Object Oriented Programming (Week 7)

2023

KWANGWOON UNIVERSITY
DEPT. OF COMPUTER ENGINEERING

Contents

ASSIGNMENT 2-3. 1

Assignment 2-3. 1

- **(Employee Class)** Implement a Class Employee with the members, as shown in the figure below, and create a program that performs actions based on the commands (insert, find, change, print, exit). The program should be able to store information of up to 10 employees.

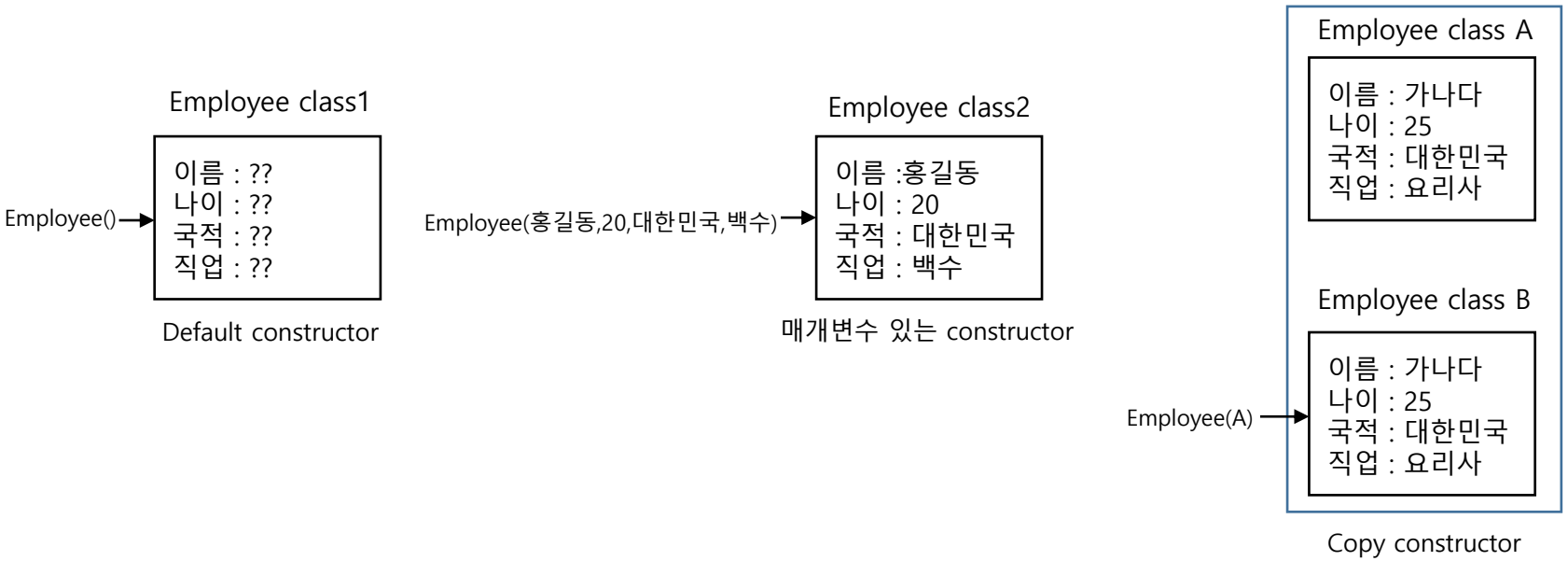
```
class Empolyee
{
private:
    char *name;
    int age;
    char *country;
    char *job;

public:
    Empolyee(char* name, int age, char* country, char* job);
    bool isNameCorrect(char* name);
    void print();
    void change(char* name, int age, char* country, char* job);
};
```

Command	Format	Description
insert	insert [name] [age] [country] [job]	Add a new Employee object that stores the input information as its member variables through a constructor.
find	find [name]	If an Employee is in the list with the same name as the input [name], print out its information.
change	change [name1] [name2] [age] [country] [job]	Change the information of an Employee in the list with the name [name1] to the input [name2], [age], [country], and [job].
print	-	Print out the information of all stored Employees.
exit	-	Exit the program.

