

# 실습 1주차

C, JAVA review

2016.09.01

# Contents

1. 수업 소개
2. C review
3. JAVA review
4. 실습

# 수업 소개

# 수업 소개

## ■ 알고리즘 (25155-01)

### ◆ 담당 조교 : 이정진

- E-mail: [like\\_jj@naver.com](mailto:like_jj@naver.com)
- Phone: 010-2551-1691
- 연구실: 분산이동컴퓨팅 연구실(공5628)

### ◆ 문의 사항 있을 시 E-mail

### ◆ 방문이 필요할 경우 사전 연락 권장

### ◆ 실습 수업 시간

- 목요일 18:00~20:00 (공5414)

### ◆ 교과목 게시판

- [http://computer.cnu.ac.kr/index.php?mid=ug\\_16\\_2\\_3\\_5](http://computer.cnu.ac.kr/index.php?mid=ug_16_2_3_5)
- 매주 수업 자료 및 안내사항 공지

# 수업 소개

## ■ 실습 방식

- ◆ MS Visual Studio 2010 사용(C Language)
- ◆ Eclipse 사용(JAVA Language)
- ◆ 개인별로 매주 실습 및 과제를 수행
- ◆ 과제 copy 적발 시 양쪽 모두 0점
  - 나중에 적발되어도 0점

# 수업 소개

## ■ 주차별 실습 계획(예정)

주차	실습 주제
1	C, JAVA review
2	Stable Marriage
3	Algorithm analysis
4	Divide and Conquer
5	Sorting
6	Greedy Algorithms
7	Huffman Coding
8	중간고사
9	Dynamic Programming
10	Fast Fourier Transform
11	String Matching
12	Depth-First Search
13	Graph Algorithms
14	NP-Completeness
15	기말고사

# 수업 소개

## ■ 알고리즘이란?

- ◆ 어떠한 문제를 해결하기 위해 명확하게 정의된 절차 및 방법

## ■ 전제 조건

- ◆ 입력: 외부에서 제공되는 자료가 있을 수 있다.
- ◆ 출력: 적어도 한 가지 결과가 생긴다.
- ◆ 명백성: 각 명령들은 명백해야 한다.
- ◆ 유한성: 한정된 단계를 처리한 후에 종료된다.
- ◆ 효과성: 모든 명령들은 명백하고 실행 가능한 것이어야 한다.

# C REVIEW



# C Language

## ■ Structure

- ◆ 다양한 타입의 변수를 한데 묶어 하나의 구조체로 정의하고 정의된 구조체를 선언하여 사용할 수 있다.

- ◆ 일반적인 변수 선언

```
int a = 7;
```

a	7
---	---

```
char b = 'k' ;
```

b	'k'
---	-----

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

# C Language

## ■ Structure

- ◆ 다양한 타입의 변수를 한데 묶어 하나의 구조체로 정의하고 정의된 구조체를 선언하여 사용할 수 있다.

- ◆ 구조체의 정의와 선언

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

```
void main() {  
    struct student st1 = {201600000, "홍길동"};  
}
```

st_num	201600000
name	"홍길동"

# C Language

## ■ Structure

- ◆ 다양한 타입의 변수를 한데 묶어 하나의 구조체로 정의하고 정의된 구조체를 선언하여 사용할 수 있다.

- ◆ typedef의 사용

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

```
void main() {  
    stu st1 = {201600000, "홍길동"};  
}
```

st_num	201600000
name	"홍길동"

# C Language

## ■ Structure

- ◆ 다양한 타입의 변수를 한데 묶어 하나의 구조체로 정의하고 정의된 구조체를 선언하여 사용할 수 있다.

- ◆ 구조체에 데이터를 저장하는 다른 방법

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

```
void main() {  
    stu st1;  
    st1.st_num = 201600000;  
    strcpy_s(st1.name, 20, "홍길동");  
}
```

st_num	201600000
name	"홍길동"

# C Language

## ■ Pointer

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

```
int a = 7;
```

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

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

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

```
value    : 7
address  : 68d8fa70
계속하려면 아무 키나 누르십시오 . . .
```

68d8fa70

7

# C Language

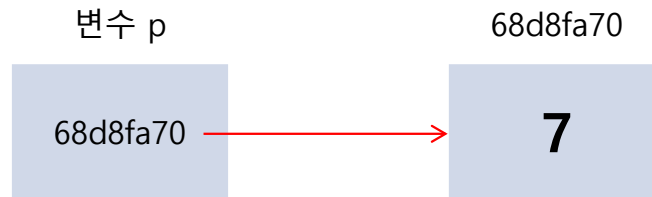
## ■ Pointer

- ◆ 포인터는 어떤 데이터 값이 아닌 주소를 저장하는 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



# C Language

## ■ Pointer

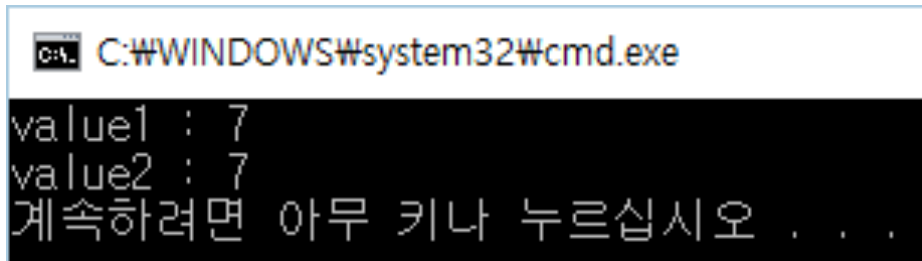
- ◆ 포인터 변수가 가리키고 있는 주소에 저장된 값을 불러올 때는 다음과 같이 기호 \* 또는 [0]를 사용한다.

```
int a = 7;
```

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

```
printf("value1 : %d \n", *p);    // *p == a
```

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



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

# C Language

## ■ Memory allocation

- ◆ 다음과 같이 포인터 변수를 먼저 선언하고, 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.



