# 2021-2학기 전공튜터링 운영결과보고서

| 학 과 | 응용소프트웨어공학과 | | | |
|---|---|---|---|---|
| 과 목 명 | 데이터구조 | | | |
| 참여학생 | 튜터 | 응용소프트웨어 공학과<br>황진주 | 튜티3 | 응용소프트웨어 공학과<br>최범진 |
| | 튜티1 | 응용소프트웨어 공학과<br>임미선 | 튜티4 | x |
| | 튜티2 | 응용소프트웨어 공학과<br>이지현 | 튜터5 | x |

# 2021-2학기 전공튜터링 운영결과보고서

- 제출장소: DOOR 전공튜터링 멘토링 게시판
- 제출서류: 운영결과보고서, 학습 일지 7부, 요약 정리 7부
- 제출기한: 2021.11.21.(일) 16:00 까지

| | | | | |
|---|---|---|---|---|
| 튜터 | 학과: 응용소프트웨어공학과 | | 학번: 20193148 | 성명: 황진주 |
| 튜티 | 1 | 학과: 응용소프트웨어공학과 | 학번: 20203218 | 성명: 임미선 |
| | 2 | 학과: 응용소프트웨어공학과 | 학번: 20203143 | 성명: 이지현 |
| | 3 | 학과: 응용소프트웨어공학과 | 학번: 20183177 | 성명: 최범진 |
| | 4 | 학과: x | 학번: | 성명: |
| | 5 | 학과: x | 학번: | 성명: |
| 학습기간 | 2021년 9월 20일 ~ 2021년 11월 21일 (2.5시간 × 8회 = 20시간) | | | |
| 학습장소 | Discord 온라인 수업 | | | |
| 학습내용 | 이수구분: (체크 ☑)<br>전공심화 ☑, 전공핵심 ☐ | 교과목번호:<br>505514 | 교과목명:<br>웹프로그래밍II | |

위와 같이 2021-1학기 전공 튜터링 프로그램 운영 결과보고서를 제출합니다.

2021년 11월 28일

튜터링 대표 튜터: 황진주 (인)

황진주

동의대학교 교수학습개발센터장 귀하

## 1. 팀 소개

학과 생활의 전반적인 지원과 개개인의 교과 능력 증대를 위한 교과동아리 DoD의 동아리 회원들의 모임이다. 다른 특성을 지닌 학생들이 다 함께 모르는 점을 공유하고, 알려주며 상호간의 발전을 위해 협력을 도모함을 목적으로 하고 있다.

## 2. 출석현황

| | 1회차 | 2회차 | 3회차 | 4회차 | 5회차 | 6회차 | 7회차 | 8회차 |
|---|---|---|---|---|---|---|---|---|
| **운영일시** | 09월 24일 x | 09월 28일 20시-22시 | 10월 01일 20시-22시 | 10월 05일 20시-22시 | 10월 28일 20시-22시 | 11월 07일 20시-22시 | 11월 11일 20시-22시 | 11월 18일 20시-22시 |
| **총인원** | x | 4 | 4 | 4 | 4 | 4 | 4 | 4 |
| **참석인원** | x | 4 | 4 | 4 | 4 | 4 | 4 | 4 |
| **결석명단** | x | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

## 3. 튜터링 진행 방법

**진행장소** : Discord 프로그램을 사용해 온라인 수업을 진행
**교재** : c++ espresso
**학습 프로그램** : Visual Studio 2019

**학습법**

- 실습 : 예시 상황을 주면 그에 대한 해결법을 추론 해보고, 추론결과로 코드를 작성하는 방식의 수업을 진행.
- 기출 풀이 : 기출 문제를 풀이 후, 풀어본 문제에 대한 새로운 문제를 만들고 해결책을 제시.
- 모듈화 : 작성 알고리즘을 도식화 하는 방법에 대해 학습해 해결하고자 하는 문제의 분할 능력을 키움.
- 퀴즈 : 뒤처지는 인원이 없도록, 학습이 진행된 내용에 대해서는 퀴즈를 진행하여 이해도를 파악함.
- 과제 풀이 : 과제가 원하는 답안, 풀이 방향에 대해 제시.
- 수업 요약지 : 튜터가 이전에 수업을 들으며 작성한 전체적인 수업 요약지를 통해 수업 흐름을 제시.
- 과목 학습지 제작 : 복습을 위한 튜터링 진행 시 핵심 키워드를 빈칸으로 만든 학습지를 제작하여 배포.
- 수행 과제 풀이 : 튜터가 수행한 해당 과목의 과제를 튜티와 함께 해결하며, 교수님의 스타일을 전수한다.

**일정진행**

- 가용시간표 : 가용 시간표를 통해 튜터링이 가능한 요일과 시간 산출
- 일정한 학습 기간 : 화요일 저녁에 꾸준한 진행으로 반복적인 학습을 할 수 있도록 함
- 횟수 이상의 학습 : 기존 8회로 기획된 수업이지만, 시험기간 및 선행 및 복습을 위한 추가적인 튜터링 진행

## 4. 우리 팀(전공 튜터링) 운영의 잘된 점(성과)와 피드백(다음에 개선할 사항)

**운영성과(튜터)**

- 메타인지(스스로 아는 것과 모르는 것의 구분)를 통해 자신이 아는 지식의 구분 가능.
- 교육의 질 관리 연구 및 개발을 통한 높은 학생 만족도 취득.
- 비교과 프로그램 효과성 분석 및 활성화 방안 연구.
- 개개인의 역량 탐색을 위한 효과적인 방안 도출을 통한 '함께하는 학습' 진행, 적극적 참여로 인한 의욕 증대.
- 튜터의 경험에 비롯한 실수를 공유 후 튜티들의 극복법을 보며, 스스로의 학습 계획 및 실수 극복에 대해 앎.
- 상대에게 질문을 할 시 어떻게 질문을 하여야 정확하고, 효율적인 답안을 얻을 수 있는지 알게됨. 스스로도
- 학습 전달을 위한 교과내용 복습을 통한 학습 성취도 증대.
- 개방되고 밝은 분위기로 선/후배 간의 친목 도모.