Assignment 2-3. 1

- Constructor : 클래스 객체가 생성 될 때 객체를 초기화하는 목적으로 실행하는 함수
  - Constructor 종류
    - Default constructor
    - 매개변수 있는 constructor
    - Copy constructor

```
Empolyee();
Empolyee(char* name, int age, char* country, char* job);
Empolyee(Empolyee& A)
```



Assignment 2-3. 1

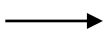
Input	Output
insert John 23 USA designer insert James 28 Korea developer insert Jessica 27 Japan manager print find James change James Jason 22 China writer print exit	=====print===== Name: John Age: 23 Country: USA Job: designer ---------- Name: James Age: 28 Country: Korea Job: developer ---------- Name: Jessica Age: 27 Country: Japan Job: manager ---------- =====find===== Name: James Age: 28 Country: Korea Job: developer ---------- =====print===== Name: John Age: 23 Country: USA Job: designer ---------- Name: Jason Age: 22 Country: China Job: writer ---------- Name: Jessica Age: 27 Country: Japan Job: manager ----------

이름 : John
나이 : 23
국적 : USA
직업 : designer

이름 : James
나이 : 28
국적 : Korea
직업 : developer

이름 : Jessica
나이 : 27
국적 : Japan
직업 : manager

이름 : James
나이 : 28
국적 : Korea
직업 : developer



이름 : James
나이 : 22
국적 : China
직업 : writer

ASSIGNMENT 2-3. 2

Assignment 2-3. 2

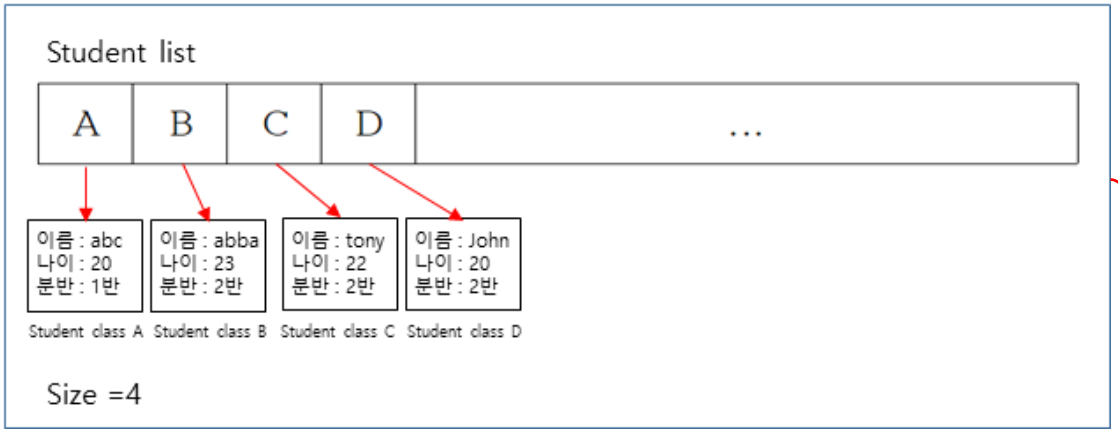
- Implement a program that stores and prints student information. To store student information, implement the Student Class below, and implement the School Class, which aims to manage multiple students' information. The Student Class should not be accessible outside of the School Class. Each class can only have the member variables shown in the figure below, and member functions can be implemented freely. The program should support four commands: new_student, sort_by_name, print_all, print_class, and exit.

```
class Student
{
private:
    char* name;
    int age;
    char* class_name;
};

class School
{
private:
    class Student* student_list[100];
    int size = 0;
};
```

command	format	description
new_student	new_student [name] [age] [class_name]	Store the information of a new student in the School class.
sort_by_name	-	Sort the student objects by name in ascending order.
print_all	-	Print information of all students stored in the School class.
print_class	print_class [class_name]	Print the information of students whose class_name is identical to [class_name]. Then print the number of the printed students.
exit	-	Terminate program.

