**알고리즘**

**설계 프로젝트**

|  |  |
| --- | --- |
| 프로젝트 명 | *Load Balancing* |
| 팀 명 | *HOI* |
| 문서 제목 | 결과보고서 |

|  |  |
| --- | --- |
| **Version** | 1.0 |
| **Date** | 2016-NOV-29 |

|  |  |
| --- | --- |
| **팀원** | 최 준형(조장) |
| 김 동훈 |
| 양 성수 |
| 한 용민 |
| **지도교수** | 최 준수 교수 |

|  |
| --- |
| **CONFIDENTIALITY/SECURITY WARNING**  이 문서에 포함되어 있는 정보는 국민대학교 전자정보통신대학 컴퓨터공학부 및 컴퓨터공학부 개설 교과목 알고리즘 수강 학생 중 프로젝트 “Load Balancing”를 수행하는 팀 “HOI”의 팀원들의 자산입니다. 국민대학교 컴퓨터공학부 및 팀 “HOI”의 팀원들의 서면 허락없이 사용되거나, 재가공 될 수 없습니다. |

**문서 정보 / 수정 내역**

|  |  |
| --- | --- |
| **Filename** | 최종보고서-Load Balancing.doc |
| **원안작성자** | 최준형 |
| **수정작업자** | 최준형, 김동훈 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 수정날짜 | 대표수정자 | Revision | 추가/수정 항목 | 내 용 |
| 2016-11-29 | 최준형 | 1.0 | 최초 작성 |  |
| 2016-11-29 | 김동훈 | 1.1 | 양식 및 오타수정 | 로드맵밸런싱->로드밸런싱, realTime함수🡪realTime변수 등등 |
| 2016-11-30 | 최준형 | 1.2 | 알고리즘 수정 | 로드밸런싱 알고리즘 오류 수정 |
| 2016-11-30 | 김동훈 | 1.3 | 내용 보강 | 스크린 샷 몇가지 추가 |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

**본 양식은 컴퓨터공학부 캡스톤 디자인 I 과목의 프로젝트 결과보고서 작성을 위한 기본 양식입니다. 문서의 필수 항목을 제시하는 것이니 폰트, 문단 구조 등의 디자인 부분은 자유롭게 설정하기 바랍니다. 양식 내에 붉은 색으로 기술한 부분은 지우고 작성하기 바랍니다.**

**목 차**

[**1** **프로젝트 목표** 4](#_Toc230855220)

[**2** **최종보고서 본문** 5](#_Toc230855224)

# **프로젝트 목표**

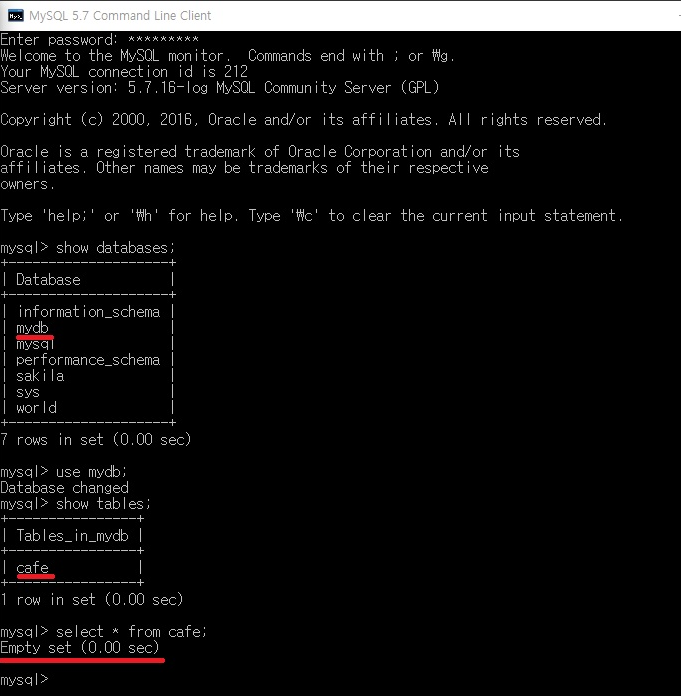
**시물레이션을 통해 카페 운영을 효율화 하기 위한 프로젝트**

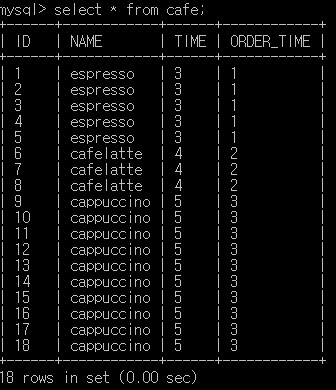
**이에 사용자가 로드밸런싱 알고리즘을 커스터 마이징 할 수 있도록 확장성 있는 업무 분배 시스템 프로토타입을 개발**

# **최종보고서 본문**

실제 프로그램을 실행시키기 위해서는 Local의 데이터베이스와 연동이 필요합니다.

