

## 객체지향 프로그래밍 5주차 실습, 과제

소프트웨어학과 2021041017 김규현

### 실습#5-1 : string (1) :

#### [코드]

```
#include <iostream>
```

```
#include <string>
```

```
using namespace std;
```

```
int main() {
```

```
    string str[5];
```

```
    for (int i = 0; i < 5; i++) {
```

```
        cout << "이름 >> ";
```

```
        getline(cin, str[i]);
```

```
    }
```

```
    string ans = str[0];
```

```
    for (int i = 0; i < 5; i++) {
```

```
        if (str[i] > ans)
```

```
            ans = str[i];
```

```
    }
```

```
    cout << "사전에서 가장 뒤에 나오는 문자열은 " << ans << "\n";
```

```
    return 0;
}
```

[실행 화면] (제 이름과 교수님 성함 넣어봤습니다.)

```
이름 >> Kim Gyu Hyeon
이름 >> Lee Tae Kyeom
이름 >> Han Won Sun
이름 >> Hwang Su Hee
이름 >> Chang Jae Young
사전에서 가장 뒤에 나오는 문자열은 Lee Tae Kyeom
```

## 실습#5-1 : string (2) :

[코드]

```
#include <iostream>
```

```
#include <string>
```

```
#include <vector>
```

```
using namespace std;
```

```
int main() {
```

```
    string s;
```

```
    cout << "여러 줄의 문자열을 입력하세요. 입력의 끝은 &문자입니다." << "\n";
```

```
    vector<string> v;
```

```
    while (1) {
```

```
        getline(cin, s);
```

```
        if (s == "&") break; // 문자열 끝에 &를 입력 받으면 반복문 종료
```

```

        v.push_back(s); // 벡터에 push
    }

    string f_str, r_str;

    cout << "find: "; // 찾을 문자열 입력
    getline(cin, f_str);

    cout << "replace: "; // 대체할 문자열 입력
    getline(cin, r_str);

    // 대체, 메모리 누수 때문에 size_t, pos, npos 사용하고 싶긴 하지만,
    // 이번주엔 배우지 않아 쓰지 않겠습니다.

    for (int i = 0; i < v.size(); i++) {
        int idx = v[i].find(f_str);

        while (idx != -1) {
            v[i].replace(idx, f_str.length(), r_str);
            idx = v[i].find(f_str);
        }
    }

    for (int i = 0; i < v.size(); i++) { // 결과 출력
        cout << v[i] << endl;
    }

    return 0;
}

```

[실행 화면]

```
여러 줄의 문자열을 입력하세요. 입력의 끝은 &문자입니다.
I
love
me
darling
&
find: me
replace: kim gyu hyeon
I
love
kim gyu hyeon
darling
```

실습#5-1 : vector :

[코드]

```
#include <iostream>

#include <string>

#include <vector>

using namespace std;

int main() {

    vector<string> str(5);           // vector 사용

    for (int i = 0; i < 5; i++) {

        cout << "0|름 >> ";

        cin >> str[i];

    }
```

```

string ans = str[0];

for (int i = 0; i < 5; i++) {
    if (str[i] > ans)
        ans = str[i];
}

cout << "사전에서 가장 뒤에 나오는 문자열은 " << ans << "\n";

return 0;
}

```

#### [실행 화면]

```

이름 >> 김규현
이름 >> 이태겸
이름 >> 홍진민
이름 >> 김서윤
이름 >> 김석용
사전에서 가장 뒤에 나오는 문자열은 홍진민

```

#### 실습#5-1 : 클래스 :

##### [코드]

```

#include <iostream>

using namespace std;

```

```

class Rectangle {
public:
    int width;    // 너비
    int height;  // 높이

    int getArea() { // 면적 구하기
        return width * height;
    }
};

int main() {
    Rectangle rect;
    rect.width = 3;
    rect.height = 5;
    cout << "사각형의 면적은 " << rect.getArea() << endl;

    return 0;
}

