

기획서

차리서 <reeseo@konkuk.ac.kr>

건국대학교 공과대학 컴퓨터공학부

<복제물에 대한 경고>

본 저작물은 **저작권법 제25조 수업목적 저작물 이용 보상금제도**에 의거, **한국복제전송저작권협회와 약정을 체결하고** 적법하게 이용하고 있습니다. 약정범위를 초과하는 사용은 저작권법에 저촉될 수 있으므로 **저작물의 재 복제 및 수업 목적 외의 사용을 금지합니다.**

2020. 03. 30.

건국대학교(서울)·한국복제전송저작권협회

<전송에 대한 경고>

본 사이트에서 수업 자료로 이용되는 저작물은 **저작권법 제25조 수업목적저작물 이용 보상금제도**에 의거, **한국복제전송저작권협회와 약정을 체결하고** 적법하게 이용하고 있습니다. 약정범위를 초과하는 사용은 저작권법에 저촉될 수 있으므로 **수업자료의 대중 공개·공유 및 수업 목적 외의 사용을 금지합니다.**

2020. 03. 30.

건국대학교(서울)·한국복제전송저작권협회

학기 전체 일정 (예정: 변동 가능)

#3-rev1
기획서

차리서

일정

요구사항
분석서와의
비교설계서와의
비교

기획서의 품질

오류 처리

Mockup과
화면/UI

사용 흐름도

기타 사항

주	월	화	수	목	금	토	일	실습	강의
1	8月	29	30	31	1	2	3 4	수강 정정, 팀 결성	과목 개요 및 팀 결성 안내
2	9月	5	6	7	8	9	10 11	팀 결성, 주제 선정	주제 선정 안내
3		12	13	14	15	16	17 18	주제 선정	기획서 안내
4		19	20	21	22	23	24 25	기획서 작성	기획서 안내
5		26	27	28	29	30	1 2	기획서 작성	구현 및 검사 안내
6	10月	3	4	5	6	7	8 9	구현 및 검사	구현 및 검사 안내
7		10	11	12	13	14	15 16	구현 및 검사	(구현 및 검사 안내)
8		17	18	19	20	21	22 23	구현 및 검사	중간 발표 안내, 개선 제안 안내
9		24	25	26	27	28	29 30	중간 발표	—
10		31	1	2	3	4	5 6	개선 제안, 기획서 수정	기획서 수정 안내, 검사 요소 준비 안내
11	11月	7	8	9	10	11	12 13	기획서 수정, 검사 요소 준비	검사 요소 준비 안내, 설계 안내
12		14	15	16	17	18	19 20	기획서 수정, 검사 요소 준비	(설계 안내)
13		21	22	23	24	25	26 27	설계	재검사 안내
14		28	29	30	1	2	3 4	설계, 구현 수정 및 재검사	(재검사 안내)
15	12月	5	6	7	8	9	10 11	구현 수정 및 재검사	기말 발표 안내
16		12	13	14	15	16	17 18	기말 발표	—

팀프로젝트: “1차 기획서 원판” 제출

#3-rev1
기획서

차리서

일정

요구사항
분석서와의
비교

설계서와의
비교

기획서의 품질

오류 처리

Mockup과
화면/UI

사용 흐름도

기타 사항

기한: **10/05(수) 오전 11시 직전까지**

- 강의자료 #1, #2 의 일정표 상의 ‘|’ 표시보다 하루 늦춰짐 (10/3(월) 이 개천절임을 고려)
- 10/24(월) 오전 11시 직전까지 지각 제출을 허용하지만, 점진적 감점 있으니 정시 제출 권장
 - 단, 해당 분반의 모든 팀들이 제출 완료하면 지각 제출 조기 마감

문서 첫 페이지 (의 전체나 상단)에 다음 사항들을 (어떤 순서로든) 잘 보이게 표시:

- **팀 이름:** ‘[AB][01][0-9]’ 형식 (예: “A08”, “B11” 등)
- **“1차 기획서 원판”**이라는 문자열
- **프로젝트 주제명**

제출 방법:

- 문서를 (.docx나 .hwp 말고) **PDF 형식**으로 저장/변환하여 업로드
- (문서만 열면 글과 그림들이 나란히 함께 보이도록) **그림들은 문서 본문 속에 요소로 삽입**
 - 특히, mockup 스크린샷들은 반드시 문서 속에 삽입
 - 사용 흐름도도 가능한 한 문서 속에 삽입하되, 정 여의치 않으면 별개의 파일(들)로 제출
- 문서(와 사용 흐름도 파일들)을 **압축하지 말고 그대로 (각각의 파일들을 따로 따로) 업로드**

팀프로젝트: 1차 기획서... “원판” 이라뇨? (미리 알아둘 점)

#3-rev1
기획서

차리서

일정

요구사항
분석서와의
비교설계서와의
비교

기획서의 품질

오류 처리

Mockup과
화면/UI

사용 흐름도

기타 사항

추후 (원판 제출 마감 후 ~ 중간 발표 하루 전) “1차 기획서 **수정판**” 제출 가능

- 원판과는 별도의 항목에 제출 (원판을 제출했던 항목에 수정/재업로드 하는 게 **아님**)
 - 해당 분반의 모든 팀들이 원판 제출을 완료하면 ‘지각 제출’을 조기 마감하는 이유
- 원판 대로 **구현할 수 있는** (즉, 모순/누락/난이도 문제가 없는) 팀은 제출할 필요 없고, **제출해선 안 됨!**
 - 원판대로 구현하는 게 (현실적으로) 가능하기만 하면, 반드시 그대로 구현해야 함!
 - 원판 내용에 대한 단순 변심/아쉬움 때문에 수정해선 안 됨! 아쉬워도 반드시 그대로 구현해야 함!
- 원판 대로 **구현할 수 없는** (즉, 모순/누락/난이도 문제가 있는) 팀은 반드시 **제출해야 함!**
- 수정판은 여러번 재제출할 수 있음 (수정 1판, 수정 2판, 수정 3판, ...)

‘원판 vs. 마지막 수정판’ 간의 차이 (수정된 항목들의 갯수, 각각의 심각성 등)에 따라 감점

- ‘어차피 나중에 수정판 제출 기회가 있으니 지금 원판은 대충 쓰자’ 했다간 손해임!
- ‘원판 대로 구현할 수 없지만, 수정판 제출하면 감점되니 수정하지 말고 버티자’ 했다간 더 손해임:
 - 기획서 (원판이든 수정판이든) ‘최종판’과 다르게 구현하면 훨씬 크게 감점됨
 - 잘못된 기획서(최종판)대로 구현하는 바람에 뻘거나 오동작해도 크게 감점됨
 - ‘검사’란 ‘구현물이 기획(설계)서대로 동작하는지 확인하는 작업’임을 유념할 것

추후 구현/검사 안내시 다시 더 자세히 안내 예정

요구사항 분석서 vs. 기획서

#3-rev1
기획서

차리서

일정

요구사항
분석서와의
비교

설계서와의
비교

기획서의 품질

오류 처리

Mockup과
화면/UI

사용 흐름도

기타 사항

소프트웨어 개발 단계의 (수많은 모델들 중) 대표적인 모델:

