



C++ Korea와 함께하는 제2회 마이크로소프트 멜팅팟 세미나

빠르게 변화하는 모든 C++에 몸을 맡겨라!

2016. 5. 28(토) 13:00 ~ 18:00
한국마이크로소프트 대회의실(11층)



C++ String 파고들기

Lyn Heo / C++ Korea FB Group
Microsoft MVP



- 2016년 현재 다국어 지원은 필수라고 할 수 있습니다
- Global One Build 얘기도 심심치 않게 나오고 있는 시대
- 하지만 아직 세계는 하나의 문자열 코드로 통일 되지 않았다
- 그럼 C++은 무엇을 도와 줄 수 있으며 최신 C++에서는 무엇이 달라졌을까요

- 1991년 시작
- 1996년 현재의 유니코드와 호환되는 최초의 버전 발표
 - 한글 재배치 이슈. 이후 재배치는 없다는 규정 추가됨
- 2010년 6.0 발표
 - 에모지(絵文字) 추가
- 2016년 5월 현재 8.0이 최신. 9.0 베타 진행중
















한국마이크로소프트 를 표현해봅시다






















빠르게 변화하는
모던 C++에 몸을 맡겨라!



















































완성형(CP949)	C7 D1 B1 B9 B8 B6 C0 CC C5 A9 B7 CE BC D2 C7 C1 C6 AE
조합형 (ks_c_5601-1992/CP1361)	D0 65 8A 82 A0 61 B7 A1 C7 61 9D A1 Ad A1 CF 61 CB 61 0A
UTF-8	ED 95 9C EA B5 AD EB A7 88 EC 9D B4 ED 81 AC EB A1 9C EC 86 8C ED 94 84 ED 8A B8
UTF-16 LE	5C D5 6D AD C8 B9 74 C7 6C D0 5C B8 8C C1 04 D5 B8 D2
UTF-32 LE	5C D5 00 00 6D AD 00 00 C8 B9 00 00 74 C7 00 00 6C D0 00 00 5C B8 00 00 8C C1 00 00 04 D5 00 00 B8 D2 00 00
첫가끝 UTF-8	E1 84 92 E1 85 A1 E1 86 AB E1 84 80 E1 85 AE E1 86 A8 E1 84 86 E1 85 A1 E1 84 8B E1 85 B5 E1 84 8F E1 85 B3 E1 84 85 E1 85 A9 E1 84 89 E1 85 A9 E1 84 91 E1 85 B3 E1 84 90 E1 85 B3

최신 유니코드에 있는 문자들

빠르게 변화하는
모던 C++에 몸을 맡겨라!

	1F00	1F01	1F02
0	 1F000	 1F010	 1F020
1	 1F001	 1F011	 1F021
2	 1F002	 1F012	 1F022
3	 1F003	 1F013	 1F023
4	 1F004	 1F014	 1F024

	1F0A	1F0B	1F0C	1F0D	1F0E	1F0F
0	 1F0A0				 1F0E0	 1F0F0
1	 1F0A1	 1F0B1	 1F0C1	 1F0D1	 1F0E1	 1F0F1
2	 1F0A2	 1F0B2	 1F0C2	 1F0D2	 1F0E2	 1F0F2
3	 1F0A3	 1F0B3	 1F0C3	 1F0D3	 1F0E3	 1F0F3

 1F553	 1F563	 1F573	 1F583	 1F593	 1F5A3	 1F5B3	 1F5C3	 1F5D3	 1F5E3
 1F554	 1F564	 1F574	 1F584	 1F594	 1F5A4	 1F5B4	 1F5C4	 1F5D4	 1F5E4
 1F555	 1F565	 1F575	 1F585	 1F595	 1F5A5	 1F5B5	 1F5C5	 1F5D5	 1F5E5
 1F556	 1F566	 1F576	 1F586	 1F596	 1F5A6	 1F5B6	 1F5C6	 1F5D6	 1F5E6
 1F557	 1F567	 1F577	 1F587	 1F597	 1F5A7	 1F5B7	 1F5C7	 1F5D7	 1F5E7

각 언어 별 유니코드 인코딩

빠르게 변화하는
모던 C++에 몸을 맡겨라!

