[C++]string 총정리 2탄(clear, empty, 문자열 추출 substr, 반복자 begin/end, 변경 replace, 제거 erase)

사용자 몰랑이말랑이 2019. 3. 23. 15:54

[C언어, C++언어, Java언어 기초 프로그래밍 완전 정복 목차!]

[C++]

저번 포스팅에서 String 사용하는 법 1탄을 살펴봤어요.

총정리가 뭔가 깔끔한거같아서 제목은 바꿨지만 저번 String 포스팅에 이어서 기능들을 살펴볼게요

String 라이브러리 총 정리 2탄!

문자열 초기화 - clear, empty

가장 먼저 문자열을 비워주는 함수를 볼게요. 안에 뭐가 들었을 지 모르니까 일단 비워주고 작업을 하는 습관을 들이는게 좋죠

clear는 문자를 비워주는 함수이고 empty 는 문자열이 비었나 확인하는 함수입니다.

```
#include <iostream>
#include <string>
using namespace std;
int main()
{
    string str1 = "abcdefghijklmnopqrstuvwxyz";
    if (!str1.empty())
    {
        str1.clear();
    }
    if (str1.empty()) //1
    {
        str1 = "new!"; //"new!"
    }
    return 0;
}
```

empty는 string 객체가 비어있으면 1을, 문자열이 저장되어 있으면 0을 반환합니다.

clear는 받는 매개변수도 없고 리턴 값도 없어요 오로지 문자열을 다 비워줄뿐!

문자열 추출하기- substr

그 다음은 엄청엄청 중요한 substr입니다!

```
string substr (size_t pos, size_t len) const;
```

문법은 위와 같아요

pos: 추출할 문자열의 시작 위치

len: 그 위치로부터 문자 몇 개까지 추출할건지~

본래 문자열을 변경시키거나 그러지는 않고 그 추출한 문자열을 리턴해줍니다.

```
#include <iostream>
#include <string>
using namespace std;
int main()
{
    string str1 = "good morning Mr Brown";
    string str2;
    str2 = str1.substr(5,7); //"moring"
    str2 = str1.substr(5); //"morning Mr Brown"
    str2 = str1.substr(str1.find("Mr")); //"Mr Brown"
    return 0;
}
```

예시는 1편에서 다뤘던 find함수와 응용해서 썼어요

만약 시작 위치를 음수로 넣으면 run-time 에러를 발생시키고 시작 위치가 문자열 길이보다 크면 빈 문자열을 리턴해줍니다. 또 만약 추출할 길이 len가 문자열의 길이보다 길면 문자열의 끝까지만 추출해줘요. len 매개변수를 안넣고 비워도도 마찬가지로 시작열부터 문자열 끝까지로 문자열을 잘라서 러턴합니다.

Iterator 반복자 - begin, end

이번에는 String함수에서 사용되는 반복자를 살펴봅시다.

front랑 back은 첫 번째 문자랑 마지막 문자를 알려줬잖아요? 여기 있는 begin과 end는 문자를 출력해주는 함수가 아니고

index를 리턴하는 반복자입니다.

```
#include <iostream>
#include <string>
using namespace std;
int main()
{
    string::iterator it;
    for (it = str.begin(); it < str.end(); it++)
    {
        cout << *it;
    }
    cout << endl;
    return 0;
}
```

for(int i=0; i<str.size(); i++) 거 쓰는 것과 무슨 차이일까요? 똑같아요 ㅎㅎ

다만 반복자는 표준입니다. 그렇기 때문에 우리가 vector와 같은 STL 컨테이너들을 사용할 때 얘네도 같은 반복자가 있기 때문에 굳이 변형하지 않고 똑같은 알고리즘을 활용할 수 있어요. 코드도 훨씬 보기 좋고요! int i, int j 이렇게 하는거보다 begin, end가 훨씬 가독성이 좋죠?! 반복자를 사용하는 습관을 들여줍시다. ㅎㅎ

rbegin, rend는 역시 reverse가 앞에 붙은 단어로 거꾸로 가는걸 말해요

특정 문자열 제거하기 - erase

erase! 특정 문자열을 제거할 때 사용하는 함수입니다.

erase는 매개변수의 개수와 타입에 따라 사용법이 좀 차이가 있는데요

1. 매개변수가 하나인데, 매개변수가 숫자일 때

```
string& string ::erase (size_type pos)
```

그 위치부터는 쭉 삭제입니다. 위치는 0부터 시작하는거 알죠? erase(2)하면 3번째 글자부터 그 뒤는 전부 삭제!