기획 (혹은 요구사항 분석) → 설계 → 구현 → 검사

요구사항 분석서: 작성자 (분석가) 외에 특정한 (즉, **직접** 의사소통한) 요구자가 따로 존재

- 작성자(분석가)는 요구자의 말을 생각없이 그대로 받아적기만 해선 안 됨
- 작성자는 요구자의 말 속의 모든 **행간들**(즉, 요구자 머리 속의 생각들)을 **명시적이고 구체적 문장**으로 밝혀서 기재해야 함. 단:
 - 요구자의 행간과 자신의 행간이 같으리라고 **함부로 짐작하지 말 것** (물어보고 확인해야 함)
 - 예시: 요구자가 “정수”라고 말하면, 그 정수의 범위로서 몇부터 몇까지를 원하는지, 표기를 다양하게 (선행 0 허용 여부 등) 할 수 있는지, 등등을 모두 확인

기획서: 작성자 자신이 요구자 (혹은 작성자가 불특정 다수의 요구자들을 간접적으로 조사¹⁾)

- 자기 생각을 자기 스스로 쓴다고 방심하지 말 것! (오히려, 한 번 놓친 요소는 끝까지 눈에 안 들어옴)
- 혼자 (혹은 팀원들끼리) 두 역할 모두 정확히 해낼 것.
- 팀 플레이의 장점 활용: 요구자/분석가 역할극, 제3자 검토 · 훈수 등

¹⁾ 회사 기획실에서 시장 동향 분석이나 설문 조사를 하는 경우 등

기획서 작성시 팀플레이 활용 팁 (제안)

#3-rev1
기획서

차리서

일정

요구사항
분석서와의
비교설계서와의
비교

기획서의 품질

오류 처리

Mockup과
화면/UI

사용 흐름도

기타 사항

팀원 A, B, C, D, E가 있고, A가 세부 기능들 중 α 라는 기능을 구상하고 제안했을 경우:

- A가 자신이 생각하는 α 의 구체적인 내용을 B에게 열심히 설명해줌 (A가 요구자)
- (A가 아닌) B가 기획서의 α 부분을 작성 (B가 분석가/작성자)
 - A가 자기 생각 ‘아’를 자기 방식으로 열심히 말함 → B가 “어”라고 이해하고 “어”라고 작성
- 작성된 기획서의 α 부분을 A가 읽어봄 (요구자 검수/확인)
 - “어”라고 쓰인 걸 발견하고, 이에 대한 정상 피드백 (왜 의미가 잘못 전달됐는지 분석/수정)
 - “어”라고 쓰인 걸 뵈히 보면서도, ‘나는 ‘아’라고 생각했었다’가 뇌리에 박혀서 ‘오케이. 내 생각대로 “아”라고 썼군.’이라고 착각할 수도 있음 (제3의 C 필요)
- 작성된 기획서의 α 부분을 C가 읽어봄 (C가 제3의 검사자)
 - A와 B 눈에는 잘 안 보이는 자체 모순이나 누락 사항, 모호한 사항을 더 잘 발견
 - 예시: “이 부분 진짜로 ‘어’를 의도한 게 맞아? 이러면 저쪽 부분이랑 앞뒤가 안 맞는데...?”
- C가 α 부분을 읽고 자신이 이해한 바를 A와 B에게 말로 설명

A: “내가 애초에 생각했던 α 기능은 그런 게 아닌데? B가 (내 말을) 잘못 들은 거야, B가 (맞게 듣고 이해해놓고) 잘못 쓴 거야, 아니면 C가 (B가 맞게 쓴 글을) 잘못 읽은 거야?”

B: “내가 작성한 문장을 그렇게 해석한다고? 어... **듣고보니** 진짜 그렇게 해석할만 하네!”

기획(요구분석)서 vs. 설계서

#3-rev1
기획서

차리서

일정

요구사항
분석서와의
비교

설계서와의
비교

기획서의 품질

오류 처리

Mockup과
화면/UI

사용 흐름도

기타 사항

소프트웨어 개발 단계의 (수많은 모델들 중) 대표적인 모델:

기획 (혹은 요구사항 분석) → 설계 → 구현 → 검사

기획(요구분석)서와 설계서의 차이가 아닌 것: 상세함

- **틀린** 생각: ‘기능을 기획서에는 대충만 설명해두고 설계할 때 더 구체적으로 자세히 정하자.’
- **틀린** 생각: ‘사용자 키 입력의 (맞고 틀리는) 세부 규칙은 기획서에는 쓰지 말고 설계문서에 쓰자.’

기획(요구분석)서와 설계서의 진짜 차이:

- **내용**: 어떤 내용을 기재하는 문서인가?
- **독자**: 누구더러 읽으라는 문서인가?

기획(요구분석)서 vs. 설계서: 내용의 차이

#3-rev1
기획서

차리서

일정

요구사항
분석서와의
비교설계서와의
비교

기획서의 품질

오류 처리

Mockup과
화면/UI

사용 흐름도

기타 사항

기획(요구분석)서: **무엇^{what}**을 만들 것인가? (요구자가 **무엇**을 요구했는가?) — **사용자 관점** (외부)

- 배포 형식(파일들), 동작 환경, 설치 방법, 실행 (및 종료) 방법
- 가능한 모든 직접적 입력 (조작) 방식들 중 무엇이 문법/의미 상 허용되고 불허되는가?
 - 허용된 입력 (조작) 각각에 대해, 무슨 정상 출력이나 정상 동작을 보이는가?
 - 불허된 입력 (조작) 각각에 대해, 무슨 오류 문구나 비상 동작을 보이는가?
- 가능한 간접적 입력과 상황 (데이터 상태, 현재 일시 등)에는 무엇이 있고, 각각 무슨 반응을 보이는가?
- 사용 흐름도: 사용자 입장에서 **사용 흐름**이 어떻게 진행되는가? (무슨 화면에서 무슨 입력을 받으면 무슨 화면으로 넘어가는가?)

설계서: 그것을 **어떻게^{how}** 만들 (즉, 실제로 구현할) 것인가? — **개발자 관점** (내부)

- 어떤 자료형/클래스의 어떤 변수/인스턴스들을 준비해야 하는가? 이들 중 무엇을 리터럴로 하드 코딩하고, 무엇을 불변의 상수^{constant}로 정의하고, 무엇을 변수에 담아야 하는가?
- 어떤 기능을 따로 함수나 서브루틴으로 빼 두어야 하는가? 함수로 만든다면 각각 어떤 타입의 매개변수들이 필요하고, 리턴 타입은 무엇인가? 함수에 부작용^{side effect}을 허용하거나 활용할 것인가?
- 어떤 변수가 지역 변수고 어떤 변수가 전역 변수인가? 전역 변수를 누구누구가 공유할 것인가?
- 순서도: 프로그램 내부에서 **계산 흐름**이 어떻게 진행되는가? (분기 지점과 조건은 무엇이고, 루프는 어디서 얼마나 돌아야 하는가?)