**운영성과(튜티)**

- 다른 과목에 비해 비교적 어렵다고 회자되는 웹프로그래밍 과목의 과제에 대한 포괄적인 개요를 전달받음으로, 제시된 과제에 의도에 맞는 정확한 답안 제출이 가능하였으며, 이를 통한 해당 자료구조에 대한 응용법을 이해함.
- 제한된 시간만이 아닌 추가적인 학습의 기회 제공으로, 시간 부담 없이 모든 내용을 이해함.
- 프로그래밍에 필수적인 내용인 자료구조에 대해 놓치는 내용 없이 이해함으로, 개발직군에 대한 전문성을 지님
- 다른 튜티와의 협력을 통한 문제해결을 통해 프로젝트형 학습의 장점과 과정을 알게됨.
- 자신이 이해한 바를 타인에게 설명하기 위해 생각을 말로 정리하는 단계에서 스스로의 모르는 점을 짚어냄. 또한 조리있게 자신의 알고리즘을 발표하기 위한 스피치 능력이 증대됨.
- 자료구조의 구현을 위한, 자료구조를 통해 구현할 수 있는 알고리즘을 도출하는 과정에서 문제해결역량을 키움.
- 교과 과목 내용을 수업만으로 끝내는 것이 아닌 통합적인 정리를 하도록하여 포트폴리오 작성법에 대해 배움.
- 개방되고 밝은 분위기로 선/후배 간의 친목 도모.

**피드백**

- 수업내용 중 학생의 이해도에 따른 즉흥적인 내용 구성이 있었는데, 이로 인해 오히려 체계가 잡히지 않은 흐름을 보여 혼동을 주게 됨. 즉흥적이고 자유로운 주제 구성을 위해서는 튜터의 개인 역량증대와 학습내용에 대한 깊은 이해도를 보여야 함을 인지함.
- 튜티의 성향이 수업의 진행 방식과 맞아 원활히 진행되었지만, 문답 방식을 부담스러워해 강의식 수업 진행 방식을 선호한다면 오히려 역효과를 볼 수 있음.

## 5. 후배들에게 전하는 전공 튜터링에 대한 제언

**튜터에게**

| 황진주(튜터) |
| --- |
| - 프로그래밍 과목에 대해 설명할 때에는 코드만을 제시하고 이해를 요구하기 보다는 글 혹은 도식화된 내용을 제시하여 명령어, 함수를 대입해 보는 방식을 적용하는 것이 이해에 효율적이다.<br>- 튜터링을 하는 이유는 교수님과의 질의응답 혹은 지도법에 대한 부담 완화를 위해 동기/선배와 함께 학습을 하는 이유도 있다는 것을 인지하고, 최대한 부담감 없는 편안한 분위기를 제공하는 것이 좋다.<br>- 한 차례의 내용이 끝나고 '이해가 되었는가?'에 대한 질문으로 끝마치치 말고, 간단한 질의응답을 통해 정확히 이해함을 다시 짚어주는 것이 좋다. 이에 대한 질의 없이 튜터링을 진행하다가 한 팀원이 튜터링 수업을 따라가지 못 하겠다며 튜터링을 돌연 중단을 선언한 튜티가 있는 팀도 있었다.<br>- 어떤 것을 가르쳐야 할지 모를 때에는 자신이 수업을 들을 때의 과제를 인용하거나 자신이 공부를 하며 헷갈렸던 내용에 대해 질문을 하는 것이 도움이 된다.<br>- 전공과목 학습이기 때문에 튜터링 과목이 아닌 과목에서도 질문이 들어올 수 있다. 또한 질 좋은 활동을 제공하기 위해선 그만한 많은 시간이 소모됨을 인지해야 한다. 자신의 시간을 튜티에게 나누는 것이므로 자신이 평소에 스케쥴링을 잘 하여야 튜티에게도 피해를 주지 않는다. |

| 최범진(튜티) |
| --- |
| - 튜티의 입장에서 튜티를 위해 고생해 주시는 튜터분들에게 많은 걸 바라지는 않습니다만, 오히려 그런 점 때문에 튜터가 힘들지는 않을까 생각해서 궁금한 점을 제대로 물어보지 않게 될 경우를 생각해야 합니다.<br>- 그래서 튜터로써 튜터링에 참가하게 되는 분들께서는 과거 자신이 힘들었던 부분이나 궁금했던 기억을 되짚어가며 수업하면 튜티들에게 큰 도움을 줄 수 있다는 생각이 듭니다. |

| 이지현(튜티) |
| --- |
| - 2학기로 접어들며, 많은 언어를 다루어야 했다. 과제또한 많아지며 학습에 부담이 컸지만, 과제 일정을 튜티보다 빠르게 판단하여 챙겨주셨던 튜터님 덕분에 훨씬 부담감이 줄었다.<br>- 튜터링의 일수에 상관 없이 매주 화요일에 고정적인 시간에 반복적으로 진행되어 복습에 도움이 되었다. 혹여나 수업에서 개념을 놓치게 되더라도 다음 주에 튜터링을 통해 극복이 가능하다는 것이 학업 부담 완화에 큰 도움이 되었다.<br>- 튜터와 튜티간 합의하에 시간을 잘 정하여 한다면 능률적인 진행이 될 수 있을 것같다.<br>- 매주 연습문제 학습지를 만들어 와주셔서, 스스로 모르는 법 짚고 다시 볼 수 있었다. |

| 임미선(튜티) |
| --- |
| - 매주 튜터링이 진행될 때 마다 그 주 진행될 내용에 대해 공지를 해주셨다. 또한 일정을 설계하기 위해 튜터링이 있기 전 수업진도를 다시 체크하고 짚어줬으면 하는 내용을 물어본 후 튜터링 내용을 정했다. 다음 튜터링을 진행 할 때도 체계적인 진행이 있으면 좋겠다.<br>- 학과 수업에 관한 근황을 수업을 듣는 튜티들보다 빠르게 접하고 알려주셨던 일이 인상 깊었다. 과제정보를 알아와 과제 코드에 대해 모두 주석을 달아주고, 혹여나 과제를 제출하지 않은 튜티가 있는지 물어봐주며 학원선생님과 같이 물어주었다. 튜티들과 소통이 자주 이루어진다면 튜티들은 여러 가지로 이점을 볼 수 있으니 다음 튜터분께서도 적극적인 소통이 이루어졌으면 좋겠다. |

**튜티에게**

| 황진주(튜터) |
| --- |
| - 튜터링 진행 중 의문이 생긴다면 적극적인 질문을 하는 것이 튜터링 진행에 매우 도움이 된다.<br>- 질문하고 이해해주는 모습이 너무나도 감사하고 기특하고 대단하고 대견하고 멋지고 최고다.<br>- 시험이 종료되고 덕분에 좋은 성과를 낼 수 있어 좋았다는 말이 너무나도 고마웠다.<br>- 미숙한 튜터지만 항상 튜터링이 종료되거나 학습지를 풀 때 수고하셨다. 유익했다. 말해주고 열심히 참여해준 우리 튜티님들께 너무 감사하였다. 튜터링의 활동은 공식적으로는 종료되었지만, 좋은 선/후배로  시험까지 계속 함께 달려나가 만족할 수 있는 성과가 나왔으면 좋겠다. |

