[C++] string 클래스, 문자열에 대해서 (총정리)

2019. 3. 29. 06:48

#C++ string #C++ string find #C++ string replace #C++ string size #C++ string substr #C++ string swap #C++ 문자열 #C++ 스트링

안녕하세요 **BlockDMask** 입니다.
오늘은 **C++의 std::string 클래스(문자열)**에 대해서 세세 하게 알아볼것 입니다.
예전 글을 보다가 제가 작성한 **이 문서**를 보게 되었는데요, 너무 내용이 빈약하다고 생각해서 리뉴얼 하게 되었습니다.
문자열 관련 **특정 함수 예제**만 보실분들은 아래 예제들에서 사용한 멤버 함수들을 적어 두었으니 **해당 예제로 바로 내려가시면** 됩니다. ~~(저 잘했쥬?)~~

<목차>

1. C++ string 클래스 헤더파일과 설명 그리고 string을 생성하는 여러가지 방법

▼ string 클래스 헤더파일, 생성방법

헤더파일 : **<string>**

생성1 : **string str("BlockDMask");**

생성2 : **string str1;**

str1 = "BlockDMask";

생성3 : **string str2(str1);**

(using namespace std; 를 추가해주어야 합니다.)

string 클래스는 말씀드린것 처럼 string을 다루는 클래스 입니다.
또한, C언어의 char*, char[] 문자열과 달리, 문자열끝에 '₩0'이런게 들어있지 않습니다.
자 그럼 C++ string 에서는 어떤 편리한 기능이 있는지 살펴보러 가겠습니다.

2. C++ string 클래스 멤버함수 정리 (거의 대부분)

std::string 에는 문자열을 다루는 정말 여러 멤버 함수들이 존재합니다. 우리는 이 멤버함수들을 적절하게 사용하여 원하는 프로그램을 만들면 됩니다.

♣ 중요중요

string str1 = "BlockDMask";

string str2 = "BlogBlogBlogBlog"; 인 상태에서 시작하겠습니다.

하나의 멤버함수 내에서 예시들은 이어지게되고.

각 멤버 변수가 시작할때 str1, str2는 처음과 같이 초기화 되었다고 가정하겠습니다.

♣ string 인자 접근, access 관련

▶ str1.at(index)

함수 원형 : char& at (size_t index);

함수 설명 : index(=인덱스) 에는 숫자가 들어가게되고, 해당 위치(n)에 해당하는 문자를 반환합니다.
index는 0부터 시작합니다. index가 string의 범위를 벗어나게 되면 예외를 뱉습니다.

함수 예시 : str1.at(0); //"BlockDMask" -> 'B'를 리턴합니다.

▶ str1.operator[index]

함수 원형 : char& operator[](size_t index);

함수 설명 : C++ string은 일반 배열처럼 대괄호를 이용해서 **string 인자에 접근**할 수 있습니다.
at과 다른점은 string의 index(인덱스)범위를 검사하지 않기 때문에 **at 함수보다는 빠릅니다**. 하지만 예외를 뱉어내지 않습니다.

index는 0부터 시작합니다. index 번째 인자를 반환합니다.

함수 예시 : str1[1]; //"BlockDMask" -> 'l'를 리턴합니다.

▶ str1.front();

함수 원형 : char& front();

함수 설명 : C++11부터 가능합니다. string의 **맨 앞 인자를 반환**합니다.

함수 예시 : str1.front(); //"BlockDMask" -> 'B'를 리턴합니다.

▶ str1.back();

함수 원형 : char& back();

함수 설명 : C++11부터 가능합니다. string의 **맨 뒤 인자를 반환**합니다.

개인적으로 front는 모르겠지만 back은 참 좋은 멤버변수인것 같습니다. 우리가 string의 첫번째 인자를 가지려고 할때는 str1[0]을 인덱스로 집어 넣으면 되지만 끝부분은 대부분 다른 멤버변수에 저장을 해놓거나 잘 모르는 경우가 많습니다. 그렇기 때문에 back() 을 이용하면 편하게 맨 뒤 인자를 가지고 올 수 있습니다.