```

#### [실행 화면]

사각형의 면적은 15

#### 실습#5-2 : 성적처리프로그램 #5 :

##### [코드]

```
#include <iostream>
```

```
#include <iomanip>
```

```
#include <string>
```

```
#include <cstring>
```

```
using namespace std;
```

```
struct Subject {          // 과목 정보

    char SubName[30];      // 과목 이름

    int Hakjum;            // 과목 학점

    char Grade[10];        // 과목 등급

    float GPA;             // 과목 평점

};
```

```
struct Student {          // 학생 정보

    char StdName[30];      // 학생 이름

    int Hakbun;            // 학번

    Subject* Sub;          // 과목

    int SubNum;            // 과목 수

    float AveGPA;          // 평균 평점

};
```

```
// 함수 프로토타입 선언
```

```
inline void PrintMenu();          // 메뉴 출력
```

```
inline void InputValue(char* str); // 문자열 입력
```

```
inline void InputValue(int& i);    // 정수 입력
```

```
void InputScores();               // 학생 성적 입력
```

```
void InputData(Student* pSt, int StudentNum); // 전체 학생 입력
```

```

void ShowAllScores();                // 전체 성적 출력

void PrintOneData(const Student& rSt);    // 한 학생 정보 출력

void PrintAllStdList(const Student* pSt, int StudentNum); // 전체 학생 목록 출력

void ModifyStdInfo(Student* pSt, int StdNum);    // 학생 정보 수정

Student* StdSearch(Student* pSt, int StudentNum, const char* name); // 학생 검색

void CalcGPA(Subject& Sub);            // 등급 -> 평점 변환

float CalcAveGPA(Subject* Sub, int SubNum);    // GPA 평균 계산

```

```

Student* student = nullptr;    // 동적 할당된 학생 배열 포인터

int StdNum = 0;                // 전체 학생 수

```

```

int main() {

    while (true) {

        PrintMenu();    // 메뉴 출력

        int choice;

        cin >> choice;    // 메뉴 선택

        cout << "\n";

        switch (choice) {

            case 1:    // 성적 입력

                InputScores();

                break;

            case 2:    // 전체 성적 입력

                ShowAllScores();

                break;

            case 3: {    // 학생 검색

                char name[30];

```

```

        cout << "검색할 학생의 이름 : ";

        cin >> name;

        Student* found = StdSearch(student, StdNum, name);

        if (found == nullptr) {

            cout << "해당 이름의 학생을 찾을 수 없습니다.\n";

        }

        else {

            PrintOneData(*found);

        }

        break;

    }

    case 4:    // 학생 목록 출력

        PrintAllStdList(student, StdNum);

        break;

    case 5:    // 정보 수정

        ModifyStdInfo(student, StdNum);

        break;

    case 6:    // 프로그램 종료

        cout << "프로그램을 종료합니다.\n";

        for (int i = 0; i < StdNum; ++i) {

            delete[] student[i].Sub;    // 과목 메모리 해제

        }

        delete[] student;    // 학생 메모리 해제

        return 0;

    default:    // 예외 처리

```

```

        cout << "잘못된 입력입니다. 다시 선택해주세요.\n";

        break;

    }

    cout << "\n";

}

return 0;

}

```

```

inline void PrintMenu() {

    cout << "===== 메뉴 =====\n"

        << "1. 학생 성적 입력\n"

        << "2. 전체 학생 성적 보기\n"

        << "3. 학생 이름 검색\n"

        << "4. 전체 학생 목록 보기\n"

        << "5. 학생 정보 수정\n"

        << "6. 프로그램 종료\n"

        << "\n원하는 기능을 입력하세요 : ";

}

```

```

inline void InputValue(char* str) {

    cin >> str;

}

```

```

inline void InputValue(int& i) {

    cin >> i;