| 최범진(튜티) |
| --- |
| - 튜티를 하면서 느끼는 점은 튜터께서 튜티를 위해 상당히 많은 시간을 할애한다는 것이고, 그러한 모습을 보면서 미안하고 고마운 마음에라도 더 열심히 수업에 임하게 된다는것을 느꼈습니다.<br>- 미래에 튜티로써 튜터링에 참가하게 되는 분들께서는 튜티를 위해 열심히 준비해서 가르쳐주는 튜터들에게 어느정도 존중과 감사를 표현하는 것이 중요하다고 생각합니다. |

| 임미선(튜티) |
| --- |
| - 선배의 도움 없이는 많이 막막했을 전공 공부를 비교적 쉽게 할 수 있어서 매우 좋았다.<br>- 수업 진도를 튜터 선배님께 알려드리면 그에 맞춰 해주신다.<br>- 튜터님이 많은 시간을 할애하고 신경써주셨다. 그 모습에 더 열심히 공부하게 되는 것도 있었다. |

| 이지현(튜티) |
| --- |
| - 비대면 수업이 진행되면서 이해도가 떨어지고 그에 따라 어떻게 해야할 지 어려울 때가 많았는데, 튜터링을 통해서 전공수업에 대한 이해도가 높아져서 좋았다.<br>- 튜터의 강의로 배우는 내용도 좋지만, 동일한 수업을 듣는 학생들이 한데 모여 소통을 나눔으로, 과제의 놓친 세세한 부분, 혹은 수업 시간의 강조한 내용을 이야기 할 수 있었다. 개인 역량 증대에도 도움이 될뿐더러, 시험 대비를 탄탄하게 할 수 있는 좋은 활동이라고 생각한다. |

# 2021-2학기 전공튜터링 ( 2 )회차 주간학습일지

작성일: 10월 10일 (일요일)

| 학 과 | 응용소프트웨어공학과 | 튜 터 명 | 황진주 |
|---|---|---|---|
| 학습일시 | 09월 28일 20시 00분~21시 30분 | 학 습 장 소 | Discord |
| 참 석 자 | 이지현, 임미선, 최범진 | 결석자(사유) | X |
| 학습주제 | 배열과 포인터 | | |
| 학습목표 | 배열과 포인터의 기능을 알고 착각하기 쉬운 요소를 점검하자. | | |
| 학습방법 (중복체크가능) | ☐ 발표 및 토의 | ☐ 강의(설명)식 | ☐ 퀴즈 및 문제풀이 |
| | ☐ 실험·실습·실기 | ☑ 기타 ( 영상 시청 및 설명 ) | |

**튜터링학습 진행 방법**

튜터가 준비한 문제를 함께 풀어보고 해설을 통해 학습함

**세부 학습 내용**

- 배열에 대한 기초 지식을 점검하고 보강하였다.

- 포인터에 대한 기초 지식을 점검하고 보강하였다.

- 배열이 포인터임을 이해하고, 그에 따른 연산의 결과를 유추할 수 있다.

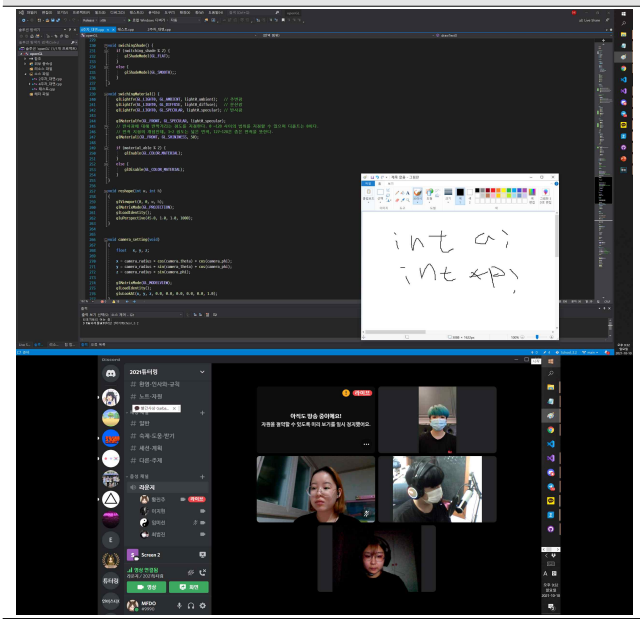
- 포인터의 특성을 알고, 포인터의 증감연산 참조 연산의 결과를 유추할 수 있다.

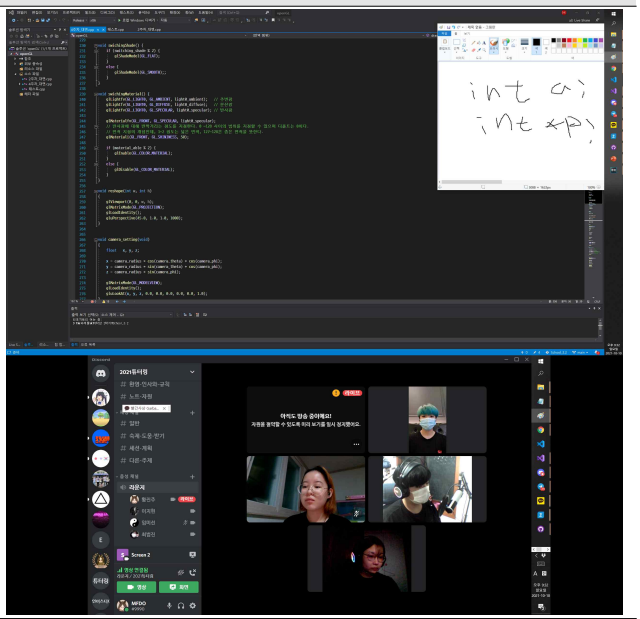
- 연산자 우선순위를 알고 그 결과가 포인터에 어떤 영향을 끼치는지 알 수 있다.

**다음 튜터링 진행 일자 및 시간**

10월 01일 / 16시 00분 / 학과동아리실

**학습활동 사진**





# 2021-2학기 전공튜터링 ( 2 )회차 주요내용 요약 정리

주제(범위) : 배열과 포인터 작성자 : 임미선

**배열(array) 이란?**

- 같은 종류의 데이터들이 순차적으로 저장되어 있는 자료구조.
- 각각 데이터들은 번호(인덱스)를 붙여서 접근한다.
- 배열의 값을 초기화 하지 않으면 기본적으로 쓰레기값이 들어있다.
- 초기화를 위해선 int arr[10] = { 0 } 과 같은 방법으로 할 수 있다.

**Q. 다차원 배열 int a[3][2][10]에는 몇 개의 원소가 존재하는가?**

답 : 60
해설 : 3*2*10의 크기를 가지기 때문.

**Q. 다차원 배열 int a[3][2][10]의 모든 요소를 0으로 초기화 하는 문장을 작성하라.**

답 : int a[3][2][10] = { 0, }
해설 : 다차원 배열도 1차원 배열과 동일한 방식으로 초기화가 가능하다.