함수 예시 : str1.back(); //"BlockDMask" -> 'K'를 리턴합니다.

♣ string size 관련

▶ str1.size();

함수 원형 : size_t size() const;

함수 설명 : string의 **사이즈를 반환**합니다.

함수 예시 : str1.size(); // "BlockDMask" 이므로 10을 반환합니다.

▶ str1.length();

함수 원형 : size_t length() const;

이를 반환합니다. size() 함수와 같다고 생각하면 됩니다.

함수 예시 : str1.length(); // "BlockDMask" 이므로 10을 반환합니다.

▶ str1.capacity();

함수 원형 : size_t capacity() const;

함수 설명 : string 객체에 **할당된 메모리 크기(bytes)를 반환**합니다.

capacity는 vector의 capacity와 마찬가지로 스트링 길이가 증가할 수 있기 때문에, 메모리 할당을 size에 대비해서 여유롭게 합니다.

size가 capacity를 넘게 될때 새롭게 더 큰 capacity(메모리)를 할당합니다.

함수 예시 : str1.capacity(); // "BlockDMask" 길이는 10인데 capacity는 15 입니다.

함수 예시 : str2.capacity(); // "BlogBlogBLogBlog" 길이는 16인데 capacity는 31 입니다.

15 -> 31 이런식으로 스트링의 capacity가 넘어가는 것을 알 수 있습니다.
길이가 10 인 스트링을 우리가 str1 += "a" 이런식으로 계속해서 길이를 늘리다가 16이 되는순간 기존의 capacity를 넘어가므로 더 큰 capacity를 할당하여서 스트링을 다루게 됩니다.

▶ str1.resize(n);

함수 원형 : void resize (size_t n);

함수 원형 : void resize (size_t n, char c);

함수 설명 : string을 **n만큼의 크기**로 만듭니다.

만약 그 크기가 원래 사이즈 보다 작다면, 남은 스트링을 버립니다.

만약 그 크기가 원래 사이즈 보다 크다면, 빈 공간으로 남은 공간을 채웁니다. 만약 c를 사용한다면 **남은 공간을 c로 채울 수 있습니다.**

함수 예시 : str1.resize(5) // "BlockDMask" -> "Block" 이 됩니다. size는 5입니다.

함수 예시 : str1.resize(6) // "Block" -> "Block " 가 됩니다. 빈칸이 있습니다. size는 6이 되었습니다.

함수 예시 : str1.resize(10, 'a') // "Block " -> "Block aaaa" 가 됩니다. 사이에 빈칸이 하나 있는거 보이시죠? q 추가했던 그것입니다.

▶ str1.shrink_to_fit();

함수 원형 : void shrink_to_fit();

함수 설명 : C++11 입니다. 이 함수는 스트링 길이에 비해 낭비되고 있는 **capacity(메모리)를 줄이는 함수**입니다.

함수 예시 : str2.resize(4); // "BlogBlogBlogBlog" -> "Blog" 가 됩니다. 아까 capacity에서 보자면 길이가 16짜리 string의 capacity는 31 이었습니다.

함수 예시 : str2.shrink_to_fit(); // "Blog" 처럼 길이가 4가 된 스트링의 capacity에 알맞게 **str2의 capacity가 31에서 16으로 줄어들게 됩니다.**

▶ str1.reserve(n);

ve(size_t n = 0);

함수 설명 : 문자열을 넣기 전에 미리 "곧 n만큼의 크기의 스트링이 들어올거니까 그에 맞는 capacity를 할당해 달라"는 함수 입니다.