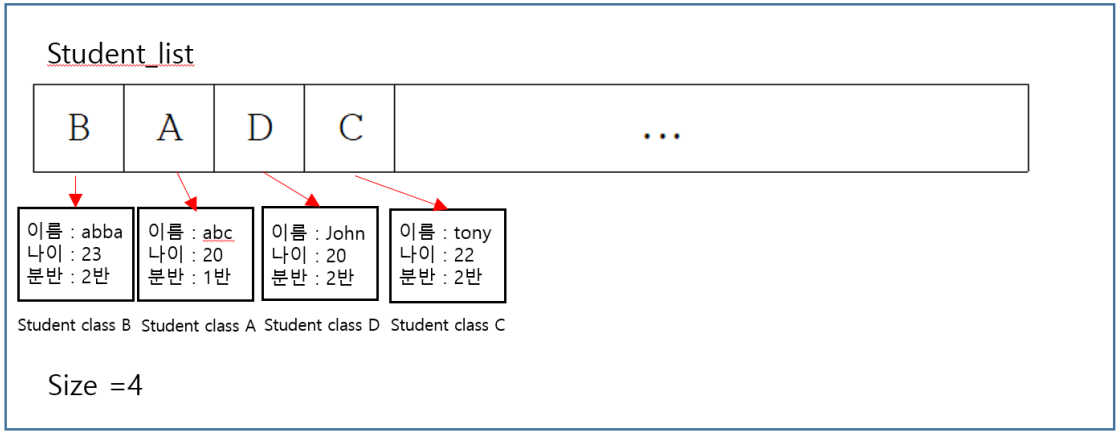
Assignment 2-3. 2

School class A



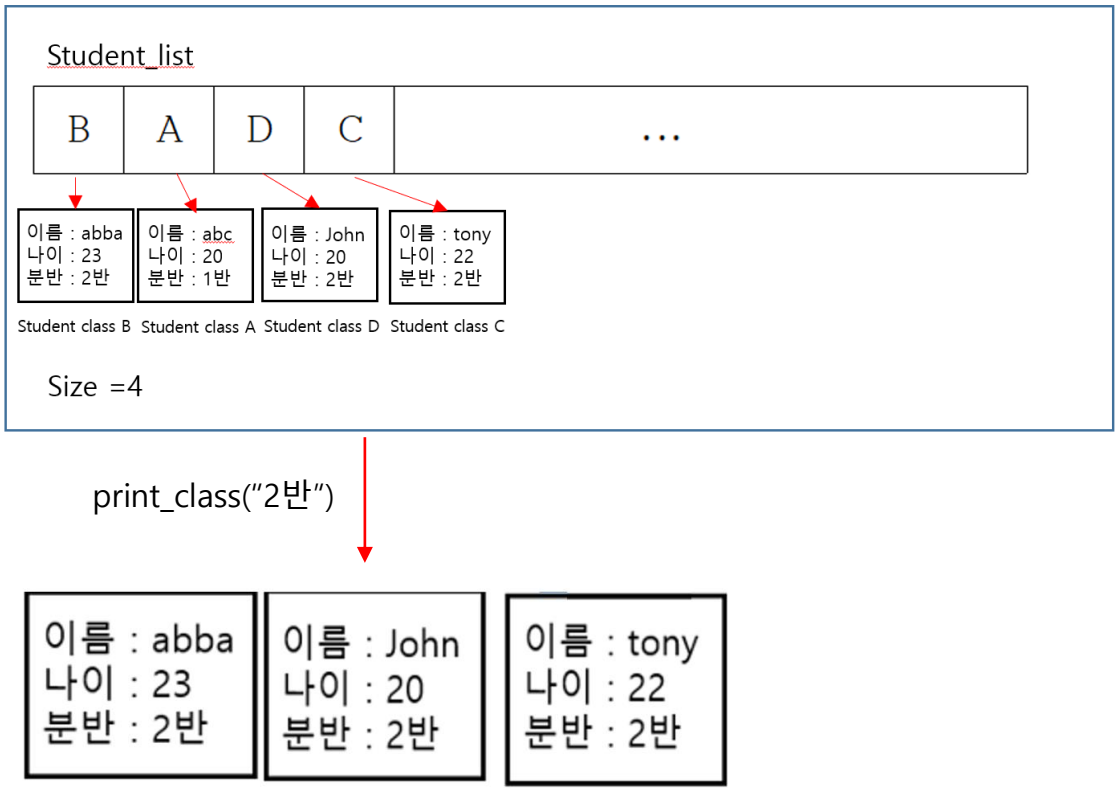
Sort_by_name()

