

코드 분석 2

Hyunchan, Park

<http://oslab.jbnu.ac.kr>

Division of Computer Science and Engineering

Jeonbuk National University



강의 일정 (updated)

주차	월 (100분)	수 (50분)
1 (9/1)	Introduction	OSS 역사 1
2	OSS 역사 2 / OSS 개요 1	추석
3	OSS 개요 2 / OSS 라이선스 1	OSS 라이선스 2
4	OSS 활용 방법 / 버전 관리 도구 Git 1	버전 관리 도구 Git 2
5 (9/30)	버전 관리 도구 Git 3	버전 관리 도구 Git 4
6	버전 관리 도구 Git 5	한글날
7	특강 1 (10/14)	코드 분석 1
8 (10/21)	중간고사	개인 프로젝트 1 : 프로젝트 선정 및 계획 마감 (휴강)
9	특강 2 (10/28)	코드 분석 2
10 (11/4)	클라우드를 이용한 개발 환경 구축 1	클라우드를 이용한 개발 환경 구축 2
11	GitHub를 이용한 협업 1	GitHub를 이용한 협업 2
12	개인 프로젝트 2: 대상 프로젝트 코드 분석 결과 발표 및 기능 구현 방향, 커뮤니티 참여 방안 발표 및 피드백	
13 (11/25)	코드 리뷰	CI/CD 구축
14	개별 프로젝트 진행	
15 (12/9)	개인 프로젝트 3: 최종 프로젝트 발표, 기능 구현 소개 및 시연, OSS 커뮤니티 활동 소개	

- 코드 분석 기능이 강한 에디터
 - 주로 C, C++ 계열의 언어용으로 많이 사용되지만, 그 외 여러 언어들도 지원함
 - 2017년, 거의 10년 만에 4.0 버전이 발표됨 (웹에는 v3.5에 대한 자료가 많음)
- vs와 비교했을 때, 몇 가지 장점들
 - 빠른 코드 분석에 좀더 집중된 기능들 제공
 - 심볼에 대한 다양한 추적 기능을 빠르게 수행
 - 심볼 하이라이트 기능
 - 단점은? 실행 환경과 연동해서 쓰기에는 불편함
- <https://www.sourceinsight.com/download/>
 - 상용 SW: \$239 / 1 copy, \$4180/20 copies

Key Benefits

Understand Code

Learn an existing code base quickly, and get up to speed on new projects. Evaluate the costs of potential changes by seeing where functions and objects are used. See class inheritance and function call trees.

Code Analysis

Source Insight has built-in dynamic analysis for C/C++, C#, Java, Objective-C, and more.

[Read More](#)

Quickly Navigate

Source Insight parses your whole project and let's you navigate and edit code like a breeze, while showing you information automatically. Jump easily to callers of functions or references to variables.

Powerful Editing

Use powerful editing features, including code snippets, symbolic auto-completion, and smart-rename.

Discover

See where functions and variables are used. Source Insight automatically shows references to functions, variables, classes, and more - almost instantly. Search across your project using advanced search features.

[Read More](#)

Syntax Formatting

See live references to variables and other declarations with Syntax Formatting. Identifiers are formatted based on their declaration, scope, and usage.

30-day free trial

Source Insight 4.0 License Management



Welcome to Source Insight.

Please select an option to enable your license.

- ☐ Enter your purchased serial number to activate a license on this machine.
Choose this option if you have a serial number for a purchased license. This will permanently activate your license.

You can [purchase an new serial number online](#).

- ☒ **Begin a 30-day Free Trial of Source Insight.**
A Trial license will be activated immediately and will run for 30 days.

