**CPP 3주차 (Vector~)**

2022/1/10 ~ 2022/1/

**Do.**

1. **Vector(가변배열)**
   * 알아서 크기를 늘려주는 배열이다.
   * push\_back(value)를 통해 가변배열 맨 뒤에 데이터를 삽입한다.
   * 사용자 정의 데이터 타입도 사용가능하다.
   * 멤버 함수
     1. assign(5, 2) : 2의 값으로 5개 원소 할당
     2. V.at(idx), V[idx] : at은 범위 탐색 O, V[idx]는 범위 탐색 X
     3. front(), back() : 처음 원소, 끝 원소를 참조한다.
     4. clear() : 모든 원소 제거, 메모리는 유지
     5. pop\_back() : 마지막 원소 제거
     6. begin(), end() : iterator와 같이 사용하며 첫번째 원소와 마지막의 다음을 가리킨다.
     7. rbegin(), rend() : begin과 end의 반대
     8. reserve(n) : n만큼 미리 동적할당을 진행함. 크기를 키움
     9. resize(n, 3) : 크기를 n으로 변경하고 늘어난 빈공간을 3으로 초기화 한다. 두번째 매개변수가 없다면 기본 값은 0으로 초기화 한다.
     10. size() : 원소의 개수를 반환
     11. capacity() : 할당된 공간의 크기를 반환
     12. v2.swap(v1) : 두 벡터의 모든 정보를 바꾼다.
     13. Insert(2, 3, 4) : 2번째 위치에 3개의 4를 삽입, 기존 값은 뒤로 밀림.
     14. erase(iter) : iterator가 가리키는 원소를 제거하고 size를 줄임.
     15. empty() : 비었는지 유무를 판단.
2. **동적할당**

* Monster\* monsterTest = new Monster() -> 이렇게 사용한다.
* 구조체의 경우 포인터를 통해 접근하기 때문에 ->를 이용해 접근한다.
* Monster\* monsterTest는 스택 영역, new Monster()은 힙 영역에 존재한다.
* monsterTest는 힙 영역에 있는 시작 주소를 가리키기 때문에 포인터 변수 형태로 만들어야 한다.
* 동적할당의 형태로 Vector를 만들어 줄 수 있다.
* Vector<Monster\*> monsters, monsters.push\_back(new Monster()), monsters[0]->name;
* 동적으로 할당된 메모리는 자체적으로 해제가 안되기 때문에 delete를 써야한다.
* 배열로 동적할당을 했을 경우 인덱스마다 delete를 해야 한다.

1. **클래스**
   * 클래스는 객체를 찍어 내기 위한 틀
   * 클래스는 기본적인 접근제한자는 private로 설정되어 있다.
   * private : public : 와 같이 접근제한자의 영역을 나누어 구간별로 설정할 수 있다.
   * 생성자 : 클래스가 객체를 찍어낼 때, 즉 변수를 선언하거나 하는 등 메모리에 할당이 될 때 실행되는 함수
   * 생성자 오버로딩 : 생성자의 인자를 다르게 하여 다른 기능을 수행하도록 할 수 있다.
   * 클래스 상속 : 기본적으로 멤버 변수와 함수를 상속하며 protected로 상속에 관여되는 클래스끼리 사용할 수 있도록 만들 수 있다.
   * 메서드 오버라이딩 : 부모클래스에서 virtual로 지정한 같은 이름의 함수를 특별한 지정자 없이 같은 이름으로 함수를 재정의 할 수 있다.
   * 업캐스팅과 다운캐스팅 : 주소값으로 할당하는 경우만 적용되고 스택으로 들어가면 그냥 형변환이 일어나게 된다.
   * Getter, Setter
   * 스코프 연산자(::) : 해당 연산자를 이용하면 부모클래스의 함수에도 접근 할 수 있음.

가상함수란 무엇인가? ->

순수 가상함수란?

추상 클래스란? -> 순수가상함수가 하나 이상 포함된 클래스를 말한다.

1. **클래스**

* Monster\*
* **클래스**
* Monster\*

1. **클래스**

* Monster\*
* **클래스**
* Monster\*

1. **클래스**

* Monster\*
* **클래스**
* Monster\*

1. **클래스**

* Monster\*
* **클래스**
* Monster\*

Vector<Monster>와 Vector<Monster\*>를 나눠 쓰는 경우가 있나요?