- C++ : 컴파일러 구현하는 사람 마음대로
 - VC++ : UTF-16
 - GCC : UTF-32
- Java, C# : UTF-16
- Python 3.0 ~ 3.2 : UTF-32
- Python 3.3 ~ : 가변 인코딩
- Delphi 2009~ : 가변 인코딩
- ~~PHP 6 : UTF-16 취소됨~~
- PHP 7 : UTF-8

OS별 유니코드 지원 여부

빠르게 변화하는
모던 C++에 몸을 맡겨라!

- MSDOS 기반 OS(DOS ~ Windows ME) : ASCII 기반 OS
- NT 기반 OS(Windows NT 3.1) : 유니코드 기반 OS
- Linux : 유니코드 기반 OS
- MacOS (~ MacOS 9.2.2) : ASCII 기반 OS
- MacOS(~ OS X 11.1) : 유니코드 기반 OS

- UTF-8
 - 로마자를 쓰는 경우 매우 효율적
 - 그 외의 경우 비효율적
- UTF-16
 - 로마자를 쓰는 경우 비효율적
 - 그 외의 경우 효율적
- UTF-32
 - 특별한 경우를 제외 하고는 매우 비효율적

인코딩 별 문자Type 범위 차이

빠르게 변화하는
모던 C++에 몸을 맡겨라!

- UTF-8
 - 로마자 / 숫자 표현 가능 (ASCII 호환성 100%)
- UTF-16
 - 모든 현대어 표현 가능
- UTF-32
 - 모든 유니코드 표현 가능

- 일반 문자열과 Wide 문자열의 2종류의 리터럴 제공
- Wide 문자 지원을 위한 L 접두사 제공
- `std::string` / `std::wstring` 지원
 - 언어의 일부가 아닌 표준 라이브러리의 일부
- `string` 으로의 변환은 `std::stringstream` 을 통함

- C++에서 유니코드 지원을 위한 스펙으로서 지원
- 표준 차원에서 type 의 크기나 인코딩 등이 정해져 있지 않음
 - 인코딩 지옥의 시작
- 결과적으로 크로스 플랫폼 라이브러리에서 String Literal 을 묶는 별도의 매크로를 제공 하게 되는 원흉이 됨
 - Ex : Unreal Engine 의 TEXT() Macro 등.

Modern C++ String

빠르게 변화하는
모던 C++에 몸을 맡겨라!

- 인코딩 별 새로운 문자, 문자열 Type 제공
- to_string, to_wstring 등의 간단한 변환 함수 제공
- move constructor 지원
- 인코딩 변경 함수 지원

Modern C++의 새로운 문자 Type

빠르게 변화하는
모던 C++에 몸을 맡겨라!

인코딩	MultiByte	UTF-8	UTF-16	UTF-32
문자열 Literal	"string"	u8"String"	u"string"	U"string"
문자 Literal	'S'		u'S'	U'S'
문자열 객체	std::string		std::u16string	std::u32string
문자 type	char		char16_t	char32_t
RAW 문자열 Literal	R"(string)"	u8R"(string)"	uR"(string)"	UR"(string)"

어???

왜 UTF-8 문자는 없나요?

Title : type.맑은 고딕 / size. 40pt

빠르게 변화하는
모던 C++에 몸을 맡겨라!

- utf-8 은 1바이트 단위의 인코딩
 1. utf-8 은 1 바이트 문자의 집합
 2. 그럼 char8_t 는 1바이트 크기의 char type 이 되어야 한다.
 3. utf-8 이 1바이트로 표현 할 수 있는 문자는 0x00~0x7F 까지
- char8_t 는 ASCII 를 표현하기 위한 char 과 표현 범위에서 달라지는 것이 하나도 없다!

빠르게 변화하는
모던 C++에 몸을 맡겨라!



그래서 ...

아직도 u8 문자는 논의 중...