Database 이름은 mydb, Table이름은 cafe입니다. 아래를 보시면 비어있는 café Table을 보실수 있습니다.



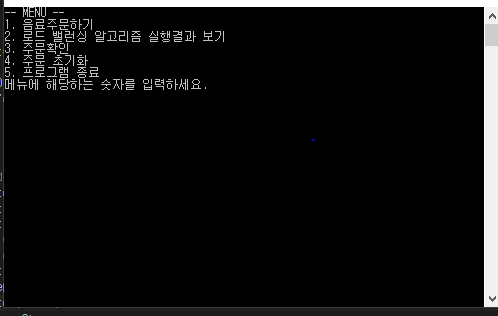


Café Table에 임의로 음료를 넣은 것을 mysql에서 출력한 것입니다.

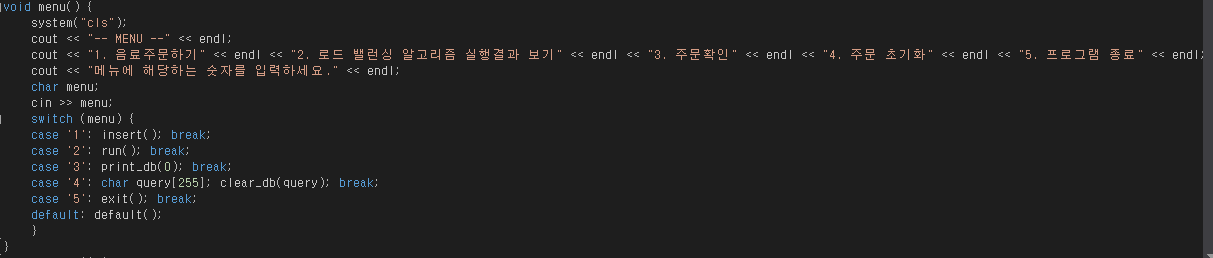
ID, NAME, TIME, ORDER\_TIME이란 attribute가 Table에 들어갑니다.

1) 메뉴 (초기 화면)