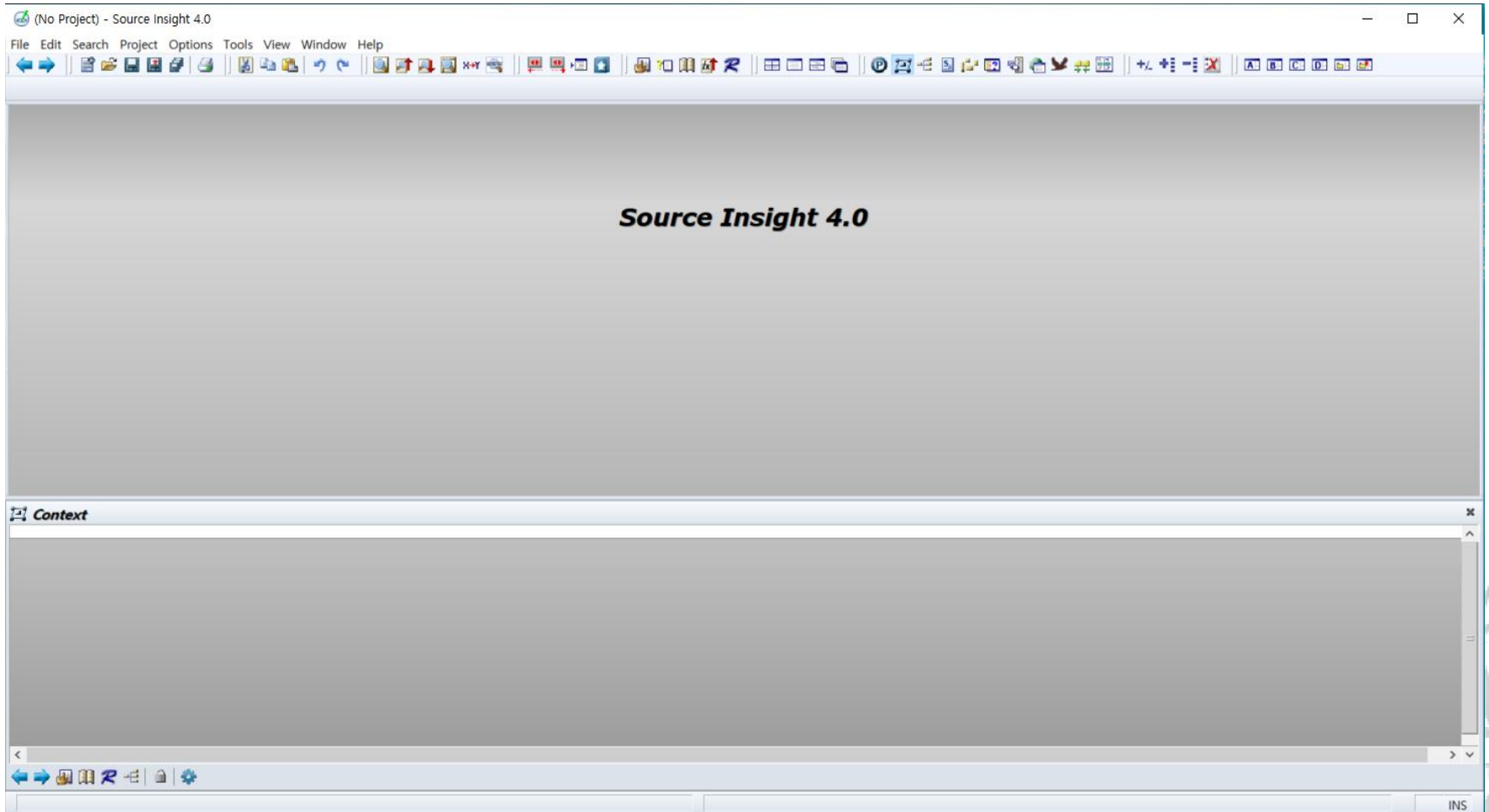
- ☐ Import a new license file.
Choose this option if a license file was sent to you.

