**2022 Spring OOP Assignment Report**

과제 번호 : ASSN2\_prob2

학번 : 20210716

이름 : 최대현

Povis ID : daehyeonchoi

**명예서약 (Honor Code)**

나는 이 프로그래밍 과제를 다른 사람의 부적절한 도움 없이 완수하였습니다.

I completed this programming task without the improper help of others.

프로그램을 하다 보면 결정해야 할 세부 사항이 많은데, 이러한 세부 사항을 처리한 방법과 이유를 보고서에 쓰십시오.

독창적인 아이디어와 추가 기능은 보너스 점수를 받을 수 있으므로, 보고서에 명확히 기재하십시오.

문제가 여러 개인 경우, 각 문제별로 정리해서 작성합니다.

1. **프로그램 개요**

개발환경: Visual Studio 2022

사용자 자료형인 class로 Customer과 Rider 자료형을 구현한 뒤, 1번 문제에서 만든Vectoer class 를 응용하여 Vector\_Customer, Vector\_Rider 자료형을 구현한 뒤 마지막으로 Pizza class를 구현하는 코드이다. Vector\_Customer.h, Vector\_Customer.cpp, Vector\_Rider.h Vector\_Rider.cpp, Pizza.h, Pizza.cpp, Customer.h, Customer.cpp, Vector.h, Vector.cpp 의 10개 파일로 구성되어 있다. 여기에 main.cpp 파일을 포함하여 컴파일한 뒤 결과를 살펴본다.

1. **프로그램의 구조 및 알고리즘**

.

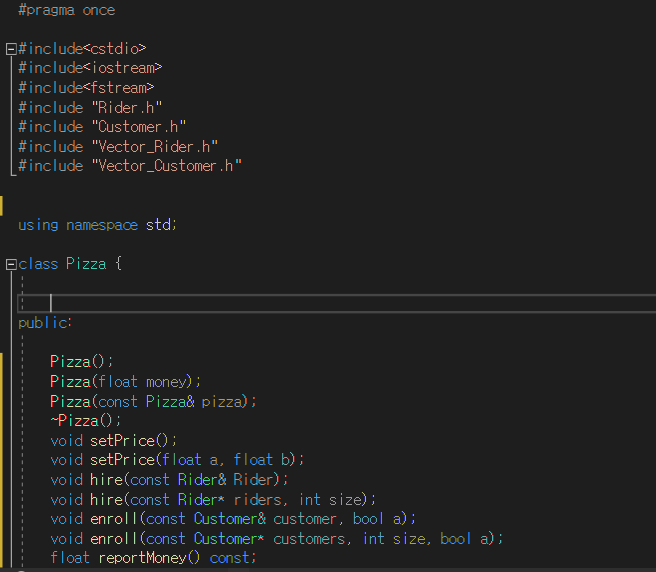
**벡터 자료형**

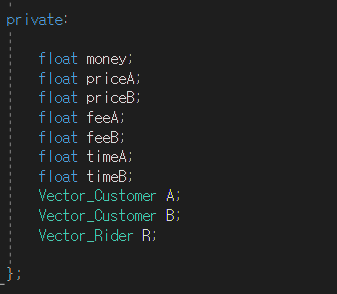
C++의 vector는 C++ 표준라이브러리(Standard Template Library)에 있는 컨테이너로 사용자가 사용하기 편하게 정의된 class이다. ctor를 생성하면 메모리 heap에 생성되며 동적할당된다. 적인 측면에서 array에 비해 성능은 떨어지지만 메모리를 효율적으로 관리하고 예외처리가 쉬우며, 템플릿으로 정의되어서 다양한 자료형에 대해 사용할 수 있어서 빈번하게 사용된다.

**사용자 정의 자료형 설명**

Customer, Rider이라는 사용자 정의 자료형을 구현했다. 그 뒤 1번 문제에서 float 타입의 vector를 구현한 방식으로 Vector\_Customer, Vector\_Rider, Pizza라는 이름을 가진 3가지 class를 구현했다. 각 클래스는 Public과 private 세션으로 나뉘어져 있고, public에는 여러 멤버 변수를 담았고, private 세션에는 문제에서 요구하는 멤버함수들을 구현하였다.

