴파일러, 디버거를 통합
 - Visual Basic, C#, C++ 등 개발 가능
 - 학생 및 개인 개발자에게 Community 2017 버전 무료 제공
- 다운로드 및 설치
 - Visual Studio 홈페이지에서 다운로드 및 설치 정보 확인
 - 홈페이지 <https://www.visualstudio.com/ko/>



Visual Studio 프로젝트 만들기

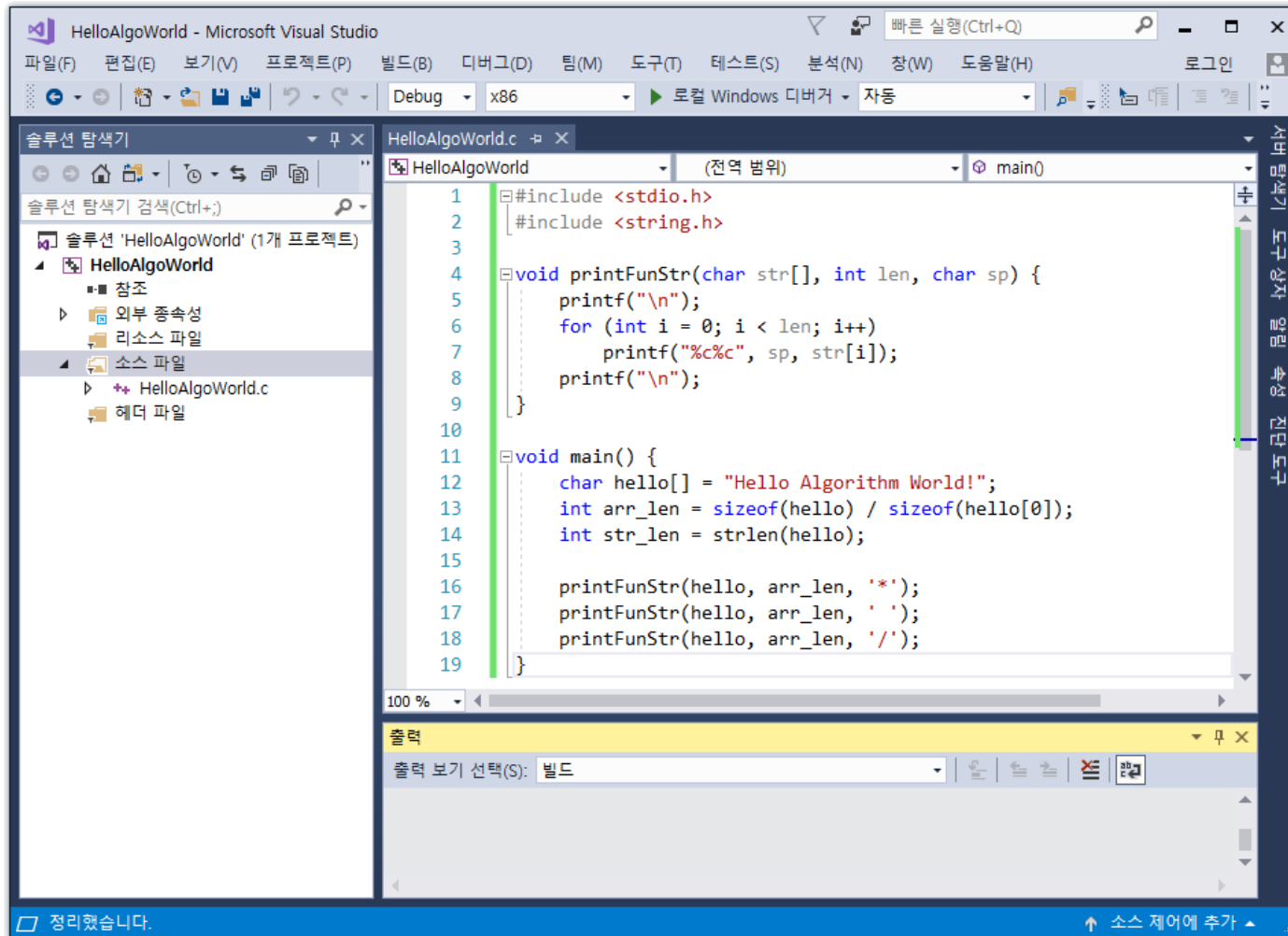
- 새 프로젝트 만들기

1. 파일(F) ➡ 새로 만들기(N) ➡ 프로젝트(P)
2. Visual C++ ➡ 빈 프로젝트 선택
3. 이름(N): **HelloAlgoWorld** 입력 ➡ 확인
4. 솔루션 탐색기에 HelloAlgoWorld 프로젝트 생성됨