이건 보통 파일을 읽을 때 사용을 하는데요, 3000자 짜리 파일을 우리가 읽게 된다하면, 파일에 있는 글을 한자씩 가지고 오게되는데 이게 while문에서 eof를 이용해서 파일의 끝 까지 읽게 됩니다. 이런 경우에는 미리 메모리를 할당해서, capacity가 사이즈에 맞게 계속 늘어나는 행위를 덜하게해서 성능 저하를 줄이게 하는 것 입니다.

만약에 reserve를 하지 않고 그냥 파일을 읽는다면 아마 이렇게 될 것 입니다. str += "글" 이렇게 계속 파일이 끝이 나올 때 까지 더해질 것이고, str의 길이가 16이 되면 새롭게 capacity를 늘리는 작업이 들어가서 비용이 들게되고, 또 str의 길이가 32가 되면 새롭게 capacity를 늘리는 작업이 들어가게 되고..... 계속 .... 계속.... 늘리는 작업을 하게 됩니다. 이런 비효율적인 작업을 하지 않게 하려고 미리 메모리 예약을 하는 것 입니다.

쉽게 말해서 노래방에 2인실을 갔는데, 2명이 더와서 4인실을 갔는데 또 여러명이 와서 8인실을 가고 이렇게 사람들이 이동하는 동안 우리는 노래를 못부르고 시간이 줄잖아요. 컴퓨터도 마찬가지 입니다 한번 할당해 놓은 메모리를 또 추가로 할당한다는 것은 그만큼 비용이 생기게 됩니다.

▶ str1.clear();

함수 원형 : void clear();

함수 설명 : 스트링에 들어있는 **문자열을 지우는 함수**입니다. 이때, **size와 length는 0**이 되고, **capacity는 그대로 남게 됩니다.**

함수 예시 : str1.clear(); //"BlockDMask" -> "" 이 됩니다. size와 lenth는 0이 되고, capacity는 15 그대로 입니다. (메모리 해제가 아닌 문자열 값들을 삭제하는것)

▶ str1.empty();

함수 원형 : bool empty() const;

함수 설명 : 스트링이 **비었는지 확인하는 함수**입니다. 비었으면 true를 반환합니다. 비었음의 기준은 size, length가 0인 것 입니다. capacity와는 관계가 없습니다.

함수 예시 : if(str1.empty()) { return true; } //이런식으로 비었는지 확인하면됩니다. clear를 사용하고 empty를 사용하면 당연히 true 겠죠?

♣ string 가지고 놀기 종류

▶ str1.c_str()

함수 원형 : const char* c_str() const;

함수 설명 : C++ 스타일의 string 문자열을 C스타일의 문자열로 변경해주는 함수입니다.

함수 예시 : const char* arr = str1.c_str(); // "BlockDMask"가 "BlockDMask₩0"로 반환해 줍니다. C언어에서 처럼 사용할 수 있습니다.

▶ str1.substr(....)

str(size_t index = 0, size_t len = npos) const;

함수 설명 : string을 **index 에서부터 len만큼 잘라서 반환**하는 함수입니다. 스트링 짤라서 반환.

두번째 인자의 len의 디폴트 npos가 의미하는 것은 "-1"입니다. size_t의 타입은 unsigned int 타입니다.

그 unsigned int에 -1이 들어온다? 그게 무슨의미 일까요? 맞습니다.

"언더플로우" 입니다. **unsigned인 타입에 음수**를 넣으면 **제일 큰 값으로 세팅**이 됩니다.

그렇기 때문에 npos에는 -1을 넣어서, 아무것도 넣지 않았을때는 항상 **문자열이 길어질 수 있는 최대의 길이**를 나타내게 됩니다.

함수 예시 : str1.substr(); //"BlockDMask" 그대로 반환합니다.

함수 예시 : str1.substr(5); // "DMask"를 반환합니다. 0부터 세기 시작해서 "5" 번째 인자부터 끝까지의 문자열을 반환합니다.

함수 예시 : str1.substr(5, 1); // "D"를 반환합니다. 5번째 인자부터, 1의 길이만큼 문자열을 반환합니다.

▶ str1.replace(....)