1. **Pizza class**





**멤버 변수 설명**

**float money**: 피자 가게의 돈

f**loat priceA:** customer A에게 받는 피자 가격

**float priceB:** customer B에게 받는 피자 가격

**float feeA:** customer A에게 받는 배달료

**float feeB:** customer B에게 받는 배달료

**float timeA:** A 구역에 배달하는 데 걸리는 시간

**float timeB:** B 구역에 배달하는 데 걸리는 시간

**Vector\_Customer A, Vector\_Customer B, Vector\_Rider R :** 각 class들을 멤버 변수로 가지고 있음

**멤버 함수 설명**

**Pizza::Pizza() :** 기본 생성자

**Pizza::Pizza(float money**): money 변수를 가져와서 생성하는 생성자

**Pizza::Pizza(const Pizza& pizza) :** 복사 생성자

**Pizza::~Pizza():** 소멸자

**void Pizza::setPrice():** 가격 지정 함수

**void Pizza::setPrice(float a, float b):** 가격 계수 a, b를 가져와서 지정해주는 가격 지정 함수

**void Pizza::hire(const Rider& rider) :** rider를 고용하는 함수 (선입선출)

**void Pizza::hire(const Rider\* riders, int size**): rider를 고용하는 함수 (rider 배열을 들여옴, 선입선출)

**void Pizza::enroll(const Customer& customer, bool a**): 특정 지역에 고객을 추가하는 함수 (선입선출)

**void Pizza::enroll(const Customer\* customers, int size, bool a):** 특정 지역에 고객 배열을 추가하는 함수 (선입선출)

**float Pizza::reportMoney() const**: 피자 가게가 현재 가지고 있는 돈을 리턴한다

**int Pizza::numRiders() const:** 현재 Rider 수를 가져오는 함수

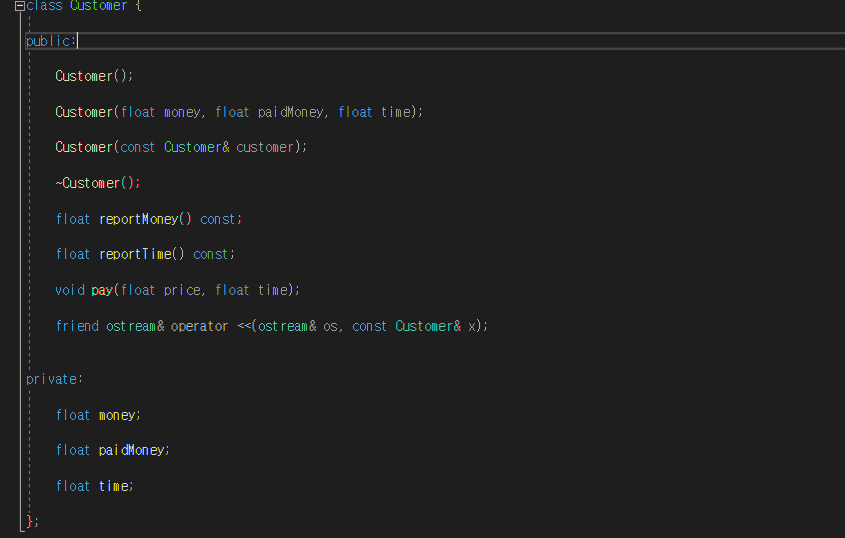
**int Pizza::numCustomers(bool a) const:** 현재 Customer 수를 가져오는 함수