```

```
}
```

```
void InputScores() {
```

```
    cout << "전체 학생 수 입력 : ";
```

```
    cin >> StdNum;
```

```
    if (student != nullptr) {
```

```
        for (int i = 0; i < StdNum; ++i) {
```

```
            delete[] student[i].Sub;    // 기존 과목 메모리 해제
```

```
        }
```

```
        delete[] student;    // 기존 학생 메모리 해제
```

```
    }
```

```
    student = new Student[StdNum];    // 학생 동적 할당
```

```
    InputData(student, StdNum);    // 학생 정보 입력
```

```
}
```

```
void InputData(Student* pSt, int StudentNum) {
```

```
    for (int i = 0; i < StudentNum; ++i) {
```

```
        cout << "₩n* " << i + 1 << " 번째 학생 이름과 학번을 입력하세요.₩n";
```

```
        cout << "이름 : ";
```

```
        InputValue(pSt[i].StdName);
```

```
        cout << "학번 : ";
```

```
        InputValue(pSt[i].Hakbun);
```

```
        cout << "₩n수강한 과목 수를 입력하세요 : ";
```

```
        InputValue(pSt[i].SubNum);    // 과목 수 입력
```

```

pSt[i].Sub = new Subject[pSt[i].SubNum]; // 과목 배열 동적 할당

pSt[i].AveGPA = 0;

for (int j = 0; j < pSt[i].SubNum; ++j) {
    cout << "교과목명 : ";
    InputValue(pSt[i].Sub[j].SubName);
    cout << "과목학점수 : ";
    InputValue(pSt[i].Sub[j].Hakjum);
    cout << "과목등급<A+ ~ F> : ";
    cin >> pSt[i].Sub[j].Grade;
    cout << "\n";

    CalcGPA(pSt[i].Sub[j]);          // 평점 계산
    pSt[i].AveGPA += pSt[i].Sub[j].GPA;
}

pSt[i].AveGPA /= pSt[i].SubNum;      // 평균 평점 계산
cout << "\n";
}
}

void ShowAllScores() {
    cout << "   전체 학생 성적 보기\n";

    for (int i = 0; i < StdNum; i++) {
        cout <<
"=====Wn";

        cout << "학번 : " << student[i].Hakbun << " 이름 : " << student[i].StdName << "\n";
    }
}

```

```

        cout <<
"=====Wn";

        cout << "과목명" << setw(10) << "과목학점" << setw(10) << "과목등급" << setw(10) <<
"과목평점" << "Wn";

        cout <<
"=====Wn";

        for (int j = 0; j < student[i].SubNum; j++) {

            cout << student[i].Sub[j].SubName

                << setw(10) << student[i].Sub[j].Hakjum

                << setw(10) << student[i].Sub[j].Grade

                << setw(10) << fixed << setprecision(2)

                << student[i].Sub[j].GPA * student[i].Sub[j].Hakjum << "Wn";

        }

        cout <<
"=====Wn";

        cout << setw(20) << "평균평점 : " << setw(10) << fixed << setprecision(2) <<
student[i].AveGPA << "WnWn";

    }

}

Student* StdSearch(Student* pSt, int StudentNum, const char* name) {

    for (int i = 0; i < StudentNum; ++i) {

        if (strcmp(pSt[i].StdName, name) == 0) {

            return &pSt[i];        // 이름 일치하면 반환

        }

    }

}

```

```

        return nullptr;                // 없으면 nullptr
    }

void PrintOneData(const Student& rSt) {
    if (rSt.StdName == nullptr) return;

    cout << "이름: " << rSt.StdName << " 학번: " << rSt.Hakbun << "\n";

    cout << "===== \n";

    cout << setw(20) << "과목명" << setw(10) << "학점" << setw(10) << "등급" << setw(10) <<
"평점" << "\n";

    cout << "===== \n";

    for (int j = 0; j < rSt.SubNum; j++) {
        cout << setw(10) << rSt.Sub[j].SubName
            << setw(10) << rSt.Sub[j].Hakjum
            << setw(10) << rSt.Sub[j].Grade
            << setw(10) << fixed << setprecision(2)
            << rSt.Sub[j].GPA * rSt.Sub[j].Hakjum << "\n";
    }

    cout << "===== \n";

    cout << setw(20) << "평균평점 : " << fixed << setprecision(2) << rSt.AveGPA << "\n";
}

```

```

void PrintAllStdList(const Student* pSt, int StudentNum) {

    cout << "\n===== \n";

    cout << setw(10) << "학번" << setw(10) << " 이름 \n";

    cout << "===== \n";

    for (int i = 0; i < StudentNum; i++) {
        cout << setw(10) << pSt[i].Hakbun

```

```

        << setw(10) << pSt[i].StdName << "\n";

    }

    cout << "=====\n";

    cout << "\n";

}

void ModifyStdInfo(Student* pSt, int StdNum) {

    string name;

    cout << "검색 할 학생 이름 : ";

    cin >> name;

    Student* target = StdSearch(pSt, StdNum, name.c_str());

    if (target == nullptr) {

        cout << "해당 학생을 찾을 수 없습니다.\n";

        return;

    }

    cout << "*" << target->StdName << ", " << target->Hakbun << "> 의 정보를 수정하세요\n";

    cout << "이름 : ";

    cin >> target->StdName;

    cout << "학번 : ";

    cin >> target->Hakbun;

}

void CalcGPA(Subject& Sub) {

    if (strcmp(Sub.Grade, "A+") == 0) Sub.GPA = 4.5f;

    else if (strcmp(Sub.Grade, "A0") == 0 || strcmp(Sub.Grade, "A") == 0) Sub.GPA = 4.0f;

    else if (strcmp(Sub.Grade, "B+") == 0) Sub.GPA = 3.5f;