- 새 소스코드 추가

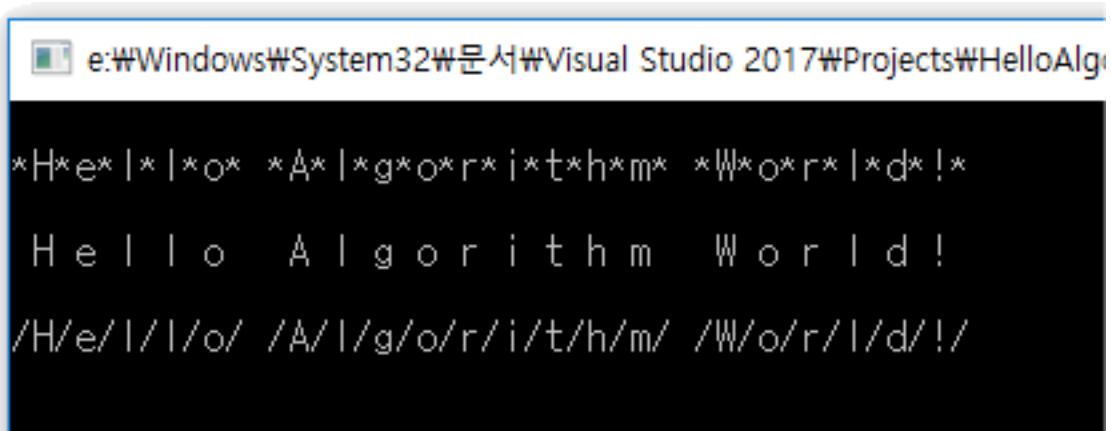
1. HelloAlgoWorld 프로젝트 – 소스파일의 옵션메뉴 ➡ 추가(D)
➡ 새 항목(W)
2. Visual C++ ➡ C++ 파일(.cpp) 선택
3. 이름(N): **HelloAlgoWorld.c** 입력 ➡ 추가
 - ◆ 확장자를 .c로 하여야 C프로그램으로 해석됨

HelloAlgoWorld.c 코드 작성



빌드 & 실행

- 프로그램 실행
 - F5를 누르거나 [▶ 로컬 Windows 디버거]를 눌러 프로그램을 빌드(컴파일 & 링크)하고 실행한다.
 - 프로그램이 실행되고 바로 종료되어 결과 확인이 어려운 경우
 - ◆ 소스코드 마지막 부분에 중단점(break-point)을 지정하거나
 - ◆ 소스코드 마지막에 `getchar()`를 추가한다.



```
e:\Windows\System32\문서\Visual Studio 2017\Projects\HelloAlg
*H*e*|*|*o* *A*|*g*o*r*i*t*h*m* *W*o*r*|*d*!*
Hello Algorithm World!
/H/e/l/l/o/ /A/l/g/o/r/i/t/h/m/ /W/o/r/l/d/!//
```

소스 레벨 디버깅

- 소스 레벨 디버깅

- 프로그램의 실행을 중단하고, 프로그램의 상태(변수, 메모리, 함수 호출 상태 등)를 확인하여 오류를 찾아내는 디버깅 방법

- 디버깅 시나리오

1. 디버깅이 필요한 지점에 중단점(break-point) 지정
2. 디버깅 모드로 프로그램 실행 ➡ 중단점에서 멈춤
3. 상태확인 창에서 프로그램 상태 확인
 - ◆ 자동 / 로컬 / 조사식1(사용자가 확인하고자 하는 수식 입력)
4. 코드 줄 단위 실행하며 프로그램 상태 확인
 - ◆ 줄 단위 또는 함수 단위 프로그램 실행
5. 버그를 찾아 수정하고, 다시 디버깅 모드로 실행

- 상태 확인창
 - 자동 – automatic variable
 - 로컬 – local variable
 - 조사식 1 – user defined expression

```

15
16 | printFunStr(hello, arr_
17 | printFunStr(hello, arr_
18 | printFunStr(hello, arr_
19 | }

```

100 %

자동

이름	값	형식
arr_len	23	int
hello	0x003afabc "Hello Algc	char[23]
hello[0]	72 'H'	char
str_len	22	int