**포인터(pointer)란?**

- 가리킨다는 뜻의 point에 er을 붙인 것, 무언가를 가리키는 변수라는 의미를 가진다.
- 포인터는 변수의 값이 아닌 변수의 주소를 저장한다.
- 포인터를 표기하는 기호는 '*' 이고 '애스터리스크'라고 읽는다.

**간접 참조(dereferencing, indirection) 연산자**

- 포인터는 메모리 주소만 저장하는 것이 아니라, 해당 주소의 값을 가져올 수 있기에 유용하다.

**Q. 다음 코드에서 (1)(2)의 출력 결과를 유추하시오. (변수 i의 주소값은 12345678이라고 가정한다.)**

```
int i = 10;     // 일반적인 변수 선언
int *p = &i;    // i의 주소값을 p가 저장하고 있다.

cout << *p;    // (1) : 10 , 간접 참조한 값을 출력
cout << p;     // (2) : 12345678 , 저장한 주소값 출력
```

**Q. 출력 결과를 유추하시오.**

```
char *pc = (char *) 10000;          pc++;
int * pi = (int *) 10000;           pi++;
double *pd = (double *) 10000;      pd++;
cout << pc << " " << pi << " " << " " pd;
```

=> 10001 10004 10008 이 출력된다. 포인터의 증감연산은 해당 변수의 타입의 크기만큼 증가되기 때문.

# 2021-2학기 전공튜터링 ( 3 )회차 주간학습일지

작성일: 10월 17일 (일요일)

| 학 과 | 응용소프트웨어공학과 | 튜 터 명 | 황진주 |
|---|---|---|---|
| 학습일시 | 10월 01일 20시 00분~21시 30분 | 학 습 장 소 | Discord |
| 참 석 자 | 이지현, 임미선, 최범진 | 결석자(사유) | X |
| 학습주제 | 객체지향 관점의 복사생성자 | | |
| 학습목표 | 복사생성자를 이용한 객체의 특성을 안다. | | |
| 학습방법 (중복체크가능) | ☐ 발표 및 토의 | ☐ 강의(설명)식 | ☐ 퀴즈 및 문제풀이 |
| | ☐ 실험·실습·실기 | ☑ 기타 ( 영상 시청 및 설명 ) | |

## 튜터링학습 진행 방법

튜터가 준비한 문제를 함께 풀어보고 해설을 통해 학습함
- **복습 학습지** : 저번 주차 내용을 복습하는 유사 문제를 풀어본다.
- **기출 학습지** : 기존 시험 기출 문제를 풀어본다.
- **주간 학습지** : 해당 주차에 학습할 주제에 대한 문제를 풀어본다.

## 세부 학습 내용

- 2017년 시험에 나왔던 문제를 풀어보며 실전 준비를 한다.

- 복사생성자의 기능과 사용 이유를 안다

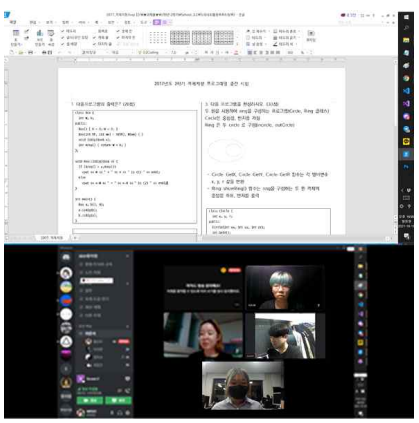
- 복사생성자가 호출되는 시기를 안다.

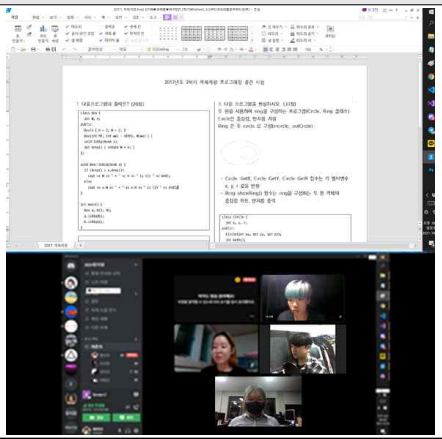
- 얕은 복사와 깊은 복사가 일어나기 위한 조건을 알고, 얕은 복사의 취약점을 안다.

## 다음 튜터링 진행 일자 및 시간

10월 05일 / 16시 00분

## 학습활동 사진



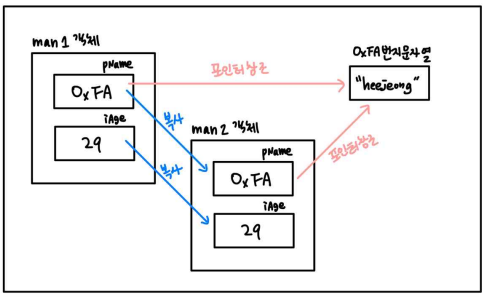


# 2021-2학기 전공튜터링 ( 3 )회차 주요내용 요약 정리

주제(범위) : 객체지향 관점의 복사생성자 작성자 : 이지현

**복사생성자란?**

: 객체의 복사가 일어날 때 이용되는 생성자의 한 종류.




**복사생성자는 언제 호출될까?**

1) 객체간의 대입이 발생한 경우.
2) 객체가 함수의 매개변수로 입력되는 경우.
3) 객체가 함수에서 반환(return)되는 경우

**Q. 아래 클래스의 복사생성자 선언하고 작성하여라.**

```
class Student {
  int num;
  string name;
  string major;

public:
  // 생성자
  Student(int n, string a, string m) : num(n), name(a), major(m) { }
  // 복사 생성자
  Student(const Student& obj) : num(obj.num), name(obj.name),
major(obj.major) { }
};
```

**복사 생성자의 특징**

- 형태는 기본 생성자의 형식을 지닌다.
- 매개변수로 const 객체 레퍼런스 타입을 받는다.
- string 형식은 : name() 형식으로 값 초기화가 가능하지만 char 배열은 불가능하다.

**기본 생성자**

- 기본 생성자 내부에서 아무것도 선언하지 않으면 객체 내부 값을 null로 자동으로 설정해준다.
  ex) ClassName() {}
- 복사생성자가 정의되지 않은 상태로 객체의 복사가 진행되면 기본 생성자가 호출된다.
- 객체의 복사 시 기본생성자가 호출되는 것을 얕은 복사, 복사 생성자가 호출되는 경우를 깊은 복사라고 한다.

**얕은 복사가 일어날 시의 문제점**

- 복사하려는 객체의 내부에 존재하는 객체는 완전한 복사가 이루어지지 않는다.
- 멤버 변수가 힙의 메모리 공간을 참조하는 경우(ex. 동적할당)에 올바른 객체공간을 갖지 못한다.