(C++ 17 예정)

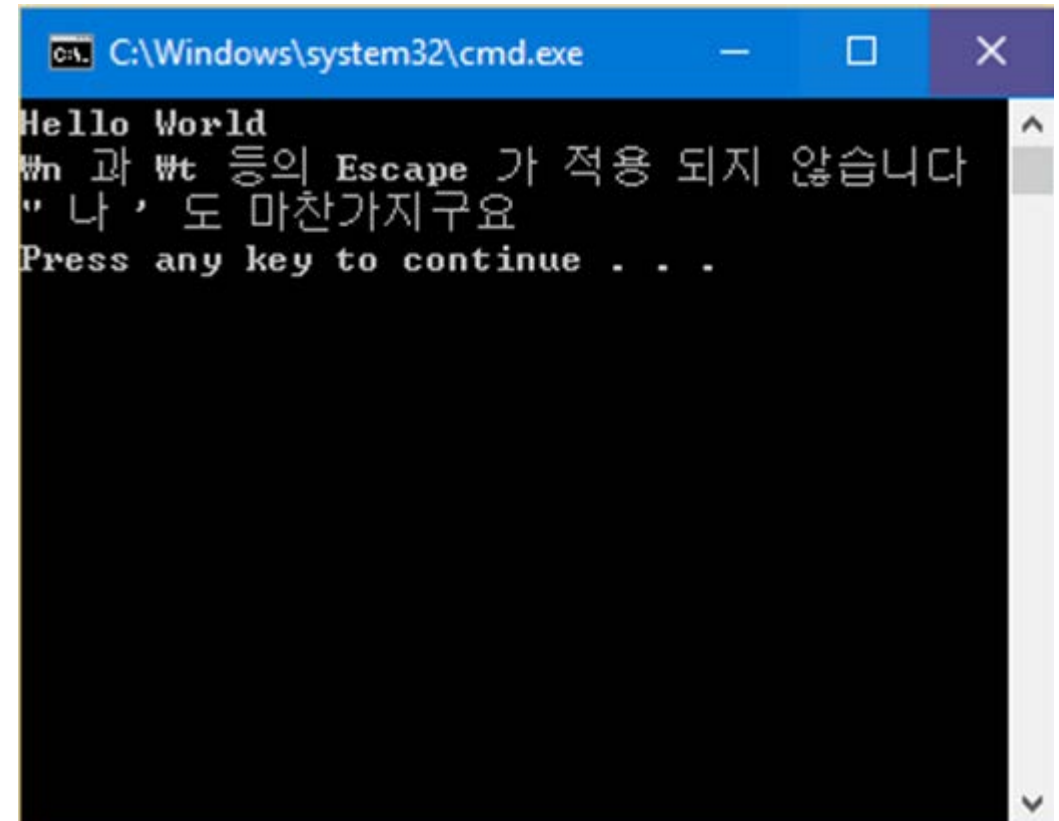
Raw String Literals

빠르게 변화하는
모던 C++에 몸을 맡겨라!

Escape sequence 가 적용 되지 않는 문자열
줄 바꿈, \w, "" 등의 문자를 모두 그대로 출력

```
int main()
{
    string s = R"(Hello World
\n과 \t 등의 Escape 가 적용
되지 않습니다
" 나 ' 도 마찬가지로요)";

    cout << s << endl;
}
```



The screenshot shows a Windows command prompt window titled "C:\Windows\system32\cmd.exe". The output of the program is displayed as follows:

```
Hello World
\n과 \t 등의 Escape 가 적용 되지 않습니다
" 나 ' 도 마찬가지로요
Press any key to continue . . .
```

The output demonstrates that the raw string literal R"(...) " correctly preserves the backslashes and quotes within the string, as well as the line breaks, without applying any escape sequence processing.

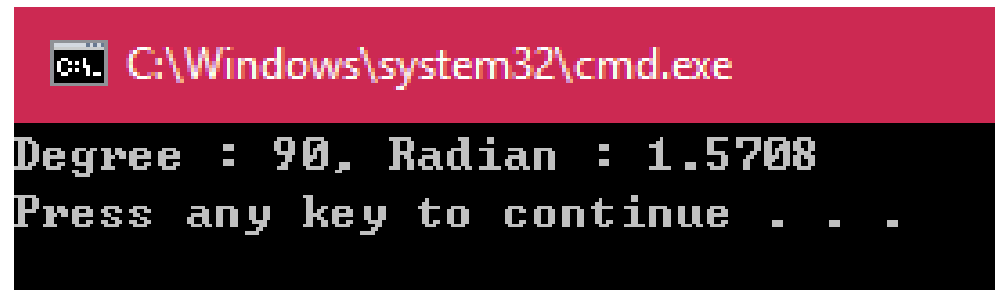
User-Defined Literals

빠르게 변화하는
모던 C++에 몸을 맡겨라!

```
constexpr long double operator"" _degree(long double value)
{
    return value * M_PI / 180;
}

int main()
{
    double rad = 90.0_degree;

    cout << "Degree : 90, Radian : " << rad << endl;
}
```



C:\Windows\system32\cmd.exe

Degree : 90, Radian : 1.5708
Press any key to continue . . .

s 접미사 추가

빠르게 변화하는
모던 C++에 몸을 맡겨라!