```

```

else if (strcmp(Sub.Grade, "B0") == 0 || strcmp(Sub.Grade, "B") == 0) Sub.GPA = 3.0f;
else if (strcmp(Sub.Grade, "C+") == 0) Sub.GPA = 2.5f;
else if (strcmp(Sub.Grade, "C0") == 0 || strcmp(Sub.Grade, "C") == 0) Sub.GPA = 2.0f;
else if (strcmp(Sub.Grade, "D+") == 0) Sub.GPA = 1.5f;
else if (strcmp(Sub.Grade, "D0") == 0 || strcmp(Sub.Grade, "D") == 0) Sub.GPA = 1.0f;
else Sub.GPA = 0.0f;
}

```

```

float CalcAveGPA(Subject* Sub, int SubNum) {
    float StudentNumGPA = 0;
    for (int i = 0; i < SubNum; i++) {
        StudentNumGPA += Sub[i].GPA;
    }
    return StudentNumGPA / SubNum;
}

```

## **[실행 화면]**

**1번 선택 :**

===== 메뉴 =====

1. 학생 성적 입력
2. 전체 학생 성적 보기
3. 학생 이름 검색
4. 전체 학생 목록 보기
5. 학생 정보 수정
6. 프로그램 종료

원하는 기능을 입력하세요 : 1

전체 학생 수 입력 : 1

\* 1 번째 학생 이름과 학번을 입력하세요 .

이름 : 김규현

학번 : 2021041017

수강한 과목 수를 입력하세요 : 2

교과목명 : 수학

과목학점수 : 3

과목등급<A+ ~ F> : A0

교과목명 : 영어

과목학점수 : 3

과목등급<A+ ~ F> : B+

2번 선택 :

1. 학생 성적 입력
2. 전체 학생 성적 보기
3. 학생 이름 검색
4. 전체 학생 목록 보기
5. 학생 정보 수정
6. 프로그램 종료

원하는 기능을 입력하세요 : 2

전체 학생 성적 보기

=====

학번 : 2021041017 이름 : 김규현

=====

과목명	과목학점	과목등급	과목평점
-----	------	------	------

=====

수학	3	A0	12.00
----	---	----	-------

영어	3	B+	10.50
----	---	----	-------

=====

평균평점 : 3.75

3번 선택 :

```
===== 메뉴 =====
1. 학생 성적 입력
2. 전체 학생 성적 보기
3. 학생 이름 검색
4. 전체 학생 목록 보기
5. 학생 정보 수정
6. 프로그램 종료
```

원하는 기능을 입력하세요 : 3

검색할 학생의 이름 : 김규현  
이름 : 김규현 학번 : 2021041017

```
=====
              과목명              학점              등급              평점
=====
수학              3              A0              12.00
영어              3              B+              10.50
=====
              평균 평점 : 3.75
=====
```

4번 선택 :

원하는 기능을 입력하세요 : 4

```
=====
              학번              이름
=====
2021041017      김규현
=====
```

```
===== 메뉴 =====
1. 학생 성적 입력
2. 전체 학생 성적 보기
3. 학생 이름 검색
4. 전체 학생 목록 보기
5. 학생 정보 수정
6. 프로그램 종료
```

원하는 기능을 입력하세요 : |

5번 선택 :

원하는 기능을 입력하세요 : 5

검색 할 학생 이름 : 김규현

\* <김규현, 2021041017> 의 정보를 수정하세요

이름 : 김홍도

학번 : 2023010403

===== 메뉴 =====

1. 학생 성적 입력
2. 전체 학생 성적 보기
3. 학생 이름 검색
4. 전체 학생 목록 보기
5. 학생 정보 수정
6. 프로그램 종료

원하는 기능을 입력하세요 : 4

```
=====
      학번      이름
=====
2023010403      김홍도
=====
```

6번 선택 :

===== 메뉴 =====

1. 학생 성적 입력
2. 전체 학생 성적 보기
3. 학생 이름 검색
4. 전체 학생 목록 보기
5. 학생 정보 수정
6. 프로그램 종료

원하는 기능을 입력하세요 : 6

프로그램을 종료합니다.