함수 원형 : string& replace(size_t index, size_t len, const string& str)

함수 설명 : 함수를 호출하는 문자열의 index위치에서 **len 길이까지의 범위**를 매개변수로 들어온 **str 전체로 대체** 하는 함수입니다.

함수 예시 : str1.replace(5, 2, str2);

// "Block**DM**ask"의 5번째 인자에서부터 2개를 str2로 대체하게 됩니다. 그러면 "Block**BlogBlogBlogBlog**ask" 이런식의 문자가 됩니다.

▶ str1.compare(....)

함수 원형 : int compare(const string& str) const;

함수 원형 : int compare(size_t index, size_t len, const string& str) const;

함수 설명 : 매개변수로 들어온 str을 비교해서 같으면 0을 반환하고, 다르면 0이 아닌 값을 반환하는 함수입니다.

호출하는 스트링의 값이 매개변수로 들어온 스트링의 값보다 작을때(사전순 빠를때) 음수(-1)를 반환하고

호출하는 스트링의 값이 매개변수로 들어온 스트링의 값보다 클때(사전순 느릴때) 양수(1)를 반환합니다

함수 예시 : str1.compare(str2); // "BlockDMask", "BlogBlogBLogBLog"는 같지 않기 때문에 0이 아닌 값을 반환하고, Blo까지는 둘이 똑같고, 그다음 c, g를 비교하게 됩니다. 이때, c가 g보다 사전상 더 앞선글자이기 때문에 c가 더 작은 글자가 됩니다. 그렇기 때문에 음수(-1)를 반환하게 됩니다.

함수 예시 : str2.compare(str1); // 이거는 그럼 양수(1)을 반환하겠죠?

함수 예시 : str1.compare("BlockDMask"); //이 경우에는 두 스트링이 같기 때문에 0을 반환합니다.

함수 예시 : str1.compare(0, 2, str2); //"BlockDMask"와 "BlogBlogBlogBlog" 스트링의 0번째 인덱스부터 2번째 인덱스까지 하면, **"Blo"**가 같기 때문에 0을 반환합니다.

▶ str1.copy(....)

함수 원형 : size_t copy(char* arr, size_t len, size_t index = 0) const;

함수 설명 : 딱봐도 복사를 하는 함수입니다. 이거는 특이하게 길이는 나타내는 len이 두번째고, index가 세번째 인자입니다. 아무래도 시작하는곳이 더 중요하지 않다고 판단하고 디폴트 인자를 설정해주려고 맨뒤로 보낸걸로 보입니다.

첫번째 매개변수 : 호출한 문자열을 첫번째 매개변수 문자열에 복사하는 함수입니다. char* 인걸 보면, C언어의 문자열(배열타입)을 받습니다.

두번째 매개변수는 복사할 문자열의 길이를 나타냅니다.

세번째 매개변수는 복사를 시작할 위치 입니다. index는 0부터 시작하는거 아시죠?

마지막으로 실제로 복사된 길이, arr의 길이를 반환합니다.

함수 예시 : char arr[10]; //문자열을 복사해서 넣을 빈 배열을 만듭니다.

함수 예시 : int arrLen = str1.copy(arr, 3, 5); //5번째 index부터 3의 길이만큼 복사 한다는 거니까 "BlockDMask" 빨간 부분이 들어갔을겁니다. 그리고 반환하는 arrLen은 3의 길이겠죠?

함수 예시 : arr[arrLen] = '₩0'; //그리고 C의 문자열의 끝에는 '₩0'이걸 넣어주어야 합니다. 그러면 문자열 복사가 깔끔하게 되었을 겁니다.

▶ str1.find(....)

함수 원형 : size_t find (const string& str, size_t index = 0) const;

함수 원형 : size_t find (const char* arr, size_t index = 0) const;

함수 설명 : 매개변수로 들어온 문자열과, 내 문자열중에 일치하는 게 있는지 확인하는 함수입니다.