문자열 리터럴이 string
객체를 반환

```
auto str = "Hello World!"s;
```

```
inline namespace literals {  
    inline namespace string_literals {  
        inline string operator "" s(const char *_Str, size_t _Len)  
        {    // construct literal from [_Str, _Str + _Len)  
            return (string(_Str, _Len));  
        }  
  
        inline wstring operator "" s(const wchar_t *_Str, size_t _Len)  
        {    // construct literal from [_Str, _Str + _Len)  
            return (wstring(_Str, _Len));  
        }  
  
        inline u16string operator "" s(const char16_t *_Str, size_t _Len)  
        {    // construct literal from [_Str, _Str + _Len)  
            return (u16string(_Str, _Len));  
        }  
  
        inline u32string operator "" s(const char32_t *_Str, size_t _Len)  
        {    // construct literal from [_Str, _Str + _Len)  
            return (u32string(_Str, _Len));  
        }  
    } // inline namespace string_literals  
} // inline namespace literals
```

- C++은 문자열 리터럴의 연결을 지원 합니다
 - `char* str = "Hello""World";`
 - `char* str = "HelloWorld";`
- 두 코드는 같습니다
컴파일 타임에 붙어있는 리터럴 을 연결 합니다
- Wide 문자열일 경우도 물론 가능합니다
 - `wchar_t* str = L"Hello"L"World";`

적어도 C++ 03 시절까진 그랬습니다

- 지금은 상황이 달라졌습니다.
- `wchar_t* str = L"Hello" L"World";`
위 코드에서 두번째 `L`이 User-Defined Literals 를 나타내는 키워드인지, 아니면 리터럴의 형식을 지정하는 접두사인지 접미사인지 구별이 불가능 합니다
- `wchar_t* str = L"Hello" L"World";` 처럼 공백이 필수

- UTF16 <-> 멀티바이트 변환함수 추가
 - mbrtoc16 / c16rtomb
- UTF32 <-> 멀티바이트 변환함수 추가
 - mbrtoc32 / c32rtomb
- UTF8 <-> UTF16 <-> UTF32 변환 스트림 필터 추가
 - codecvt_utf8, codecvt_utf16, codecvt_utf8_utf16,

codecvt 를 이용한 인코딩 변환

빠르게 변화하는
모던 C++에 몸을 맡겨라!

```
int main()
{
    wstring wstr = L"한국마이크로소프트";

    wstring_convert<codecvt_utf8_utf16<wchar_t>> convertor;
    string u8str = convertor.to_bytes(wstr);

    for (auto i = 0; i < u8str.size(); ++i)
    {
        printf("%#02x\n", (uint8_t)u8str[i]);
    }
}
```

```
0xed 0x81
0x95 0xac
0x9c 0xeb
0xea 0xa1
0xb5 0x9c
0xad 0xec
0xeb 0x86
0xa7 0x8c
0x88 0xed
0xec 0x94
0x9d 0x84
0xb4 0xed
0xed 0x8a
      0xb8
```

imbue 와의 조합

빠르게 변화하는
모던 C++에 몸을 맡겨라!

```
int main()
{
    std::wifstream file2("korea.ms");
    file2.imbue(std::locale(file2.getloc(),
        new std::codecvt_utf8_utf16<wchar_t>));
    std::cout << "파일을 읽음과 동시에 변환할 수도 있습니다";
}
```

지금까지 cout 을 3번, printf 를 한번 썼습니다
왜 printf 가 한번 있을까요?

아주 간단하고 심플한 출력문 한줄

```
printf("%#02x\n", i);
```

이걸 C시절의 함수를 사용하지 않고 C++ 표준 라이브러리 만으로 만들면?

```
cout << hex << showbase << setw(2) << setfill('0') << i << endl;
```

빠르게 변화하는
모던 C++에 몸을 맡겨라!