School class A



Assignment 2-3. 2

School class A



Assignment 2-3. 2

Input	Output
new_student Jim 8 Hope new_student Alice 9 Love new_student Claude 8 Hope new_student Megan 11 Wish print_all print_class Hope sort_by_name print_all exit	=====print_all===== Name: Jim Age: 8 Class: Hope ---------- Name: Alice Age: 9 Class: Love ---------- Name: Claude Age: 8 Class: Hope ---------- Name: Megan Age: 11 Class: Wish ---------- =====print_class===== Name: Jim Age: 8 Class: Hope ---------- Name: Claude Age: 8 Class: Hope ---------- Number of classmates : 2 =====print_all===== Name: Alice Age: 9 Class: Love ---------- Name: Claude Age: 8 Class: Hope ---------- Name: Jim Age: 8 Class: Hope ---------- Name: Megan Age: 11 Class: Wish ----------

ASSIGNMENT 2-3. 3

Assignment 2-3. 3

- **(String Class)** Implement a simple string class which is a simplified version of std::string based on given class definitions. When you need any member variables or member functions which are not defined in the class definition, feel free to add them to your implementation. You can see the C++ documentation to understand the detailed behavior.

C++ string class documentation : https://cplusplus.com/reference/string/string/

Assignment 2-3. 3

```
namespace oopstd {


class string
{
public:
  string();
  string(const char* s);
  string(const string& str);
  ~string();

  string& assign(const string& str);
  char& at (size_t pos);
  const char* c_str() const;
  void clear();
  void push_back(char c);
  int compare(const string* str) const;
  string& replace(size_t pose, size_t len, const string* str);
  string substr(size_t pos, size_t len) const;
  size_t find(const string& str, size_t pos) const;

private:
  char* s;
  size_t len;

};

int stoi(const string& str, size_t* idx, int base);
float stof(const string& str, size_t* idx);

string to_string(int val);
string to_string(float val);
}
```

Assignment 2-3. 3

- **Constructor**
  - string()
  - string(const char* s)
  - string(const string& str)
    - Constructs a string object, initializing its value depending on the constructor version used

- **Destructor**
  - ~string()
    - Destroys the string object

Assignment 2-3. 3

- **assign**
  - string& assign(const string& str);
    - Assigns a new value to the string, replacing its current contents
  - Return value
    - *this

```
a.assign("HI");
cout << a << endl; // 결과 : "HI"
```

- **at**
  - char& at(size_t pos);
    - Returns a reference to the character at position *pos* in the string
  - Return value
    - The character at the specified position in the string.

```
string example = "Hello World";
cout << example.at(2) << endl; // 결과 : "l"
```

Assignment 2-3. 3

- **c_str**
  - const char* c_str() const;
    - Returns a pointer to an array that contains a **null-terminated sequence of characters (i.e., a C-string)** representing the current value of the string object
  - Return value
    - A pointer to the **c-string** representation of the string object's value.

```
b = example.c_str();
cout << b << endl; // 결과 : "Hello World"
```

- **clear**
  - void clear();
    - Erases the contents of the string, which becomes an empty string
  - Return value
    - none

```
example.clear();
cout << example << endl;
```

Assignment 2-3. 3

- **push_back**
  - void push_back(char c);
    - Appends character *c* to the end of the string, increasing its length by one.
  - Return value

```
example.push_back('b');
cout << example << endl; // 결과 : "Hellow Worldb"
```

- **compare**
  - int compare(const string* str) const;
    - Compares the value of the string object (or a substring) to the sequence of characters specified by its arguments.
  - Return value

value	relation between *compared string* and *comparing string*
0	They compare equal
<0	Either the value of the first character that does not match is lower in the ***compared string***, or all compared characters match but the ***compared string*** is shorter.
>0	Either the value of the first character that does not match is greater in the ***compared string***, or all compared characters match but the ***compared string*** is longer.