**float Pizza::meanFee() const:** 현재 고용 중인 라이더가 배달을 통해 번 돈의 평균을 리턴한다.

**float Pizza::meanTimeRiders() const** : 현재 고용 중인 라이더가 배달하는데 걸린 시간의 평균을 리턴한다.

**ostream& operator <<(ostream& os, const Pizza& x) :** 콘솔창 출력을 << 연산자 오버로딩을 통해 구현함.

1. **Customer class**



**멤버 변수   
float money:** ustomer가 현재 가지고 있는 돈

**float paidMoney:** customer가 지불한 돈

**float time:** customer가 기다린 시간

**멤버 함수**

**Customer::Customer() :** Customer 기본 생성자

**Customer::Customer(float money, float paidMoney, float time):** Customer 생성자, 매개변수에 해당하는 값을 각각 멤버 변수에 대입함.

**Customer::Customer(const Customer& customer):** Customer 복사 생성자

**Customer::~Customer():** Customer 소멸자

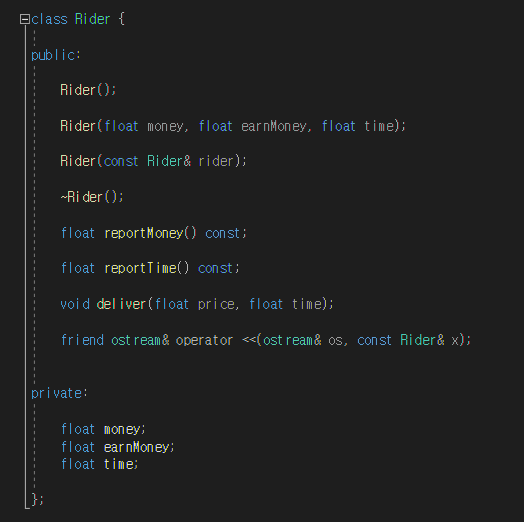
**float Customer :: reportMoney() const:** Customer가 값을 지불하는 함

**float Customer :: reportTime() const**: Customer가 값을 지불하는 함수

**void Customer :: pay(float price, float time):** Customer가 값을 지불하는 함

**ostream& operator <<(ostream& os, const Customer& x) :** << operator 오버로딩; class 출력할 수 있게 만듦

1. **Rider class**



**멤버 변수**

**float money:** rider가 가진 돈

**float earnMoney:** rider가 번 총 돈

**float time:** rider가 배달한 총 시간

**멤버 함수**

**Rider::Rider() :** Rider 기본 생성자

**Rider::Rider(float money, float earnMoney, float time):** Rider 생성자, 매개변수에 해당하는 값을 각각 멤버 변수에 대입함.

**Rider::Rider(const Rider& Rider):** Rider 복사 생성자

**Rider::~Rider():** Rider 소멸자

**float Rider :: reportMoney() const:** Rider가 번 돈을 리턴하는 함수

**float Rider :: reportTime() const**: Rider가 총 배달한 시간을 리턴하는 함수

**void Rider :: deliver(float price, float time):** 피자 가격과 배달 시간을 매개변수로 가져와서 pizza 배달을 해주는 함수

**ostream& operator <<(ostream& os, const Rider& x) :** << operator 오버로딩; class 출력할 수 있게 만듦.

**4), 5)** **Vector\_Rider, Vector\_Customer Class**의 경우 모든 멤버 함수와 멤버 변수가 1번 문제의 Vector class와 동일하고, datatype만 각각 Rider과 Customer로 바뀜.

1. **토론 및 개선**

과제를 다 끝내고 나서 여러 멤버 함수를 만들 때 새로 만들 멤버함수에 이미 만들어둔 멤버함수를 접목해서 코드를 간단히 구현할 수 있다는 사실을 깨달았다. 이를 참고하여 다음 과제 때는 코드를 더 간결히 짜보고 싶다.

또한, 처음에 “생성자의 리턴값은 없다’ 라는 것을 간과하고 문법을 잘못 사용해서 과제를 마무리할 때 오랜 시간이 걸렸었는데, 이 부분을 빨리 인지했으면 더 빨리 끝낼 수 있었을 것 같다.

**Template 에 대한 설명을 검색해 본 후, Template 을 사용해 문제 2 번을 구현한다면 어떤 점이 좋을지 Report Form 에 자신의 의견을 작성하시오.**

또한, 이번 과제에 대한 설명에도 적혀있듯이 Vector class 자체를 template으로 구현하면 문제에서 구현해야 하는 vector\_rider, vector\_customer에 대해 코드를 거의 바꾸지 않고 간단하게 쓸 수 있기 때문에 편리하다. 즉, 다형성을 구현할 수 있다.

1. **참고 문헌**