

*1주차 : C Language review*

# 알 고 리 즈

2015. 09. 03

충남대학교 컴퓨터공학과 임베디드 시스템 연구실  
조교 권진세

# Overview

## ▶ 수업 소개

- 1) 담당 조교, 연구실
- 2) 수업 흐름 및 실습 방식
- 3) 강의 소개 + 주차별 실습 과정 (예정)

## ▶ Review : C Language

Structure, **Pointer**, Memory allocation, **File I/O**

## ▶ 실습

# About us

## ▶ 조교 : 권진세

- E-mail : kwonse@cnu.ac.kr
- Phone : 010-3422-2466

## ▶ Lab : 임베디드 시스템 연구실 (공5533)

- 내선번호 : 042-821-7446
- 방문이 필요할 경우 사전 연락 권장

# Actual

## ▶ 교과목 게시판 (<http://cse.cnu.ac.kr>)

- 매주 수업 자료 및 안내사항을 공지

## ▶ 실습 방식

- **MS Visual Studio 2008 & 2010** 사용 (C Language)
- **개인별**로 매주 실습 및 과제를 수행
- **과제 복사 시 적발 시점에 관계없이 0점**
  - Copy : 동일한 코드 소유자 모두 0점
  - 기한 : 차주 실습 시간 전까지 제출
  - 제출방법 : 조교 메일로 프로젝트 폴더 압축 후 제출  
**[알고리즘00반]\_학번\_이름\_실습번호.zip**

# Lecture

## ▶ 알고리즘이란?

– 어떠한 문제를 해결하기 위해 명확하게 정의된 절차 및 방법

## ▶ 전제 조건

- ① 입력 : 외부입력이 존재할 수 있다
- ② 출력 : **1개 이상의 결과값**이 도출되어야 한다
- ③ 명확성 : 각 연산의 의미가 명확해야 한다
- ④ 효과성 : 모든 연산은 **실행 가능**한 것이어야 한다
- ⑤ 유한성 : 일정 횟수의 연산이 이루어진 후 끝나야 한다

# Lecture

## ▶ 주차별 실습 과정 (예정)

주차	실습 주제
1	Introduction : 강의 소개 및 C Language Review
2	Stable Marriage
3	Algorithm analysis : Notation
4	<b>Divide and Conquer</b>
5	Sorting : Insertion, Merge, Quick Sorting
6	<b>Greedy Algorithms</b>
7	Huffman Coding
8	중간고사

# Lecture

## ▶ 주차별 실습 과정 (예정)

주차	실습 주제
9	<b>Dynamic Programming</b> : Longest Common Subsequence (LCS)
10	Fast Fourier Transform (FFT)
11	String Matching
12	Depth-First Search
13	<b>Graph Algorithms</b> : Dijkstra's Algorithm
14	<b>NP-Completeness</b>
15	기말고사

# Review : C Language

## (1) Structure

다양한 타입의 변수를 한데 묶어 **하나의 구조체로 정의**하고  
**정의된 구조체를 선언하여 사용**할 수 있다.

e.g. 1. 일반적인 변수 선언

```
int a = 7;  
char b = 'k';
```

a	7
b	'k'

- ▲ 이미 정의가 되어 있는 기본 data type 이므로  
사용자가 직접 정의하지 않아도 선언만 하여 사용할 수 있음



# Review : C Language

## (1) Structure

다양한 타입의 변수를 한데 묶어 **하나의 구조체로 정의**하고 **정의된 구조체를 선언하여 사용**할 수 있다.

e.g. 2. 구조체의 정의와 선언

```
struct student {  
    int st_num;  
    char name[20];  
};
```

st_num	201500000
name	"홍길동"

```
void main() {  
    struct student st1 = {201500000, "홍길동"};  
}
```

# Review : C Language

## (1) Structure

다양한 타입의 변수를 한데 묶어 **하나의 구조체로 정의**하고  
**정의된 구조체를 선언하여 사용**할 수 있다.

e.g. 3. typedef의 사용

```
typedef struct student {  
    int st_num;  
    char name[20];  
} stu;  
  
void main() {  
    stu st1 = {201500000, "홍길동"};  
}
```

# Review : C Language

## (1) Structure

다양한 타입의 변수를 한데 묶어 **하나의 구조체로 정의**하고  
**정의된 구조체를 선언하여 사용**할 수 있다.

e.g. 4. 구조체에 데이터를 저장하는 다른 방법

```
void main() {  
    struct student st1;  
  
    st1.st_num = 201500000;  
    strcpy_s(st1.name, 20, "홍길동");  
}
```