자동 로컬 스레드 모듈 조사식 1

```

15
16 | printFunStr(hello, arr_
17 | printFunStr(hello, arr_
18 | printFunStr(hello, arr_
19 | }

```

100 %

로컬

이름	값	형식
arr_len	23	int
hello	0x003afabc "Hello Algc	char[23]
str_len	22	int

자동 로컬 스레드 모듈 조사식 1

```

15
16 | printFunStr(hello, arr_
17 | printFunStr(hello, arr_
18 | printFunStr(hello, arr_
19 | }

```

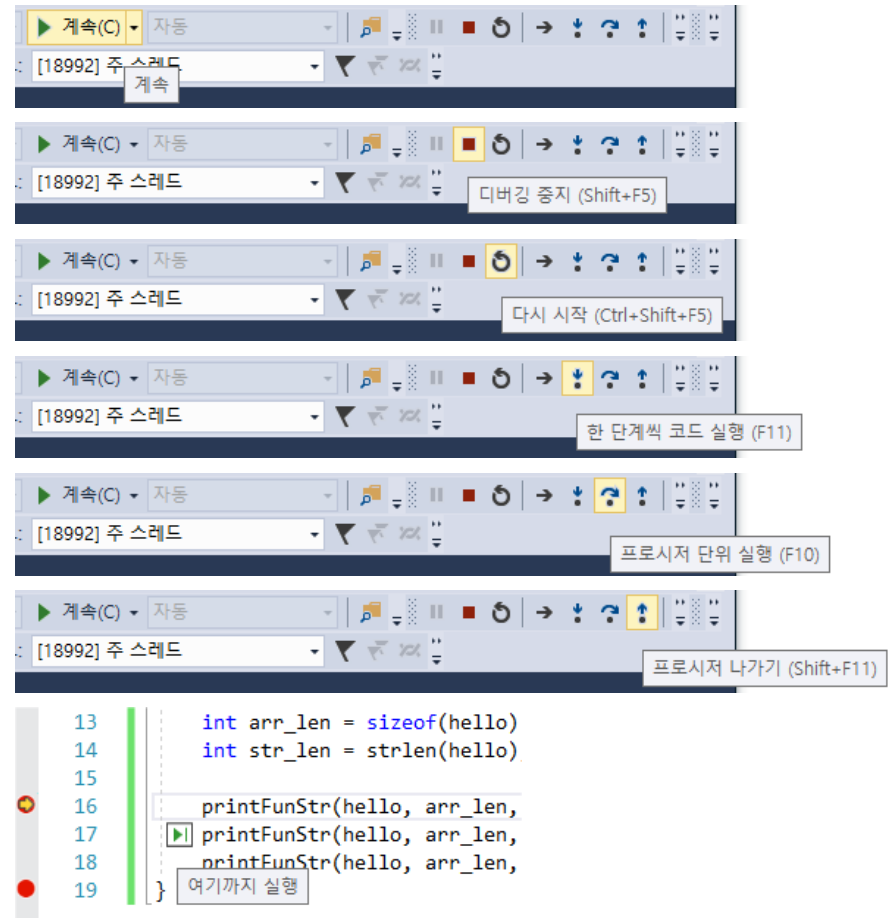
100 %

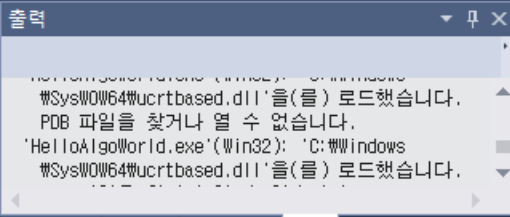
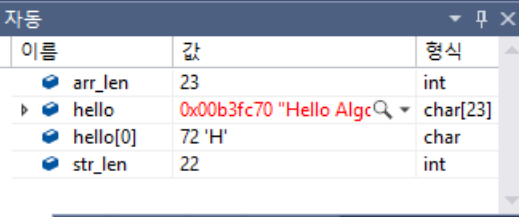
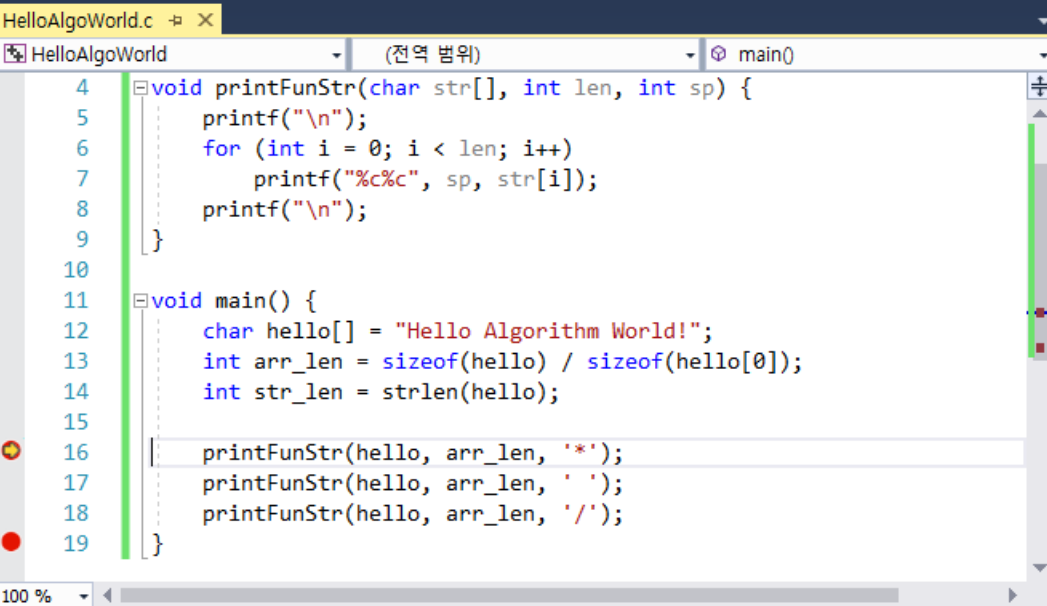
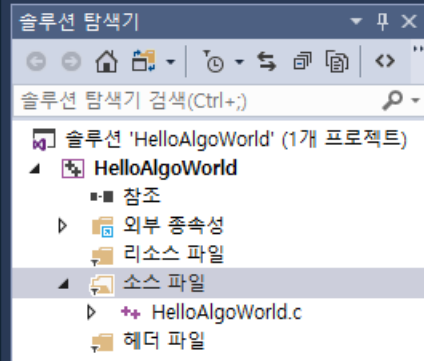
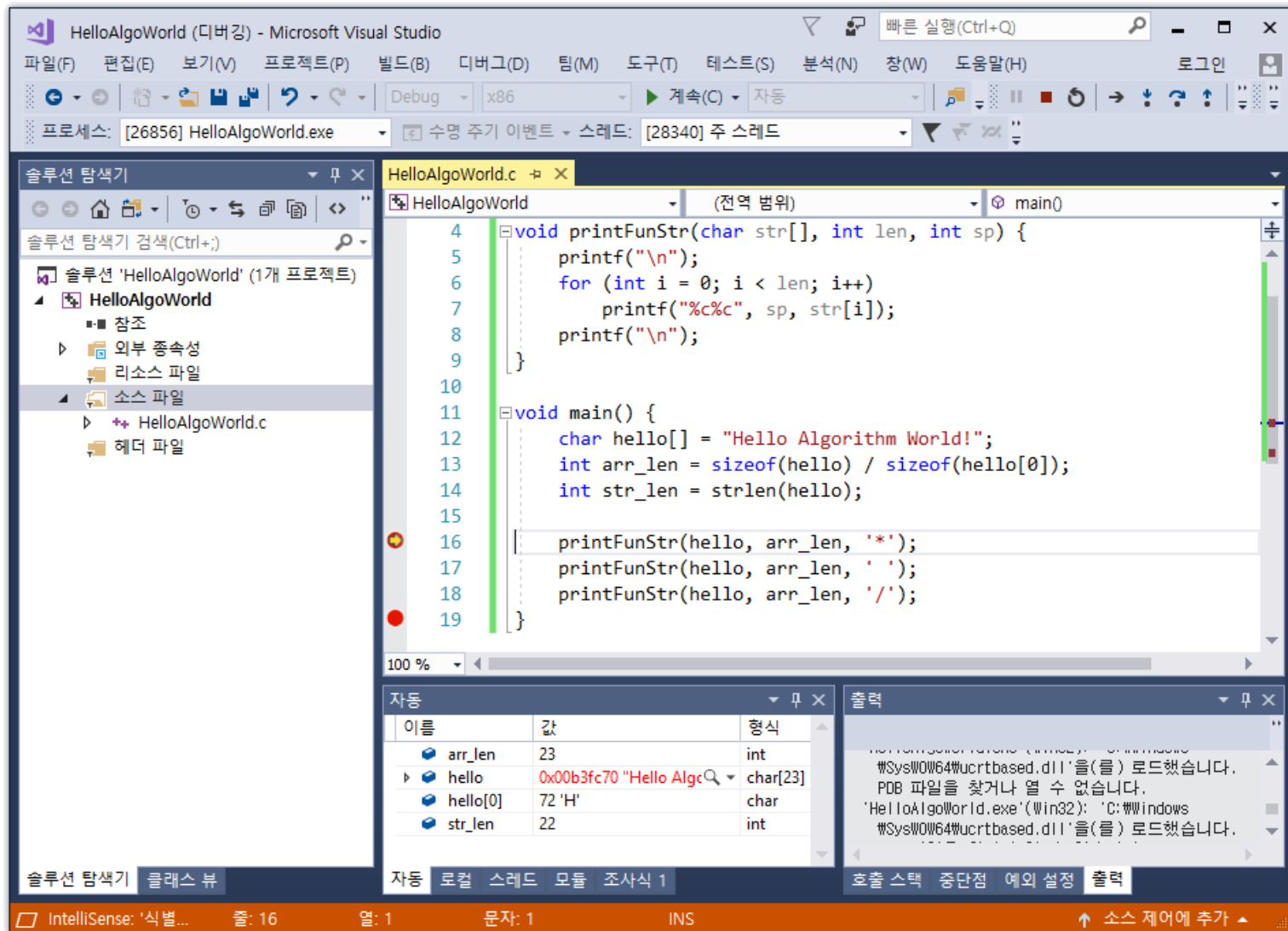
조사식 1

