2022 년 1 학기 C++프로그래밍 (CD0064-01,02,03)

[개인 프로젝트]

String 클래스 만들기

개요

- ✓ 이번 학기동안 C++을 공부하면서 String 클래스를 작성/완성한다.
  - string 과 String 은 다르다. (**대소문자 구분!**)
  - string 은 표준 라이브러리에서 제공하는 클래스
  - String 은 우리가 만들 클래스 : string 을 일부 구현한다.
- ✓ C++ 표준 라이브러리에 있는 std::string 클래스 일부분을 목표로 하며 약간의 차이가 있을 수 있다.
  - 새로운 멤버 추가, 기존 멤버 변경/삭제
- ✓ 기본 코드는 수업자료의 예제로 나오는 String 클래스를 참고한다.
- ✓ std::string reference
  - http://www.cplusplus.com/reference/string/string/

프로젝트 진행 방법

- ✓ 본 문서에서 단계적으로 설명하는 클래스의 기능, 사용법, 사용 예제 등을 보고 그 내용에 맞게 클래스를 작성한다.
- ✓ 클래스를 테스트하는 간단한 프로그램(main)을 작성한다.

개발 방법

✓ 윈도우, 비주얼 스튜디오

또는

✓ 리눅스, g++ 컴파일러

작성 방법

✓ 프로젝트 소스 파일의 구성 : 다중 파일

- main.cpp , String.h , String.cpp

✓ 사용 설명서 (manual)

- 클래스의 사용법을 설명하는 문서를 나름대로 작성한다.
- 문서의 형식은 std::string 의 레퍼런스를 참고한다.
- 기본 서식 : 첨부 파일

제출

✓ 소스 파일과 설명서를 첨부하여 제출 기한 안에 LMS 에 제출한다.

✓ 소스 파일 이름 : **파일 이름을 임의로 바꾸지 않는다.**

자신의 “학번”, “이름”을 파일의 이름으로 사용하지 않는다.

✓ 소스 파일 : .cpp(.h) 파일을 그대로 업로드한다. (압축 파일 아님)

✓ 설명서 : 일반 과제의 형식으로 임의 편집하지 않는다.

채점 방법

✓ 1~3 단계 제출물 : 제출 기한 준수 여부를 가장 크게 평가함

✓ 4 단계 제출물 (최종) : 제출 기한 준수, 실행 결과

1 단계. 기본 코드 구현하기

제출 기한 : LMS 에 설정된 기한

1. 아래의 멤버함수를 추가한다.

생성자/소멸자
String(); String(const char* s); String(const String& str);
– String 객체를 생성한다. – 입력 값이 없으면 “” 문자열 기본적으로 갖는다. (빈 문자열) – 입력 값이 리터럴 문자열(s)이면 이를 this 의 문자열로 복사한다. – 입력 값이 String 객체(str)면 이 객체의 내부 문자열을 this 의 문자열로 복사한다.
~String();
– 문자열 저장 공간을 동적으로 삭제한다.

기본 멤버 함수

void print();
int size();
int length();
int capacity();

– print : 내부 문자열을 표준출력한다. (개행 문자 출력함) – size : 문자열의 길이를 반환한다. – length : 문자열의 길이를 반환한다. – capacity : 문자열을 저장하는 동적 배열의 크기를 반환한다.

2. 설명문

없음

주의!

- 제시된 것 이외에 `public` 멤버 함수를 임의로 추가하지 않는다.
- `private` 멤버는 필요한 경우 추가할 수 있다.
- 만들어야 할 것이 프로젝트를 진행하면서 점점 늘어난다. 기대하자!

2 단계. 멤버함수 추가하기 : 연산자 오버로딩 (1)

제출 기한 : LMS 에 설정된 기한

1. 아래의 멤버함수를 추가한다.