기획(요구분석)서 vs. 설계서: 독자의 차이

#3-rev1
기획서

차리서

일정

요구사항
분석서와의
비교설계서와의
비교

기획서의 품질

오류 처리

Mockup과
화면/UI

사용 흐름도

기타 사항

기획(요구분석)서의 독자들: 비 개발자(요구자, 투자자, 경영자²⁾)도 반드시 알아볼 수 있어야 함

- (내부 단계의) 검사자: 자체 모순이 없는지 확인함.³⁾
- (이전 단계의) **요구자**: 요구사항을 보고, 자신의 의도 대로 쓰여있는지 확인함.
- (외부 단계의) 투자자: 기획 내용을 보고, 투자하면 이익을 볼만한지 확인함.
- (외부 단계의) 경영자: 기획 내용을 보고, 진행시키면 회사에 도움이 될지 확인함.
- (다음 단계의) **설계자**: 기획 내용을 보고, 기획 대로 설계함.
- (최종 단계의) **검사자**: 최종 구현물이 기획서에 온전히 부합하는지 확인함.

설계서의 독자들: 비 개발자는 못 알아보는 것이 정상

- (내부 단계의) 검사자: 설계에 자체 모순이 없는지, 설계서가 기획(요구분석)서에 부합하는지 확인⁴⁾
- (다음 단계인) **구현자**: 설계도를 보고, 설계도 대로 구현함.
- (최종 단계의) **검사자**: 최종 구현물이 설계서에 온전히 부합하는지 확인함.

구현자가 요구분석/기획서를 직접 읽고 곧바로 (설계 없이) 코드로 구현하는 방식은 바람직하지 않지만...
→ 현실에서는 흔히 일어남. 이 과목에서도 이렇게 진행.

²⁾ 사실은, 투자자/경영자에게 보이는 용도의 “상업적 기획서”는 이 수업의 “기술적 기획서”와는 성격이 다를 수 있음.

³⁾ 검사자는 (이 수업에서 그렇듯이) 작성자 자신이나 자기 팀원일 수도 있지만, 때로는 다른 직원이나 다른 팀일 수도 있음.

⁴⁾ 기획(요구분석)서 작성자가 설계서를 읽을 기술적 소양도 있을 경우, 기획/요구 부합성 검사에 작성자도 동참

기획(요구분석)서 vs. 설계서: 화면/UI는 어디에?

#3-rev1
기획서

차리서

일정

요구사항
분석서와의
비교

설계서와의
비교

기획서의 품질

오류 처리

Mockup과
화면/UI

사용 흐름도

기타 사항

Q: 화면/UI ‘설계’는 설계서에 쓰는 게 맞나요?

A: 아니오! 화면/UI는 (비록 흔히 설계^{design} 라고도 불리지만) **기획서**에 쓰는 게 맞습니다.

- 화면/UI도 결국 프로그램의 (사용자가 밖에서 다루고 볼 수 있는) 외부 입/출력의 일종
- 요구분석/기획서에 쓸 때에는 **화면/UI ‘구성’**이라는 표현이 (설계나 디자인이라는 표현보다) 더 적절
- 뒤에서 설명할 mockup이 바로 화면/UI 구성을 기획서에 담는 방식들 중 하나

사실은, 각 단계의 구체화 정도와 관점에 따라, 간혹 이를 설계로 간주할 수도 있긴 있음:

- 요구자: “뒤로가기를 가장 자주 써요. 편한 게 좋아요. 오른손잡이고 한 손으로만 써요. 끝.”
- 분석가: “오른손으로 폰을 들었을 때 오른손 엄지로 가장 편하게 터치할 수 있는 화면 상의 위치에 뒤로가기 버튼을 배치한다.”
- 설계자: “여기(화면 하단, 약간 우측)에 이런 크기/모양으로 뒤로가기 버튼을 그린다.”

좋은 기획 (요구분석) 서의 요건 (검사자 관점에서, 문서 자체로서)

무모순성 논리적으로 틀리거나 서로 충돌하는 요소가 없을 것

- 반례: “입력은 네 개 이하의 숫자들이어야 합니다. 그리고, 입력은 65535보다 커야 합니다.”

명확성 하나의 서술이 둘 이상의 서로 다른 의미로 해석되지 않을 것
(특히, 작성자의 의도보다 더 포괄적/제한적인 의미로도 해석되지 않도록 주의)

- 반례: “앞뒤의 공백을 허용합니다.”
 - 한 개만 허용하나요, 여러 개도 허용하나요?
 - 양쪽 다 있어야 허용하나요, 한 쪽만 있어도 허용하나요, 한 쪽만 있어야 허용하나요?
- 반례: “메뉴를 표시합니다.”
 - 프로그램 (기능 선택) 메뉴요, (판매할) 음식 메뉴요?
- 반례: “네 자리 정수”
 - 0042는 네 자리 정수인가요, 두 자리 정수인가요?

완비성 확실히 정해야 하는 모든 사항들을 빠짐 없이 전부 서술할 것

- 예: 정수 입력시 선행 0(들) 허용 여부, 길이 제한, 공백 허용 여부 등

간결성 짧고 단순 명료하게 서술할 수 있는 사항을 괜히 길거나 복잡하게 꼬아서 서술하지 않을 것

좋은 기획(요구분석)서의 요건 (설계자, 구현자, 경영자, 투자자 관점에서)

실현성 실제로 설계/구현이 가능할 것^{feasibility}

- 반례: 지정한 일시(미래)와 장소에 일어날 일을 미리 영상으로 보여주는 프로그램 기획서
- 실현 불가능한 것과 논리적으로 ‘틀린’ 것은 (비슷해 보일 수 있지만) 별개의 개념
- 실현 불가능한 것과 (아래에서 설명할) ‘어려운’ 것도 서로 별개의 개념

난이도 현실적으로 주어진 시간, 비용, 인력, 능력 기준으로 설계/구현이 너무 어렵지 않을 것

- 반례: (학부생 6 명, 한 학기 기준) 자동차 Level 5 자율 주행 프로그램 기획서
- 일단 실현 가능한 것들에 대해서만 난이도를 따짐

수익성 이 문서에서 말하는 물건을 만들면 돈이 될 것인가?

기술성 우리만의 독자 기술로 보호하여 장기간 경쟁 없이 수익을 유지할 수 있을 것인가?

독창성 지적 재산권을 확보하고 로열티를 받아먹을 수 있을 것인가?