Exit Program

Next >

Help

첫 화면

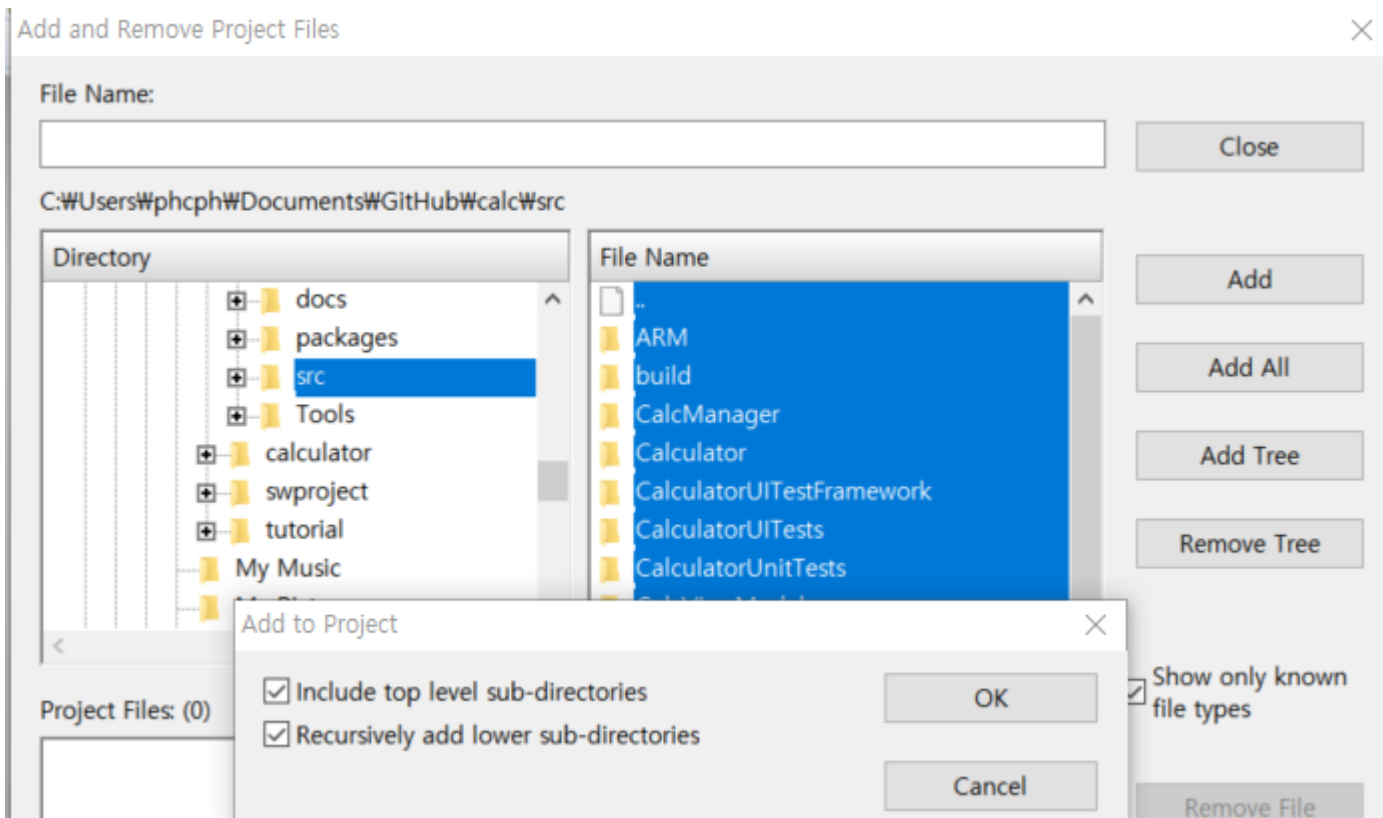


프로젝트 생성

- Project → New project
 - 프로젝트 이름 입력, 관련 데이터 저장 폴더 선택 (소스 폴더 아님)
- New Project Settings
 - Project source directory
 - 소스 코드가 존재하는 경로 지정
 - 예) 계산기 소스 코드: GitHub\Calculator\src
 - 나머지는 기본 옵션
- Add and Remove Project Files
 - 분석에 사용할 소스 파일을 지정