Assignment 2-3. 3

- **replace**
  - string& replace(size_t pose, size_t len, const string* str);
    - Replaces the portion of the string that begins at character *pos* and spans *len* characters
  - Return value
    - *this

```
d = example.replace(0, 5, "Hi");
cout << d << endl; // 결과 : "Hi World"
```

- **substr**
  - string substr(size_t pose, size_t len) const;
    - Returns a newly constructed string object with its value initialized to a copy of a substring of this object.
  - Return value
    - A string object with a substring of this object.

```
c = example.substr(5, 10);
cout << c << endl; // 결과 :" World"
```

Assignment 2-3. 3

- **find**
  - size_t find(const string& str, size_t pos) const;
    - Searches the string for the first occurrence of the sequence specified by its arguments.
  - Return value
    - The position of the first character of the first match.
      If no matches were found, the function returns string::npos.

```
cout << example.find("World") << endl; // 결과 : 6
```

Assignment 2-3. 3

- **stoi**
  - int stoi(const string& str, size_t* idx, int base = 10);
    - Parses str interpreting its content as an integral number of the specified base, which is returned as an int value
    - If *idx* is not a null pointer, the function also sets the value of *idx* to the position of the first character in *str* after the number
  - Return value
    - On success, the function returns the converted integral number as an int value

```
string number2 = "10bus";
size_t sz;
int num = stoi(number2, &sz);
cout << num << endl; // 결과 : 10
cout << sz << endl; // 결과 : 2
```

Assignment 2-3. 3

- **stof**
  - float stof(const string& str, size_t* idx) const;
    - Parses str interpreting its content as a floating-point number, which is returned as value of type float
  - Return value
    - On success, the function returns the converted floating-point number as a value of type float

```
string float_number= "1.5abc";
float float_num = stof(float_number, &sz);
cout << float_num << endl; // 결과 : 1.5
cout << sz << endl; // 결과 : 3
```

- **to_string**
  - ① string to_string (int val); ② string to_string (float val);
    - Returns a string with the representation of *val*
  - Return value
    - A string object containing the representation of *val* as a sequence of characters

ASSIGNMENT 2-3. 4

Assignment 2-3. 4

- **(Matrix Class)** Define a Matrix Class consisting of 2D pointer(data) and the number of rows(rows) and columns(cols) and then implement functions of its operations:

  Matrix - Matrix Operation: Addition, Subtraction, Multiplication

  Matrix - Scalar Operation: Addition, Subtraction, Multiplication, Division

  Matrix Operation: Transpose, Adjoint, Inverse

  There are no restrictions on implementation of matrix operation.

Assignment 2-3. 4

- Matrix - Matrix Operation: Addition, Subtraction, Multiplication

2	2	3
1	1	2
1	2	3

Matrix **A**

1	2	3
4	5	6
7	8	9

Matrix **B**

A + B =

3	4	6
5	6	8
8	10	12

- a11 = 2*1 + 2*4 + 3*7 = 31
- a12 = 2*2 + 2*5 + 3*8 = 38
- a13 = 2*3 + 2*6 + 3*9 = 45
- a21 = 1*1 + 1*4 + 2*7 = 19
- a22 = 1*2 + 1*5 + 2*8 = 23
- a23 = 1*3 + 1*6 + 2*9 = 27
- a31 = 1*1 + 2*4 + 3*7 = 30
- a32 = 1*2 + 2*5 + 3*8 = 36
- a33 = 1*3 + 2*6 + 3*9 = 42

A - B =

1	0	0
-3	-4	-4
-6	-6	-6

A X B =

a11	a12	a13
a21	a22	a23
a31	a32	a33

Assignment 2-3. 4

- Matrix - Scalar Operation: Addition, Subtraction, Multiplication, Division

1	2	3
4	5	6
7	8	9

Matrix **A**

A + 4=

5	6	7
8	9	10
11	12	13

A * 2=

2	4	6
8	10	12
14	16	18

A - 2=

-1	0	1
2	3	4
5	6	7

A / 2=

0.5	1	1.5
2	2.5	3
3.5	4	4.5

- Matrix Operation: Transpose

1	2	3
4	5	6
7	8	9

Matrix **A**

1	4	7
2	5	8
3	6	9

Matrix $\mathbf{A}^T$

aij = aji

Assignment 2-3. 4

- Matrix Operation: Determinant

$$D = \begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix}$$

$$= a_{11}\begin{vmatrix} a_{22} & a_{23} \\ a_{32} & a_{33} \end{vmatrix} - a_{12}\begin{vmatrix} a_{21} & a_{23} \\ a_{31} & a_{33} \end{vmatrix} + a_{13}\begin{vmatrix} a_{21} & a_{22} \\ a_{31} & a_{32} \end{vmatrix}$$

$$= a_{11}(a_{22}a_{33} - a_{23}a_{32}) - a_{12}(a_{21}a_{33} - a_{23}a_{31}) + a_{13}(a_{21}a_{32} - a_{22}a_{31})$$