위 요건들 중:

- 이 과목에서는, 당연히 **실현 가능**하고 구현 **난이도**가 **적당한** 기획서를 만들어야 함
- 이 과목에서는, 수익성, 독창성, 기술성은 **전혀 안** 따짐!
- 현실에서는 (특히 직장 상사나 투자자에게 ‘제안서’로서의 기획서를 쓸 때에는) 모든 요소가 아주 중요

좋은 요구사항 분석서의 요건 (요구자 관점에서)

요구 부합성 요구자의 진짜 의도와 다르게 (혹은 적어도, 모자라게) 서술된 사항이 없을 것. (요구자가 읽자마자 모든 사항들 각각에 대해

“예, 제가 의도했던 게 분명히 이거 맞아요. 적어도 제가 의도했던 건 다 들어있어요.”
라고 **즉각적으로, 단호하게** 말하는 분석서가 좋은 분석서)

종지 **않은** 분석서의 단계:

- **중:** 요구자가 분석서 읽고 (즉각적으로, 단호하게) “**아니요, 제 말 뜻은 이게 아니었어요!**”
 - 문서가 그나마 명확하고 간결하기에 나올 수 있는 반응
 - 설계/구현으로 넘어가기 전에, 잘못된 부분을 확실히 알아내고 재빨리 수정 가능
- **하:** 요구자가 분석서 읽고 “**잘 모르겠어요.**” 혹은 “**음... 아마 대충 맞는 것 같아요.**”
 - 문서가 복잡해서 요구자가 제대로 해석하지 못했을 가능성
 - 잘못된 부분을 제대로 짚어내기 힘들어짐
- **최하:** 요구자가 분석서 읽고 “**예, 맞아요!**” → 구현물 보고 (진심으로) “**어? 이거 아니었잖아요?**”
 - 문서가 불명확해서 (작성자과) 다르게 해석하고 동상이몽
 - 잘못된지 모르고 설계/구현한 후에야 잘못된 줄 알게 됨 (시간, 비용 손해)

[참고] 기술/학술 문서 vs. 문학 작품

#3-rev1
기획서

차리서

일정

요구사항
분석서와의
비교설계서와의
비교

기획서의 품질

오류 처리

Mockup과
화면/UI

사용 흐름도

기타 사항

기술/학술 문서에서는 (문학 작품과는 달리):

- 널리 공통적으로 확고하게 정의된 **수식**이나 **기호**, **도식** ^{diagram} 등이 자연어 문장보다 유리한 경우가 많음
- 자연어인 경우에도, **개조식 목록**이나 **표** 등이 길고 장황한 서술형 문장들보다 유리한 경우가 많음
- 서술형 문장들도 **무미건조**한 게 유리 (불필요한 화려한 수사, 도치, 강조, 은유, 관용구 등은 **피할 것!**)
- 감각(느낌)에 호소하려고 시도하지 말고, 철저하게 **이성에만 호소**할 것!
 - 감각에 호소하려 들면 (독자/청중에 따라서는) 오히려 의심하고 경계 태세를 취하며 읽을/들을 수도 있음 (예: 학회 발표)
 - 감각적인 글꼴보다 **가독성** 높은 글꼴! / 화려하고 장식적인 도식보다 **명시성** 높은 도식!
(알아보기 힘들어도 내용만 맞으면 된다는 뜻은 아님! 한 눈에 혼동 없이 쉽게 알아볼 수 있어야 함!)

단, 기술 문서이기도 하지만 동시에 ‘상업’ 문서이기도 한 경우 (독자가 경영자, 투자자, 고객 등인 경우):

- 수식이나 기호를 (그나마 다행으로) 못 알아보거나 (심지어) 잘못 전달될 수도 있음에 유의
- 화려한 수사나 도치, 강조, 은유, 관용구 등을 활용한 ‘멋들어진/인상적인’ 표현 하나가 간혹 더 강력할 수도 있음
- 느낌(감각)에 호소하는 데에 (당장은) 대체로 반발하지 않으며, 일단은 통할 수도 있음
→ 물론, 합리적이지 않은 내용을 감각적으로 포장한 경우, 결국 나중에 언젠가는 댓가를 치르게 됨

(프로그램) 오류의 종류

#3-rev1
기획서

차리서

일정

요구사항
분석서와의
비교설계서와의
비교

기획서의 품질

오류 처리

Mockup과
화면/UI

사용 흐름도

기타 사항

(절대적이진 않지만, 대체로) 아래 목록에서 번호가 클수록 (즉, 아래에 있을수록) 악성 오류:

- 1 정확히 뭐가 문제인지 구체적인 문구를 보여주며 확실히 경보를 보낸 상태로, 내성^{tolerance}을 발휘해 ‘정상에 준하는 우회/대체^{fallback} 작동’을 지속하는 경우
- 2 정확히 뭐가 문제인지 구체적인 메시지를 보여준 후, 작동을 멈추거나 특별한 ‘비상 동작’을 하는 경우
- 3 문제가 있다는 사실만 간단히 통보하고, 내성을 발휘해 정상에 준하는 대체 작동을 계속하는 경우
- 4 문제가 있다는 사실만 간단히 통보하고, 작동을 멈추거나 특별한 비상 동작을 하는 경우
- 5 아무 통보 없이 (정상 종료한 것과 구별되지 않거나 힘들게) 작동을 멈추는 경우
- 6 아무 통보 없이 (정상 작동 중인 것과 구별되지 않거나 힘들게) 오작동을 계속 하는 경우
- 7 아주 가끔 특정한 상황에서만 아무 통보 없이 오작동을 하며, 그 외에는 정상 작동하는 경우
- 8 아주 가끔 불특정한 상황에서 오작동을 하는데, 그 상황과 오작동을 재현하기 힘든 경우 (하이젠버그)

[참고] 내성의 예:

- 일부 프로세스가 멈췄지만, 운영체제가 그에 굴하지 않고 (해당 프로세스만 죽이고) 계속 정상 작동
- 엔진들 중 하나가 터졌지만, 항공기가 그에 굴하지 않고 남은 엔진으로 정상 비행
- F15 lands with one wing

이 과목에서의 (프로그램) 오류 처리

#3-rev1
기획서

차리서

일정

요구사항
분석서와의
비교설계서와의
비교

기획서의 품질

오류 처리

Mockup과
화면/UI

사용 흐름도

기타 사항

구체적인 오류 메시지 출력

- “오류 발생” < “E028392 번 오류” < “입력 중에 숫자가 아닌 문자(‘h’)가 들어있음”
- 구체적인 오류 메시지는 사용자 뿐만 아니라 개발자 본인들에게도 매우 중요 (물론 디버거가 더 좋음)
- 무조건 길고 장황한 메시지가 아니라, **정확히 뭐가 문제인지 구체적으로** 짚는 것이 중요!
- **기획서**에도 오류 메시지가 출력될 ‘장면’들을 (그 메시지의 ‘내용’과 함께) 명시
 - 특히, 오류가 출력될 ‘**장면**’들은 설계/구현 단계에서 바꾸면 안 됨 (처음부터 **빠짐없이 명시**)
 - 단, 추후 설계/구현시 오류 메시지 ‘내용’의 **표현 방식만 살짝** 바꾸는 건 용인

적절한 비상 동작 혹은 종료

- 적절한 비상 동작: 입력 다시 받기, 다른 선택지 주기, 상위 메뉴로 되돌아가기 등
- 종료가 필요한 경우라고 판단되면 종료해도 됨 (상황에 맞다면, 비상 동작 vs. 종료 간 점수 차이 없음)
- **기획서**에도 정확히 명시!
 - 비상 동작이든 종료든, **기획서에 명시한 대로** 실제 구현물이 동작/정지해야 함