만약에 일치하는게 있다면, 일치하는 부분의 첫번째 순서(index)를 반환합니다.

두번째 매개변수로 들어온 index는 어느 위치에서 부터 찾을까 입니다.

함수 예시 : str2.find("Blog"); // "BlogBlogBlogBlog" -> 0 을 반환합니다.

함수 예시 : str2.find("Blog", 5); // 5번째 인자부터 blog를 찾게되니 BlogBlogBlogBlog 빨간색 I에서부터 찾게되므로 그 다음에 나오는 b의 위치인 8을 반환할 것 입니다.

▶ str1.push_back(c)

함수 원형 : void push_back(char c);

함수 설명 : 함수를 호출하는 스트링의 맨뒤에 문자 c를 더하는 함수 입니다.

함수 예시 : str1.push_back('a'); //"BlockDMask" -> "BlockDMaska" 'a'가 하나 더해집니다.

▶ str1.pop_back()

함수 원형 : void pop_back()

함수 설명 : 함수를 호출하는 스트링의 맨뒤에 있는 문자 하나를 없애는 함수 입니다.

함수 예시 : str1.pop_back(); // "BlockDMask" -> "BlockDMas" 이렇게 k가 하나 빠집니다.

♣ string iterator 종류

▶ str1.begin();

함수 원형 : iterator begin();

함수 원형 : const_iterator begin() const;

함수 설명 : 문자열의 첫 번째 문자를 가리키는 반복자(iterator 포인터)를 반환합니다.

▶ str1.end();

함수 원형 : iterator end();

함수 원형 : const_iterator end() const;

함수 설명 : 문자열의 마지막의 바로 다음을 가리키는 반복자(iterator 포인터)를 반환합니다. (잘 봐야 해요 문자열 끝이 아니라 그 다음!)

iterator는 보통 순회를 할때 많이 사용합니다. 아래와 같은 방식으로 많이 쓰죠.

아래와 같이 하면, B 엔터 I 엔터 이런식으로 한글자씩 나오게 됩니다.

```
for (string::iterator iter = str1.begin(); iter != str1.end(); ++iter)
{
    cout << *iter << endl;
}
```

Colored by Color Scripter cs

♣ string의 기타등등.

▶ swap(str1, str2);

함수 원형 : void swap(string& str1, string& str2);

를 바꾸는 것 입니다.

스왑을 할때 복사를 해서 스왑을 하는것이 아니라 **서로 참조(reference)를 교환**해서 스왑을 합니다.
그렇기 때문에 복사에 의한 성능저하를 우려할 필요가 없습니다.

함수 예시 : swap(str1, str2); //str1이 "blog~~"가 되고, str2가 "BlockDMask"가 됩니다.

▶ operator+

함수 설명 : 이거는 오퍼레이터 +인데요. string끼리 더할 수 있습니다. 더한다는 의미는 이어 붙인다는 것 입니다.

이미 만들어져 있는 것이라, 우리가 그냥 string끼리 더해서 사용하면 됩니다.

함수 예시 : str1 += str2; //str1은 "**BlockDMaskBlogBlogBlogBlog**" 이런식으로 만들어집니다.

3. C++ string 클래스 예제1 (생성과 출력, 문자열 연결, push_back, pop_back)

```cs
//C++ string example1
//BlockDMask.
#include<iostream>
#include<string>
using namespace std;

int main()
{
    string str1 = "BlockDMask";
    string str2("BlogBlogBlogBlog");
    string str3(str1);

    //string을 생성하는 다양한 방법.
    cout << "==> onstructor" << endl;
    cout << "str1 = \"BlockDMask\" \n : " << str1 << endl << endl;
    cout << "str2(\"BlogBlogBlogBlog\") \n : " << str2 << endl << endl;
    cout << "str3(str1) \n : " << str3 << endl << endl;

    //push_back, pop_back
    cout << endl;
    cout << "==> string front, back example." << endl;
    str1.push_back('1');
    cout << "str1.push_back('1') : " << str1 << endl;
    str1.push_back('2');
    cout << "str1.push_back('2') : " << str1 << endl;
    str1.pop_back();
    cout << "str1.pop_back() :  " << str1 << endl;
    str1.pop_back();
    cout << "str1.pop_back() :  " << str1 << endl;
    str1.pop_back();
    cout << "str1.pop_back() :  " << str1 << endl;

    //                r+ stirng 덧셈.
    //                endl;
```

```
    cout << "==> str1 += str2" << endl;
    str1 += str2;
    cout << str1 << endl;

    cout << endl;
    system("pause");
    return 0;
}
```