이름	값	형식
hello[arr_len-1]	0 '\0'	char
hello[str_len-1]	33 'I'	char
sizeof(hello)	23	unsigned

자동 로컬 스레드 모듈 조사식 1

- 소스 레벨 디버깅 기능
 - 계속 실행(C)
 - 디버깅 중지(Shift+F5)
 - 다시 시작(Ctrl+Shift+F5)
 - 한 단계씩 코드 실행(F11)
 - 프로시저 단위 코드 실행(F10)
 - 프로시저 나가기(Shift+F11)
 - 여기까지 실행





3.2 배열(Array)

- 배열(array)
 - 같은 종류의 데이터를 순차적으로 저장하는 자료구조
 - 원소(element): 배열의 개별 데이터, 배열이름[인덱스]로 표시
 - 인덱스(index): 배열 원소의 번호, 인덱스 범위는 0~(배열크기 - 1)
- 배열의 선언 및 초기화

```
int  a1[100];  
int  a2[5] = { 10, 20, 30, 40, 50 };  
int  a3[] = { 1, 2, 3, 4, 5 };  
int  a4[5] = { 0 };  
char s1[10] = {'K', 'o', 'r', 'e', 'a', '\0'};  
char s2[] = "Korea";
```

배열 연산

- 배열은 반복문과 함께 사용하여 많은 수의 원소를 간편하게 처리할 수 있음

```
#define SIZE 100

void main() {
    int a[SIZE], b[SIZE], c[SIZE], sum[SIZE], avg[SIZE];
    int i;

    // read score a, b, c

    for (i = 0; i < SIZE; i++) {
        sum[i] = a[i] + b[i] + c[i];
        avg[i] = sum[i]/3;
    }
}
```

배열의 복사

- 배열의 크기 구하기

```
size = sizeof(grade)/sizeof(grade[0]);
```

- 배열의 복사

```
void main() {  
    int a[SIZE], b[SIZE];  
    int len = sizeof(a)/sizeof(a[0]);  
    int i;  
  
    b = a;      // 컴파일 오류  
  
    for (i = 0; i < len; i++) // 올바른 방법  
        b[i] = a[i];  
}
```