=> 극복법 : 깊은 복사를 이용한다.
=> 동작 원리 : 복사 생성자를 재정의해줌으로써 객체간의 같은 메모리 공간 참조를 막아주기 때문.
=> 코드 작성법 : 복사생성자를 객체 내부에 직접 작성해준다.

# 2021-2학기 전공튜터링 ( 4 )회차 주간학습일지

작성일: 10월 24일 (일요일)

| 학 과 | 응용소프트웨어공학과 | 튜 터 명 | 황진주 |
|---|---|---|---|
| 학습일시 | 10월 05일 20시 00분~21시 30분 | 학 습 장 소 | Discord |
| 참 석 자 | 이지현, 임미선, 최범진 | 결석자(사유) | X |
| 학습주제 | 디폴트 멤버변수를 이용한 초기화 기법 | | |
| 학습목표 | 디폴트 멤버변수의 역할과 이를 이용한 초기화를 진행해본다. | | |
| 학습방법 (중복체크가능) | □ 발표 및 토의 | ☑ 강의(설명)식 | ☑ 퀴즈 및 문제풀이 |
| | □ 실험·실습·실기 | □ 기타 ( ) | |

## 튜터링학습 진행 방법

튜터가 준비한 문제를 함께 풀어보고 해설을 통해 학습함
- **주간 학습지** : 해당 주차에 학습할 주제에 대한 문제를 풀어본다.

## 세부 학습 내용

- 디폴트 멤버 함수를 이용하여 초기화하는 방법을 이해한다.

- 디폴트 멤버 함수가 적용 될 수 있는 범위를 안다.

- defult, delete 키워드를 이용해 멤버 함수의 초기화, 멤버 함수의 변경 금지 선언을 할 수 있다.

- 변수를 초기화 하기 위한 기법을 알고 적용할 수 있다.

## 다음 튜터링 진행 일자 및 시간

10월 08일 / 16시 00분

## 학습활동 사진


# 2021-2학기 전공튜터링 ( 4 )회차 주요내용 요약 정리

**주제(범위) : 디폴트 멤버변수를 이용한 초기화 기법** **작성자 : 최범진**

**디폴트 멤버 함수란?**

: 개발자가 따로 정의하지 않으면 컴파일러에 의해서 제공하는 함수를 의미한다.

- 컴파일러가 제공하는 디폴트 멤버 함수 : 디폴트 생성자, 디폴트 소멸자, 디폴트 복사 생성자, 디폴트 대입연산자

* 일반적으로 제공하는 디폴트 복사생성자는 얕은 복사(shallow copy)가 이루어진다.

**default 키워드**

: 개발자가 따로 정의하지 않으면 컴파일러에 의해서 제공하는 함수를 의미

- C++11에서부터는 default 키워드를 이용하여 명시적으로 default 생성자를 선언 가능
- C++ 클래스는 내부를 작성하지 않아도 기본생성자, 복사생성자, 대입연산자, 소멸자 멤버함수를 생성 가능

```
//물론 default한 정의를 하기 때문에 깊은 복사는 이루어지지 않는다.
class MyClass {
public:
    MyClass() = default; // MyClass(){}와 동일한 의미
private:
    MyClass(const MyClass& class) = default; // 따로 복사생성자의 내부를 정의할 필요가 없다.
};
```

**delete 키워드**

- 복사 및 대입이 불가능한 클래스를 생성 시 사용 가능
- C++11부터는 delete 키워드를 사용하여 명시적으로 특정 함수에 대한 정의를 금지

```
class MyClass {
public:
    MyClass() {}
    ~MyClass() {}
// 아래의 생성자 및 연산자가 생성되지 않도록 명시
// private에 선언하는 방식보다 더 의도가 명확해짐
    MyClass(const MyClass& class) = delete;
    MyClass& operator = (const MyClass& class) = delete;
private:
// C++11 이전 사용 방식
    MyClass(const MyClass& class);                  // 1.  private에 선언하여 외부에서의 접근을 막고
    MyClass& operator = (const MyClass& class); // 2. 정의부를 제외해 실수로 public에 위치하는 경우에도 링크 에러를 유발
};
```

**변수 초기화 방법**

- 복사 초기화 (copy initialization) : 대입 연산자('=')를 이용한 초기화 기법
- 직접 초기화 (direct initialization) : 괄호 연산자('()')를 이용한 초기화 기법
- 유니폼 초기화 (uniform initialization) : 중괄호 연산자('{}')를 이용한 초기화 기법

```
int value1 = 1;        // 복사 초기화
double value2(2.2); // 직접 초기화
char value3 {'c'};    // 유니폼 초기화
```

# 2021-2학기 전공튜터링 ( 5 )회차 주간학습일지

작성일: 11월 1일(월요일)

| 학 과 | 응용소프트웨어공학과 | | 튜 터 명 | 황진주 |
|---|---|---|---|---|
| 학습일시 | 10월 28일 20시 00분~21시 30분 | | 학 습 장 소 | Discord |
| 참 석 자 | 이지현, 임미선, 최범진 | | 결석자(사유) | X |
| 학습주제 | 소멸자/파괴자를 이용한 객체 구성 | | | |
| 학습목표 | 소멸자/파괴자를 이용하여 객체 활용의 종료시점을 알고 올바른 파괴자 호출을 한다. | | | |
| 학습방법 (중복체크가능) | ☐ 발표 및 토의 | ☑ 강의(설명)식 | ☑ 퀴즈 및 문제풀이 | |
| | ☐ 실험·실습·실기 | ☐ 기타 ( ) | | |

## 튜터링학습 진행 방법

튜터가 준비한 문제를 함께 풀어보고 해설을 통해 학습함
- **주간 학습지** : 해당 주차에 학습할 주제에 대한 문제를 풀어본다.

## 세부 학습 내용

- 파괴자의 역할을 이해한다.

- 파괴자가 호출되는 시점을 짚어낼 수 있다.

- 생성자와 파괴자의 생명 주기를 안다.

- RAII 프로그래밍 기법을 알고, 그에 맞는 프로그램을 작성할 수 있다.

## 다음 튜터링 진행 일자 및 시간

11월 04일 / 16시 00분

## 학습활동 사진


# 2021-2학기 전공튜터링 ( 5 )회차 주요내용 요약 정리

주제(범위) : 디폴트 멤버변수를 이용한 초기화 기법 작성자 : 임미선

**소멸자 (Destructor)란?**

: 소멸자는 객체가 소멸될 때 자동으로 실행되는 클래스의 멤버 함수다.

**생성자와의 차이점**

: 생성자는 클래스 초기화를 위해 사용되지만, 소멸자는 클래스 즉, 객체의 소멸/삭제를 돕는다.

