야구 스코어보드

코드 발전 분석

2021658042 황보정

사용한 예제 : 응용8_2_1

2025-10-16

전체 아키텍처 변화

데이터 구조 확장

정적 배열 → 동적 배열

```
int baseball[12][2]={0};
// 12개의 고정 이닝
```

↓

```
struct GameData {
  vector<int> R[2], H[2], E[2], B[2];
  void push_one_inning(); // 연장 가능
}
```

장점: 연장전 유연 지원

렌더링 방식 구조화

좌표 기반 → 함수 기반

```
gotoxy(x, y);
printf("문자열");
// 개별 좌표 출력
```

↓

```
display_scoreboard();
// 표 기반 일관성 출력
```

장점: 코드 가독성/재사용성

프로그램 흐름 제어

단순 루프 → 모드 기반

```
while(1) {
  scanf 입력;
  게임 진행;
}
```

↓

```
switch(mode) {
  case 1: 수동 모드;
  case 2: 랜덤 모드;
  case 3: 수정 모드;
  case 4: 테마 모드;
}
```

장점: 명확한 상태 관리

입력 UX 변화

입력 방식 개선

scanf 입력

이전: 버퍼링 문제

```
scanf("%d", &input);
// 엔터 키 입력 필수
```

_getch() 즉시 반응

현재: 즉각적 처리

```
input = _getch();
// 키 입력 즉시 감지
```

개선된 사용자 경험

- 즉각적인 피드백 제공
- 엔터 키 입력 없이 바로 반응
- 사용자 실수율 감소

메뉴 및 안전 입력

표준 메뉴

이전: 기본 메뉴

```
printf("1. 입력\\n");
printf("2. 종료\\n");
```

친화적 메뉴

현재: 향상된 메뉴

```
printf("1. 입력 (ESC: 취소)\\n");
printf("2. 수정 (-999: 메뉴)\\n");
```

안전 입력 처리

scan_int_safe()
잘못된 입력 방지

```
if(scanf("%d", &input) != 1) {
  printf("잘못된 입력입니다.");
  return scan_int_safe();
}
```

메뉴 복귀 키 (-999)
任何时候 메뉴로 돌아가기

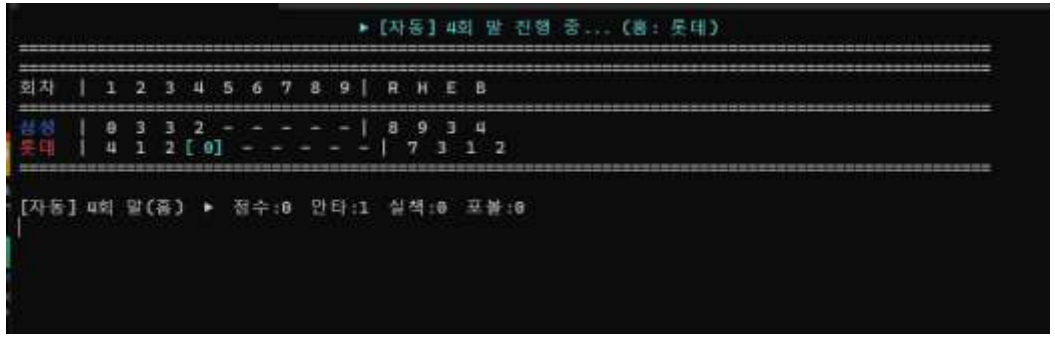
랜덤 시뮬레이션 모드

자동화된 시뮬레이션 흐름

반이닝 단위 자동 진행
1회초 → 잠시 표시 → 1회말 → ...

진행 속도 조절
자동 진행 속도에 따른 경기 흐름 감상

연장전 지원
9회 이후 자동으로 연장전 진행



상단 진행 배너

KBO CLASSIC 1회 초 홈팀

실시간 게임 상태 표시
현재 이닝, 초/말, 팀명

한눈에 파악 가능한 디자인
사용자가 시뮬레이션 상태를 쉽게 인지

ESC 키로 즉시 메뉴 복귀

ESC → 메뉴로 복귀

사용자 제어권
자동 진행 중에도 언제든지 사용자가 메뉴로 돌아갈 수 있도록 제어권을 제공

연장전 지원 및 스코어보드 시각화

연장전 지원

9회 이후 자동 연장전
최대 20회 이닝까지 자동으로 연장 가능

사용자 확인
수동 모드에서는 사용자 확인 후 진행

동적 이닝 처리
vector 구조로 인한 유연한 이닝 확장

```
// 연장전 처리 코드
if (inning > 9) {
  printf("연장전 %d회차\\n", inning-9);
  push_one_inning(); // 동적 이닝 추가
}
```