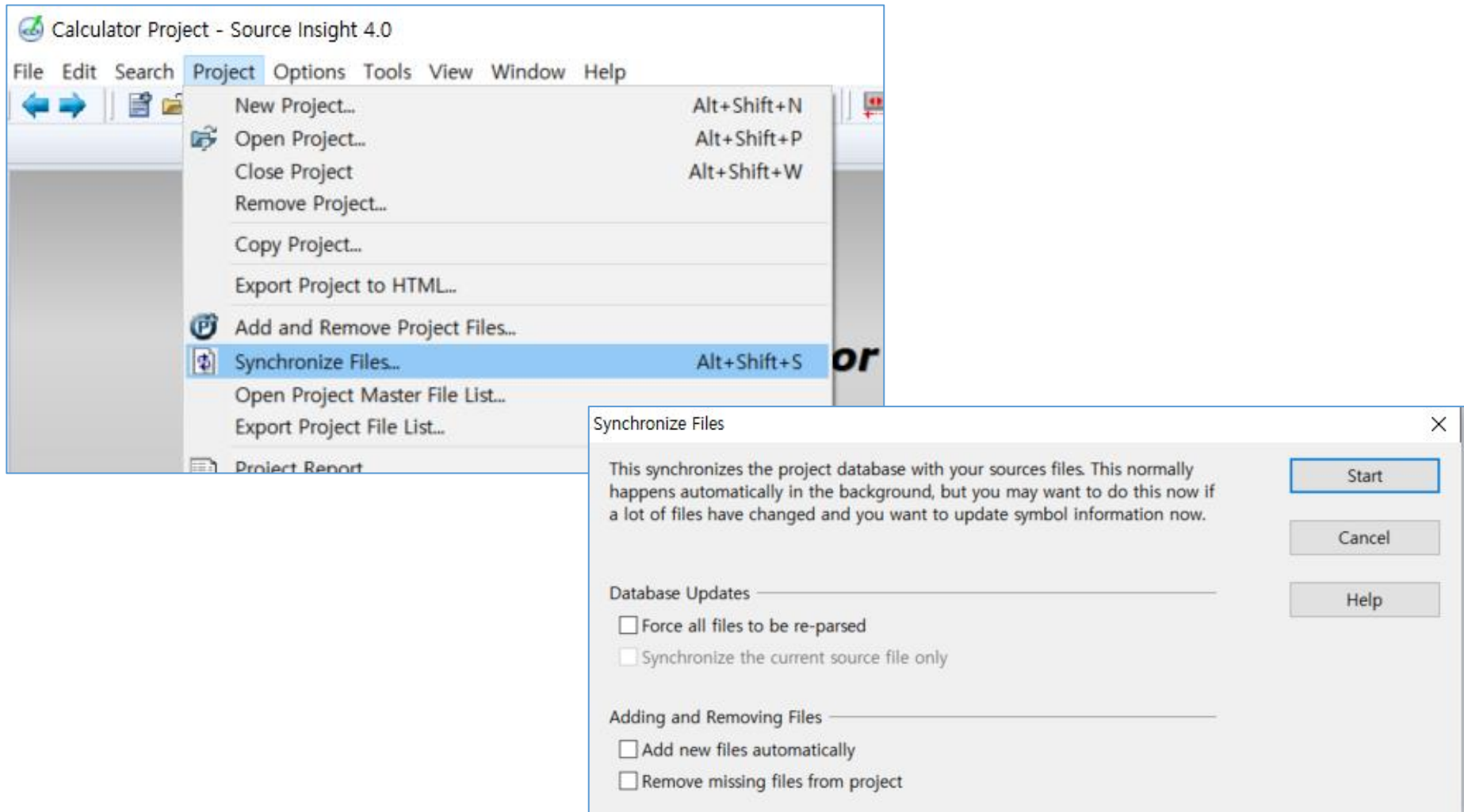
Add and Remove Project Files

- 각 파일 별로 고를 수도 있고, 소스 코드 전체를 한 번에 추가할 수 있음
 - Add All 선택 후, 하위의 모든 폴더 내용을 포함 후, Close