Colored by Color Scripter

▲ C++ string 예제1 (생성과 출력, 문자열 연결, push_back, pop_back)

stirng을 생성하는 3가지 방법에 대한 예제

그리고 string끼리 연결하기, 더하기, 맨뒤에 문자 추가하기, 맨뒤에있는 문자 빼기

4. C++ string 클래스 예제2 (size, capacity, length, clear, shrink_to_fit)

```
//C++ string example2
//BlockDMask.
#include<iostream>
#include<string>
using namespace std;

int main()
{
    string str2 = "BlogBlogBlogBlog";

    cout << "==> str2 original" << endl;
    cout << "str2 : " << str2 << endl;
    cout << "str2.size() : " << str2.size() << endl;
    cout << "str2.length() : " << str2.length() << endl;
    cout << "str2.capacity() : " << str2.capacity() << endl;
```

cs

```
    cout << endl;
    str2.resize(4);
    cout << "==> str2.resize(4)" << endl;
    cout << "str2 : " << str2 << endl;
    cout << "str2.size() : " << str2.size() << endl;
    cout << "str2.length() : " << str2.length() << endl;
    cout << "str2.capacity() : " << str2.capacity() << endl;

    cout << endl;
    str2.shrink_to_fit();
    cout << "==> str2.shrink_to_fit()" << endl;
    cout << "str2 : " << str2 << endl;
    cout << "str2.size() : " << str2.size() << endl;
    cout << "str2.length() : " << str2.length() << endl;
    cout << "str2.capacity() : " << str2.capacity() << endl;

    cout << endl;
    str2.clear();
    cout << "==> str2.clear()" << endl;
    cout << "str2 : " << str2 << endl;
    cout << "str2.size() : " << str2.size() << endl;
    cout << "str2.length() : " << str2.length() << endl;
    cout << "str2.capacity() : " << str2.capacity() << endl;

    cout << endl;
    system("pause");
    return 0;
}
```

Colored by Color Scripter

```
==> str2 original
str2 : BlogBlogBlogBlog
str2.size() : 16
str2.length() : 16
str2.capacity() : 31

==> str2.resize(4)
str2 : Blog
str2.size() : 4
str2.length() : 4
str2.capacity() : 31

==> str2.shrink_to_fit()
str2 : Blog
str2.size() : 4
str2.length() : 4
str2.capacity() : 15

==> str2.clear()
str2 :
str2.size() : 0
str2.length() : 0
str2.capacity() : 15
```

▲ C++ string 예제2 (size, capacity, length, clear, shrink_to_fit)

1. string에서 size와 length가 같은지.

y가 무엇인지.

3. resize(), shrink_to_fit() 함수와 clear() 함수도 살펴봐 주세요.