**소멸자의 호출 시기**

아래의 경우에 객체가 메모리에서 제거되기 전에 필요한 정리를 수행하기 위해 클래스는 소멸자가 있다면, 객체는 자동으로 소멸자를 호출한다.

1) 지역에서 생성된 객체가 지역 범위를 벗어남.
2) 동적으로 할당된 객체가 삭제 키워드를 사용해 명시적으로 삭제됨.

**소멸자 작성 규칙**

1) 물결표('~')를 작성하고 그 뒤에 클래스 이름과 동일하게 작성한다.
2) 소멸자는 인수가 없다.
3) 소멸자는 반환 값이 없다.

```
class IntArray {
private:
        int* m_Array;
public:  // 생성자
        IntArray(int length) {
                m_Array = new int[length]{};
        } // 파괴자
        ~IntArray() {
                delete[] m_Array; // 동적으로 할당한 배열을 삭제한다.
        }
};
```

**디폴트 소멸자**

: 생성자와 마찬가지로 소멸자를 생성하지 않으면 디폴트 소멸자가 자동 삽입된다. 이 때 소멸자가 삽입되었으나, 소멸자는 아무런 기능도 하지 않는다.

```
class AAA {
int num;
public:
    ~AAA() {}
};
```

**RAII (Resource Acquisition Is Initialization)**

: RAII는 객체의 수명과 리소스 관리와 연관있는 프로그래밍 디자인 기법이다. C++ 에서 RAII는 클래스의 생성자와 소멸자로 구현되었다.

- 리소스는 일반적으로 객체의 생성자에서 획득된다.
- 리소스는 객체가 살아있는 동안 사용될 수 있다.
- 객체가 소멸하면 자원이 소멸된다.

=> 리소스를 보유한 객체가 자동으로 정리됨에 따라 리소스 누수를 방지.

# 2021-2학기 전공튜터링 ( 6 )회차 주간학습일지

작성일: 11월 7일(월요일)

| 학 과 | 응용소프트웨어공학과 | 튜 터 명 | 황진주 |
|---|---|---|---|
| 학습일시 | 11월 07일 20시 00분~21시 30분 | 학 습 장 소 | Discord |
| 참 석 자 | 이지현, 임미선, 최범진 | 결석자(사유) | X |
| 학습주제 | 임시객체의 형태와 생성 시기 | | |
| 학습목표 | 임시객체에 대해 알고, 생성 시기를 앎으로 코드를 보고 생성 시기를 짚어낼 수 있다. | | |
| 학습방법 (중복체크가능) | ☐ 발표 및 토의 | ☑ 강의(설명)식 | ☑ 퀴즈 및 문제풀이 |
| | ☐ 실험·실습·실기 | ☐ 기타 ( ) | |

## 튜터링학습 진행 방법

튜터가 준비한 문제를 함께 풀어보고 해설을 통해 학습함
- **주간 학습지** : 해당 주차에 학습할 주제에 대한 문제를 풀어본다.

## 세부 학습 내용

- 임시객체의 정의를 안다.

- 임시객체의 용도와 활용에 대해 한다.

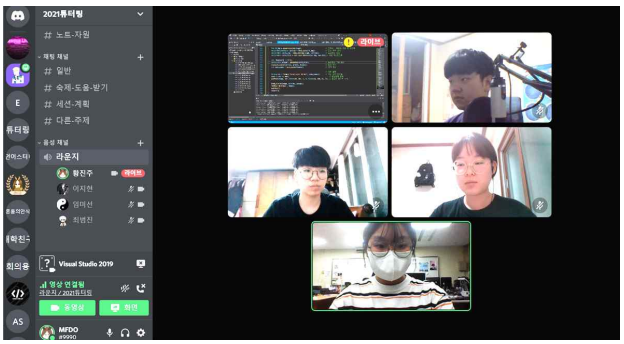
- 값에 의한 복사(Call by value), 참조에 의한 복사(Call by reference)의 차이와 코드적인 구현 방법을 안다.

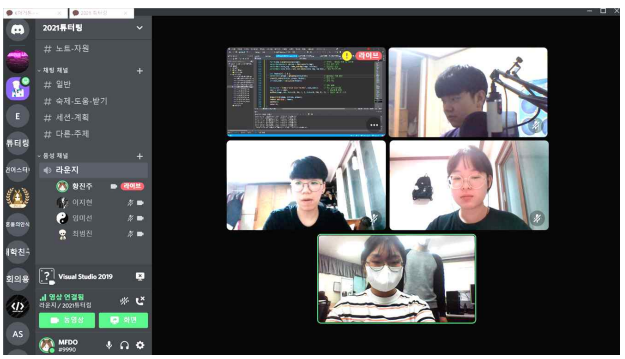
- 이전까지 실습했던 코드에서 임시객체가 호출되는 시기를 짚어낼 수 있다.

## 다음 튜터링 진행 일자 및 시간

11월 12일 / 20시 00분

## 학습활동 사진





# 2021-2학기 전공튜터링 ( 6 )회차 주요내용 요약 정리

주제(범위) : 임시객체의 정의와 구현 작성자 : 이지현

**임시객체란?**

: 실행 도중에 잠깐만 사용되는 객체로, 소스 코드에도 없는 힙 이외의 공간에 생성되는 것이 임시객체이다.

```
int A(int a){
    cout<<"임시객체"<<endl;  // 객체가 호출되며 임시객체 생성
    return a;  // 반환하기 위한 임시객체 생성
}
int main() {
    int n = 1;
    A(n);  // A를 출력하게 됨
}
```

>> **main함수에서, A함수에 n을 인자로 넘겨**주면, A함수의 통용범위에서만 사용가능한 **int a라는 임시객체를 생성**하게 된다. **임시객체 a는 n의 값인 1을 담게 되는데 이 때 복사**가 일어난다. 이를 **값에 의한 복사(Call by value)**라고 한다. 마찬가지로, return a를 할 때도, **리턴 값은 int형 임시객체를 생성**하고 담게 된다.

**임시객체의 호출**

- **int A()** : 값 리턴, 임시객체가 생성된다.
- **int & A()** : 참조 리턴, 리턴용 임시객체를 만들지 말라는 의도를 말한다.

* 임시객체는 성능저하를 유발하는 원인이 되기 때문에, 참조에 의한 복사(Call by reference)를 해주는 것이 좋다.

**임시객체 정의법**

- **생성자 객체 생성** : 객체의 이름이 존재하는 생성자 호출 ex) Temporary Tem(100);
- **임시객체 생성** : 객체의 이름이 존재하지 않고 생성자 호출 ex) Temporary(100);.