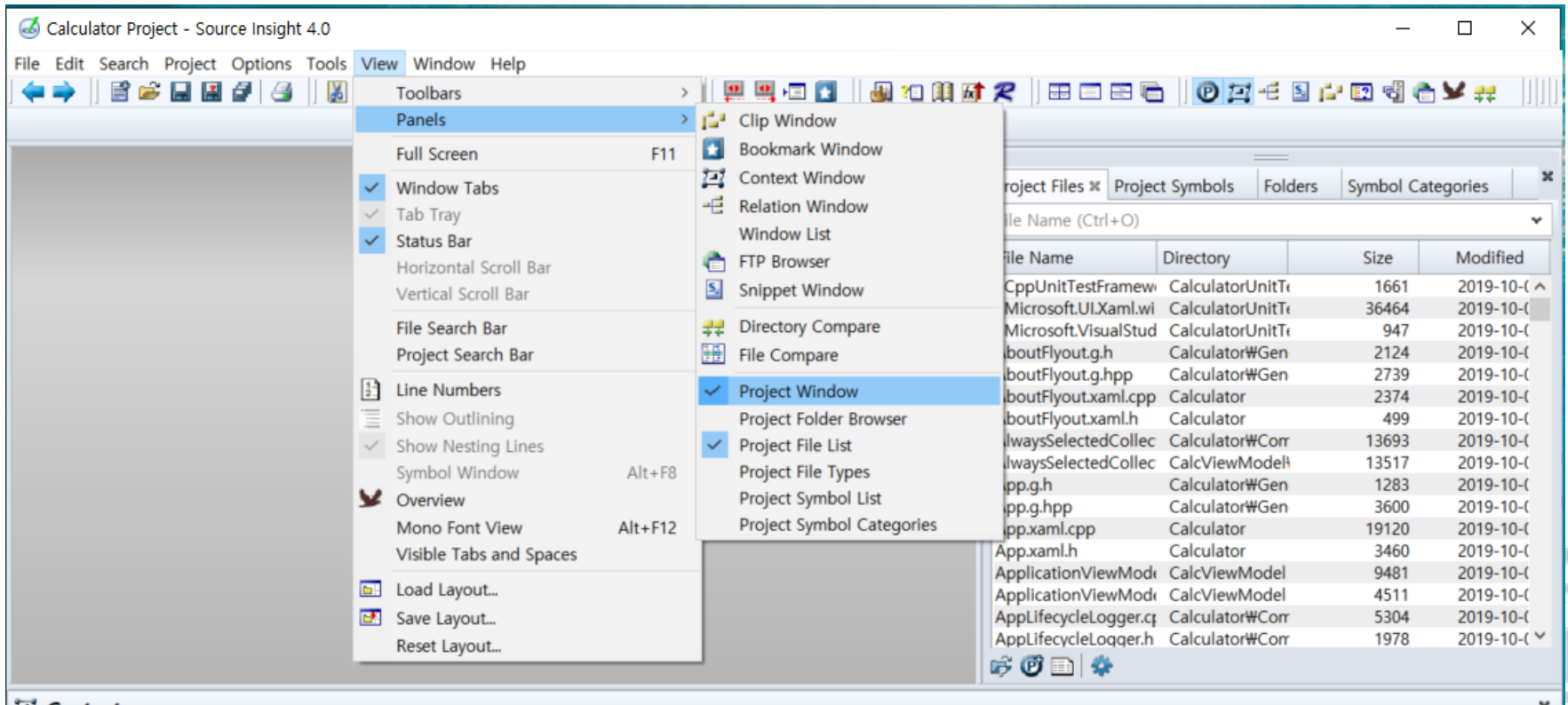


Synchronize files

- 변수, 함수 등의 관계를 분석하고, 차후 빠르게 접근할 수 있도록 함

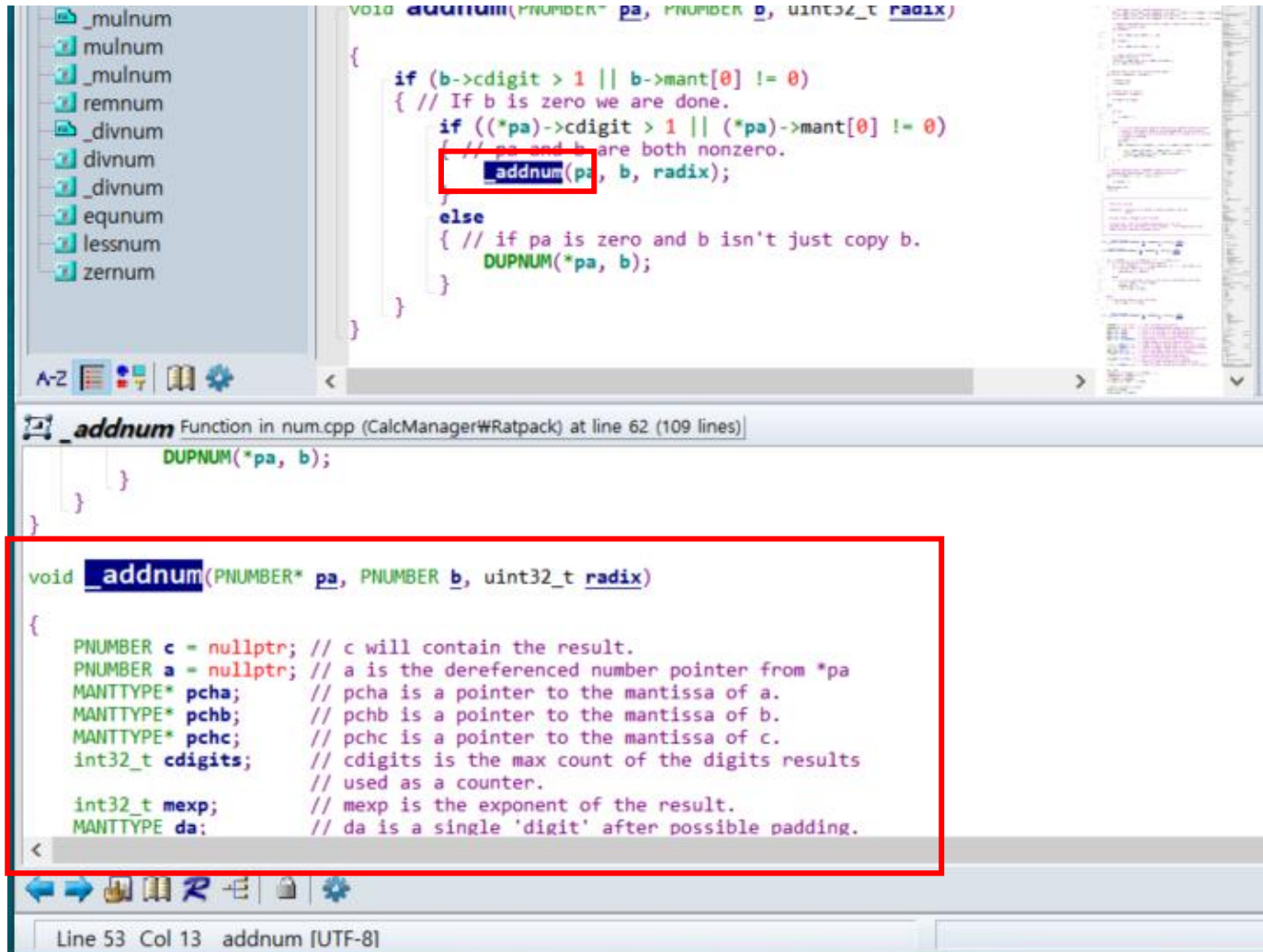


프로젝트 윈도우 패널 Enable



기능: Context Window

- 선택한 심볼의 정의를 아래 분리된 화면에 표시해주어,
정의를 빠르게 참조할 수 있음 (심볼 위에 커서를 두고 잠시 기다려보자)