예제 3-1: 배열 원소의 출력

- printArray() 함수를 만들어보자.

```
#include <stdio.h>
#define MAX_LENGTH 100

void printArray(char *name, int a[], int len) { . . . }

void main() {
    int list[MAX_LENGTH];
    int size = 5;

    for (int i = 0; i < size; i++)
        list[i] = 10 + i * 10;

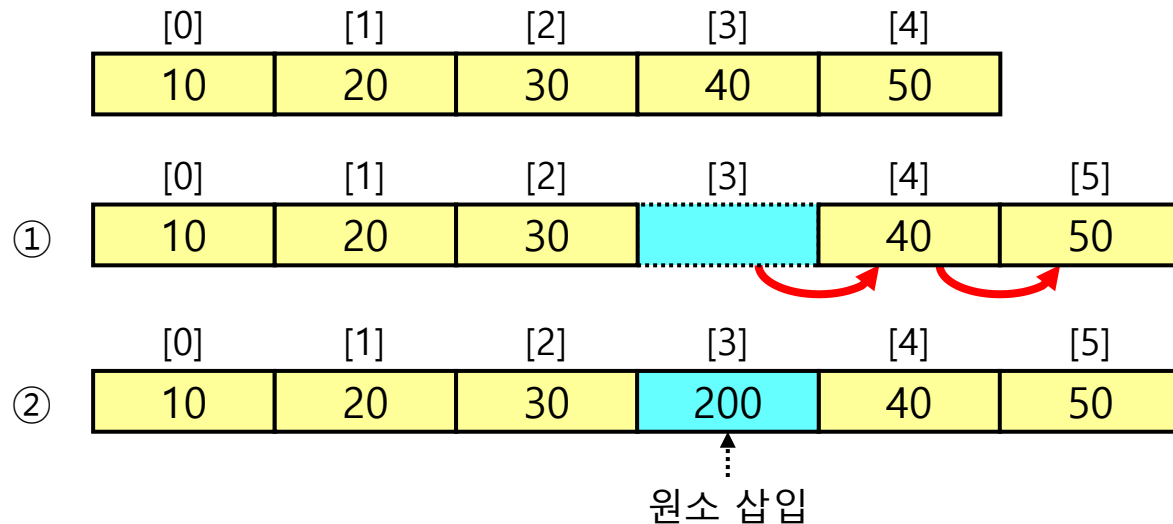
    printf("printArray() Test\n");
    printArray("list", list, 0);
    printArray("list", list, 1);
    printArray("list", list, size);
    printArray("list", list, size + 1);
}
```

```
printArray() Test
list[0] = { }
list[1] = { 10 }
list[5] = { 10, 20, 30, 40, 50 }
list[6] = { 10, 20, 30, 40, 50, -858993460 }
```

배열 원소의 삽입

- 원소 삽입 방법

- ① 원소를 삽입할 자리 이후의 원소들을 한자리씩 뒤로 이동
- ② 빈 자리에 원소 삽입



예제 3-2: 배열 원소의 삽입

- insertElem() 함수를 만들어보자.

```
#include <stdio.h>
#define MAX_LENGTH 100

int insertElem(int a[], int size, int index, int value) { . . . }

void main() {
    int list[MAX_LENGTH];
    int size = 5;

    for (int i = 0; i < size; i++)
        list[i] = 10 + i * 10;

    printf("insertElem() Test\n");
    printArray("list", list, size);
    size = insertElem(list, size, 3, 200);
    printArray("list", list, size);
    size = insertElem(list, size, 0, 300);
    printArray("list", list, size);
    size = insertElem(list, size, 7, 400);
    printArray("list", list, size);
}
```

insertElem() Test

list[5] = { 10, 20, 30, 40, 50 }

list[6] = { 10, 20, 30, 200, 40, 50 }

list[7] = { 300, 10, 20, 30, 200, 40, 50 }

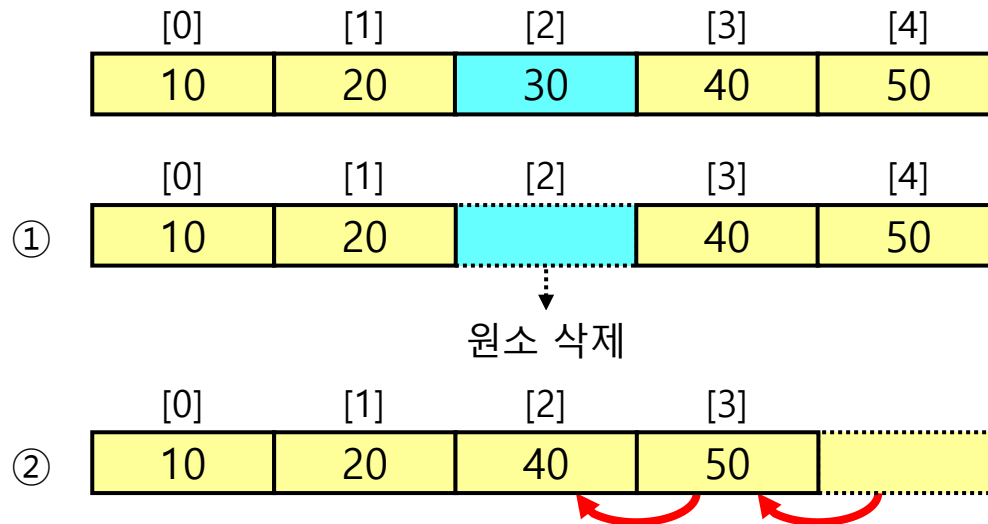
list[8] = { 300, 10, 20, 30, 200, 40, 50, 400 }