2. 매개변수가 하나인데, 매개변수가 반복자일 때

```
string& string ::erase (iterator pos)
```

그 반복자가 가리키는 지점만 삭제입니다. 즉 한 글자 삭제예요! 밑에 예제에 다시 설명해놨어요 ㅎㅎ

3. 매개변수가 두 개인데, 매개변수가 숫자일 때

```
string& string ::erase (size_type idx, size_type len )
```

범위 삭제예요. 첫 번째 매개변수는 시작지점, 두 번째 매개변수는 거기부터 얼마나 잘라서 없앨건지 크기를 나타내요

4. 매개변수가 두 개인데, 반복자일 때

```
string& string ::erase (iterator beg, iterator end )
```

마찬가지로 범위 삭제인데, 각 매개변수는 반복자가 가리키는 점을 시작으로 두 번째 매개변수 반복자가 가르키는 지점까지입니다. 길이가 아니라 끝나는 지점이예요 ㅎㅎ

```
#include <iostream>
#include <string>
using namespace std;

int main()
{
    string str, str2, str3;
    //a부터 n개 삭제
    str.assign("12345678");
    str.erase(0, 4); //"5678"
    //전체 삭제
    str.erase();

    //index 4 즉 e부터 4개 글자 삭제
    str2.assign("abcdefgh");
    str2.erase(4, 4); // "abcd"
```

```
    //n위치 이후로 다 삭제
    str2.erase(2); //"ab"

    //매개변수가 하나일 때는 그 위치에 있는 한 문자 삭제.
    //반복자가 가르키는 게 begin에서 하나 뒤에 있는거니까 a
    //즉 a삭제
    str3.assign("1a2b3c4d5e");
    str3.erase(str3.begin() + 1); //"12b3c4d5e"

    //반복자가 두 개 일경우는 한 글자가 아니라 범위 삭제
    //처음부터 처음에서 3칸 떨어진 지점까지이니까 12b삭제
    str3.erase(str3.begin(), str3.begin()+3); //"3c4d5e"
    return 0;
}
```

특정 문자열 변경하기 -replace

말그대로 특정 문자열을 찾아 한 번에 변경시켜주는 replace함수입니다!

```
template <class ForwardIterator, class T>
  void replace (ForwardIterator first, ForwardIterator last, const T& old_value, const T& new_value
);
```

replace는 다양하게 오버로딩 되어있어서 사용 방법이 많으므로 주석으로 정리해줄게요 ㅎㅎ

문법은 erase와 비슷한데 erase는 삭제할 문자열, 즉 문자열이 하나만 필요하다면

replace는 기존 문자열에서 변경하고 싶은 문자열과, 새로운 문자열 즉 두 문자열이 필요해서 매개변수의 개수가 좀 늘어나서 헷갈릴 수 있습니다. 순서는 무조건 old 문자열 + new 문자열 순!

```
#include <iostream>
#include <string>
using namespace std;
int main()
{
    string str("hello my name is Ella!");
    string str2 = str;
    //매개변수가 3개일 때 ex replace(a, b, "c");
    //a는 str 문자열에서 변환할 문자열의 시작점, b는 변환할 문자열의 길이이다.
    //즉 str문자열의 a지점부터 b길이만큼의 문자열이 "c"라는 문자열로 대체된다.
    str.replace(str.find("my"), 2, "your"); //"hello your name is Ella!"

    //replace의 매개변수에 숫자가 아닌 반복자 iterator가 오면
    //replace (it1, it2, "c"); -> it1부터 it2까지 "c"로 바꾼다.
    //즉 두번째 매개변수는 길이가 아니예요! 끝나는 지점임. 참고로 6번째 글자는 'm'입니다.
    str2.replace(str2.begin() + 6, str2.begin() + 8, "her"); //"hello her name is Ella!"

    string str3 = "his friend";
    //replace(str_it1, str_it2, str2_it1, str2_it2)
    //-> it1에서 it2까지 문자열을 다른 문자열(str2)의 it1에서 it2까지 문자열로 교체
    str2.replace(str2.begin() + 6, str2.begin() + 9, str3.begin(), str3.begin() + 3);
    //결과 | "hello his name is Ella!"
    cout << str2 << endl;
```

```
    //6시작점부터 3길이까지 문자열을 its toy의 0시작점부터 3길이인 its로 변환!
    str2.replace(6, 3, "its toy", 0, 3); //"hello its name is Ella!"

    str2.replace(6, 3, "my toy", 2); //"hello my name is Ella!"
    cout << str2 << endl;
    return 0;
}
```

