**2022 Spring OOP Assignment Report**

과제 번호 : 2

학번 : 20210054

이름 : 정하우

Povis ID : howru0321

**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.

문제가 여러 개인 경우, 각 문제별로 정리해서 작성합니다.

각 문항별 설명은 편의를 위한 것으로, 삭제하고 제출한다.

1. **프로그램 개요**
   * C++에 구현되어 있는 vector class의 중요한 기능들을 구현한다. Vector class에 실재로 존재하는 여러 함수들을 직접 구현한 함수들이 담겨있는 헤더파일을 만들고 직접 사용해 본다.
2. **프로그램의 구조 및 알고리즘**

private:

* + - float\* arr : 배열
    - int len : 데이터 개수
    - int msize : 할당된 메모리 사이즈

public:

* + - typedef float\* iterator : float\*를 iterator 형으로 선언.
    - typedef const float\* const\_iterator : const float\*를 const\_iterator 형으로 선언.
    - Vector() : 기본 생성자. 모든 멤버 변수를 0 혹은 NULL 로 초기화한다.
    - Vector(int size) : 데이터 크기와 메모리 할당량은 size 로 초기화. size 개수만큼 0 으로 초기화된 데이터 저장.
    - Vector(int size, const float& init) : float init 으로 초기화된 데이터를 저장. 데이터 크기와 메모리 할당량은 size 로 초기화.
    - Vector(int size, const float\* init) : float 배열 init 으로 초기화된 데이터 저장. 데이터 크기와 메모리 할당량은 size 로 초기화.
    - Vector(const Vector& v) : 복사 생성자.
    - ~Vector() : 소멸자. 할당된 메모리를 삭제한다.
    - int capacity() const : 데이터에 할당된 메모리 사이즈를 리턴하는 함수.
    - int size() const : 데이터 개수를 리턴하는 함수.
    - iterator begin() : 데이터의 맨 처음 주소를 리턴하는 함수.
    - const\_iterator begin() const : 데이터의 맨 처음 주소를 리턴하는 함수. (const 버전)
    - iterator end() : 데이터의 맨 마지막 주소를 리턴하는 함수.
    - const\_iterator end() const : 데이터의 맨 마지막 주소를 리턴하는 함수. (const 버전)
    - float& front() : 데이터의 맨 처음 데이터를 리턴하는 함수.
    - const float& front() const : 데이터의 맨 처음 데이터를 리턴하는 함수.(const 버전)
    - float& back() : 데이터의 맨 마지막 데이터를 리턴함수.
    - const float& back() const : 데이터의 맨 마지막 데이터를 리턴하는 함수. (const 버전)
    - void allocate(int capacity) : 데이터에 할당된 메모리를 capacity 만큼 동적 할당한다. 재할당할때는 사용하지 않으며, 모든 데이터를 0으로 초기화한다.
    - void resize(int size) : size 만큼 데이터에 할당된 메모리를 동적 할당하며, 데이터 개수는 size 만큼 갖는다. 재할당할때 사용하며, 기존 데이터를 가지고 나머지 빈 부분은 0으로 초기화한다. 데이터 개수 보다 작은 크기로 재할당하면 범위 밖 부분은 날라간다. 데이터 개수를 size로 맞추었지만, 이는 실제로 사용했을 때 데이터 개수는 데이터 개수보다 적은 범위로 재할당하는 경우를 제외하고 기존 데이터 개수와 동일하므로, 이 함수를 사용할 때 데이터 개수는 다시 돌려놓는 알고리즘을 항상 적용했다.
    - void push\_back(const float& value) : 데이터의 맨 뒤에 새로운 데이터 value 를 추가한다. 2.문제 세부 조건 규칙에 따라 메모리를 동적 할당하는 함수
    - void pop\_back() : 맨 마지막 데이터를 삭제하는 함수. 이때 메모리 할당은 유지한다.
    - void insert(iterator position, const float& value) : 데이터의 특정 위치(position)에 value 를 삽입하는 함수. 2.문제 세부 조건 규칙에 따라 메모리를 동적 할당한다.
    - void insert(iterator position, const float\* first, const float\* last) : : 데이터에서 특정 위치(position)에 float array 를 삽입하는 함수. 이때 first, last 는 각각 float array 의 맨 처음 주소, 맨 마지막 주소이다. 2.문제 세부 조건 규칙에 따라 메모리를 동적 할당한다.
    - void erase(iterator position) : 데이터에서 특정 위치(position)의 값을 삭제하는 함수. 이때 메모리 할당은 유지한다.
    - void clear() : 할당된 메모리 사이즈, 데이터 개수를 0 으로 설정하며, 데이터를 NULL 로 설정하는 함수. 이때 재할당하지 않는다. 실제 vector class에 있는 clean()함수 역시 재할당하지 않는다.
    - float& operator[](int index) : 연산자 ‘[]’를 오버로딩한다. 데이터에서 index 위치에 해당하는 값을 리턴한다.
    - const float& operator[](int index) const : 연산자 ‘[]’를 오버로딩한다. 데이터에서 index 위치에 해당하는 값을 리턴한다. (const 버전)
    - Vector& operator=(const Vector& v) : 연산자 ‘=’를 오버로딩한다. 현재 멤버변수를 입력 Vector 의 멤버변수로 치환한다.

