190401

**\*Namespace**

함수를 이름과 파라미터를 똑같이 하면서 두 함수를 구별해서 사용하고 싶다.

<Def>

Namespace Myspace1{

int Myfunc1(int a, int b){

return a+b;

}

}

Namespace Myspace2{

int Myfunc1(int a, int b){

return a\*b;

}

}

<How to use>

Myspace1::Myfunc1(3, 4);

Myspace2::Myfunc1(5, 6);

매번 Myspace1:: 또는 Myspace2::를 앞에 붙이기 귀찮다.

* 사전에 using namespace Myspace1; 을 써주면 해결된다.

중첩도 가능

<Def>

namespace Myspace{

namespace Innerspace{

int MyFunc(void){

return 0;

}

}

}

<How to use>

Myspace::Innerspace::MyFunc();

그리고

Using namespace Myspace::Innerspace;

MyFunc();

**\*Preprocessor** //빌드 전에 처리한다는 것이 핵심

#define //문서편집기능 역할. 다음에 오는 문자를 만나면 다음 형태로 바꿔준다. 이것을 **매크로** 라고 부른다.

정수, 실수, 문자, 문자열, 심지어 함수까지도 올 수 있다.

Ex)

#define MAX(a, b) (( a > b ) ? a : b)

TIP. 간단한 알고리즘(최댓값 출력, 스왑, 같은 지 여부, sort …)은 algorithm에 include 되어 있으니 참고하여 사용할 것

#define LIKE\_APPLE //매크로 정의

#ifdef LIKE\_APPLE //다음 매크로가 정의되어 있으면

…

#endif

#ifndef LIKE\_APPLE //정의되어 있지 않으면

…

#endif

매크로 전처리기의 효력 범위는 해당 cpp 파일에만 영향이 미친다.

Q. Unsigned와 signed 의 구분을 두는 이유

* 두 타입의 저장방식이 다르다. 그리고 Unsigned가 계산속도가 더 빠르다. 상황에 따라 구분해서 쓸 필요가 있음.

**\*Fundamental Data Types(= Built-in Data type)**

float 초기화시...

float fVal = 3.141592**f**;

Q. f를 쓰는 이유?

* 일반 소수를 입력하면 컴파일러가 double로 인식한다. Float 초기화시 f를 명시하지 않으면 정밀도가 높은 수를 float에 저장하려고 하기 때문에 절삭 될 수 있는데 이것을 Truncation이라고 한다.

Auto 자료형 = 초기화 시 자료형이 결정된다.

Q. 변수 선언을 언제 하면 좋을까?

* 요즘 트렌드는 변수를 사용하기 직전에 선언과 초기화를 하는 것이 추세고 이점도 있다.

(1) 디버깅 하기가 좋다.

(2) 기능을 하는 부분과 변수는 묶여 있어야 나중에 코드가 길어 졌을 때 Code Refactoring(함수의 갯수와 반복이 많아 질 경우 클래스나 네임스페이스로 묶어 재구성(?)할 필요가 생겨지는데 그때 하는 작업을 말함.)하기가 편해진다.

**\*Integers**

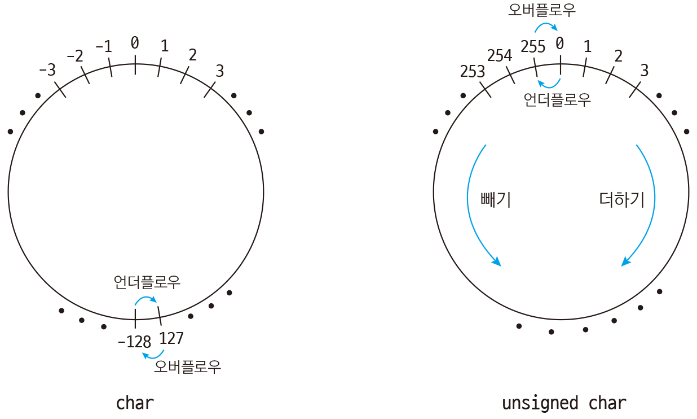
#include<limits>

//std::numeric\_limits<Datatype>::max()

//std::numeric\_limits<Datatype>::min()

//std::numeric\_limits<Datatype>::lowest()

…



**unsigned int i = -1;** //42억

**\*Fixth-Width Integers**

C++ 11부터 고정 너비 정수 기능이 추가 됐다.

#include <cstdint> //iostream에 포함되어 있으므로 생략 가능

int8\_t i(65); //char 1byte

int16\_t j(5); //short 2byte

int32\_t k(1); //int 4byte

int64\_t l(3); //long long 8byte

int\_fast8\_t fi(5); //signed char //8비트 자료형 중에 최상의 산술 성능을 가진 자료형.

int\_least64\_t fl(5); //long long //적어도 64비트를 갖는 데이터 타입

**\*Void, 보이드형, 보이드 무치형** 이라고도 함