3x3 matrix Determinant

$$\begin{vmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{vmatrix} = 1 * \begin{vmatrix} 5 & 6 \\ 8 & 9 \end{vmatrix} \text{ -2 } * \begin{vmatrix} 4 & 6 \\ 7 & 9 \end{vmatrix} + 3 * \begin{vmatrix} 4 & 5 \\ 7 & 8 \end{vmatrix}$$

det **A**

= 1*(-3) + -2*(-6) + 3*(-3) = 0

2x2 matrix Determinant

$$\begin{vmatrix} a & b \\ c & d \end{vmatrix} = ad\text{-}bc$$

Assignment 2-3. 4

- Matrix Operation: Adjoint matrix

1	4	1
2	3	1
0	2	1

Matrix **A**

adj **A** =

$$\begin{pmatrix} + \begin{vmatrix} 3 & 1 \\ 2 & 1 \end{vmatrix} & - \begin{vmatrix} 2 & 1 \\ 0 & 1 \end{vmatrix} & + \begin{vmatrix} 2 & 3 \\ 0 & 2 \end{vmatrix} \\ - \begin{vmatrix} 4 & 1 \\ 2 & 1 \end{vmatrix} & + \begin{vmatrix} 1 & 1 \\ 0 & 1 \end{vmatrix} & - \begin{vmatrix} 1 & 4 \\ 0 & 2 \end{vmatrix} \\ + \begin{vmatrix} 4 & 1 \\ 3 & 1 \end{vmatrix} & - \begin{vmatrix} 1 & 1 \\ 2 & 1 \end{vmatrix} & + \begin{vmatrix} 1 & 4 \\ 2 & 3 \end{vmatrix} \end{pmatrix}^T$$

$$= \begin{pmatrix} 1 & -2 & 4 \\ -2 & 1 & -2 \\ 1 & 1 & -5 \end{pmatrix}^T = \begin{pmatrix} 1 & -2 & 1 \\ -2 & 1 & 1 \\ 4 & -2 & -5 \end{pmatrix}$$

Assignment 2-3. 4

- Matrix Operation: Inverse matrix

$$\mathbf{A}^{-1} = \frac{\text{adj}\mathbf{A}}{|\mathbf{A}|} \quad (\text{단, } |\mathbf{A}| \neq 0\text{이 아님})$$

1	4	1
2	3	1
0	2	1

Matrix **A**

det **A** = -3

1	−2	1
−2	1	1
4	−2	−5

adj **A**

−1/3	2/3	−1/3
2/3	−1/3	−1/3
−4/3	2/3	5/3

Inverse matrix **A**

Assignment 2-3. 4

```
namespace ooplinalg {

// Definition of Matrix
class Matrix
{
public:
  Matrix();
  Matrix(const Matrix& mat);
  Matrix(int rows, int cols);
  ~Matrix();

  float getElement(int row, int col) const;
  float** getData() const;
  void setElement(const int row, const int col, float value);
  void setData(const int rows, const int cols, float** data);
  int getRows() const;
  int getCols() const;
  void setRows(const int rows);
  void setCols(const int cols);

  float determinant();

private:
  float** data;
  int rows;
  int cols;
};

// definition of Matrix operations
Matrix& add(Matrix& r, Matrix& a, Matrix& b);
Matrix& sub(Matrix& r, Matrix& a, Matrix& b);
Matrix& mul(Matrix& r, Matrix& a, Matrix& b);

Matrix& elementAdd(Matrix& r, Matrix& a, float v);
Matrix& elementSub(Matrix& r, Matrix& a, float v);
Matrix& elementMul(Matrix& r, Matrix& a, float v);
Matrix& elementDiv(Matrix& r, Matrix& a, float v);

Matrix& transpose(Matrix& r, Matrix& m);
Matrix& adjoint(Matrix& r, Matrix& m);
Matrix& inverse(Matrix& r, Matrix& m);

}
```

