1.동영상 재생기

string이라는 클래스는 <string>이라는 헤더 파일에 저장되어 있다.

string str1 = “ I like “ ;

이러면 string 객체가 생성되는 거고 생성자가 호출 될 것이다.

위 문장은 string str1(“ I like “ )와 동일한 문장이다.

string은 동적할당을 통해 문자열을 저장한다. 왜냐하면 입력 받게 될 문자열의 길이를 잘 모르기 때문이다. 그러면 동적할당 때문에 소멸자도 필요하게 된다. new와 delete사용을 생각해보자.

깊은 복사를 하는 복사 생성자와 대입 연산자까지 정의되어 있어야 한다.

string \* commands 와 string& commands의 차이점

{ \*은 포인터다, 그래서 nullptr인지 확인이 필요하고, 포인터를 받겠다는 것,, 이거 또한 밖에 있는 원본 데이터를 가리키는 포인터를 잘 매개변수로 넣으면 함수 밖의 원본 데이터도 바꿀 수 있다}

{&은 참조다. 이건 string객체를 매개변수로 넣었을 때 유효하고, 원본 데이터를 변경하는 것을 허락하겠다는 말이다. 그리고 저 참조를 넣을 때 참조가 가리키는 대상이 있어야 함. 왜냐하면 참조는 선언과 동시에 초기화 되어야 하기 때문이다. 그리고 저 참조,,는 참조가 다른 것을 가리키게 만들 수 없음. 참조가 별칭이라는 것을 알아 두길 바람.}

const string \* commands, const string & commands의 차이점

{먼저 const는 선언과 동시에 const로 선언해야만 상수로 사용가능. 나중에 변수를 const로 바꾸는 것은 불가능하다. const바로 뒤에 오는 변수를 상수화 시킨다. 그런데 바로 뒤에 변수가 아니라 자료형이 오면 = 뒤에 있는 것을 상수화 시키는 문법이다. 예를 들어

const int \* ptr = &num2라고 하자 그러면 이건 ptr을 상수화 하는 게 아니라 &num2를 상수화 하는 거다. 그러니까 num2에 저장된 값을 바꾸는 게 안된다는 말이다 다른 예시로

int \* const ptr = &num2;를 하게 되면 이건 ptr에 저장된 값을 바꾸는 게 안된다는 말이다. 왜냐면 const는 자기 자신 바로 뒤에 오는 것을 상수화 시키는 것을 기본으로 하기 때문이다. 또 다른 예시로

const string & commands=str2; 를 하게 되면 이것 역시 str2라는 객체를

string은 vector와 비슷, 시퀀스, 배열 기반 컨테이너이다

string컨테이너는 이것도 벡터처럼 역시나 클래스로 만든 객체이고, 다른 문자열들과 달리 /0을 붙일 필요가 없다.

-매개변수에 배열을 집어 넣는 방법 -🡪 보통 vector<int>& 이런 식으로 넣어서 참조까지 하는 방법을 사용한다

-string은 동적할당이 필요 없다. 왜냐면 내부적으로 벡터처럼 동적할당 하여 사용하기 때문이다.

만약 동적할당을 한다고 하면, string \*answer = new string; 한 이후에 \*answer= “ I like”;이런 식으로 해결 할 수 있다.

-함수로 들어오는 매개변수를 따로 함수 내에서 정의하거나 하는 등의 작업은 필요 없고 바로 사용 가능하다. 매개변수 중 그 자리로 들어오는 순간 선언이 된 것이나 마찬가지라고 할 수 있겠다.

-함수에 const video\_len으로 들어왔기 때문에 video\_len을 변경할 수는 없다. 그 값을 함수 내부에서 변경하는 것은 불가능 하다 위에서 말했듯이 함수 자체가 내부에서 사용할 때 상수로 선언이 된 것이다. 그리고 또한 const가 없을 때는 내부에서 값을 변경하더라도 참조가 포인터를 통한 매개변수가 아니므로, 밖에서의 값이 변경되지 않는다. 포인터와 참조의 다른점은 포인터가 매개변수로 정의되어 있다면 실제로 포인터를 함수에 집어넣는 것이고 참조가 있을 때는 참조는 별칭이므로 같은 자료형이 들어오면 그것의 참조를 선언하게 되고 그것을 통해 외부에 있는 본체의 값을 바꿀 수 있는 것이다.