실행화면



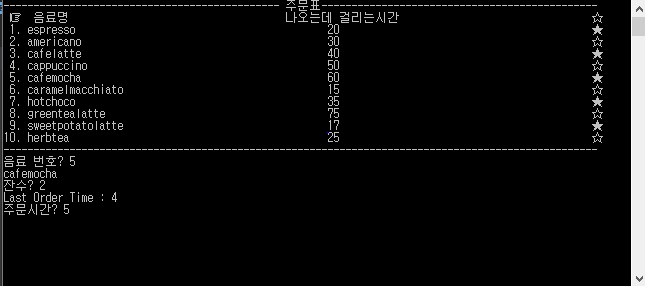
소스 코드



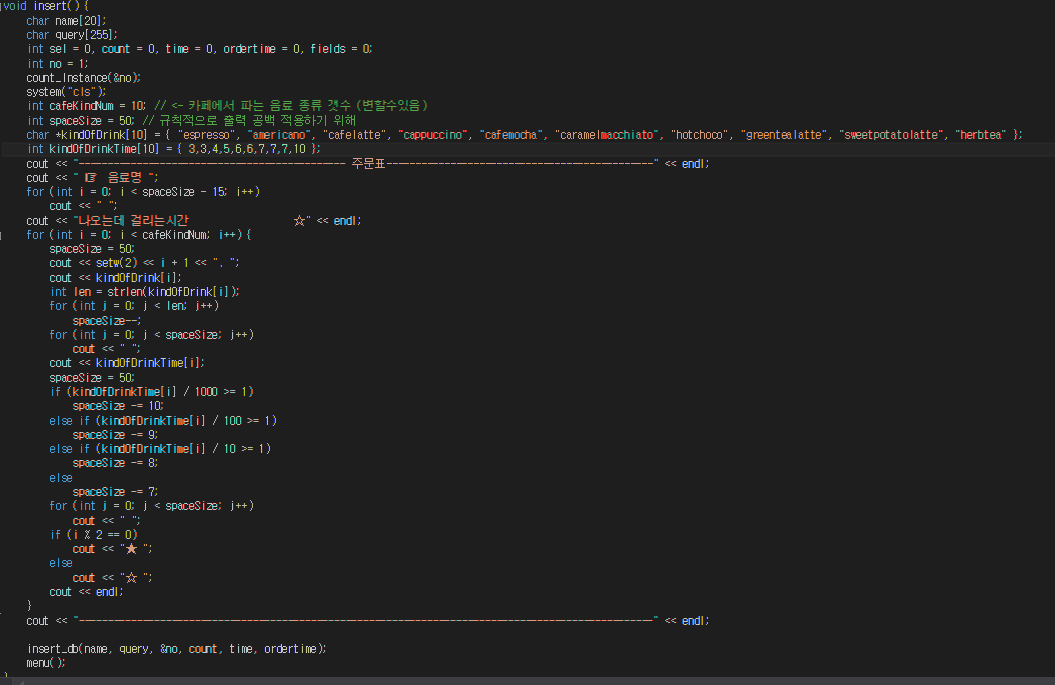
초기 화면 관련 메뉴 함수를 만들어 입력에 따른 함수를 호출한다.