대입/할당
String& assign(const String& str);
– 입력 객체 str 의 문자열을 this 의 문자열로 복사한다.
String& assign(const char* s);
– 입력 문자열 s 를 this 의 문자열로 복사한다.
String& operator=(const String& str);
– String str1, str2; 일 때 str1 = str2;를 지원한다.
String& operator=(const char* s);
– String a;일 때 a = “literal”;을 지원한다.
누적/연결
String& append(const String& str);
– this 의 문자열 뒷부분에 입력 객체 str 의 문자열을 추가한다.
String& append(const char* s);
– this 의 문자열 뒷부분에 입력 문자열 s 를 추가한다.
String& operator+=(const String& str);
– this 의 문자열 뒷부분에 입력 객체 str 의 문자열을 추가한다.
String& operator+=(const char* s);
– this 의 문자열 뒷부분에 입력 문자열 s 를 추가한다.
원소로 접근
char& operator[](int index);
– this 객체의 저장 문자열을 char []로 다룬다. – 입력값 index 를 첨자로 하여 해당하는 원소(char)를 참조로 반환한다. – 만약, 인덱스가 유효 범위를 벗어나면 내부에서 다음과 같이 조정한다. (1) index 가 0 미만이면 0 으로 조정 (2) index 가 문자열 길이 이상인 경우 마지막 문자의 인덱스 (‘\0’ 아님) – //이 부분은 나중에 예외(exception)를 발생하도록 변경해야 한다.

문자열 저장 공간 정리
`void shrink_to_fit();`
– 문자열이 저장된 공간이 문자열 길이보다 클 때 그 크기를 문자열에 맞게 재조정한다. (저장소의 크기는 문자열 길이보다 +1 크다.)

2. 함수 수정

– 1 단계에서 만든 `String::print` 함수를 수정

일반 함수
`const char* print(bool show = true);`
– show 가 `true` 면 객체의 문자열을 표준 출력한다. – show 가 `false` 면 출력하지 않는다. – show 에 상관없이 객체 문자열의 시작 주소를 반환한다.

3. [1단계] 생성자를 수정하고, [2단계]의 대입/할당 함수도 이에 맞게 작성하라.

✓ `String` 클래스의 생성자를 살펴보면 모두 비슷한 코드로 만들었을 것이다.

✓ 3개의 생성자에서 공통된 부분을 별도의 멤버 함수로 만들자.

✓ 1단계에서 작성한 생성자들이 이 함수를 호출하도록 수정하자.

✓ 이 함수는 `String`의 멤버만 사용해야 하므로 `private` 영역에 속한다.

✓ 이 문제는 현재 단계에서 수행하지 않고 [3단계]로 미루어도 된다.

4. 설명서 작성

✓ 없음

※ 이번 단계에서 작성하는 함수는 다음 단계에서 매우 유용하므로 반드시 작성하자.

3 단계. 멤버함수 추가하기 : 연산자 오버로딩 (2)

제출 기한 : LMS 에 설정된 기한

1. 멤버 함수 만들기

연결 입출력
`String operator+(const String& str);`
– this 의 문자열과 str 의 문자열을 연결해서 새로운 문자열을 만들고, 이 문자열로 String 객체를 만들어 반환한다.
`String operator+(const char* s);`
– this 의 문자열과 문자열 s 를 연결해서 새로운 문자열을 만들고, 이 문자열로 String 객체를 만들어 반환한다.

2. 전역 함수 만들기

표준 입출력
`std::ostream& operator<<(std::ostream& os, const String& str);`
– String str;일 때 cout << str;을 지원한다. – str 의 문자열을 cout 로 출력한다. – 저장 문자열만 출력한다. 이외의 문자는 출력하지 않는다.
`std::istream& operator>>(std::istream& is, String& str);`
– String str;일 때 cin >> str;을 지원한다. – cin 으로 입력받은 문자열(char *)을 str 에 저장한다. – 표준 입력한 문자열만 저장한다.

※ 1 단계, 2 단계의 함수를 활용하면 friend 설정은 생략할 수 있다.

3. 설명서 작성

✓ 지금까지 만든 public 함수 중에서 가장 자랑스러운 하나를 선택하고 이 함수의 사용 설명서를 작성하자.

✓ 제외: size/length, capacity, shrink_to_fit

✓ 설명서는 과제에 첨부한 서식 파일을 사용하여 작성한다. 설명서의 구체적인 구성과 내용은 아래의 링크에 연결된 페이지를 참고한다.