# Review : C Language

## (2) Pointer

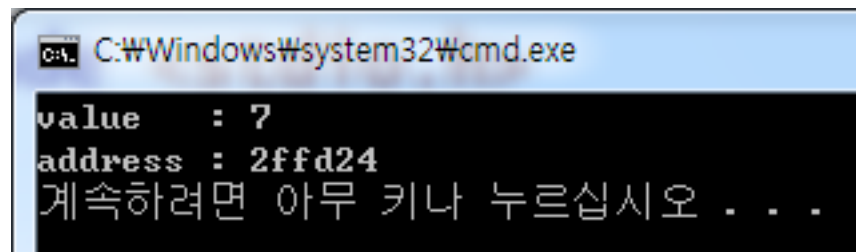
변수를 선언하고 나면, 메모리에 데이터를 저장할 수 있으며  
이때 변수는 데이터를 보존할 **저장 공간**과  
그 저장 공간의 위치(**메모리 주소**) 정보를 필요로 한다.

```
int a = 7;
```

```
printf("value    : %d \n", a);  
printf("address  : %x \n", &a);
```

2ffd24

7



```
C:\Windows\system32\cmd.exe  
value    : 7  
address  : 2ffd24  
계속하려면 아무 키나 누르십시오 . . .
```

# Review : C Language

## (2) Pointer

포인터는 어떤 데이터 값이 아닌 **주소를 저장**하는 변수이며,  
다음과 같이 선언하였을 때,  
**“포인터 변수 p는 변수 a(의 주소)를 가리킨다”**라고 한다.

```
int a = 7;  
int* p = &a;
```

**int\*는 int형 변수의 주소를 저장하는 타입이라 보면 좋음**



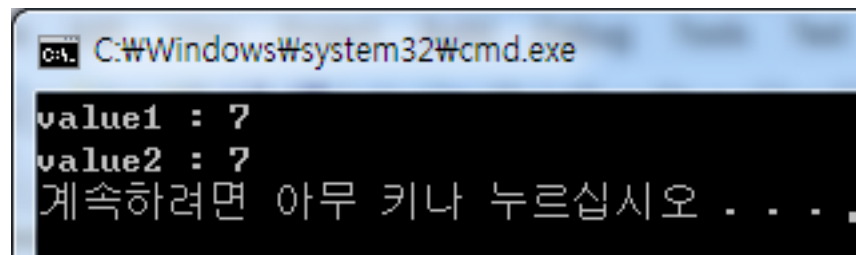
# Review : C Language

## (2) Pointer

포인터 변수가 가리키고 있는 주소에 저장된 값을 불러올 때는 다음과 같이 **기호 \* 또는 [0]**를 사용한다.

```
int a = 7;  
int* p = &a;
```

```
printf("value1 : %d \n", *p);    // *p == i  
printf("value2 : %d \n", p[0]);  // p[0] == i
```



```
C:\Windows\system32\cmd.exe  
value1 : 7  
value2 : 7  
계속하려면 아무 키나 누르십시오 . . .
```

# Review : C Language

## (3) Memory allocation

다음과 같이 **포인터 변수를 먼저 선언**하고,  
그 다음에 사용할 저장 공간을 할당할 수 있다.

```
int* p = NULL;  
p = (int*)malloc(sizeof(int)*ARRAY_SIZE);  
  
...  
  
free(p);
```

**변수 사용이 끝난 뒤에는 반드시 메모리 할당을 해제한다.**

# Review : C Language

## (3) Memory allocation

포인터를 이용하여 구조체를 선언할 경우,  
다음과 같이 메모리 할당을 한다.

```
// typedef을 사용한 경우
stu* s1 = (stu*)malloc(sizeof(stu));

...

free(s1);
```

단, 구조체의 member 사용시 `s1.name`이 아니라  
`s1->name` 또는 `(*s1).name`과 같이 사용하여야 한다.



# Review : C Language

## (4) File I/O

① 파일 입출력 변수 선언과 파일 열기

```
#define FILENAME "data.txt"
```

```
FILE* fp;
```

```
fp = fopen(FILENAME, "rt");    // 쓰기는 "wt"
```

```
// 파일 열기/생성에 실패했을 경우 프로그램 종료
```

```
if (fp == NULL) {  
    printf("**** File open error ****\n");  
    exit(1);  
}
```

# Review : C Language

## (4) File I/O

② 파일에 기록된 문자 수(bytes)만큼 메모리 할당

```
int len;  
char* data;
```

```
// 단위는 byte  
fseek(fp, 0, SEEK_END);    // 파일의 끝 + 0으로 이동  
len = ftell(fp);           // 현재 위치를 저장  
fseek(fp, 0, SEEK_SET);    // 파일 시작 + 0으로 이동
```

```
data = (char*)malloc(sizeof(char)*len);
```

# Review : C Language

## (4) File I/O

③ 파일의 끝(EOF)에 도달할 때까지 파일을 읽기

```
int cnt = 0;

while(!feof(fp)) {
    fscanf(fp, "%c", &data[cnt++]);
}
```

④ 더 이상 사용하지 않는 파일 닫기

```
fclose(fp);
```

# Review : C Language

## (4) File I/O

- ⑤ "wt"로 파일을 열거나 생성했을 경우  
배열의 끝에 도달할 때까지 파일 쓰기

```
int cnt = 0;
int len = _msize(data) / sizeof(*data);

for(cnt=0; cnt<len; cnt++) {
    fprintf(fp, "%c", data[cnt++]);
}
```