이건 뭐...



아직 C++은 문자열 자체를 조작하는 기능은
매우 취약
~~CString~~ 만세

C++ string 에서 지원 하지 않는 기능

빠르게 변화하는
모던 C++에 몸을 맡겨라!

- Format
- Upper / Lower
- Tokenize
- Trim
- Replace
- 기타 등등...

흔하게들 쓰는 기능인데 제공 되지 않습니다
MFC / ATL의 CString 도 지원 하던 기능인데...

~~최신언어표준은 무슨 ...~~

그렇다고 안 쓸 수는 없으니 대안이 필요합니다

fmt a.k.a. cppformat

빠르게 변화하는
모던 C++에 몸을 맡겨라!

- 공식 주소 : <http://fmtlib.net/>
- 장점
 - 속도가 빠름(printf랑 비슷. boost::format 대비 약 5배 빠름)
 - 기능이 다양함
 - 최신 C++을 이용한 편리한 사용법 제공
- 단점
 - 익숙하지 않은 Format 문법 (C# / python 스타일의 format 문법)
 - C함수와의 이름 충돌(namespace 안에 있긴 하지만..)

```
char buf[100];  
sprintf(buf, "%#x\n", i);
```

를 fmt 스타일로 다시 쓰면

```
auto s = format("{0:#x}\n", i);
```

Index를 생략 하는 것도 가능합니다

```
auto s = format("{:#x}\n", i);
```

이름을 가진 Index

빠르게 변화하는
모던 C++에 몸을 맡겨라!

```
int main()
{
    string Company = "마이크로소프트";
    string Country = "한국";

    auto s = format("{Country} 의 {Company}",
        arg("Country", Country), arg("Company", Company));

    cout << s;
}
```

User-Defined Literals 을 사용하여

빠르게 변화하는
모던 C++에 몸을 맡겨라!

```
int main()
{
    string Company = "마이크로소프트";
    string Country = "한국";

    auto s = format("{Country} 의 {Company}",
        "Country"_a = Country, "Company"_a = Company);

    cout << s;
}
```

```
int main()
{
    string Company = "마이크로소프트";
    string Country = "한국";

    cout << "{} 의 {}".format(Country, Company);
}
```

설마 모르시는 분은 없겠죠?

Modern C++ 이후에는 활용도가 많이 줄어들었지만, 아직도 사실상의 C++ 표준 라이브러리 확장으로 많이 사용

다음 표준에 들어 갈 라이브러리를 미리 테스트 해 보는 역할도

단지 기능 위주라 전반적으로 성능이 안좋은 편입니다...