배열 원소의 삭제

- 원소 삭제 방법

- ① 원소 삭제하기

- ② 삭제한 자리 이후의 원소들을 한자리씩 앞으로 이동 시키기



예제 3-3: 배열 원소의 삭제

- deleteElem() 함수를 만들어보자.

```
#include <stdio.h>
#define MAX_LENGTH 100

int deleteElem(int a[], int size, int index) { . . . }

void main() {
    int list[MAX_LENGTH];
    int size = 5;

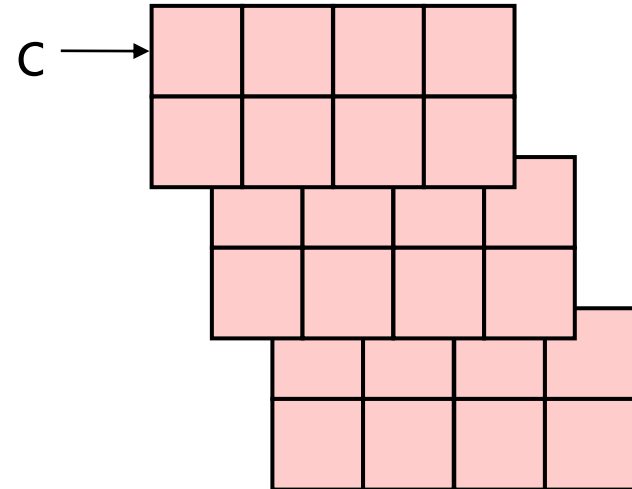
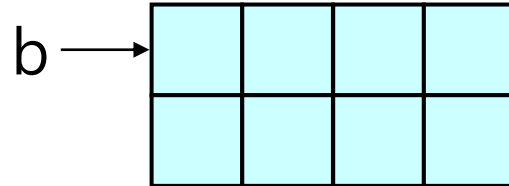
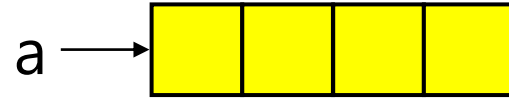
    for (int i = 0; i < size; i++)
        list[i] = 10 + i * 10;

    printf("deleteElem() Test\n");
    printArray("list", list, size);
    size = deleteElem(list, size, 2);
    printArray("list", list, size);
    size = deleteElem(list, size, 3);
    printArray("list", list, size);
    size = deleteElem(list, size, 0);
    printArray("list", list, size);
}
```

```
deleteElem() Test
list[5] = { 10, 20, 30, 40, 50 }
list[4] = { 10, 20, 40, 50 }
list[3] = { 10, 20, 40 }
list[2] = { 20, 40 }
```

다차원 배열

- `int a[4];`
 - 일차원 정수 배열
 - 4개의 정수변수로 구성
- `int b[2][4];`
 - 이차원 정수 배열
 - 2개의 행과 4개의 열로 구성
- `int c[3][2][4];`
 - 삼차원 정수 배열
 - 3 x 2 x 4 개의 정수로 구성



3.3 포인터(Pointer)

- 포인터: 주소를 값으로 가지는 변수
 - 변수는 이름, 저장되는 데이터의 자료형과 값으로 구성됨
 - 포인터 변수는 메모리 주소를 데이터로 갖는 변수
- 포인터 변수의 선언과 사용

```
void main() {  
    int a = 3, b = 4;  
    int *p, *q;           // 포인터 변수 선언 (int*)p, (int*)q  
  
    p = &a;               // 변수a의 주소  
    q = p;  
    b = *q;               // 포인터 변수q의 값(주소)에 저장된 값  
    printf("%d %d\n", a, b);  
}
```