5. C++ string 클래스 예제3 (substr, replace, swap)

```
1  //C++ string example3.
2  //BlockDMask.
3  #include<iostream>
4  #include<string>
5  using namespace std;
6
7  int main()
8  {
9      string str1 = "BlockDMask";
10     string str2 = "BlogBlogBlogBlog";
11
12     cout << "str1 : " << str1 << endl;
13     cout << "str2 : " << str2 << endl;
14     cout << endl;
15
16     //string substr example.
17     cout << "str1.substr(5) : " << str1.substr(5) << endl;
18     cout << "str1.substr(5,1) : " << str1.substr(5,1) << endl;
19
20     //string replace example.
21     cout << "str1.replace(5, 2, str2) : " << str1.replace(5, 2, str2) << endl
22
23     //string swap example.
24     cout << endl;
25     string str3 = "C++ example";
26     cout << "str2 : " << str2 << endl;
27     cout << "str3 : " << str3 << endl;
28
29     cout << endl;
```

```
    cout << "swap(str2, str3)" << endl;
    cout << "str2 : " << str2 << endl;
    cout << "str3 : " << str3 << endl;

    cout << endl;
    system("pause");
    return 0;
}
```

Colored by Color Scrip

```
str1 : BlockDMask
str2 : BlogBlogBlogBlog

str1.substr(5) : DMask
str1.substr(5,1) : D
str1.replace(5, 2, str2) : BlockBlogBlogBlogBlogask

str2 : BlogBlogBlogBlog
str3 : C++ example

swap(str2, str3)
str2 : BlogBlogBlogBlog
str3 : C++ example
```

▲ C++ string 예제3 (size, capacity, length, clear, shrink_to_fit)

1. str1의 substr(5) : index가 5번째인 값부터 끝까지 문자열을 반환.
2. str1의 substr(5,1) : index가 5번째인 값부터 1 길이의 문자열을 반환.
3. str1.replace(5, 2, str2) : str1의 5번째인 자리부터 2개의 문자열을 str2문자열 전체로 대체.
4. swap(str2, str3) : str2와 str3의 참조를 서로 바꾸어줌.

6. C++ string 클래스 예제4 (find, operaotr[], at, front, back)

```
//C++ string example4.
//BlockDMask.
#include<iostream>
#include<string>
using namespace std;

int main()
{
    string str1 = "BlockDMask";
    string str2 = "BlogBlogBlogBlog";

    cout << "str1 : " << str1 << endl;
    cout << "str2 : " << str2 << endl;
    cout << endl;

    //find
    ==> string find example." << endl;
    cout << "str1.find(\"DM\") : " << str1.find("DM") << endl;
```

```
    cout << "str2.find(\"Blog\") : " << str2.find("Blog") << endl;
    cout << "str2.find(\"Blog\", 5) : " << str2.find("Blog", 5) << endl;

    //operator[], at
    cout << endl;
    cout << "==> string operator[], at() example." << endl;
    cout << "str1[0] : " << str1[0] << endl;
    cout << "str1[3] : " << str1[3] << endl;
    cout << "str1[str1.size()-1] : " << str1[str1.size() - 1] << endl;
    cout << endl;
    cout << "str1.at(0) : " << str1.at(0) << endl;
    cout << "str1.at(3) : " << str1.at(3) << endl;
    cout << "str1.at(str1.size()-1) : " << str1.at(str1.size() - 1) << endl;

    //front, back
    cout << endl;
    cout << "==> string front, back example." << endl;
    cout << "str1[0] : " << str1[0] << endl;
    cout << "str1.at(0) : " << str1.at(0) << endl;
    cout << "str1.front() : " << str1.front() << endl;
    cout << endl;
    cout << "str1[str1.size()-1] : " << str1[str1.size() - 1] << endl;
    cout << "str1.at(str1.size()-1) : " << str1.at(str1.size() - 1) << endl;
    cout << "str1.back() : " << str1.back() << endl;

    cout << endl;
    system("pause");
    return 0;
}
```

Colored by Color Scripter

```
str1 : BlockDMask
str2 : BlogBlogBlogBlog

==> string find example.
str1.find("DM") : 5
str2.find("Blog") : 0
str2.find("Blog", 5) : 8

==> string operator[], at() example.
str1[0] : B
str1[3] : c
str1[str1.size()-1] : k

str1.at(0) : B
str1.at(3) : c
str1.at(str1.size()-1) : k

==> string front, back example.
str1[0] : B
str1.at(0) : B
str1.front() : B

str1[str1.size()-1] : k
str1.at(str1.size()-1) : k
```

▲ C++ string 예제4 (find, operator[], at, front, back)

1. find의 사용법.