※ 내성 및 정상에 준하는 우회/대체 작동 지속은 이 과목에서는 대체로 요구하지 않음

문법 오류 vs. 의미 오류

#3-rev1
기획서

차리서

일정

요구사항
분석서와의
비교설계서와의
비교

기획서의 품질

오류 처리

Mockup과
화면/UI

사용 흐름도

기타 사항

문법^{syntax} 과 의미^{semantics} 는 100% 칼같이 구분되지는 않지만

- 대부분의 경우 상식적인 선에서 구별 가능 (문법은 문자열 속 문자들의 종류와 배치, 갯수 등)
- 간혹, 본인들이 의도한 사항들 중 어디까지를 문법 규칙으로 표현하고 어디서부터를 의미 규칙으로 표현할지 본인들이 정하기 나름인 경우도 있음
 - 문법 규칙을 뽁뽁하게 잘 정하면, 어느 정도까지는 ‘문법 규칙이 의미 규칙을 포함하게’ (문법 규칙에 맞으면 의미도 자동적으로 맞게) 정할 수 있음.
 - 단, 사안에 따라서는 완벽하게 문법으로만 표현하기는 어렵거나 불가능 (날짜의 윤년 처리 문제 등)

	의도	문법 오류	의미 오류
사칙연산 산술식 (중위 연산자와 괄호))46+abc*---((3 + 5) / (2 - 2) (0으로 나눔)
날짜 (YYYY-MM-DD 형식)		202-10A1*-	2021-02-30 (2월에 30일 없음)
일기장 용 날짜 (YYYY-MM-DD 형식)			2027-10-28 (미래)
가입시 ID 결정 (로마자만, 3~12글자)		12monkeys	exid (시스템에 이미 해당 ID 존재)
메뉴 항목 번호 (주어진 번호만, 숫자만)		1.00	807 (메뉴에 없는 항목 번호)

(넓은 의미의) Mockup

#3-rev1

기획서

차리서

일정

요구사항
분석서와의
비교설계서와의
비교

기획서의 품질

오류 처리

Mockup과
화면/UI

사용 흐름도

기타 사항

아직 만들어지지 않은 어떤 것의 (예정된, 완성 후의) 겉모습을 미리 보여주는 **가상의 예시**

- (완성품이 아니면서) 일부분은 완성품처럼 생겼고 동작하며, 그 부분만 예시로 보여주기 위한 모형
- “나중에 완성되면, 이 부분은 이렇게 보이고 이렇게 동작할 예정입니다.”라는 의미
- 겉으로 드러나 보이는 부분/기능 만큼은 (향후 만들) 실물과 똑같아야 함
 - 바꿔 말하면, 일단 mockup을 보여준 부분 만큼은 나중에 최대한 그대로 만들어야 함!
- 겉으로 드러나지 않는 부분/기능은 실물과 완전히 달라도 됨
- mockup과 사기의 차이:
 - mockup: “이 부분만 이렇게 보이게 해둔 겁니다”, “이건 mockup입니다”
 - 사기: “이 부분 외에 안 보이는 다른 부분들도 모두 실제와 같이 동작하고 있습니다”
 - mockup으로 보여준 부분을 나중에 (심하게) 다르게 만드는 것도 일종의 사기

예: 모델하우스

- 전등이 켜지지만, 한전 전기 끌어오고 두꺼비집 공사를 완료한 게 아니라, 벽 뒤에 배터리가 있음
- 가스 레인지가 켜지지만, 도시가스 끌어오고 계량기 공사를 완료한 게 아니라, 벽 뒤에 LPG 통 있음
- 수도꼭지가 있지만, 간이 물탱크와 펌프로는 수압이 너무 낮아서 (안 좋아 보일까봐) 일부러 물탱크도 빼버리고 그냥 “수도꼭지는 위치/형태만 참고하시라는 mockup입니다” 해버림

(일반적인 컴퓨터 프로그램에 대한) Mockup

GUI의 경우:

- 내부 엔진 없이 GUI 캡데기만 약식으로 구현 → 스크린샷 → 기획서에 삽입
- 그림판 등으로 가짜 GUI 그리기 → 기획서에 삽입
- 메세지 상자에 출력되는 문자열을 기획서 본문에 직접 타이핑
- 텍스트 입력 필드에 입력하는 문자열을 기획서 본문에 직접 타이핑

TUI, CLI의 경우:

- 내부 엔진 없이 터미널 출력만 약식으로 구현 → 스크린샷 → 기획서에 삽입
- 텍스트 편집기로 터미널 출력처럼 생긴 (여러 줄의, 2차원적) 문서 작성 → 스크린샷 → 기획서에 삽입
- 기획서 본문에 고정폭 글꼴로 터미널 출력처럼 생긴 (여러 줄의, 2차원적) 문자들 직접 타이핑
- 데이터 (텍스트) 파일의 구문 형식 예시도 일종의 mockup → 스크린샷 삽입 혹은 텍스트 복붙
- (정상/오류) 출력이나 키 입력 문자열 한 줄도 일종의 mockup → 기획서에 직접 타이핑

(이 과목에서의) Mockup: 기획서에 화면/UI 구성 및 입출력 예시 기재

GUI, TUI의 경우: 화면 배치 mockup을 충분히 보이고 조작 방법도 구체적으로 명시해야 함

CLI의 경우: 출력 위치가 항상 마지막 줄 커서 위치고 자동 스크롤되므로, ‘화면 배치’ mockup은 덜 필요

■ mockup이 권장되는 경우:

- 복잡한 구성의 기능 선택 메뉴나 정렬된 표 형식의 결과 등 ‘설명’만으로는 곤란한 출력
- 사용자가 직접 편집 가능한 텍스트 데이터 파일의 (올바른/틀린) 내용 형식 예시
- 복잡한 규칙을 요구하는 키 입력 문자열의 (올바른/틀린) 예시

■ mockup이 (있어도 되지만) 굳이 필요 없는 경우:

- 프롬프트나 간단한 메시지 출력
- 아주 짧고 간단한 규칙의 키 입력 문자열 (예: “한 글자 이상의 연이은 숫자(들) 입력”)
- 그 외에, 글로 설명할 수 있는 출력 (예: “...하고나면, 전체 회원의 ID 목록이 알파벳 사전식 순서로 한 줄에 하나씩 출력되고, 그 밑에 ID 하나를 입력받는 프롬프트가 표시된다.”)

공통: 명시한 mockup 화면과 조작 방법에 부합하도록 설계/구현해야 함

- 단, 버튼의 크기나 오류 메시지의 세부 문구 등 아주 사소한 변경은 허용

사용 흐름도

#3-rev1
기획서

차리서

일정

요구사항
분석서와의
비교설계서와의
비교

기획서의 품질

오류 처리

Mockup과
화면/UI

사용 흐름도

기타 사항

사용자 입장(관점)에서 프로그램 외부를 관찰한 흐름을 나타낸 도표


- 프로그램 내부의 계산/데이터/논리 흐름이 **아님** → 이걸 (설계서에 들어갈) 순서도!
- 사용 흐름도에 포함되는 요소들: 시작, 종료, 사용자 키 입력 장면, 데이터 파일 입출력, (중요한) 화면 출력 장면, 사용 흐름이 갈라지는 분기점 등
- **아주 드물게**, 프로그램의 사용 흐름이랄 게 (분기, 반복 등이) **전혀 없을** 경우, 기획서에 넣지 않아도 됨
 - 예: 실행시키면, 현재 일시를 화면에 한 줄로 출력한 후 곧바로 종료되는 프로그램
 - 예: 0개 이상의 명령행 인자⁵⁾(들)을 주어 실행시키면, 각 인자들을 한 줄에 하나 씩 화면에 출력한 후 곧바로 종료되는 프로그램
 - 실질적으로, 이런 주제는 (입력 규칙 등 다른 훈련 요소들이 유난히 훌륭하지 않는 한) 대부분 부적합

