[C++]string 라이브러리 사용법 1탄, 비교(compare),추가(append),찾기(find),크기(size,length)등 함수(메서드) 예제

사용자 몰랑이말랑이 2019. 3. 23. 13:48

[C언어, C++언어, 자바 언어 프로그래밍 강좌 목차]

[C++]

String Library 알아보기~~!!

1탄입니다. ㅎㅎ

String은 문자열을 나타내기 위한 클래스입니다. 문자를 조작하는 일은 정말정말 많이 쓰이죠 ㅎㅎ

한 번 정리해서 머리 속에 넣을 필요성이 있어요 ㅎㅎ 아주 쉬운거부터 고럼 가볼까요

string 초기화

string을 생성하는 건 여러 방법이 있는데요.

```
#include <iostream>
#include <string>
using namespace std;
int main()
{
    //동시에 선언과 초기화
    string str1 ="str1";
    string str2("str2");
    //선언과 초기화 각각
    string str3;
    str3="str3";
    //string 멤버함수 사용
    string str4;
    str4.assign("str4");
    //동작 할당
    string *pstr5 = new string("str5");
    delete pstr5;
    return 0;
}
```

이렇게 객체를 생성해줄 수 있습니다!

크기 (size, length)

먼저 크기 관련해서~

공백도 다 포함시켜서 셉니다. 대표적으로 size함수와 length 함수가 있어요. 둘이 똑같은 함수입니다.

```
#include <iostream>
#include <string>
using namespace std;
int main()
{
    string sentence;
    sentence = "Hello this is a sentence";
    cout << "문자열의 크기" << sentence.size() << endl;; //참고로 한글은 2byte 영어는 1byte로 인식합
니다
    cout << sentence.length() << endl; //이름만 다를 뿐 size랑 같은 역할을 합니다.
    return 0;
}
```

"Hello this is a sentence"는 24글자네요!

인덱스, at, front, back

특정 위치에 있는 글자를 빼오려면? 그 위치에 있는 값을 [] 넣어서 cout으로 출력해주면 됩니다.

위치는 배열의 특성 때문에 위의 예제의 경우 0~23까지 즉 *'글자수 -1'*의 위치 값들을 갖습니다.

이렇게 특정 위치의 문자를 변경할 수도 있어요.

at 함수 또한 같은 역할을 수행해요. Java에서 String.charAt()과 같아요 ㅎㅎ

```
#include <iostream>
#include <string>
using namespace std;
int main()
{
    string sentence;
    sentence = "Hello this is a sentence";

    cout << sentence[0] << endl; //"H"
    cout << sentence.at(1) << endl; //"e"
    sentence.at(5) = '~'; //"Hello~this is a sentence"
    sentence[23] = '!'; //"Hello~this is a sentenc!"
    cout << sentence.front() << endl;  //"H"
    cout << sentence.back() << endl; //"!"
    return 0;
}
```

여기서 front함수는 at(0)과 같아요 즉 맨 처음 나오는 character를 말하고 back은 맨 마지막에 나오는 character를 말합니다.

앞뒤글자만 뽑을 때에는 size를 세가지고 그 index적어주기 힘들잖아요 ㅎㅎ front와 back 함수를 통해서 간편화 할 수 있습니다.

문자열 비교

다음 두 개의 문자열을 비교하는 함수들을 알아볼게요

비교로는 '==', '!=', '<', '<=', '>', '>=' 등이 있습니다.

아스키코드를 기준으로 하기 때문에 비교할때 대문자도 구별해요! 대문자가 소문자보다 앞에 위치해서 "A"<"a"가 성립합니다.

참고로 A가 아스키코드로 65고 소문자 a가 32를 더한 97이라는 거 몰랐던 분들은 알아두세요~!

```
#include <iostream>
#include <string>
using namespace std;
int main()
{
    string str1 = "Hello";
    string str2 = "Hello";
    cout << (str1 == str2) << endl;
    cout << (str1 != str2) << endl;

    string str3 = "apple";
    string str4 = "banana";
    if (str3 < str4) {
        cout << "에플은 a로 시작하니까 b인 바나나보다 사전적으로 먼저 위치!" << endl;
        cout << "<는 사전적으로 앞에 있다 >는 이후에 있다를 의미" << endl;
    }
```

```
    cout << str1.compare(str2) << endl; //0
    cout << str3.compare(str4) << endl; //-1
    cout << str4.compare(str3) << endl; //1
    return 0;
}
```

같은 역할을 하는 string 함수로는 compare이 있어요 ㅎㅎ

결과는 -1, 0, 1로 알려주는데요 -1이면 왠지 앞에 있을 거같죠? 넴 ㅎㅎ 사전적으로 비교하고자 하는 문자열보다 앞에 위치하면 -1을, 같으면 0을, 더 뒤에 위치하면 1을 리턴해줍니다.

글자 추가 (+, +=, append, insert)

연산자가 오버로딩 되어있어서 '+', '+='연산자 사용 가능합니다. 빼기는 오버로딩 되어있지 않아요 ㅎㅎ

함수로는 append함수가 있습니다. append를 default로 사용하시면 문자열 맨 마지막부터 추가가 돼요.