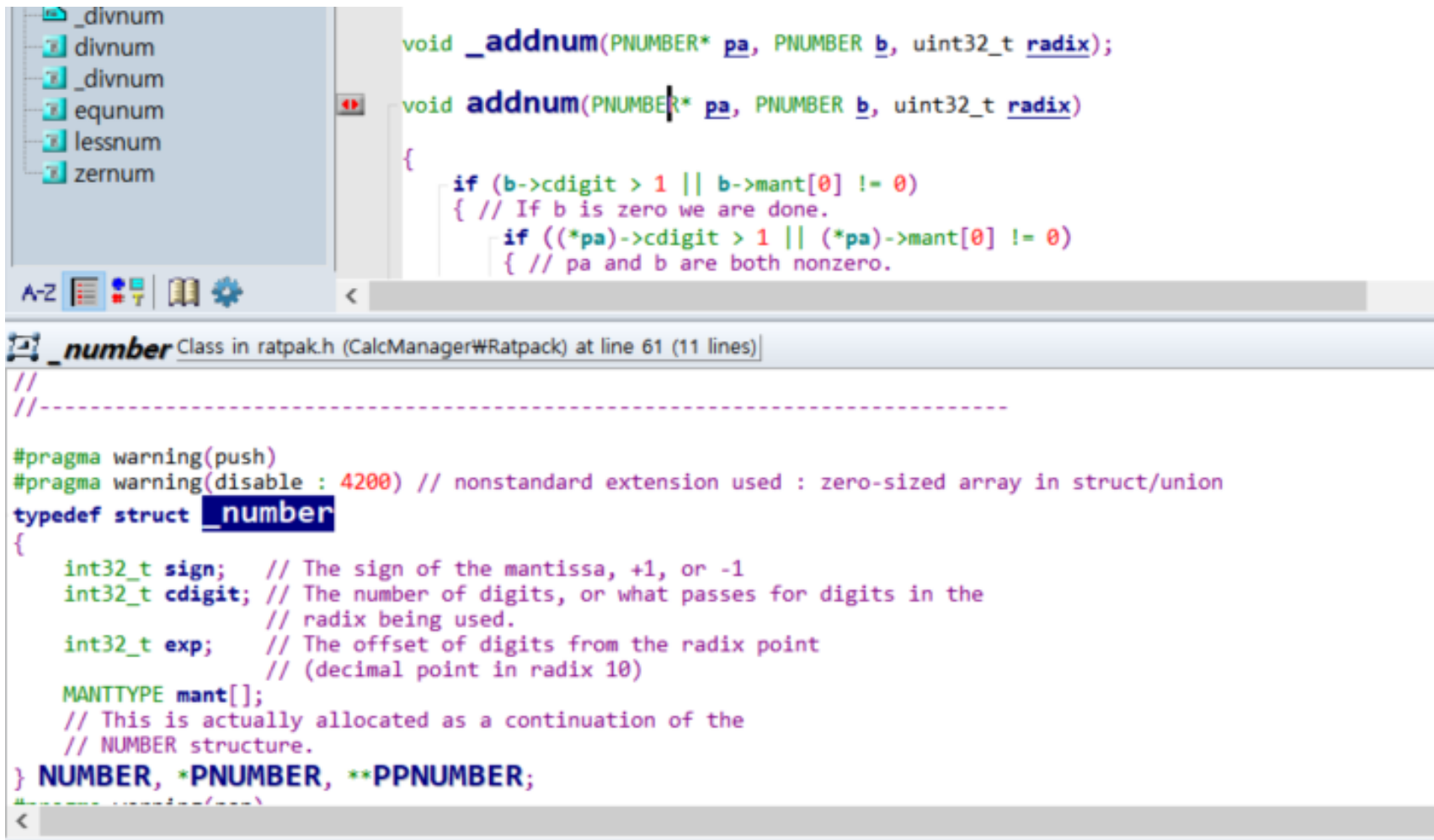


```
void addnum(PNUMBER* pa, PNUMBER b, uint32_t radix)
{
    if (b->cdigit > 1 || b->mant[0] != 0)
    { // If b is zero we are done.
        if ((*pa)->cdigit > 1 || (*pa)->mant[0] != 0)
        { // pa and b are both nonzero.
            addnum(pa, b, radix);
        }
        else
        { // if pa is zero and b isn't just copy b.
            DUPNUM(*pa, b);
        }
    }
}

void addnum(PNUMBER* pa, PNUMBER b, uint32_t radix)
{
    PNUMBER c = nullptr; // c will contain the result.
    PNUMBER a = nullptr; // a is the dereferenced number pointer from *pa
    MANTTYPE* pcha; // pcha is a pointer to the mantissa of a.
    MANTTYPE* pchb; // pchb is a pointer to the mantissa of b.
    MANTTYPE* pchc; // pchc is a pointer to the mantissa of c.
    int32_t cdigits; // cdigits is the max count of the digits results
                  // used as a counter.
    int32_t mexp; // mexp is the exponent of the result.
    MANTTYPE da; // da is a single 'digit' after possible padding.
```