과제 제출 방법

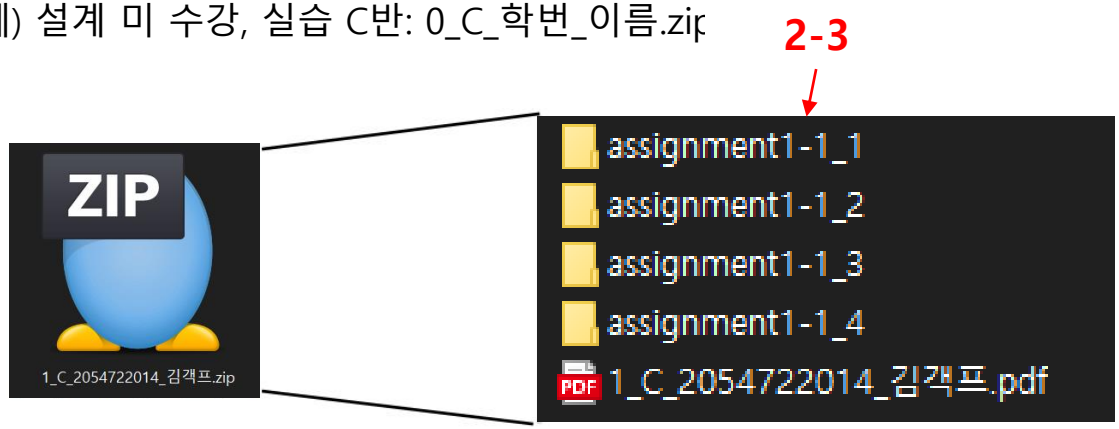
과제 제출 방법

- **FTP Upload (Klas 과제 제출 X)**
  - Address : ftp://223.194.8.1:1321
  - username : IPSL_OBJ
  - password : ipslobj_2023

- **Due date**
  - Soft copy: 마감일 4/28(금) 23:59:59까지 제출 (서버시간 기준)
  - Delay
    - 마감일 이후 +7일까지 제출 가능
    - 단, 1일 초과마다 과제 총점의 10%씩 감점

과제 제출 방법

- Soft copy
  - 과제(보고서, 소스 코드)를 압축한 파일 제출
    - 설계반_실습반_학번_이름.zip
      - 예) 설계1반 수강, 실습 A반: 1_A_학번_이름.zip
      - 예) 설계 수강, 실습 미수강: 2_0_학번_이름.zip
      - 예) 설계 미 수강, 실습 C반: 0_C_학번_이름.zip



  - 과제 수정하여 업로드 시 버전 명시
    - 설계반_실습반_학번_이름_verX.zip

과제 제출 방법

- Soft copy
  - 과제 보고서
    - 영문 또는 한글로 작성
    - **반드시 PDF**로 제출 (PDF 외 파일 형식으로 제출시 0점 처리)
    - 보고서 양식
      - 문제 및 설명(문제 capture 금지) / 결과 화면 / 고찰
      - 보고서 양식은 아래 경로에서 참고
        - https://www.ipsl.kw.ac.kr/post/1%EC%B0%A8-%EA%B3%BC%EC%A0%9C
    - **소스코드 제외**
    - 분량 제한 없음
    - **표절 적발 시 0점 처리**
  - 소스 코드
    - Visual Studio 2022 community 사용 필수
      - https://docs.microsoft.com/ko-kr/visualstudio/install/install-visual-studio?view=vs-2022
    - STL (Standard Template Library) 사용 금지 (vector, map, algorithm 등)
    - Debug 폴더를 제외한 모든 파일 제출
      - .sln 파일 포함(.cpp 만 제출하지 말것)
    - **각 문제마다 프로젝트 파일 생성 필수**
    - **주석 반드시 달기**
    - **소스코드 표절 적발 시 0점 처리**

END OF PRESENTATION

Q&A