vector.h

이번에 구현한 Vector 클래스는 **T* 타입의 elements 배열**을 통해 요소를 저장하게 된다. 배열 자체의 저장 가능 공간은 **vecCapacity** 변수에, 현재 저장된 요소의 수는 **vecSize** 변수에 저장하여 사용하게 된다. iterator는 포인터를 사용하여 vector를 가리킬 수 있도록 하였고, const나 reverse 같은 특징을 반영하여 연산자를 구현해주었다. 그 이외에도, Vector 클래스의 생성자를 오버로딩하여 또 다른 vector의 reference, const refernce를 통해 복사할 수 있도록 해주었다.

다음은 Vector 클래스의 **멤버 함수의 구현 과정**에 대한 설명이다.

- `int capacity() const`

private 변수인 vecCapacity 값을 리턴할 수 있도록 해주는 함수이다. 함수 내에서 변경하는 과정이 없으므로 안전성을 위해 const 키워드를 붙여주었다.

- `int size() const`

위와 마찬가지로, private 변수인 vecSize 값을 리턴할 수 있도록 해주는 함수이다.

- `bool empty() const`

vector의 size가 0이면 true를, 아니면 false를 리턴해주는 함수이다. 삼항 연산자를 이용하여 간결하게 구현해 보았다.

- `void clear()`

vector에 저장된 요소를 모두 지워주는 함수이다. elements에 할당된 메모리를 해제해주고, 새로운 메모리를 할당해주는 방식으로 구현해 보았다. vecCapacity와 vecSize 모두 새롭게 초기화해주어야 한다.

- `iterator insert(const iterator &pos, const T &content)`

iterator를 사용하여 pos 위치에 content를 삽입해주는 함수이다. 메모리 낭비를 최소화하기 위해 함수 인자를 const reference 타입으로 받아주었다. capacity가 가득 찬 상태(vecCapacity == vecSize)라면, capacity를 2배로 늘려 새로운 메모리를 할당해준 이후에, 삽입을 진행하도록 구현하였다. 이 함수는 삽입한 위치를 가리키는 iterator를 리턴한다.

- `iterator erase(const iterator &pos)`

iterator를 사용하여 pos 위치에 있는 요소를 삭제해주는 함수이다. 역시 메모리 낭비를 최소화하기 위해 함수 인자를 const reference 타입으로 받아주었다. pos 뒤에 있는 요소들을 한 칸씩 앞으로 당겨 저장하는 식으로 구현하였다. 이 함수는 삭제한 요소의 바로 다음 요소를 가리키는 iterator를 리턴한다.

- `void push_back(const T &content)`

content를 vector의 맨 뒤에 저장해주는 함수이다. capacity가 가득 찬 상태라면, insert와 마찬가지로 capacity를 2배 늘려 새로 메모리를 할당해주는 작업을 진행해주었다.

- `void pop_back()`

vector 맨 뒤의 요소를 삭제해주는 함수이다. 실제로는, size를 하나 줄여, 해당 요소에 접근할 수 없도록 간결하게 구현하였다.

- `void resize(int newSize, const T &content = 0)`

vector의 사이즈를 재조정해주는 함수이다. content를 default 인자로 만들어, 현재 사이즈보다

크게 조정할 경우, content 값이 존재한다면 사이즈 조정 후 content 값으로 남은 공간을 채워주도록 하였다(default는 0).

• `T &operator[](int idx) const`
vector의 요소에 접근할 수 있도록 하는 연산자를 오버로딩하였다. 해당 idx가 0보다 크고, size보다 작은지 체크하여 유효한 idx에만 반응하도록 구현하였다.

• `friend std::ostream &operator<<(std::ostream &out, Vector<T> &vector)`
`friend std::istream &operator>>(std::istream &in, Vector<T> &vector)`
vector 요소를 한 번에 출력해줄 수 있도록 ostream의 <<를, vector에 쉽게 push_back할 수 있도록 istream의 >>를 오버로딩하였다. 출력시에는, “, ”를 삽입하여 가독성을 높였다.

• `Vector<T> &operator=(const Vector<T> &originVec)`
기존 vector를, 인자로 들어온 벡터로 교체시켜주는 함수이다. 메모리 공간이 동시에 가리키지 않도록, 기존 메모리의 해제와 새로운 메모리 할당을 직접 구현해주었다.

• `void assign(int count, const T &content)`
count 개수만큼, content를 저장해주는 함수이다. 기존 vector 메모리를 해제하고 새로 할당해주는 방식으로 구현하였고, capacity와 size 역시 새로 초기화해주었다.

•`iterator begin() const`, `iterator end() const`, ...
iterator를 반환하는 여러 가지 함수들이다. iterator의 종류에 따라, iterator, const_iterator, reverse_iterator, const_reverse_iterator 타입을 모두 구현하였다. 가장 대표적인 예로, begin의 경우에는 vector의 첫 번째 요소를, end의 경우에는 vector의 마지막 요소의 다음 요소를 가리키도록 만들어주었다.