기능: Context Window

- 구조체, 변수 정의 등을 한 화면에서 바로 보여주어 편리함



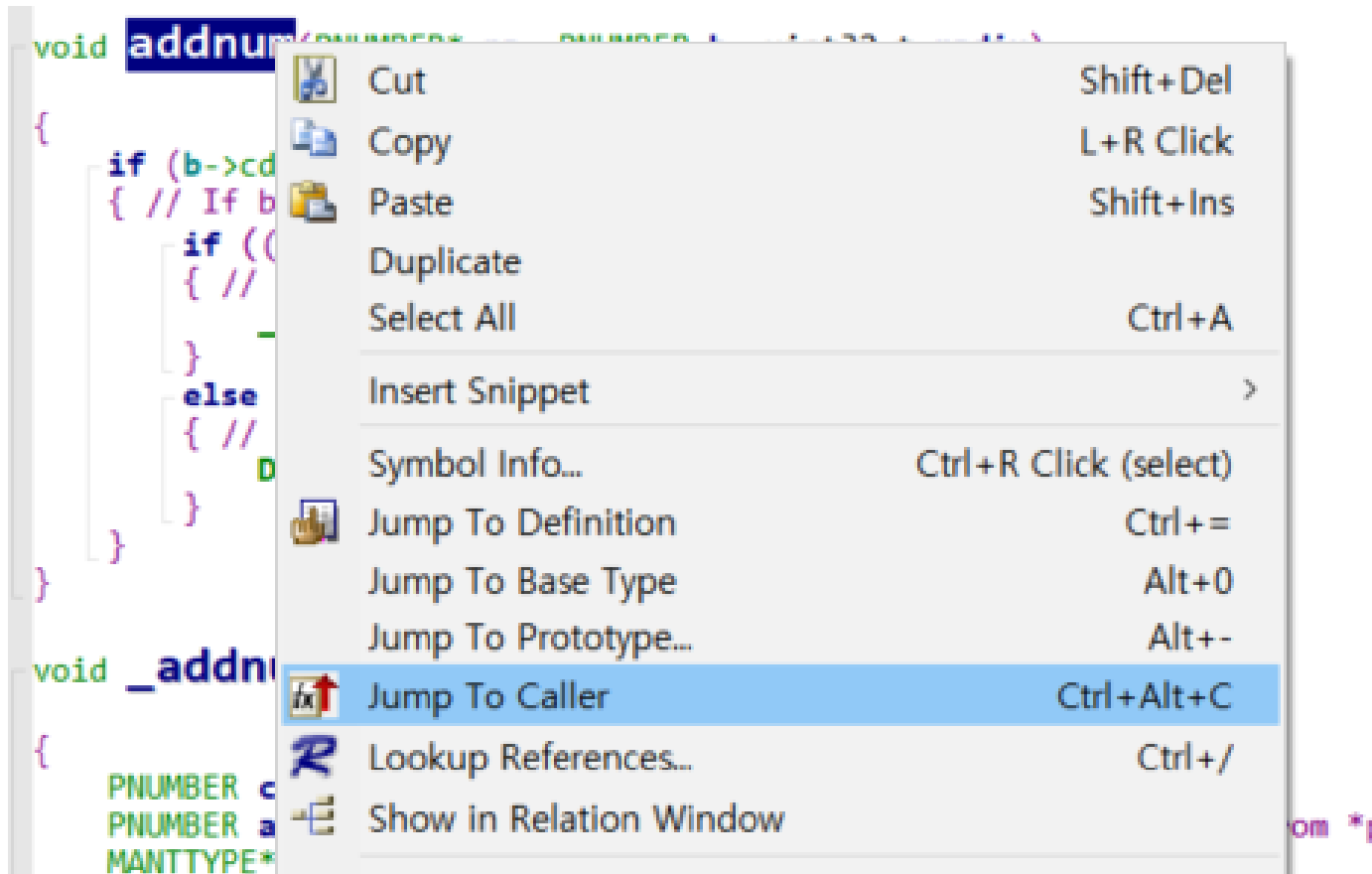
```
void _addnum(PNUMBER* pa, PNUMBER b, uint32_t radix);  
void addnum(PNUMBER* pa, PNUMBER b, uint32_t radix)  
{  
    if (b->cdigit > 1 || b->mant[0] != 0)  
    { // If b is zero we are done.  
        if ((*pa)->cdigit > 1 || (*pa)->mant[0] != 0)  
        { // pa and b are both nonzero.  
            // ...  
        }  
    }  
}
```

_number Class in ratpak.h (CalcManager\Wratpack) at line 61 (11 lines)