예제 3-4: 포인터 변수

```
#include <stdio.h>

void printVars(char *title, int *a, int *b, int **p, int **q)
{
    printf("%s\n", title);
    printf("a %X: %d\n", (int)a, (int)*a);
    printf("b %X: %d\n", (int)b, (int)*b);
    printf("p %X: %X\n", (int)p, (int)*p);
    printf("q %X: %X\n", (int)q, (int)*q);
    printf("\n");
}

void main() {
    int a = 3, b = 4;
    int *p = NULL, *q = NULL;

    printVars("a = 3, b = 4;", &a, &b, &p, &q);
    p = &a;      // 변수a의 주소
    printVars("p = &a;", &a, &b, &p, &q);
    q = p;        // 포인터 변수p의 값(변수 a의 주소)
    printVars("q = p;", &a, &b, &p, &q);
    b = *q;       // 포인터 변수q의 값(변수 a의 주소)에 저장된 값
    printVars("b = *q;", &a, &b, &p, &q);
}
```

```
a = 3, b = 4;
a 4FF9F0: 3
b 4FF9E4: 4
p 4FF9D8: 0
q 4FF9CC: 0
```

3.4 구조체 (Structure)

- 구조체
 - 여러 개의 변수를 하나의 그룹으로 묶어주는 방법
 - 구조체 안에 있는 변수를 구조체의 멤버라 한다.
- 구조체 정의
 - 예약어 struct를 사용하여 구조체 자료형 정의

```
struct name_card {  
    char  name[20];  
    char  phone[20];  
    int   birthday;  
};  
  
struct name_card card1;
```

구조체 정의

- typedef으로 구조체 자료형 단순화
 - typedef는 새로운 이름의 자료형을 정의하는 기능
 - 구조체 자료형 이름을 한 단어로 단순화 할 수 있음

```
struct name_card {  
    char    name[20];  
    char    phone[20];  
    int     birthday;  
};  
  
typedef struct name_card CardType;  
  
CardType  card2;
```

```
typedef struct name_card {  
    char    name[20];  
    char    phone[20];  
    int     birthday;  
} CardType;  
  
CardType  card3;
```

구조체 변수

- 구조체 변수 선언
 - 구조체 자료형의 변수 선언
- 구조체 변수 사용
 - 구조체 변수명에서 '.' 연산자로 멤버 참조

```
CardType c1 = { "홍길동", "010-123-4567", 710101 };  
CardType c2;  
  
strcpy(c2.name, "임꺽정");  
strcpy(c2.phone, "010-888-9999");  
c2.birthday = 750308;
```


예제 3-5: 구조체 사용

```
#include <stdio.h>

typedef struct name_card {
    char  name[20];
    char  phone[20];
    int    birthday;
} CardType;

void printCard(CardType c) {
    printf("%s, %s, %d\n", c.name, c.phone, c.birthday);
}

void main() {
    CardType c1 = { "홍길동", "010-123-4567", 710101 };
    CardType c2, c3;

    strcpy(c2.name, "임꺽정");
    strcpy(c2.phone, "010-888-9999");
    c2.birthday = 750308;

    c3 = c1;
    strcpy(c3.phone, "010-8765-4321");

    printCard(c1);
    printCard(c2);
    printCard(c3);
}
```

홍길동, 010-123-4567, 710101
임꺽정, 010-888-9999, 750308
홍길동, 010-8765-4321, 710101

구조체 포인터

- 구조체 포인터 변수 사용
 - 구조체 포인터의 변수명에서는 '->' 연산자로 멤버 참조

```
CardType c1 = { "홍길동", "010-123-4567", 710101 };  
CardType c2 = { "임꺽정", "010-888-9999", 750308 };  
CardType *pc3;  
  
pc3 = &c1;  
strcpy(pc3->phone, "010-8765-4521");
```

예제 3-6: 구조체 포인터

```
#include <stdio.h>

typedef struct name_card {
    char name[20];
    char phone[20];
    int birthday;
} CardType;

void printCard(CardType c) {
    printf("%s, %s, %d\n", c.name, c.phone, c.birthday);
}

void main() {
    CardType c1 = { "홍길동", "010-123-4567", 710101 };
    CardType c2 = { "임꺽정", "010-888-9999", 750308 };
    CardType *pc3;

    pc3 = &c1;
    strcpy(pc3->phone, "010-8765-4521");

    printCard(c1);
    printCard(c2);
    printCard(*pc3);
}
```

홍길동, 010-8765-4521, 710101
임꺽정, 010-888-9999, 750308
홍길동, 010-8765-4521, 710101

Q&A