-size\_t는 부호없는 정수라는 뜻으로 C++표준 라이브러리에서 메모리의 크기나 배열의 인덱스를 표현할 때 사용된다. size\_t len = str.size();등으로 문자열의 길이를 반환하는 len

-프로그래머스에서 solution을 작성하기 위해서 안에 함수를 따로 정의하는 것은 불가능하다. 즉 함수안에 함수를 정의하는 것은 불가능하다. C++에서는 불가능하다.

-int main() { std::string str = "Hello, world!"; std::string sub1 = str.substr(7, 5); 5는 5칸을 출력하라는 말이다 // "world" std::string sub3 = str.substr(7); // "world!" (끝까지)->함수 문법이 이렇다.

이런 식으로 substr()을 사용해 볼 수 있다. substr은 string의 서브 문자열이다.

-string을 배열처럼 사용 할 수 있나? 그렇다!! str[0], str[1]이런 식으로 접근하는 게 가능하다.

stoi라는 함수는 문자열을 정수형으로 바꾸는 방법이다. str=”12345”라면 int num=stoi(str)을 하게 되면 num이 12345가 된다.

vector<string>이라면 **vector의 각 요소**는 **string 객체**들이 들어갑니다. vector는 **동적 배열**의 일종으로, **string 객체들을 순차적으로 저장**하는 컨테이너입니다.

**✅ vector<string>의 동작:**

* \*\*vector<string>\*\*은 std::string 객체들을 **동적으로** 저장합니다. 즉, vector는 **여러 개의 문자열을 순차적으로 저장**하는 **컨테이너**입니다.

**✅ vector<string>에 값 넣을 때 공백 구분:**

* vector<string>에 값을 넣을 때 **공백으로 구분된 문자열**을 자동으로 나누는 것은 아닙니다. 공백을 기준으로 **수동으로 문자열을 나누어** 각 string 객체에 넣어야 합니다. 🡪수동으로 넣는다??만약 vector<string>commands에 "prev" "next" 이렇게 넣는다면 vector의 두 칸에 각각 string 객체가 들어가게 되는것인가요?
* 네, 맞습니다! vector<string>에 "prev"와 "next"를 넣는다면, **vector의 각 요소**에는 **string 객체**가 들어가게 됩니다

범위기반 for 문이란 for( string com : commands){} 이런식으로 작성하는 것인데, commands는 vector<string>으로 선언 되었기 때문에 commands에 들어있는 한칸(요소)들을 com에 하나씩 집어넣으면서 요소들을 다 넣을 때 까지 for문을 도는것이다. commands의 요소가 string이기 때문에 string com으로 놓아야 한다.

-string 에서의 compare함수 🡪 com.compare(“prev”)를 하게 되면 com이 prev와 같은지를 확인하고 같다면 0 틀리다면 1을 반환한다. 즉 compare함수는 서로 다른 애들을 비교하는, 다른게 맞는지를 확인하는,, 다르다면 1을 반환하는 함수인 것이다.

-삼항 연산자 : 조건문을 간단히 표기하는 방법을 말한다.

세개의 피 연산자를 사용한다.

(조건 ? 참일때 실행할 코드 : 거짓일 때 실행할 코드;) 이렇게 쓸 수 있다.

answer = (temp / 60 < 10 ? "0" + to\_string(temp / 60) : to\_string(temp / 60)) + ":" + (temp % 60 < 10 ? "0" + to\_string(temp % 60) : to\_string(temp % 60));

여기서 to\_string(정수형이나 다른 기본형 데이터)라는 함수를 사용하여 다른 자료형을 문자로 바꾼다.-->stoi와 비슷