2) 음료주문하기

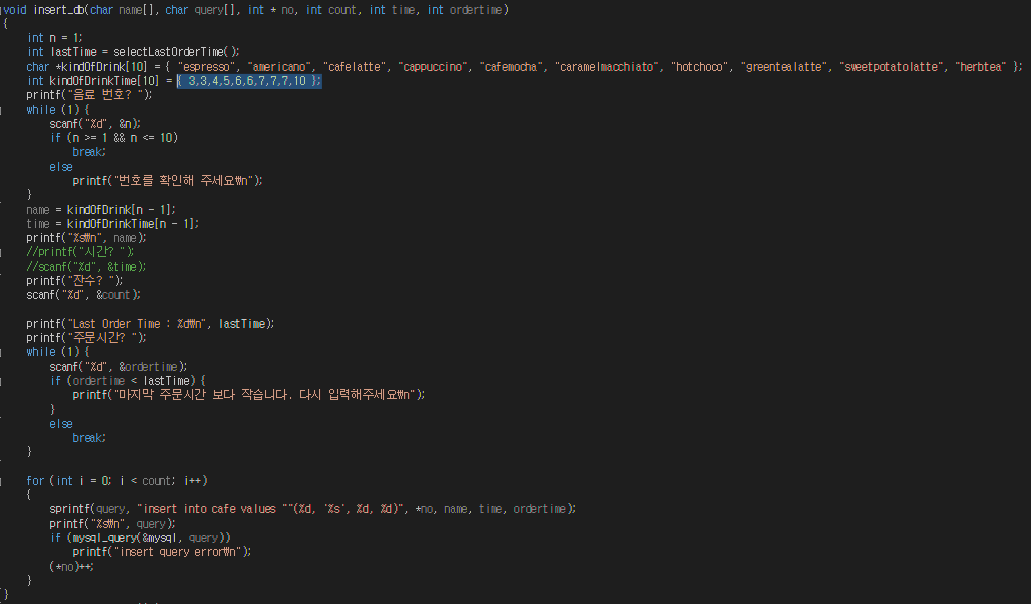
실행화면



소스 코드

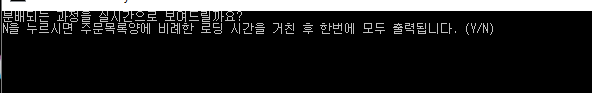


**음료의 이름과 음료제조시간을 배열에 저장하여 UI사용의 간편성을 유도하였고, 메뉴판의 공백을 구해주는 spaceSize변수를 통하여 공백을 계산하여 메뉴판을 출력해주었습니다.**



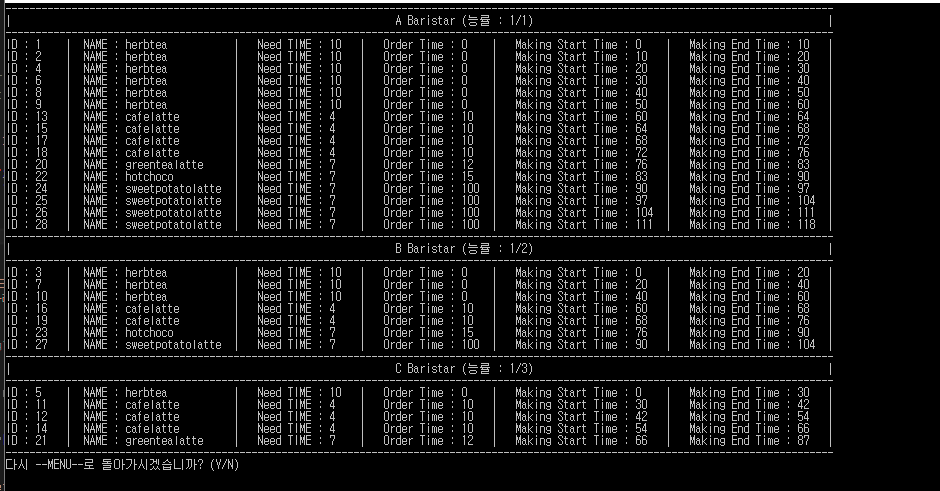
**Insert\_db함수를 제작하여 입력받은 변수들을 query문을 제작하여 DataBase에 바로 넣을수 있게 만들었습니다. 또한 잘못된 주문을 방지하기위해 조건을 반복하여 체크하도록 하였습니다.**

3) 로드 밸런싱 알고리즘 실행결과 보기



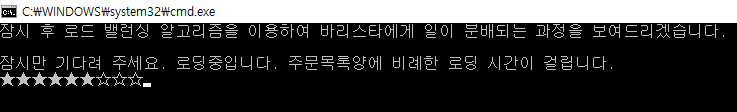
분배되는 과정을 자세히 볼것인지 아니면 한번에 결과만 출력할것인지를 선택합니다.

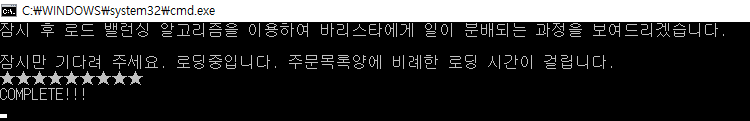


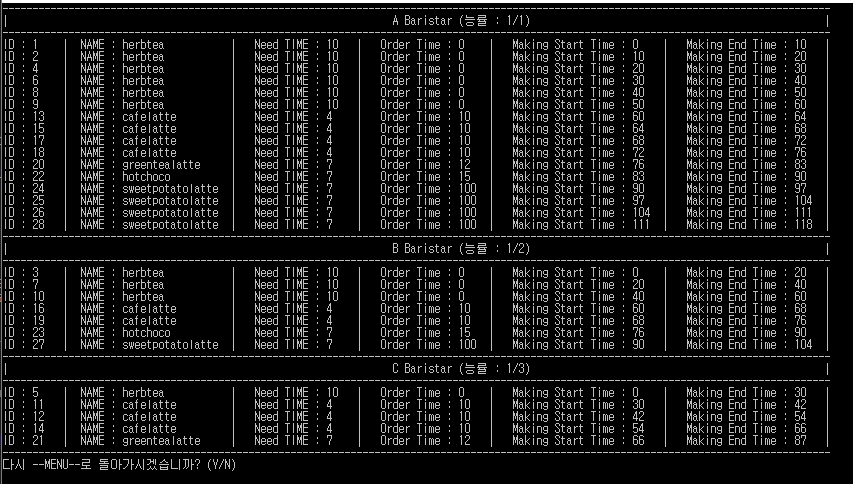


Y를 입력하면 위와 같이 음료의 순서대로 출력이 됩니다. N을 입력하면 아래와 같이 목록 양에 비례한 약간의 로딩시간을 거친 후 한번에 모두 출력되는 화면을 볼 수 있습니다.

주문받은 목록의 양이 많을수록 로딩시간이 늘어 납니다. (빈 별 개수가 많아짐)



