

함수

(Function)

들어가기 전에

- 오늘의 소스코드를 받아봅시다.
 - 중요한 코드는 꼭 다른 곳으로 복사해두세요.
 - phycom2015-2 디렉토리에서,
 - git stash
 - git pull
 - 디렉토리가 사라졌다면,
 - git clone
<https://github.com/geonmo/phycom2015-2.git>

함수란 무엇인가?

- 함수의 정의
 - 어떤 목적을 가진 프로그램 명령어들의 묶음
 - 사용자 정의 함수를 이용하여 특정 계산식 가능
 - 예) $f(x) = 4x^2 + 2x + 1$
 - C언어의 라이브러리를 통해 특정기능을 사용가능
 - 헤더파일을 include한 후 사용 가능
 - » 예) `abs`(절대값), `fabs`(float 절대값), `pow`(제곱)
 - 해당 기능의 자세한 내용은 cppreference.com을 이용하자!

함수의 사용법

- 사용 형태
 - 변수와 마찬가지로 "선언"과 "정의" 과정 필요
- 함수의 선언(Declare)
 - 함수의 형태만 알려주면 됨.
 - `float abs(float arg);`
 - float형을 입력 받아 float형의 반환(return)값을 돌려주는 함수
- 함수의 정의(Define)
 - `float abs(float arg) {
 if (arg<0) return -arg;
 else return arg; }`
 - 함수 자체의 소스코드 부분을 scope 안에 기입

함수의 사용법

- 함수의 위치
 - 사용하기 전에 반드시 "선언(Declare)" 작업을 할 것!
 - 함수 "선언" 예제 (5_UOS_abs2.cpp)
 - 다른파일에 "선언"부분을 만들고 include로 사용가능
 - 예제: 6_UOS_header.cpp
 - "선언"만 들어 있는 파일을 "헤더" 파일이라고 부른다.(UOSAbs.h)

4_UOS_abs.cpp

```
#include<iostream>
### std::abs와 이름이 같기 때문에 UOS::abs로.
namespace UOS {
    float abs( float arg) {
        if ( arg>=0. ) return arg;
        else return -arg;
    }
}

int main()
{
    float a = 3.02;
    float b = -22.21;

    std::cout<<UOS::abs(a)<<std::endl;
    std::cout<<UOS::abs(b)<<std::endl;

    return 0;
}
```