사용흐름도의 기본 구성 요소: 노드와 간선

노드 node, vertex 상태나 작업을 나타내는 상자: 출력, 판정, (파일 입출력 등의) 동작 등**간선** edge, arc 노드 간의 전이를 나타내는 선 (항상 방향이 있음)

⁵⁾터미널 프롬프트에 실행파일 이름을 입력하여 실행시킬 때, 실행파일 이름 옆에 나란히 입력하는 인자들. 다음과 같이 입력하면:

```
D:\work\SoPrj> myProgram.exe foo bar baz
```

프로그램에 전달되는 명령행 인자들은 foo와 bar, baz임. 이렇게 전달된 인자들은 프로그램 내부에서 특정한 방식으로 받아서 처리하며 각 언어마다 이를 위한 방법을 제공하는데, C 언어의 경우 main 함수의 매개변수들이 바로 이 인자(문자열)들의 갯수와 배열임. 

사용 흐름도 예시

#3-rev1
기획서

차리서

일정

요구사항
분석서와의
비교

설계서와의
비교

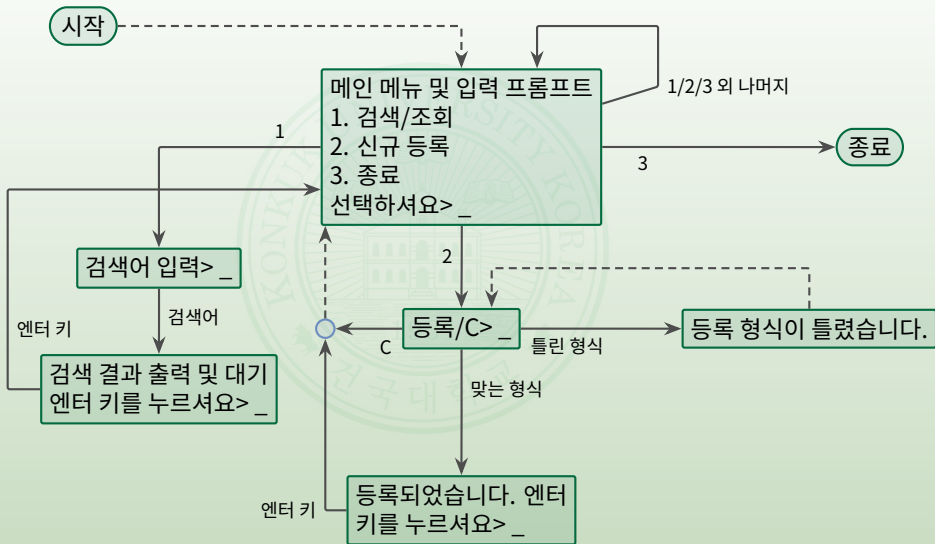
기획서의 품질

오류 처리

Mockup과
화면/UI

사용 흐름도

기타 사항



사용 흐름도 작성시 주의 사항

#3-rev1
기획서

차리서

일정

요구사항
분석서와의
비교설계서와의
비교

기획서의 품질

오류 처리

Mockup과
화면/UI

사용 흐름도

기타 사항

- **[필수]** 시작 노드는 반드시 딱 한 개, 종료 노드는 반드시 한 개 이상 있어야 함⁶⁾
- **[필수]** 모든 간선에는 반드시 **단방향 화살표**가 있어야 함 (끝나는 부분에 화살촉)
- **[필수]** 한 노드에서 나가는 간선이 둘 이상이면:
 - 모든 나가는 간선에 ‘어떤 경우(입력/판정 등)에 그 간선을 타고 나가는지’를 나타내는 **표찰label**을 (해당 간선에 바짝 붙여서, 출발 노드에 가까운 쪽에) 반드시 표시
 - 모든 나가는 간선들의 **표찰들의 합집합**은, 해당 노드에서 발생/판정 가능한 이벤트들의 **전체 집합**이어야 함
- 한 노드에서 나가는 간선이 한 개 뿐일 때, 그 간선에 표찰이:
 - 있으면, 그 노드에서 그 표찰을 입력받거나 그 표찰로 판정되는 등의 이벤트가 있어야만 전이
 - 없으면, 그 노드에서 할 일이 끝나자마자 (아무 이벤트가 없어도) 해당 간선을 따라 무조건 자동 전이⁷⁾
- 노드 모양에는 (순서도에서 주로 사용하는) 통상적인 관례가 존재하지만, 이 과목에서는 굳이 세간의 관례를 뽁뽁하게 따를 필요 없음
 - 기능적으로 서로 달리 구별해야 할 노드들이 실제로 잘 구별되고
 - 기능적으로 동일/유사한 노드들이 일관성 있게 표현되기만 하면 됨

⁶⁾종료 노드도 가능한 한 딱 한 개만 있는 것이 더 좋지만, 부득이한 상황이면 둘 이상도 허용됨.

⁷⁾자동 전이 간선은 표찰이 없는 것만으로도 이미 구별되지만, 점선이나 파선으로 그려서 더 잘 구별되게 만들면 좀 더 좋음. (단, 자동 전이 간선에 점선/파선을 쓰는 건 이 수업 밖에서는 통용되지 않는 비표준 표현법임.)

간선끼리의 접촉: 횡단과 합류

#3-rev1
기획서

차리서

일정

요구사항
분석서와의
비교설계서와의
비교

기획서의 품질

오류 처리

Mockup과
화면/UI

사용 흐름도

기타 사항

간선의 횡단: 두 간선이 서로의 흐름과 무관하게 시각적으로만 교차되는 것

- 아무 표시 없이 그냥 교차되게 그릴 것 (방향이 있으니 gap이나 jog 불필요)
- 단, 교차점에서 미분가능해야 함 (교차점에서 꺾이면 안 됨: 서로를 직선이나 일정한 곡률로 통과)
- 한 지점에서 셋 이상의 간선이 서로를 횡단하게 그리지 말 것 (둘까지만 허용)

간선의 합류: 둘 이상의 간선의 흐름이 모여서 한 흐름으로 합쳐지는 것

- 합쳐지는 자리에 내용 없는 빈 ‘합류 노드’를 그릴 것 (작은 동그라미 권장)
- 둘 이상의 간선들이 빈 노드로 들어가고, 그 빈 노드에서 하나의 간선이 나온다는 개념
- 빈 노드로 들어가는 간선들 끝에 화살촉 필수 (이 화살촉으로 흐름 방향 구별)
- 빈 노드에서 나가는 간선은 반드시 딱 한 개고, 자동 전이임 (표찰 있으면 안 됨)



횡단에
'gap' 불필요



횡단에
'jog' 불필요



횡단은 그냥
이거면 충분!



대신, 횡단은
절대 꺾이면 **안 됨!!**



대신, 합류는
반드시 이렇게!