# Review : C Language

## (4) File I/O

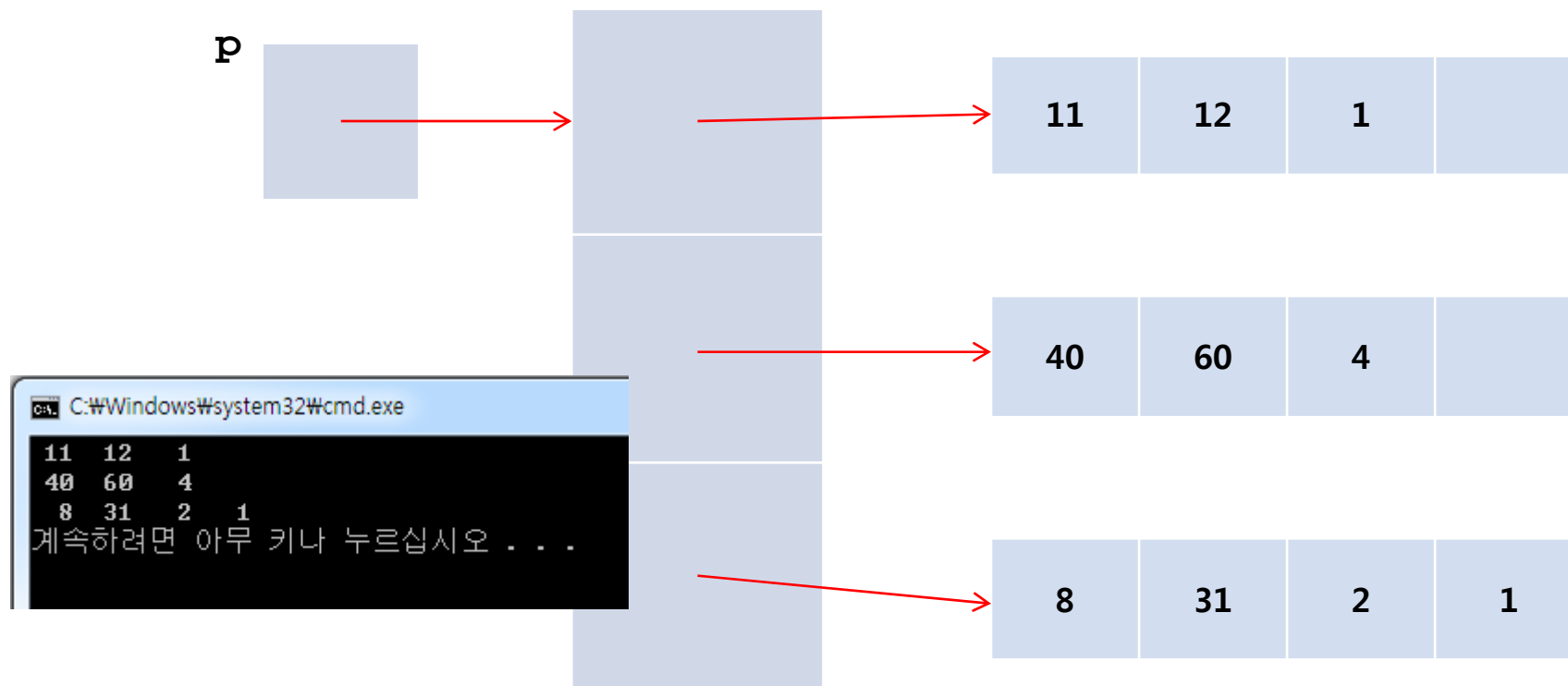
⑥ 파일을 읽을 때마다 메모리를 재할당 하는 방법

```
int* data = (int*)malloc(sizeof(int));
int cnt = 0;

while(!feof(fp)) {
    data = (int*)realloc(data, sizeof(int)*(cnt+1));
    fscanf(fp, "%d", &data[cnt++]);
}
```

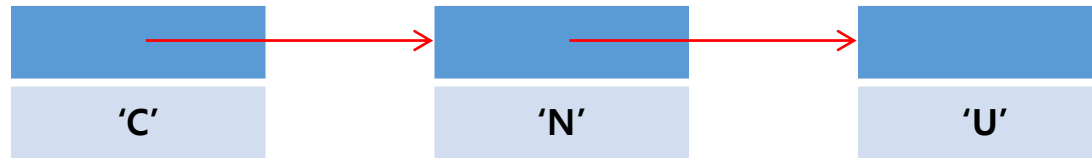
# Practice (1)

1. 포인터와 동적 할당을 사용하여  
다음과 같은 저장 공간을 생성할 수 있는가?



# Practice (2)

2.포인터와 동적 할당을 사용하여  
다음과 같이 연결 가능한 구조체(linked list)를 생성할 수 있는가?



```
C:\ C:\Windows\system32\cmd.exe
```

```
The List Contains : CNU  
계속하려면 아무 키나 누르십시오 . . .
```

- 포함내용 : 프로젝트 파일 전체 압축
- 제출이름 : [알고리즘00반]\_201500000\_홍길동\_1주차.zip
- 제출기한 : 2015-09-09 18:00까지
- 메일주소 : [kwonse@cnu.ac.kr](mailto:kwonse@cnu.ac.kr)