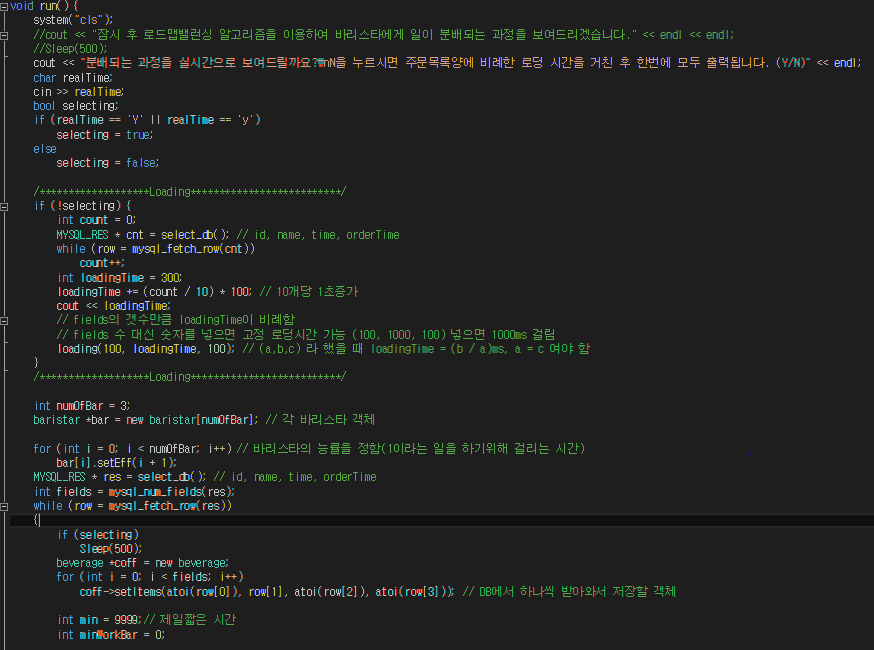




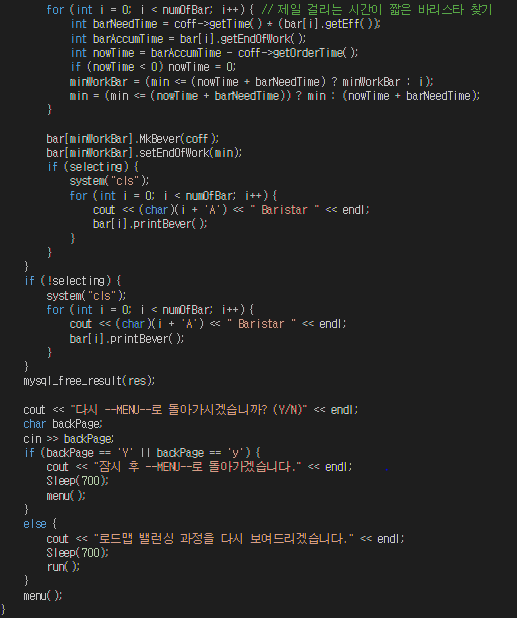
출력이 모두 완료되면 MENU로 돌아갈것인지 로드 밸런싱 과정을 다시볼것인지 확인합니다.

소스코드

1.



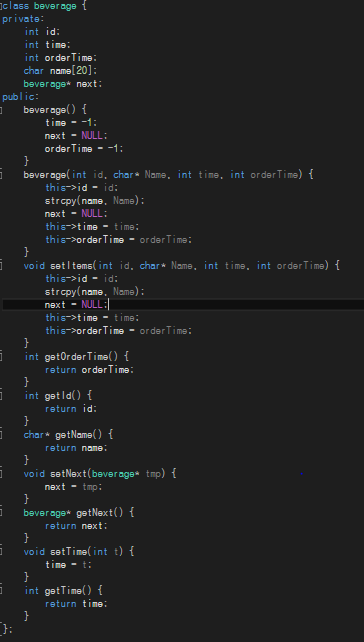
2.