✓ 아래의 항목은 설명서 내용에서 생략한다.

- Complexity, Iterator validity, Data races, Exception safety

참고 링크: http://cplusplus.com/reference/string/string/operator+=/

4. 지금까지 만든 함수의 동작을 확인하고 수정하기

- 다음의 함수를 점검하자.

```
String::operator=          // assign
String::operator+=         // append
String::operator+
```

- 아래와 같이 사용할 수 있어야 한다. 혹시 문제가 있다면 수정하라.

```
String s1("apple");
s1 = s1;
s1 += s1;
s1 + s1;
```

5. 지금까지 만든 함수의 코드를 간결하게 정리하기

✓ 이름은 다르지만 동작이 동일한 함수들이 많다.

: assign 시리즈, append 시리즈

✓ 함수들의 일부 코드가 비슷하다면 이를 별도의 부함수로 작성하고, 함수들이 이 부함수를 호출하도록 구조를 변경한다.

✓ 본 항은 [2 단계]의 3 번 항과 동일하다.

4 단계. Text 클래스 만들기

제출 기한 : LMS 에 설정된 기한

✓ 지금까지 작성한 String 클래스를 재사용(상속)하여 **Text 클래스를 만들자.**

1. Text 클래스 개념

✓ String 클래스를 상속하여 작성한다.

✓ 기존 String 의 문자열에 멀티라인 개념을 추가한다.

- 라인 수가 얼마인가? (=몇 줄 짜리인가?)
- 마지막에 한 줄 끼워 넣기
- 중간 어디쯤에 한 줄 끼워 넣기
- 특정 위치의 한 줄을 지우기
- 그 외 다양한 함수(연산자)를 생각해볼 수 있다.

✓ 문자열 특징

- '\n'로 구분하는 다수의 줄(line)을 갖는다.
- 초기 상태가 아니라면 "" 와 같은 빈 문자열은 가지지 않는다. 빈 문자열은 길이가 0 으로 문자가 전혀 없는 것을 의미한다.

  반면에 "\n"는 빈 문자열이 아니다. 개행(줄바꿈) 문자가 하나 있기 때문이다. 이는 빈 줄 하나가 있다는 의미기도 하다.
- Text 객체가 저장하는 문자열의 마지막 문자는 항상 '\n'이다.

  "apple is red\n"

  "apple is red\nbanana is yellow\n"

- 다음과 같이 빈 문자열이 아닌데 마지막 문자가 '\n'이 아니면 잘못된 문자열이다. 제대로 구성된 Text 의 문자열이라면 마지막 문자는 항상 '\n'이어야 한다.

  "apple is red"

2. 멤버 함수 작성

생성자/소멸자
Text(); Text(const char* str); Text(const String& str); Text(const Text& txt); ~Text();
– 입력 값이 없으면 빈 문자열("\n")을 만든다. – 입력 문자열이 Text 문자열의 조건을 만족하지 않는다면 이를 수정한다.
표준 출력 함수
ostream& operator<<(ostream& out, Text &txt);
– cout 객체로 표준 출력한다.
일반 함수
int lines();
– 저장 문자열의 줄 수를 반환한다.
Text& append(const char* s); Text& append(const String& str); Text& append(const Text& text);
– 입력 문자열을 this 객체의 문자열 뒷부분에 추가한다. – 연산의 결과는 Text 문자열의 조건을 만족해야 한다.

3. String 멤버 함수의 재사용

✓ ostream& operator<<(ostream& out, Text& txt);

- 이 함수는 Text 클래스에 friend 로 등록할 필요없이 구현할 수 있다.

- Text 객체를 String 객체로 형변환하여
  `ostream& operator<<(ostream& out, String& str);`
  를 호출한다.

✓ append 함수들

- String 에서 구현한 함수와 Text 생성자를 적절히 재사용하면 쉽게 만들 수 있다.

4. 설명서 (manual)

- 3단계에서 작성한 설명서에서 미흡한 부분을 보완해서 제출한다.