스코어보드 시각화

gotoxy() → 표 기반
display_scoreboard() 함수로 전체 스코어보드 표시

하이라이트 표시
현재 진행 중인 이닝 또는 중요한 정보 강조

자동 정렬
점수 및 정보의 자동 정렬로 시각적 안정감 제공

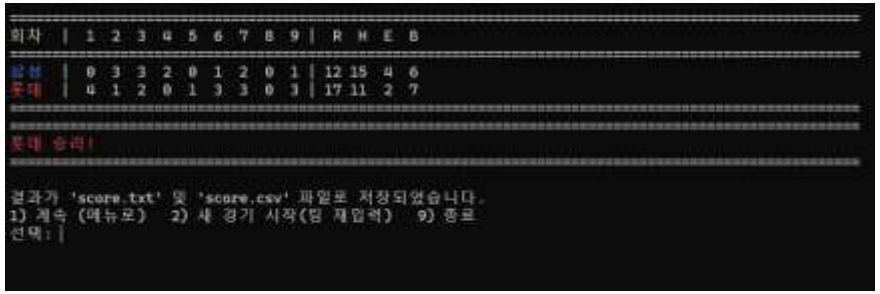


표 기반 출력으로 인한 더 좋은 가독성

인트로 및 사운드 / 저장 포맷

인트로 및 사운드

인트로 텍스트 중앙 정렬

프로그램 시작 시 인트로 텍스트가 화면 중앙에 정렬되어 시각적인 안정감을 제공합니다.

interruptibleBeep() 기능

배경 음악(BGM)을 재생하며, 사용자가 원할 경우 언제든지 즉시 중단할 수 있도록 제어권을 부여합니다.

인트로 텍스트 출력:

```
  printf("\033[H\033[J"); // 화면 지우기
  printf("\033[%d;%dH%s", rows/2, cols/2 - strlen(intro)/2,
intro);
```

사운드 재생:

```
  interruptibleBeep(); // 재생 중지 가능
```

저장 포맷

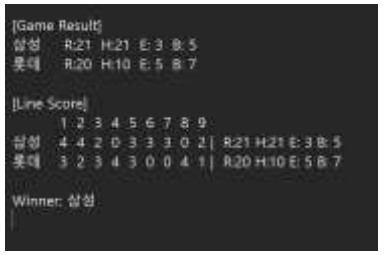
자동 파일 생성

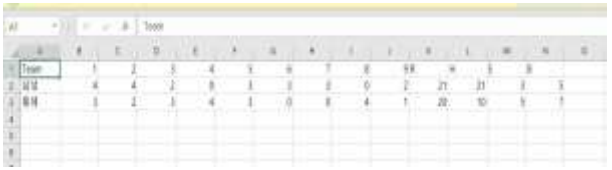
경기 종료 후 라인스코어, 합계, 승자 정보 등을 포함한 경기 결과를 자동으로 저장합니다.

다양한 파일 형식

결과를 **score.txt**와 **score.csv** 두 가지 파일 형식으로 저장하여 다양한 용도로 활용할 수 있습니다.

파일 내용 예시





오류/예외 처리 및 UX 디테일

오류/예외 처리

안전 입력

```
scan_int_safe() 함수
// 잘못된 입력 방지
```

특징:

- 유효한 정수 입력만 허용
- 잘못된 입력 시 재요청

입력 안정성 오류 방지

자동 진행 중단

```
if(key == ESC) return MENU;
// ESC 키로 즉시 중단
```

특징:

- ESC 키로 자동 진행 중단
- 즉시 메뉴로 복귀

사용자 제어 응답성

UX 디테일 개선

상단 진행 배너

```
display_banner()
// 현재 이닝, 초/말 표시
```

특징:

- 현재 이닝, 초/말 표시
- 팀명, 진행 상황 한눈에

정보 가시성 사용자 안내

하이라이트 표시

```
highlight_cell()
// 현재 이닝 강조
```

특징:

- 현재 진행 중인 이닝 강조
- 중요 정보 시각적 표시

사용자 집중 정보 강조

업그레이드 내용 간단 정리

프로젝트 종합 평가

스코어보드 게임 코드는 초기 콘솔 기반의 단순한 구조에서 점진적인 아키텍쳐 개선과 사용자 경험(UX) 강화 업데이트로 시뮬레이션 게임으로 발전

초기 버전

고정 이닝 배열, scanf 입력, 원시 화면 출력

최종 버전

동적 이닝 처리, 즉각적 입력 반응, 배너/테마/자동 진행 등 게임 루프 완성

아키텍처의 구조화

- 정적 배열에서 (Vector) 기반 동적 배열로의 전환
- 모듈 단위 함수 분리로 유지보수성과 확정성 향상
- 콘솔 출력, 입력, 저장 기능을 분리해 코드 안정화

UX의 혁신

- 키 입력 실시간 반응
- 안전한 입력 검증 및 예외 방지
- 사용자 제어권 강화
- 직관적인 상단 배너 하이라이트 표시

기능적 확장

- 랜덤 시뮬레이션 모드
- 연장전 지원

참고자료

ChatGPT (GPT-5)
Claude 3.5 (보조 검증 버전)