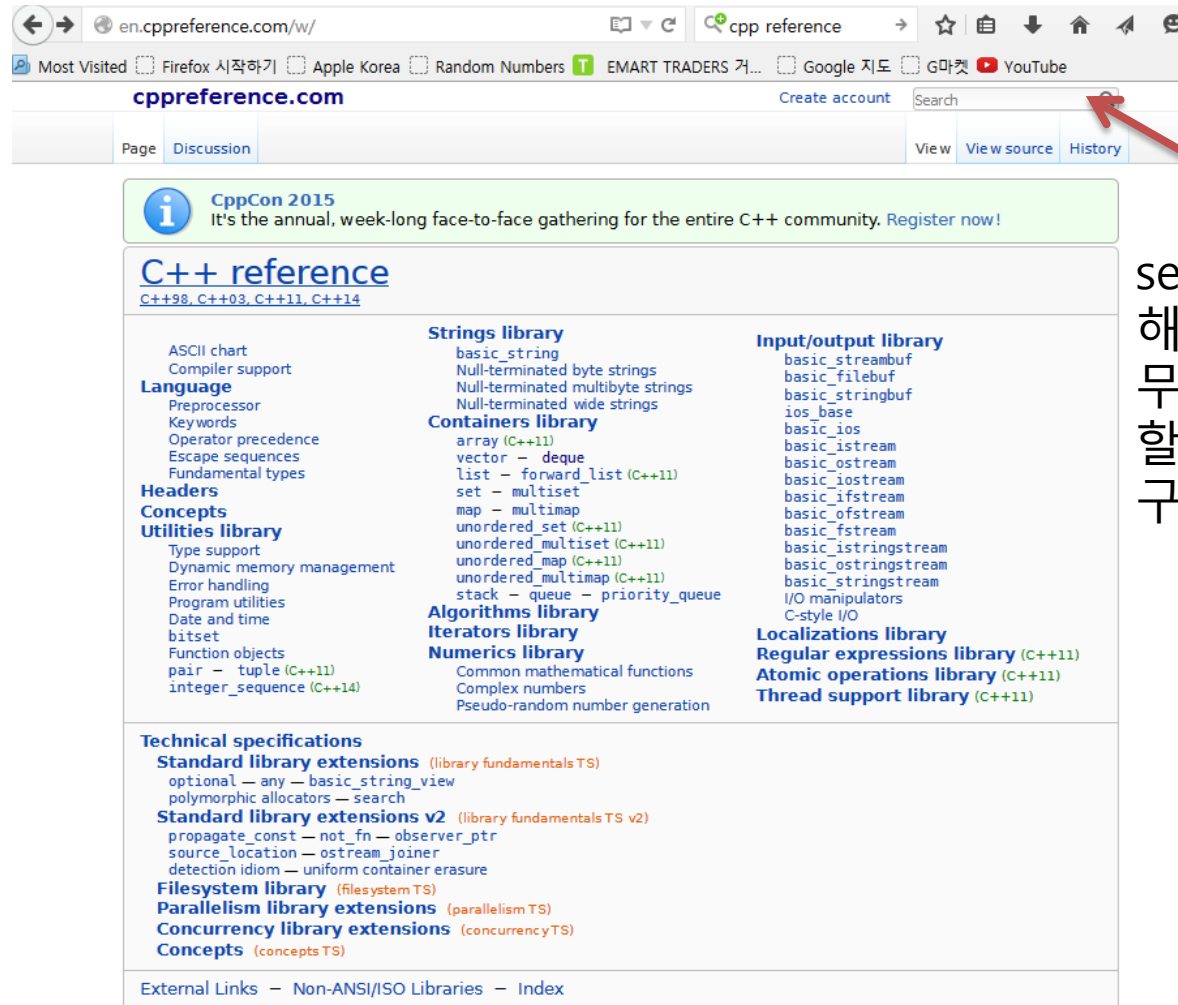
함수의 종류

- 함수가 선언된 위치에 따라 종류가 정해짐
 - 전역함수
 - main함수 밖 namespace 안에서 설정된 함수
 - `std::abs` 등
 - 클래스 멤버함수("메소드"라고 부름)
 - 클래스 안에 선언 및 정의된 함수
 - 클래스의 정보를 이용한 함수를 만들 수 있음
 - `std::cout.flush()`

CPP Reference 보는 법

- C++의 라이브러리의 함수들을 영어 단어 처럼 모두 외우는 건 의미가 없음
 - 기본함수들의 이름은 배우는 데로 외워보자
 - 이름은 외우더라도 사용법까지 전부 외우기는 어려움
 - 따라서, 사용법을 잘 찾아볼 수 있는 방법이 중요
- 필요한 함수들을 빠른 시간에 찾을 수 있도록 en.cppreference.com 페이지를 숙달해보자.

C++ Reference 보는 법



The screenshot shows the C++ Reference website in a web browser. The address bar shows 'en.cppreference.com/w/'. The browser's search bar contains 'cpp reference'. A red arrow points to the search bar. The website header includes 'cppreference.com', 'Create account', and a search bar. The main content area is titled 'C++ reference' and lists various C++ topics. A green banner at the top promotes 'CppCon 2015'. The main content is organized into columns: Language, Headers, Concepts, Utilities library, Strings library, Containers library, Algorithms library, Iterators library, Numerics library, Input/output library, Localizations library, Regular expressions library, Atomic operations library, and Thread support library. Technical specifications are listed at the bottom, including Standard library extensions, Standard library extensions v2, Filesystem library, Parallelism library extensions, Concurrency library extensions, and Concepts. External links for Non-ANSI/ISO Libraries and Index are provided at the bottom.