**임시객체 특징**

- 임시객체는 주소연산자로 주소를 구할 수 없다.
  ex ) Temporary *t1 = &Temporary(100); // 에러 발생.
- 임시객체는 lvalue가 될 수 없다. ( =의 왼쪽에 올 수 없다.)
  ex ) Temporary(100) = 10; // 에러 발생
- 임시객체는 일반적인 참조가 불가능하다. 그러나, 상수 참조는 가능하다.
  ex ) Temporary &ref = Temporary(100); // 에러 발생
  const Temporary & ref = Temporary(100); // 실행 가능, 즉 ref가 파괴될 때까지 임시객체는 존재한다.
- 임시객체는 다음 행으로 넘어가면, 바로 소멸되게 된다. 참조자에 참조되는 임시객체는, 바로 소멸되지 않는다.

**클래스의 외부에서, 객체의 맴버함수를 호출하기 위해 필요한 것**

- 객체에 붙여진 **이름**
- 객체의 **참조 값** (객체 참조에 사용되는 정보)
- 객체의 **주소** 값

**〉〉 위 세가지 중 한 가지 경우**

# 2021-2학기 전공튜터링 ( 7 )회차 주간학습일지

작성일: 11월 14일(일요일)

| 학 과 | 응용소프트웨어공학과 | 튜 터 명 | 황진주 |
|---|---|---|---|
| 학습일시 | 11월 11일 20시 00분~21시 30분 | 학 습 장 소 | Discord |
| 참 석 자 | 이지현, 임미선, 최범진 | 결석자(사유) | X |
| 학습주제 | 상속의 형태와 활용 | | |
| 학습목표 | 다른 객체의 요소를 사용하기 위한 여러 형태를 알고 그 동작 원리를 이해한다. | | |
| 학습방법 (중복체크가능) | ☐ 발표 및 토의 | ☑ 강의(설명)식 | ☑ 퀴즈 및 문제풀이 |
| | ☐ 실험·실습·실기 | ☐ 기타 ( ) | |

## 튜터링학습 진행 방법

튜터가 준비한 문제를 함께 풀어보고 해설을 통해 학습함
- **주간 학습지** : 해당 주차에 학습할 주제에 대한 문제를 풀어본다.

## 세부 학습 내용

- 상속의 정의와 필요성을 안다.

- 다른 객체를 이용하기 위한 두 가지 형태인 is-a 관계와 has-a 관계를 알고, 적합한 상황에 둘을 구분해 이용할 수 있다.

- 상속 관계에서의 생성자와 파괴자의 호출 순서을 알아 그 실행 결과를 유추 할 수 있다.

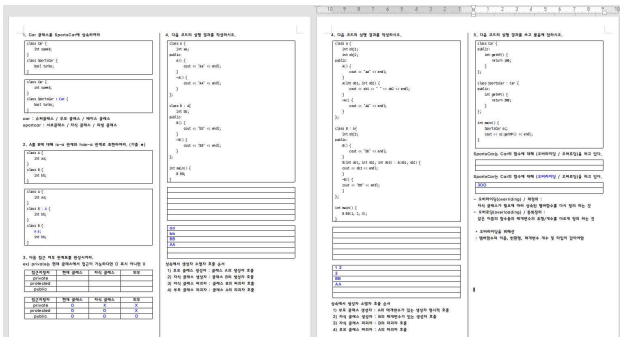
- 오버로딩과 오버라이딩의 차이를 알고 둘의 이용 상황을 구분지을 수 있다.

## 다음 튜터링 진행 일자 및 시간

11월 18일 / 20시 00분

## 학습활동 사진

사진 한 장이 깨져 사용한 학습지 사진으로 대체하였습니다.



# 2021-2학기 전공튜터링 ( 7 )회차 주요내용 요약 정리

주제(범위) : 상속의 형태와 활용 작성자 : 최범진

**상속이란?**

: 기존에 존재하는 클래스로부터 **속성과 동작을 이어받아** 자신이 **필요한 기능을 추가**하는 기법

**상속의 장점**

- 상속을 통해 기존 클래스의 필드와 메소드를 **재사용**
- 기존 클래스의 일부 변경이 가능
- 상송을 이용하면 복잡한 GUI 프로그램을 순식간에 작성
- 상속은 이미 작성된 **검증된 소프트웨어**를 재사용
- 신뢰성 있는 소프트웨어를 손쉽게 개발, 유지 보수에 적합
- 코드의 **중복 감소**

```
class Car { // 부모 클래스
    int speed;
}
// SportsCar가 부모객체인 Car를 상속받았다.
class SportsCar : class Car { // 자식 클래스
    bool turbo;
}
```

**상속시 이용 가능 관계**

- **부모 클래스에 선언된 접근 지정자**에 따라 접근이 가능한 범위가 달라진다.
- protected는 상속받은 객체의 접근만 허용한다.

| 접근지정자 | 현재 클래스 | 자식 클래스 | 외부 |
|---|---|---|---|
| private | O | X | X |
| protected | O | O | X |
| public | O | O | O |

**Is-a 관계와 Has-a 관계**

- **Is-a 관계** :
  > 추상화(형식이나 클래스와 같은)들 사이의 **포함 관계**를 의미
  > 한 클래스 A가 다른 클래스 B의 **서브클래스**(파생클래스)임을 의미
  ex) 사자는 동물이다. 트럭은 자동차다.
- **Has-a 관계** :
  > 구성 관계를 의미하며 한 오브젝트가 다른 오브젝트에 "**속함**"
  > 객체의 **멤버 필드**라고 불리는 객체
  > 도서관에는 책이 있다. 도형은 점, 선, 면으로 구성된다.

```
class A {
    int aa;
}
class B : A {  // Is-a 관계
    int bb;
}
class B {      // Has-a 관계
    A a;
    int bb;
}
```

**상속 관계에서의 생성자와 파괴자 호출 순서**

```
class A {
    int aa;
public:
    A() { cout << "aa" << endl; }  // ①부모의 생성자
    ~A() { cout << "AA" << endl; } // ④부모의 파괴자
};
class B : A{
    int bb;
public:
    B() { cout << "bb" << endl; }  // ②자식의 생성자
    ~B() { cout << "BB" << endl; } // ③ 자식의 파괴자
};
int main() {
    B BB;
}
```

1) **부모 클래스 생성자** : A의 생성자 호출
2) **자식 클래스 생성자** : B의 생성자 호출
3) **자식 클래스 파괴자** : B의 파괴자 호출
4) **부모 클래스 파괴자** : A의 파괴자 호출

**코드 실행 결과**

```
aa
bb
BB
AA
```

**자주 틀리는 문제) 오버라이딩과 오버로딩**

- **오버라이딩(overriding) – 재정의** : 자식 클래스가 필요에 따라 **상속된 멤버함수**를 다시 정의 하는 것
- **오버로딩(overloading) – 중복정의** : **같은 이름의 함수**들의 매개변수의 유형/개수를 다르게 정의 하는 것

```
class Car {
public:
    int getHP() {
        return 100;   // 기존 함수
    }
};

class SportsCar : Car {
public:
    int getHP() {     // 함수의 오버라이딩
        return 300;   // 기존 함수의 재정의
    }
};
```

▲ 함수의 오버라이딩

```
class Car {
public:
    int getHP(int a) {    // 기존 함수
        return a;
    }
    int getHP(double a) { // 함수의 오버로딩
        return a;         // 기존 함수의 중복정의
    }
};
```

▲ 함수의 오버로딩

※ 오버라이딩을 위해서는 **멤버함수의 이름, 반환형, 매개변수 개수 및 타입이 같아**야한다.