너무 많아서 어려워보이지만 사실 단순해요

방법은 두 가지! (erase 를 생각해보세요)

1. 단순 index 즉 위치 값으로 바꿀 문자열 범위를 지정하느냐

index일 경우, 두 번째 매개변수는 길이를 의미하고

2. 반복자를 이용해서 문자열 범위를 지정하느냐

반복자일 경우 두 번째 매개변수는 끝나는 위치를 의미합니다.

string -> C 스타일의 문자열로 변환 - c_str()

[C/C++]cstring vs string.h vs string 스트링클래스 차이(C-strings vs std::string) <- 이 포스팅에서 문자열을 저장하는 방식 2가지에 대해서 설명했었어요. 기존 C문자열 저장 방식 vs String 방식의 ! 못본 사람은 보고 오기

간단하기 다시 정리하면 C-style string은 char* 포인터를 문자열을 저장하는 데 사용하고 '₩0'으로 끝맺습니다.
반면 string은 문자열을 클래스 형태로 저장하고 문자열을 조작하기 위한 다양한 기능을 포함하고 있습니다. C에서 기존에 헷갈릴 만한 사항들을 정리해서 사용하기 한층 편하게 만들었어요.ㅎㅎ 둘 다 알아야하고 혼합해서 쓰기도 하지만 보통 이점때문에 string의 문자열을 많이 씁니다.
지금 하고 있는 string 함수들이 결국 string 스타일인데요 c_str() 멤버함수를 이용해서 C-Style 문자열 방식으로 변환할 수 있습니다. 파일 입출력 해보신 사람들은 다 한번씩 써봤을 거에요 그 밖에도 두 스타일을 혼합해서 사용해야할 경우에는 필수적인 함수~

```
#include <iostream>
#include <string>
#include <fstream>
using namespace std;
int main()
{
    ifstream fs;
    string filename = "hello.txt";
    //파일을 여는 open함수는 c_style 문자열을 매개변수로 받기 때문
    fs.open(filename.c_str());
    if (fs.fail()) return -1;
    fs.close();
    return 0;
}
```

습관적으로 썼던 문자열 배열에 string 옮기기 ↓

```
#include <iostream>
#include <string>
using namespace std;
```

```
int main()
{
    string str = "str";
    char *strAry = new char[str.length()+1];

    // strcpy ( strAry , str );
    // 이렇게 하면 틀려요 왜냐 strAry는 C 스타일 방식인데
    // str은 string방식이기 때문이예요.
    strcpy ( strAry , str.c_str() ); //이렇게 해줘야 합니다.
    return 0;
}
```

여기까지 입니다. ㅎㅎ 자주쓰는 것들 위주로 알아봤어요

혹시 string클래스에 앞 뒤 공백을 제거해주는 trim이라는 함수가 없다는건 눈치채셨나요? trim은 엄청 자주쓰는 함수인데 말이죠! 자바나 자바스크립트 등 다른 언어에서 자주 사용해서 있다고 착각하고 쓰려고 멤버함수들을 확인해보니까 없어서 놀란(?) 분들도 있을거예요. C++의 string 멤버함수 목록에는 trim이 없어서 정의해줘야합니다. 이걸 다음 포스팅에서 알아볼게요