boost tokenize

빠르게 변화하는
모던 C++에 몸을 맡겨라!

```
int main()
{
    typedef tokenizer<char_separator<char>> TOKC;

    string str = "Hello,World,Microsoft";
    char_separator<char> sep("/", "");

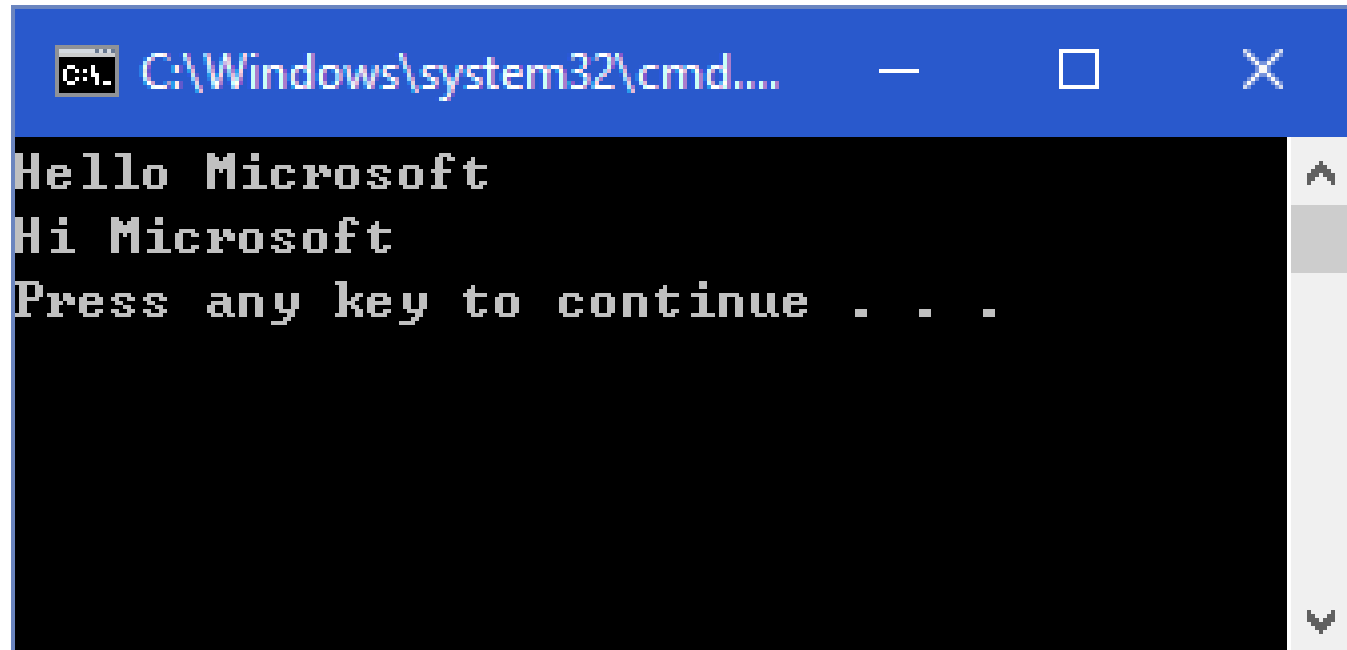
    TOKC tok(str, sep);

    for (TOKC::iterator i = tok.begin(); i != tok.end(); ++i)
    {
        cout << *i << endl;
    }
}
```


Trim Replace

빠르게 변화하는
모던 C++에 몸을 맡겨라!

```
int main()
{
    string str = " Hello Microsoft ";
    trim(str);
    cout << str << endl;
    replace_all(str, "Hello", "Hi");
    cout << str << endl;
}
```



A screenshot of a Windows command prompt window. The title bar is blue and contains the text "C:\Windows\system32\cmd...". The window has standard minimize, maximize, and close buttons. The command prompt itself has a black background with white text. It displays the output of the C++ program: "Hello Microsoft" on the first line, "Hi Microsoft" on the second line, and "Press any key to continue . . ." on the third line. A vertical scrollbar is visible on the right side of the window.

```
C:\Windows\system32\cmd...
Hello Microsoft
Hi Microsoft
Press any key to continue . . .
```

어? Upper / Lower는요?

Upper / Lower

빠르게 변화하는
모던 C++에 몸을 맡겨라!

```
int wmain()
{
    wstring str = L"wächst";
    wstring str2 = L"wächst";
    wstring str3 = L"wächst";
    icu::UnicodeString str4 = L"wächst";

    std::transform(str.begin(), str.end(), str.begin(), toupper);
    to_upper(str2);
    CharUpper((const_cast<wchar_t*>(str3.c_str())));
    str4.toUpper();
}
```

변환 결과는?

- toupper : WäCHST
- boost::to_upper : WäCHST
- Win32API CharUpper : WÄCHST
- icu::toUpper : WÄCHST

앞부분에 나왔지만 재방송 합니다

C++의 문자열 Type 은 언어의 일부가 아닙니다!

- International Components for Unicode ~~중환자실 아닙니다~~
 - <http://icu-project.org>
 - 2016년 5월 유니코드 컨소시엄에 참가
- 유니코드 관련 API를 가장 빠르게 반영 하는 단체
- 자바와 C++ 을 지원합니다
- 자체적인 UnicodeString Type 을 제공하며, 문자열에 관한 매우 많은 기능을 제공합니다
 - 개인적으로는 아직 까지는 기능 모자란 C++의 기본 문자열 보다는 icu의 UnicodeString을 사용하는 것을 추천

Thanks

빠르게 변화하는
모던 C++에 몸을 맡겨라!

감사합니다

첫가끝 코드

빠르게 변화하는
모던 C++에 몸을 맡겨라!

//