CppCon 2015
It's the annual, week-long face-to-face gathering for the entire C++ community. [Register now!](#)

C++ reference

[C++98](#), [C++03](#), [C++11](#), [C++14](#)

- ASCII chart
- Compiler support
- Language**
 - Preprocessor
 - Keywords
 - Operator precedence
 - Escape sequences
 - Fundamental types
- Headers**
- Concepts**
- Utilities library**
 - Type support
 - Dynamic memory management
 - Error handling
 - Program utilities
 - Date and time
 - bitset
 - Function objects
 - [pair - tuple \(C++11\)](#)
 - [integer_sequence \(C++14\)](#)
- Strings library**
 - [basic_string](#)
 - Null-terminated byte strings
 - Null-terminated multibyte strings
 - Null-terminated wide strings
- Containers library**
 - [array \(C++11\)](#)
 - [vector - deque](#)
 - [list - forward_list \(C++11\)](#)
 - [set - multiset](#)
 - [map - multimap](#)
 - [unordered_set \(C++11\)](#)
 - [unordered_multiset \(C++11\)](#)
 - [unordered_map \(C++11\)](#)
 - [unordered_multimap \(C++11\)](#)
 - [stack - queue - priority_queue](#)
- Algorithms library**
- Iterators library**
- Numerics library**
 - Common mathematical functions
 - Complex numbers
 - Pseudo-random number generation
- Input/output library**
 - [basic_streambuf](#)
 - [basic_filebuf](#)
 - [basic_stringbuf](#)
 - [ios_base](#)
 - [basic_ios](#)
 - [basic_istream](#)
 - [basic_iostream](#)
 - [basic_ifstream](#)
 - [basic_ofstream](#)
 - [basic_ostream](#)
 - [basic_fstream](#)
 - [basic_istreamstream](#)
 - [basic_ostreamstream](#)
 - [basic_stringstream](#)
 - I/O manipulators
 - C-style I/O
- Localizations library**
- Regular expressions library (C++11)**
- Atomic operations library (C++11)**
- Thread support library (C++11)**

Technical specifications

- Standard library extensions** ([library fundamentals TS](#))
 - [optional](#) - [any](#) - [basic_string_view](#)
 - [polymorphic allocators](#) - [search](#)
- Standard library extensions v2** ([library fundamentals TS v2](#))
 - [propagate_const](#) - [not_fn](#) - [observer_ptr](#)
 - [source_location](#) - [ostream_joiner](#)
 - [detection idiom](#) - [uniform container erasure](#)
- Filesystem library** ([filesystem TS](#))
- Parallelism library extensions** ([parallelism TS](#))
- Concurrency library extensions** ([concurrency TS](#))
- Concepts** ([concepts TS](#))

External Links - [Non-ANSI/ISO Libraries](#) - [Index](#)

search에
해당함수를 입력
무슨 함수를 써야
할지 모르면
구글 검색

C++ Reference 보는 법

cppreference.com [Create account](#)

Special page

Search results

C++	C
std::abs(int)	abs
std::abs(float)	
std::abs(std::complex)	
As-if rule	
std::fabs	
std::labs	
std::forward_as_tuple	
std::cauchy_distribution::a	
std::cauchy_distribution::b	
std::weibull_distribution::a	
std::weibull_distribution::b	
std::lognormal_distribution::s	
std::uniform_int_distribution::a	
std::uniform_int_distribution::b	
std::uniform_real_distribution::a	
std::uniform_real_distribution::b	
std::extreme_value_distribution::a	
std::extreme_value_distribution::b	
std::chrono::treat_as_floating_point	
std::experimental::treat_as_floating_point_v	

예제와 비슷한
abs(float)를 선택

C++ Reference 보는 법

`std::abs(float)`, `std::fabs`

Defined in header <code><cmath></code>		
<code>float</code>	<code>abs(float arg);</code>	(1)
<code>double</code>	<code>abs(double arg);</code>	(2)
<code>long double</code>	<code>abs(long double arg);</code>	(3)
<code>float</code>	<code>fabs(float arg);</code>	(4)
<code>double</code>	<code>fabs(double arg);</code>	(5)
<code>long double</code>	<code>fabs(long double arg);</code>	(6)
<code>double</code>	<code>fabs(Integral arg);</code>	(7) (since C++11)

abs(float)의 종류
float, double,
long double 형을 지원함을 확인
가능

Parameters

`arg` - Value of a floating-point or `Integral` type

파라미터 혹은 인자(argument) 정보
부동소수점 혹은 정수형을 입력

Return value

If successful, returns the absolute value of `arg` (`|arg|`). The value returned is exact and does not depend on any rounding modes.