간선끼리의 접촉: 분기

#3-rev1
기획서

차리서

일정

요구사항
분석서와의
비교설계서와의
비교

기획서의 품질

오류 처리

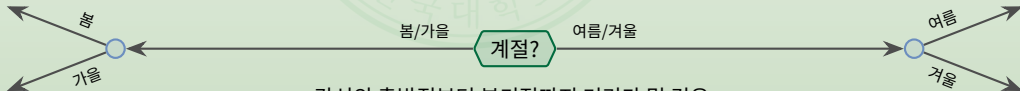
Mockup과
화면/UI

사용 흐름도

기타 사항

간선의 분기: 한 간선의 흐름이 여러 갈래로 나뉘는 것

- 문제점: 노드에서 ‘어떤 간선을 타고 나갈지’를 출발시에 (노드 근처 표찰로) 빨리 확인할 수 없음
- 그림의 복잡도 때문에 부득이한 경우에만 허용 (웬만하면 분기는 피할 것)
- 일단 빈 ‘분기 노드’로 들어갔다가 거기서 다시 **각각의 표찰을 달고** 나가는 방식으로 그림

가능한 한
분기 없이 이렇게!부득이 분기할 경우,
분기 직후에 표찰 필수!간선의 출발점부터 분기점까지 거리가 멀 경우,
출발점 근처에도 분기될 ‘통합 표찰’들 필수!

(페이지) 연결자 노드

#3-rev1
기획서

차리서

일정

요구사항
분석서와의
비교설계서와의
비교

기획서의 품질

오류 처리

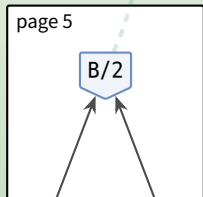
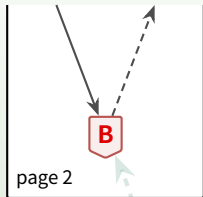
Mockup과
화면/UI

사용 흐름도

기타 사항

스타크래프트의 땅굴관(Nydus canal)과 거의 같은 개념 (종이가 점막(creep))

- 단, 단방향으로만 작동하며, 양방향으로는 쓸 수 없음
- 노드 형태는 관례적으로 야구 홈베이스 모양
- 다른 페이지에서 건너뛰어와서 착지하는 ‘도착 연결자’는
 - 노드 내용(이름): 주로 로마자 대문자(A~Z) 한 글자
 - 한 페이지 속의 모든 ‘도착 연결자’들끼리는 이름이 서로 달라야 함 (서로 다른 페이지에는 서로 이름이 같은 도착 연결자도 존재 가능)
 - 노드에서 나가는 간선은 꼭 있어야 하고, 딱 하나고, 반드시 자동 전이
 - 노드로 들어오는 간선은 있어도 되고 없어도 됨
 - 모종의 방법(색상 등)으로 눈에 잘 띄게 표시 (뒤져서 찾아야 하기 때문)
- 다른 페이지로 건너가기 위해 밟고 뛰어오르는 ‘출발 연결자’는
 - 노드 내용(이름): 가려는 도착 연결자의 이름과 페이지 번호 조합 (예: 2 페이지의 [B]로 건너뛰기 위한 출발 연결자 이름은 [B/2])
 - 서로 이름이 같은 ‘출발 연결자’들은 어디에든 얼마든지 등장할 수 있음
 - 노드에서 나가는 간선이 있으면 안 됨
 - 노드로 들어오는 간선은 하나 이상 있어야 함
 - 페이지 번호가 자기 페이지일 수 있음: 자기 페이지의 다른 지점으로 건너뛸



사용 흐름도: 작성 및 제출 팁

#3-rev1
기획서

차리서

일정

요구사항
분석서와의
비교

설계서와의
비교

기획서의 품질

오류 처리

Mockup과
화면/UI

사용 흐름도

기타 사항

Q: 손으로 그려도 되나요?

A: [Diagrams.net \(draw.io\)](#)이나 파워포인트 등 벡터 기반 편집기로 작성해서 PDF로 저장하기를 가장 권장하지만, 뭔가 여의치 않으면 그림판 등 화소^{pixel} 기반 편집기로 작성해서 PNG 이미지⁸⁾로 저장해도 되고, 그조차 여의치 않다면 손으로 그려서 스캔/촬영해도 됩니다. (단, 도형/글자들을 확실히 잘 알아볼 수 있어야 하고, 이 조건만 지킨다면 점수 차이는 없습니다.)

Q: 흐름도가 너무 크고 복잡해서 기획서 한 페이지에 넣기 곤란합니다.

A: 다음 중 한 가지 방법을 사용하세요. (위쪽에 먼저 나오는 방법을 더 권장합니다):

- 앞 슬라이드에서 소개한 **페이지 연결자**를 써서, 글자를 충분히 읽을 수 있는 크기로 **여러 페이지에 나눠서** 그리고, 기획서 한 페이지마다 그림 한 페이지 씩 **삽입**하세요.
- 흐름도를 **벡터** 도형/글꼴로 작성해서 **PDF**로 export하고, 해당 PDF를 기획서 한 페이지에 **삽입**하세요. (확대시 화질/가독성 확보)
- **화소** 편집기나 **손으로**, 글자를 충분히 읽을 수 있는 크기로 그린 후, 저장한 (혹은 스캔/촬영한) **이미지** 파일을 기획서에 삽입하지 말고 그냥 **별도 파일**로 (기획서 파일과 함께) 업로드하세요.

⁸⁾ 사진은 어차피 연속적인 변화가 많아서 JPEG 등 손실 압축 포맷으로 저장해도 화질 열화가 크게 눈에 띄지 않지만, 세밀한 다이어그램과 작은 글자들을 JPEG으로 저장하면 화질 열화에 의해 가독성이 크게 줄어듭니다. 다이어그램은 PNG 저장을 추천합니다. 🔍🔍🔍

표제heading

#3-rev1
기획서

차리서

일정

요구사항
분석서와의
비교설계서와의
비교

기획서의 품질

오류 처리

Mockup과
화면/UI

사용 흐름도

기타 사항

목차에 포함되는, 각 절 및 (재귀적으로) 그 하위 소절들의 번호와 제목⁹⁾

■ **표제 번호 형식:** 해당 번호 속에 **모든 ‘상위’ 표제 번호들이 포함**된 형식 강력 권장

- 그 표제 번호 하나만 보고도 문서 전체 구조 중 정확히 어느 부분인지 즉시 파악 가능 (상위 번호가 들어있지 않으면, 문서를 거슬러 올라가면서 뒤져서 파악해야 함)
- 특정 표제를 찾아야 할 경우, 눈 앞에 어떤 표제가 보이더라도 그보다 앞에 있는지 뒤에 있는지 바로 알 수 있음

■ 하위 표제 및 그 표제에 딸린 본문(내용)의 **들여쓰기**를 권장하지 **않음**

- 표제 번호를 위 권장 형식으로 붙이면 들여쓰기가 필요 없음
- 본문(내용)을 쓸 공간(좌우 폭)이 불필요하게 줄어듦

강력 권장	다소 권장	비추천	금지 (감점!)
2 ...	II ...	2 ...	II ...
2.1 ...	II.1 ...	2.1 ...	1 ...
2.1.1 ...	II.1.A ...	2.1.1 ...	A ...
2.1.2 ...	II.1.B ...	2.1.2 ...	B ...
2.2 ...	II.2 ...	2.2 ...	2 ...