2. string 인자 접근법.

3. 그리고.. 맨앞과 맨 끝 접근법.

front vs str1[0]

back vs str1[str1.size()-1]

어떤걸 사용하시든, 개발하시는분 마음이지만 이왕이면 저는 front와 back을 추천합니다.

7. C++ string 클래스 예제5 (begin, end, copy, compare)

```cpp
//C++ string example5
//BlockDMask.
#include<iostream>
#include<string>
using namespace std;

int main()
{
    string str1 = "BlockDMask";
    string str2 = "BlogBlogBlogBloc";

    cout << "str1 : " << str1 << endl;
    cout << "str2 : " << str2 << endl;
    cout << endl;

    //begin, end
    cout << "==> begin, end" << endl;
    string::iterator iter = str1.begin();
    for (; iter != str1.end(); ++iter)
    {
        cout << *iter << " ";
    }
    cout << endl;
```

```
26      cout << endl;
27      cout << "==> copy " << endl;
28      char arr[10];
29      int arrLen = str1.copy(arr, 3, 5);
30      cout << "str1.copy(arr, 3, 5)" << endl;
31      cout << "arrLen : " << arrLen << endl;
32      cout << arr << endl;    //error.
33      arr[arrLen] = '\0';
34      cout << arr << endl;    //ok.
35
36      //compare
37      cout << endl;
38      cout << "==> compare" << endl;
39      cout << "str1.compare(\"BlockDMask\") : " << str1.compare("BlockDMask")
40      cout << "str1.compare(str2) : " << str1.compare(str2) << endl;
41      cout << "str2.compare(str1) : " << str2.compare(str1) << endl;
42      cout << endl;
43      system("pause");
44      return 0;
45  }
```

Colored by Co

```
str1 : BlockDMask
str2 : BlogBlogBlogBloc

==> begin, end
B l o c k D M a s k

==> copy
str1.copy(arr, 3, 5)
arrLen : 3
DMa薇薇薇薇薇薇薇薇薇缺妻
DMa

==> compare
str1.compare("BlockDMask") : 0
str1.compare(str2) : -1
str2.compare(str1) : 1
```

▲ C++ string 예제4 (find, operator[], at, front, back)

1. begin, end 반복자를 이용한 순회

2. copy를 했을때는 꼭 맨뒷자리에 '₩0'을 잊지말아주세요. 저기 DMa 뒤에 쓰레기값이 들어가는 거 보이시죠?

C언어에서, char 배열에서 문자열의 끝은 '₩0' 한번 더 강조합니다.

3. compare는 같으면 0, 다르면 -1, 1 입니다.

-1은 불러주는 스트링이 사전순 앞

1은 불러주는 스트링이 사전순 뒤

감사합니다. 이번 포스팅은 역대급으로 기네요.

"구독 중이오"는 제게 큰 힘이 됩니다.

'<개인공부> > [C++]' 카테고리의 다른 글

SECRE WRITE

PRIMI *2019.10.15 11:39*

정리하신 내용이 큰 도움이 되었습니다! 감사합니다. 공부하는데 잘쓰겠습니다!

 # Delete Reply

사용자 BlockDMask *2019.11.18 10:27 신고*

감사합니다 또방문해주세요!

 # Delete

Kyun2da *2019.11.14 18:13*

항상 잘보고 있습니다. 덕분에 STL 잘 배우고 있어요 복받으세요!

 # Delete Reply

사용자 BlockDMask *2019.11.18 10:28 신고*

네 복 감사히 받겠습니다
더 좋은 정보를 가지고 기다리겠습니다
또오세요~!

 # Delete

+ 방문 감사합니다. 좌측에 더 많은 글이 있습니다.