# C Language

## ■ Memory allocation

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

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

...

free(s1) ;
```

- ◆ 단, 구조체의 member 사용시 s1.name이 아니라 s1->name 또는 (\*s1).name과 같이 사용하여야 한다.

# C Language

## ■ File I/O

### ◆ 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);
}
```

# C Language

## ■ File I/O

- ◆ 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);
```

# C Language

## ■ File I/O

- ◆ 파일의 끝(EOF)에 도달할 때까지 파일을 읽기

```
int cnt = 0;
```

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

- ◆ 더 이상 사용하지 않는 파일 닫기

```
fclose(fp);
```

# C Language

## ■ File I/O

- ◆ “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++]);
}
```

# C Language

## ■ File I/O

### ◆ 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++]);  
}
```

# JAVA REVIEW

# JAVA Language

## ■ Class

- ◆ 객체의 행동과 특성 규정짓기 위한 틀로써 기본형과 구별되는 새로운 데이터형으로 인식
- ◆ 멤버: 객체의 특성을 나타내는 변수
- ◆ 메소드: 행동을 나타내는 함수
- ◆ 생성자: 생성한 클래스의 이름과 똑같은 메소드



# JAVA Language

## ■ Class

- ◆ 객체에서 사용할 변수 선언(클래스 멤버)

```
public class student {  
    Private int studentNum;  
    Private String name;  
  
}
```

# JAVA Language

## ■ Class

- ◆ 객체에서 행동을 나타내는 함수 선언(클래스 메소드)

```
public int getStudentNum() {  
    return studentNum;  
}  
  
public String getName() {  
    return name;  
}
```

## ■ 생성자

- ◆ 클래스의 이름과 동일한 이름을 가지는 특별한 메소드를 생성자라 명명한다.
- ◆ 객체가 생성될 때 호출되며 초기 멤버의 값을 지정해주는 용도로 사용한다.

- ◆ 일반적인 객체 생성(생성자가 없을 때)

```
student stu1 = new student();
```

studentNum	-
name	-

- ◆ 생성자가 선언 되어있지 않은 경우 객체의 멤버가 초기화 되지 않는다.

# JAVA Language

## ■ 생성자

### ◆ 생성자를 이용한 객체 생성

```
public class student {  
    Private int studentNum;  
    Private String name;  
    public student(int studentNum,String name){  
        this.studentNum=studentNum;  
        this.name=name;  
    }  
}
```

studentNum	201600000
------------	-----------

name	홍길동
------	-----

```
student stu1 = new student(201600000, "홍길동");
```

### ◆ 생성자를 이용하여 객체의 멤버를 초기화할 수 있다.

# JAVA Language

## ■ 접근 제어자

- ◆ 변수나 메소드를 선언 할 때 접근 제어를 설정할 수 있다.

	modifier	설명
접근 권한	public	모든 클래스에서 접근이 가능함
	protected	동일 패키지에 속하는 클래스와 하위 클래스 관계의 클래스에 의해 접근 가능
	private	클래스 내에서만 접근이 가능하다

# JAVA Language

## ■ 접근 제어자

종류	클래스	하위 클래스	동일 패키지	모든 클래스
private	○	X	X	X
(default)	○	X	○	X
protected	○	○	○	X
public	○	○	○	○

# JAVA Language

## ■ 메소드 호출

- ◆ Public으로 정의된 메소드 호출

- ◆ "." 연산자 이용

```
System.out.println(stu1.getStudentNum());
```

```
System.out.println(stu1.getName());
```

# JAVA Language

## ■ File I/O

### ◆ 파일 입출력 변수 선언과 파일 열기

- Byte 단위
  - ✓ InputStream, FileInputStream, BufferedInputStream
  - ✓ OutputStream, FileOutputStream, BufferedOutputStream
- 문자 단위
  - ✓ Reader, FileReader, BufferedReader
  - ✓ Writer, FileWriter, BufferedWriter

```
BufferedReader reader=new BufferedReader(new  
FileReader("data.txt"));
```

```
FileWriter writer=new FileWriter("data.txt");
```



# JAVA Language

## ■ File I/O

### ◆ Byte 단위로 파일 읽기

```
FileInputStream fileStream = null;  
fileStream = new FileInputStream("data.txt");  
int c=0;  
while((c = fileStream.read()) != -1)  
{  
    System.out.print((char)c);  
}
```

### ◆ 한 바이트씩 읽으므로 한글과 같은 2바이트 문자는 정상적으로 사용 못함

# JAVA Language

## ■ File I/O

### ◆ 문자 단위로 파일 읽기

```
FileReader reader=new FileReader("data.txt");  
int c=0;  
while ((c=reader.read()) != -1) {  
    System.out.print((char)c);  
}
```

# JAVA Language

## ■ File I/O

### ◆ 한 줄씩 파일 읽기

```
BufferedReader reader=new BufferedReader(new
                                   FileReader("data.txt"));

String str;
while((str=reader.readLine()) != null){
    System.out.println(str);
}
```

### ◆ 더 이상 사용하지 않는 파일 닫기

```
reader.close();
```

### ◆ JAVA 에서 close()는 생략가능

# JAVA Language

## ■ File I/O

### ◆ 파일 쓰기

```
FileWriter writer=new FileWriter("data.txt");  
String data="Algorithm Class";  
writer.write(data);  
writer.close();
```

# Q&A

Thank you!!