# 2021-2학기 전공튜터링 ( 8 )회차 주간학습일지

작성일: 11월 21일(월요일)

| 학 과 | 응용소프트웨어공학과 | 튜 터 명 | 황진주 |
|---|---|---|---|
| 학습일시 | 11월 18일 20시 00분~21시 30분 | 학 습 장 소 | Discord |
| 참 석 자 | 이지현, 임미선, 최범진 | 결석자(사유) | X |
| 학습주제 | Friend 키워드의 다양한 활용과 적용 범위 | | |
| 학습목표 | Friend가 선언되는 위치에 따른 의미를 이해한다. | | |
| 학습방법 (중복체크가능) | ☐ 발표 및 토의 | ☑ 강의(설명)식 | ☐ 퀴즈 및 문제풀이 |
| | ☑ 실험·실습·실기 | ☐ 기타 ( ) | |

## 튜터링학습 진행 방법

튜터가 준비한 문제를 함께 풀어보고 해설을 통해 학습함
- 예제 코드를 준비해와 함께 작성해본다.

## 세부 학습 내용

- Friend 예약어의 역할을 안다.

- Friend 예약어의 적용 가능한 범위를 안다.

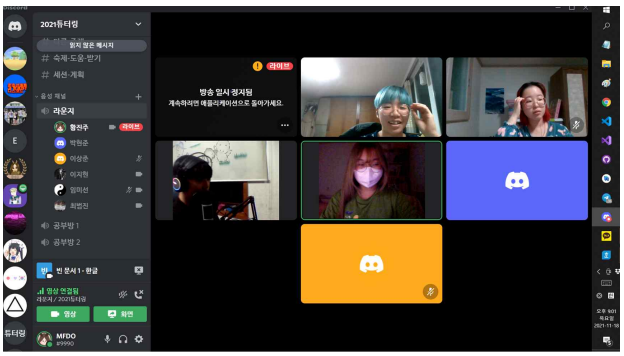
- 클래스, 멤버 함수, 전역 함수에 사용된 Friend 예약어의 차이를 안다.

- Friend 예약어의 사용 시 주의점을 안다.

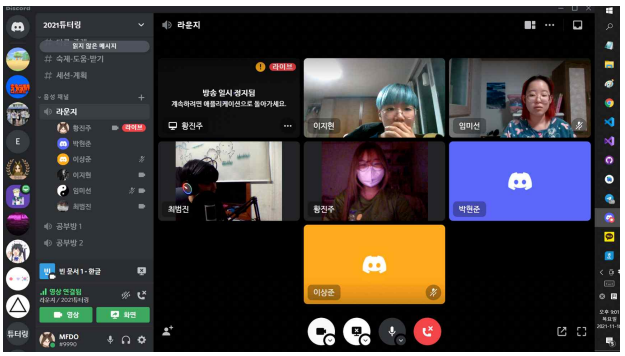
## 다음 튜터링 진행 일자 및 시간

11월 24일 / 20시 00분

## 학습활동 사진



함께 듣고싶어 하시는 다른 학우분도 참여했습니다.



# 2021-2학기 전공튜터링 ( 8 )회차 주요내용 요약 정리

주제(범위) : Frind 키워드의 다양한 활용 작성자 : 임미선

**Frind 키워드**

: private 혹은 protected로 선언되어 있는 비공개적인 객체 혹은 요소에 대해 friend 키워드를 붙여 public처럼 사용 할 수 있도록 해 줌.

```
class Box; // friend로 선언될 클래스
class Point {
    int line[8][8];
    friend Box; // Box를 friend 선언
};
class Box {
   void setL(Point& p, int x, int y, int v) {
        // private 요소에 접근 가능
        p.line[x][y] = v;
   }
};
```

**Frind 키워드 활용방안 3가지**

- friend 클래스
- friend 멤버 함수
- friend 전역함수

**Frind 클래스**

: 다른 클래스를 자신의 친구(friend)로 선언. 즉, 선언이 된 클래스는 해당 객체의 요소를 public처럼 사용 가능

ex) Box는 Point의 요소인 line에 접근 가능하다.

**Frind 멤버 함수**

: 클래스 전체가 아닌 클래스의 특정 멤버 함수만을 친구로 선언. 친구가 된 일부 함수만이 원본 클래스에 접근 가능하다.

ex) 다른 애들은 안되는데, 회사 내부의 하나의 부서만 허용

```
class Friend1 {
    string name;
    // friend로 선언될 함수
    friend void set_name(Friend1&, string);
};
void set_name(Friend1& f, string s) {
    f.name = s; // 내부 요소 접근
    cout << f.name << "\n";
}
```

**Frind 전역 함수**

: 전역함수에 대해 frind 선언. 전역함수이지만, 객체의 private한 요소에 대해 접근하기 위함

ex) 내 계좌에 돈을 넣어야 겠어

```
class Rect {
private:
    double height_;
    double width_;
public:
    Rect(double height, double width);
    void DisplaySize();
    Rect operator*(double mul) const;
    // 프렌드 함수
    friend Rect operator*(double mul,
const Rect& origin);
};
```

**!! 주의사항 !!**

- 친구의 친구는 friend 키워드로 명시하지 않으면 친구 관계가 형성X
- 친구의 자식도 friend 키워드로 명시하지 않은 경우 친구 관계 X

**요약**

- private 맴버에 대한 접근은 내부 접근만 가능하고 외부 접근은 불가능하지만 friend에 의한 접근은 가능하다.
- 특정 클래스 에서 다른 전역 함수, 클래스, 다른 클래스의 멤버 함수를 자신의 friend 키워드로 선언 가능하다.
- friend 설정한 함수에서만 접근 가능하다.

**참고사항**

- friend 키워드는 함수 선언의 위치에 따라 변하는 것이 아님. 선언하는 영역(private, public)과 무관
- 클래스 내에서 friend 키워드 이용 함수 선언 시에만 friend 키워드를 추가해야 함. 그 외에 특별한 기능이 없음.
- friend 선언은 멤버 함수의 선언이 아님. 단지 기존 함수에 대한 friend 선언일뿐 객체를 통한 접근 불가능하다.