Error handling

This function is not subject to any of the error conditions specified in `math_errhandling`.
If the implementation supports IEEE floating-point arithmetic (IEC 60559),

- If the argument is ± 0 , $+0$ is returned
- If the argument is $\pm \infty$, $+\infty$ is returned
- If the argument is NaN, NaN is returned

반환값이 무엇인지 표시

C++ Reference 보는 법

`std::abs(float)`, `std::fabs`

Defined in header <code><cmath></code>		
<code>float</code>	<code>abs(float arg);</code>	(1)
<code>double</code>	<code>abs(double arg);</code>	(2)
<code>long double</code>	<code>abs(long double arg);</code>	(3)
<code>float</code>	<code>fabs(float arg);</code>	(4)
<code>double</code>	<code>fabs(double arg);</code>	(5)
<code>long double</code>	<code>fabs(long double arg);</code>	(6)
<code>double</code>	<code>fabs(Integral arg);</code>	(7) (since C++11)

abs(float)의 종류
float, double,
long double 형을 지원함을 확인
가능

1-6) Computes the absolute value of a floating point value `arg`.

7) A set of overloads or a function template for all combinations of arguments of `arithmetic type` not covered by 4-6). If any argument has `integral type`, it is cast to `double`. If any other argument is `long double`, then the return type is `long double`, otherwise it is `double`.

Parameters

`arg` - Value of a floating-point or `Integral type`

Return value

If successful, returns the absolute value of `arg` (`|arg|`). The value returned is exact and does not depend on any rounding modes.

Error handling

This function is not subject to any of the error conditions specified in `math_errhandling`.

If the implementation supports IEEE floating-point arithmetic (IEC 60559),

- If the argument is ± 0 , $+0$ is returned
- If the argument is $\pm \infty$, $+\infty$ is returned
- If the argument is NaN, NaN is returned

파라미터 혹은 인자(argument) 정보
부동소수점 혹은 정수형을 입력

반환값이 무엇인지 표시

C++ Reference 보는 법

- 예제 또한 제공하니 참고!
 - 특히, 필요한 헤더파일이 무엇인지 확인하는 것이 중요
 - 예)
`#include <cmath>`

Example

Run this code

```
#include <iostream>
#include <cmath>

int main()
{
    std::cout << "abs(+3.0) = " << std::abs(+3.0) << '\n'
               << "abs(-3.0) = " << std::abs(-3.0) << '\n';
    // special values
    std::cout << "abs(-0.0) = " << std::abs(-0.0) << '\n'
               << "abs(-Inf) = " << std::abs(-INFINITY) << '\n';
}
```

Possible output:

```
abs(+3.0) = 3
abs(-3.0) = 3
abs(-0.0) = 0
abs(-Inf) = inf
```

Quiz

- UOS 구구단 프로그램을 만들어 봅시다.
 - 명세(Spec)
 - File Name : UOSgugu.cpp + (UOSgugu.h)
 - Restrict
 - 반드시 UOS namespace의 gugu라는 함수로 작성할 것
 - Parameter
 - » x : x 단 구구단 출력
 - Return value
 - » 임의대로 판단
 - Feature
 - 1단은 "1단은 너무 쉽지."라고 출력
 - 2~9단은 일반적인 구구단으로 출력
 - $x \geq 10$ 은 마지막 x 라인까지 출력
 - » 예) 14단의 경우. $14 \times 14 = 196$ 까지 출력
 - 제출 : ry840901@gmail.com 으로 보낼 것
 - 제목 : 본인 학번
 - 내용 : 본인 이름 및 기타 내용
 - 첨부 : UOSgugu.cpp or UOSgugu.cpp & UOSgugu.h