⁹⁾이 과목에서 다루는 기획서 정도 규모의 문서에서는 대체로 ‘장’ 단위를 사용하지 않으며 최상위 표제가 ‘절’입니다.

#3-rev1
기획서

차리서

일정

요구사항
분석서와의
비교설계서와의
비교

기획서의 품질

오류 처리

Mockup과
화면/UI

사용 흐름도

기타 사항

연속으로 나란히 나열된 항목^{item} 들

- 한 표제에 딸린 본문 내용 내부에 첫 항목부터 마지막 항목까지 모두 나열됨 (표제를 경계로 잘리지 않음)
- (필수는 아니지만) 대체로 한두 페이지 이내에 모두 나열되는 분량 → 만일 목록 전체 규모가 더 크고 한 항목 당 내용도 충분히 길면, 목록이 아닌 하위 표제들로 바꾸기를 권장함
- 목록의 종류와 용도
 - 서술 목록** 용어를 정의하거나, 각 항목별로 서술적인^{descriptive} 제목이 필요할 때 (예: 지금 이 목록)
 - 번호 목록** 진행 단계의 순서를 표시하거나, 특정 항목(들)을 다른 곳에서 쉽게 지목(참조)해야 할 때
 - 무순 목록** 순서도 없고, 항목별 제목이나 참조 번호도 필요 없을 때
- 목록 항목 속에 하위 목록이 (재귀적으로) 포함될 수 있으며, 이때 포함된 깊이에 따라:
 - **들여쓰기**를 명확하게 하고 (필수)
 - 번호/볼릿을 서로 다른 양식으로 붙일 것 (권장)

번호 목록의 각 항의 번호는 (표제의 경우와는 달리) **상위 표제 번호 없이 단순 번호로** 붙이길 권장:

- 예: 2.4.1 절 본문의 번호 목록의 각 항목 번호는 “2.4.1.1”, “2.4.1.2”, ... 대신 그냥 “1”, “2”, ...를 권장)
- 만일 한 표제 속에 서로 독립적인 번호 목록이 둘 이상 존재하면? → 단순 번호여야 안 꼬임!

목차와 참조

#3-rev1
기획서

차리서

일정

요구사항
분석서와의
비교설계서와의
비교

기획서의 품질

오류 처리

Mockup과
화면/UI

사용 흐름도

기타 사항

목차: 문서 속 모든 표제들 각각의 “표제 번호, 표제 제목, **페이지 번호**”들의 나열

- 문서 초반에 목차를 **반드시 기재**할 것 (**페이지 번호**도 필수!)
- 목차에서는 하위 표제의 들여쓰기 허용 (꼭은 충분, 가독성 유리)
- 목록은 목차에 포함되지 **않음** (목록은 본문 내용의 일부로 간주)
- 하이퍼링크는 (있으면 좋지만) 없어도 됨

참조: 문서 속의 특정 요소(주로 표제, 표, 그림, 식 등)를 문서 속 다른 부분에서 지목하여 말하는 것

- 표제를 참조할 때: 표제 번호는 필수, 표제 제목은 권장, (표제가 있는) 페이지 번호는 옵션
 - 표제 제목만으로 그 표제 찾기는 어려움 vs. 표제 번호만으로는 그 표제 찾기 쉬움
 - 자주 편집하다보면, 표제 번호는 (레이블 기능 안 쓰면) 꼬일 수 있음 → 표제 제목 병기로 보강
 - 페이지 번호는 (레이블 기능 안 쓰면) 대체로 더 혼하게 꼬임
- 목록 항목을 참조할 때: 그 목록이 들어있는 표제와 항목 번호(혹은 제목)를 병기
 - (어디까지나) 제안 형식: “{표제 번호} 절 ‘{표제 제목}’의 (n 번째) 목록 중 {항목 번호} 항”
 - 예시: 2.4.1 절 ‘날짜 입력 문법’의 첫번째 목록 중 3항
- 표/그림/식 등을 참조할 때: 그 요소가 들어있는 표제와 요소 번호를 병기 (요소에 번호 필요)
- 하이퍼링크는 (있으면 좋지만) 없어도 됨

■ 목차에 표제별 페이지 번호를 일것 매겨놓고, 정작 **각 페이지에 페이지 번호**를 안 붙인 경우도 있었음

■ 문서 속 **용어**에 혼동이 없도록 (가능한 한 초반에 몰아서) **잘 정의**하고, **꼭 그렇게만 사용할 것**:

메뉴 프로그램 실행 중 고를 수 있도록 제시되는 명령이나 데이터, 혹은 하위 메뉴¹⁰⁾들의 목록

차림표 프로그램을 사용하는 가게에서 손님들에게 판매하는 음식들의 종류와 각각의 가격들의 목록

■ “...는 보장하지 않습니다.”

- 기획서를 쓸 때에는 **가능한 한** 긴지 아닌지 제대로 확인해서 “기다”, “아니다”라고 단언해야 함!
- 단, 그렇게 호언장담한 사항들은 반드시 정말로 그래야만 하므로, 현실적으로 도저히 제대로 확인하고 단언하기 어려운 사항에 한해서는 차라리 “보장하지 못한다”고 기재하는 게 (호언장담했다가 틀리는 것보다는) 나음!
- 그래도 역시, 너무 남발하지는 말 것!

■ **예시는 어디까지나 예시일 뿐!** (무조건 모든 요소를 항상 똑같이 따라하라는 의미가 아님!)

- 이 슬라이드나 별도로 소개할 예시용 기획서는 대부분 특정 주제/상황을 가정
- 예시로 보여주는 구성, 형식, 내용, 서술 방식 등은 대부분 이 특정 주제/상황에 특화됨
- 그런데, 각 팀별로 주제/상황이 모두 다름
- 따라서, 강의에서 “웬만하면 꼭 이렇게 하라”고 (강력히) 추천/권장한 요소들 외에는, 각자 **자기 팀 주제/상황에 더 적합한** 구성, 형식, 내용, 서술 방식이 떠오르면 그 방식을 사용할 것

¹⁰⁾재귀적 정의

<복제물에 대한 경고>

본 저작물은 **저작권법 제25조 수업목적 저작물 이용 보상금제도**에 의거, **한국복제전송저작권협회와 약정**을 체결하고
적법하게 이용하고 있습니다. 약정범위를 초과하는 사용은 저작권법에 저촉될 수 있으므로
저작물의 재 복제 및 수업 목적 외의 사용을 금지합니다.

2020. 03. 30.

건국대학교(서울)·한국복제전송저작권협회

<전송에 대한 경고>

본 사이트에서 수업 자료로 이용되는 저작물은 **저작권법 제25조 수업목적저작물 이용 보상금제도**에 의거,
한국복제전송저작권협회와 약정을 체결하고 적법하게 이용하고 있습니다.
약정범위를 초과하는 사용은 저작권법에 저촉될 수 있으므로
수업자료의 대중 공개·공유 및 수업 목적 외의 사용을 금지합니다.

2020. 03. 30.

건국대학교(서울)·한국복제전송저작권협회