1. **토론 및 개선**

C언어만을 배운 나에게 Vector class는 너무 편리한 클래스라고 생각했다. 어떻게 이런 기능을 구현하는지 궁금했는데, 직접 주요 기능들을 구현해보니까 정말 모든 경우를 다 고려한 클래스라는 것을 알게 되었다. 그리고 vector class를 구현함으로서 class에 대한 개념이 확실하게 잡혔다. 그런데 실제로 vector class에 없는 allocate가 resize와 어떤 기능차이가 있는지 모호해서 그냥 기존 데이터가 있는 배열을 재할당 기능을 구현하는 함수는 resize로 하고 allocate는 기본 연산자로 할당된 배열을 원하는 크기만큼 할당하는 함수로 구현하였다. 그리고 resize 함수에서 데이터 개수를 size와 같게해야했었는데, 실제로든 데이터 개수는 데이터 개수보다 적은 범위로 재할당하지 않는 이상 이전과 같다. 이 조건으로 resize를 구현했으면 좀더 수월하게 구현하였을 것 같다. 왜냐하면 resize를 구현하고 난 뒤 항상 데이터 개수를 다시 바꿔줘야하는 수고로움이 있었기때문이다. 살제로 데이터 개수를 늦게 맞춰주어 중간에 할당된 범위 밖에 값을 넣어서 delete를 할 때 heap error가 떴다. 오류를 해결하는 과정은 어렵고 힘들었지만 이 오류를 해결하면서 이전에는 전혀 잡히지 않은 heap에 대한 개념이 어느정도 잡힌 것 같다.

1. **참고 문헌**
   * + <http://naver.me/5jYFQhvp> -동적할당
     + http://naver.me/FmOqlfyJ -매개변수 참조
     + <https://hwan-shell.tistory.com/119> - c++ vector
     + <https://dydtjr1128.github.io/cpp/2019/08/05/Cpp-const-after-function.html> - 함수 const
     + <https://m.blog.naver.com/PostView.naver?isHttpsRedirect=true&blogId=kks227&logNo=60205418298> – 함수 const
     + <https://siminq.tistory.com/80> - 참조자 리턴 함수
     + <http://naver.me/G4wZ4XUQ> - lvalue rvalue
     + <https://micropilot.tistory.com/3068> - 연산자 오버로딩
     + <https://docs.microsoft.com/ko-kr/cpp/standard-library/vector-class?view=msvc-170#op_at> – vector class
     + <https://lazyren.github.io/devlog/vector.html> - vector class 구현
     + <https://blog.joonas.io/151> - vector class 구현
     + <https://oceancoding.blogspot.com/2020/12/c-custom-stdvector.html> - vector class 구현
     + <https://2jinishappy.tistory.com/67> - vector class 구현