중간에 삽입하고 싶은 경우에는 insert 함수를 사용해주시면 됩니다. insert(추가할 위치, 문자열)!

append 문법↓

string& string::append (const string& str, size_type str_idx, size_type str_num)

str: 추가할 문자열

str_idx : 추가할 문자열의 시작 지점

str_num :시작 지점으로부터 몇 개의 글자의 크기를 추가할건지!

두번째, 세번째 따로 지정 안해주면 맨 첫 문자열부터 마지막 문자열까지 str전체를 붙일 문자열 맨 뒤에 추가합니다.

```
#include <iostream>
#include <string>
using namespace std;
int main()
{
    //직접 돌려보기 귀찮으신 분들을 위해 주석에 결과를 달아드릴게요
    string str1 = "Hello";
    string str2 = "friend";
```

```
    string str3 = str1 +" "+ str2;// "Hello friend"
    str3 += "! "; //"Hello friend! "

    string str4 = "nice to meet you";
    //str4의 0 index부터 4칸 붙이기
    str3.append(str4, 0, 4); //"Hello friend! nice"
    str3.append(" to meet you"); //"Hello friend! nice to meet you"
    str3.append(5, '.'); //"Hello friend! nice to meet you....."
    cout << str3 << endl;
    return 0;
}
```

append에 또 오버로딩 되어있는 것 중 하나가

```
string& string::append (size_type num, char c)
```

입니다.ㅎㅎ

첫 매개변수에 숫자를 넣어주면 c 문자열을 그만큼 반복해서 붙이라는 뜻이예요

그래서 str3.append(5, '.');는 점을 뒤에 5개 붙이라는 것을 의미하는 코드가 됩니다.

특정 문자 탐색

마지막으로 특정 문자를 탐색하는 find함수를 보고 2편으로넘어갈게요.

find는 특정 문자열을 찾아주는! 검색해주는 함수입니다.

```
#include <iostream>
#include <string>
using namespace std;
int main()
{
    string str1 = "find a word 'computer' in this sentence!";
    string str2 = "computer";
    cout << str1.find(str2) << endl; //13
    cout << str1.at(13)<<endl; //c

    size_t pos;
    string str3 = "I love my friends, my friends are sooo nice";
    pos = str3.find("friend"); //10
    pos = str3.find("friend", pos + 1); //22
    return 0;
}
```

find는 특정 문자열을 찾으면 그 문자열이 시작하는 문자의 위치를 리턴해줍니다.ㅎㅎ 만약에 문자열이 없으면 string::npos를 리턴해줍니다.

보면 computer라는 단어를 찾아서 str1 문장에서 c가 처음으로 위치한 13을 리턴해줬네요

at함수 배웠던거 활용해서 13번째 위치에 어떤 문자가 오는지 역으로 다시 확인해줬어요 ㅎㅎ

str3처럼 한 문장 내에 같은 단어가 여러번 들어 있을 경우, 탐색을 한 후에 그 다음번부터 다시 탐색함으로써 순차적으로 해당 문자열을 찾을 수 있어요. 저는 두 개밖에 안들어가있는걸 아니까 find를 그냥 두 번 써줬지만, 모를 경우 반복문으로 탐색해주면 되겠죠?!

```
#include <iostream>
#include <string>
using namespace std;
int main()
{
    size_t pos;
    string str3 = "friend, I love my friends, my friends are sooo nice";
    pos = str3.find("friend");
    while (pos != string::npos) {
        cout << pos << " ";
        pos = str3.find("friend", pos + 1);
    }
    return 0;
}
```

find랑 비슷한 함수로는 find_fisrt_of, find_first_not_of, find_last_of, find_last_not_of 가 있습니다.

find_first_of(): find랑 같으나 주어진 문자가 첫번째로 나타나는 위치를 찾습니다.

find_last_of(): 마찬가지로 얘는 처음으로 나타나는 문자 말고 마지막으로 나타나는 문자의 위치를 찾아요

find_first_not_of(): not에서 유추할 수 있듯이 주어진 문자가 처음으로 안타나는 위치를 알려줍니다.

find_last_not_of(): find last of() 과 같으나 주어진 문자가 아닌 문자를 찾습니다.

rfind(); reverse의 약자로 r이들어갔네요 거꾸로 탐색입니다.

```
#include <iostream>
#include <string>
using namespace std;
int main()
{
    string str("aa bb aa bb aa bb");
    cout << "길이:" <<str.length() << endl; //17
    cout << str.find("aa") << endl; //0
```

```
    cout << str.find("bb") << endl; //3
    cout << str.rfind("bb") << endl; //15
    cout << str.find_first_of("aa") << endl; //0
    cout << str.find_first_not_of("aa")<<endl; //2
    cout << str.find_first_of("bb") << endl;; //3
    cout << str.find_last_of("aa") << endl; //13
    return 0;
}
```

이어서 2탄에서 string에 관한 나머지를 살펴볼게요 ~!!