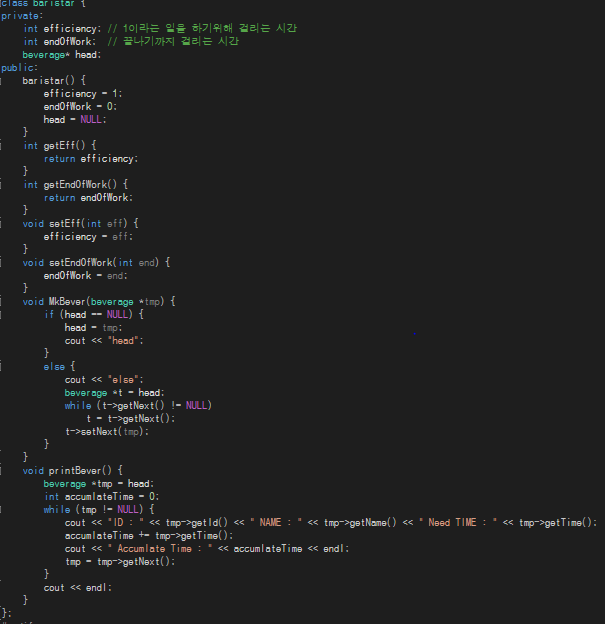
run()함수는 분배 과정을 출력해주기위한 함수이다. 처음에는 음료마다 분배되는 과정을 출력해줄것인지 아니면 한번에 출력해줄것인지 realtime 이라는 변수에 저장을하여 selecting을 참,거짓으로 하여 거짓일경우 select\_db를 통해 데이터를 탐색후 잠깐의 로딩시간후에 로드밸런싱 알고리즘으로 진입합니다. 바리스타의 개수를 임의로 3을 주었고 각 바리스타의 능률은 임의로 자신의 번호+1만큼 주었습니다. (set.eff(i+1)) db에서 받아온 정보를 row에 저장하여 반복문을 통해 beverage타입의 coff객체에 저장하였으며, 이 coff객체를 바리스타에게 분배하였습니다. 분배 알고리즘은 (바리스타의 누적 제조하는데 걸리는시간 – 커피 주문시간) =>주문 한 시간에 이미 모든 제조가 완료될수있으므로 음수이면 0과 동일 취급

**주문시간에 현재 남은 누적 제조음료 + (음료제조시간 \* 바리스타의 능률)** 이 가장 작은 바리스타에게 분배해였습니다.

로드 밸런싱에 사용된 class



음료의 id, 제조시간, 주문시간, 이름, 로드 밸런싱으로 다음에 올 음료를 Linked List로 저장할 next를 저장하는 클래스 각각 set, get함수가 존재한다.

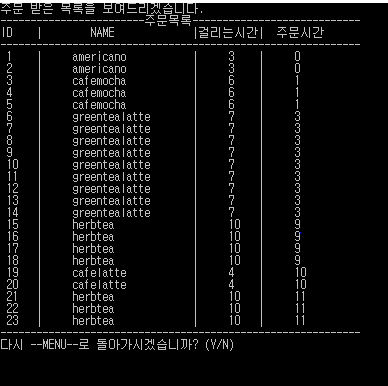


바리스타의 능률,음료의 누적시간, 음료 Linked List의 헤더를 저장하는 클래스 각각의 set,get함수와 Linked List의 마지막 tail에 음료를 저장할 MkBever함수, Linked List를 한번에 출력해줄 printBever()함수를 만들었습니다.

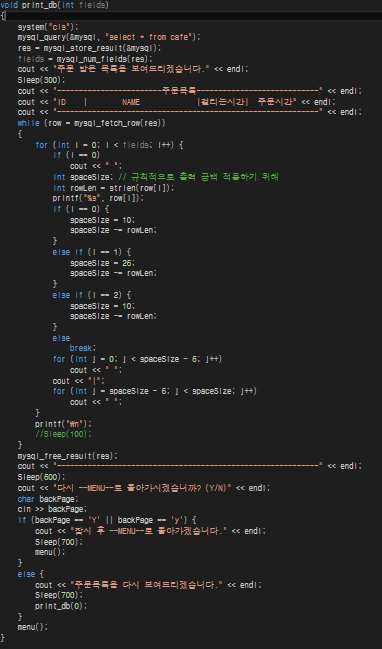
4) 주문 확인

데이터베이스에 저장되어있는 주문내역들을 출력해주는 메뉴

실행화면

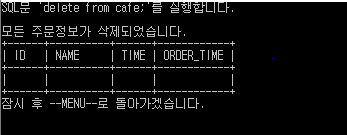


소스 코드



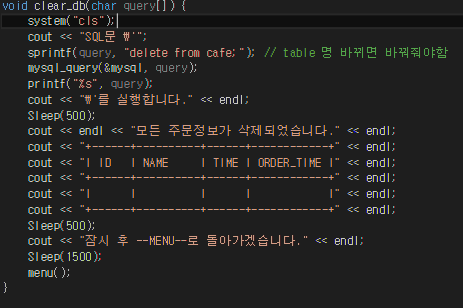
데이터베이스에 받아온 데이터들을 메뉴에 깔끔하게 출력후 메뉴로 돌아가는 화면

5) 주문 초기화



Delete from café 쿼리를 실행하는 화면

소스 코드



쿼리문을 실행함

6) 프로그램 종료

 종료 확인 후 프로그램을 종료합니다

소스코드