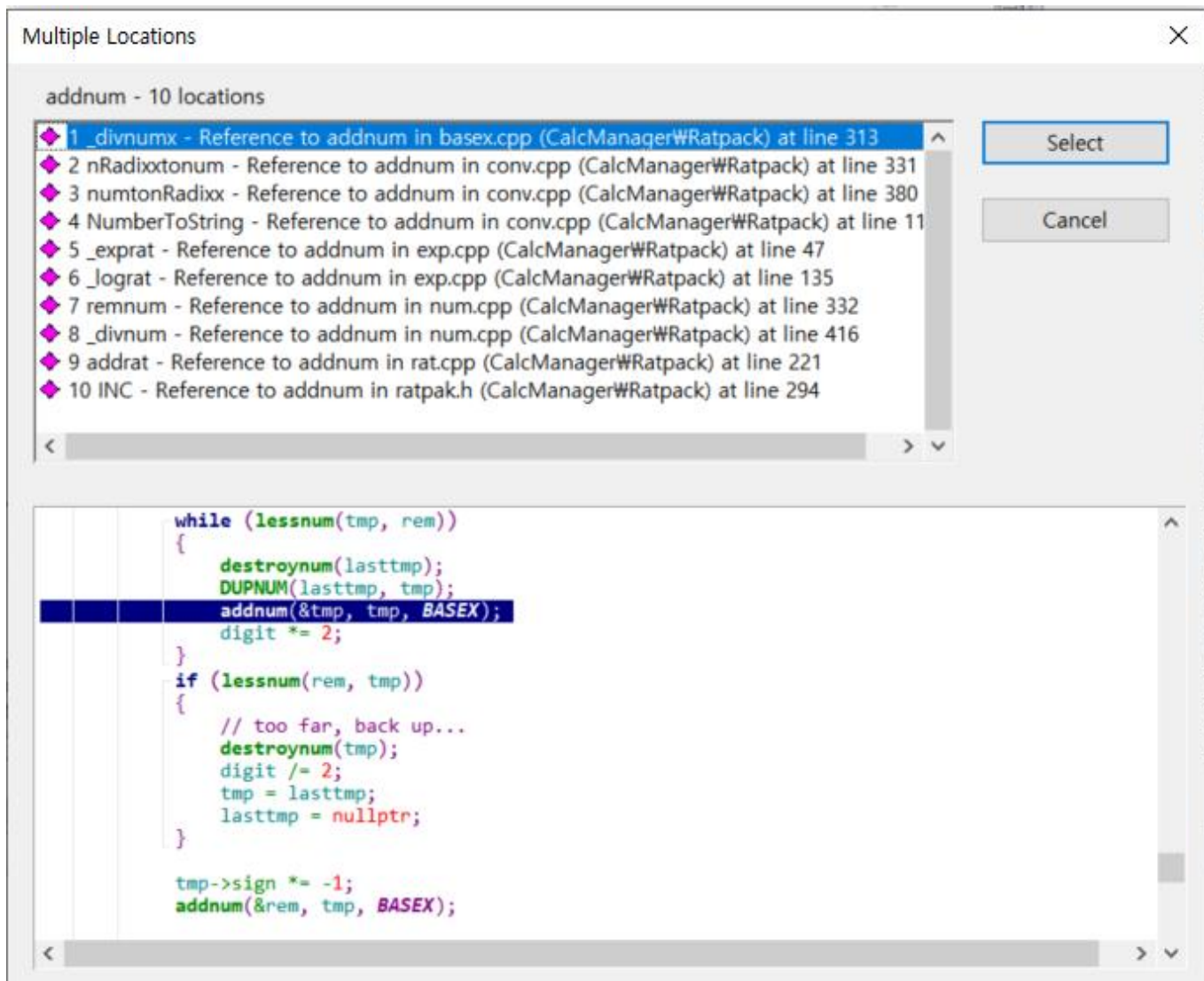
```
//  
//-----  
#pragma warning(push)  
#pragma warning(disable : 4200) // nonstandard extension used : zero-sized array in struct/union  
typedef struct _number  
{  
    int32_t sign; // The sign of the mantissa, +1, or -1  
    int32_t cdigit; // The number of digits, or what passes for digits in the  
                  // radix being used.  
    int32_t exp; // The offset of digits from the radix point  
               // (decimal point in radix 10)  
    MANTTYPE mant[];  
    // This is actually allocated as a continuation of the  
    // NUMBER structure.  
} NUMBER, *PNUMBER, **PPNUMBER;  
//-----  
<
```

기능: Jump to caller

- 선택한 심볼을 호출하는 위치를 바로 모두 검색하여 보여줌



기능: Jump to caller



기능: Jump to caller

- VS 2019 에서도 되는데, 상당히 시간이 걸림
 - 어? 9개 밖에 못 찾았네? 시간도 오래 걸렸으면서...

호출 계층 구조

내 솔루션

addnum(PNUMBER * pa, PNUMBER b, uint32_t radix)

- ▶ 'addnum'에 대한 호출
 - ▶ NumberToString(PNUMBER & pnum, int format, uint32_t radix, int32_t precision)
 - ▶ _divnum(PNUMBER * pa, PNUMBER b, uint32_t radix, int32_t precision)
 - ▶ _divnumx(PNUMBER * pa, PNUMBER b, int32_t precision)
 - ▶ _exprat(PRAT * px, int32_t precision)
 - ▶ _lograt(PRAT * px, int32_t precision)
 - ▶ addrat(PRAT * pa, PRAT b, int32_t precision)
 - ▶ nRadixtonum(PNUMBER a, uint32_t radix, int32_t precision)
 - ▶ numtonRadix(PNUMBER a, uint32_t radix)
 - ▶ remnum(PNUMBER * pa, PNUMBER b, uint32_t radix)
- ▶ 'addnum'에서의 호출

호출 사이트

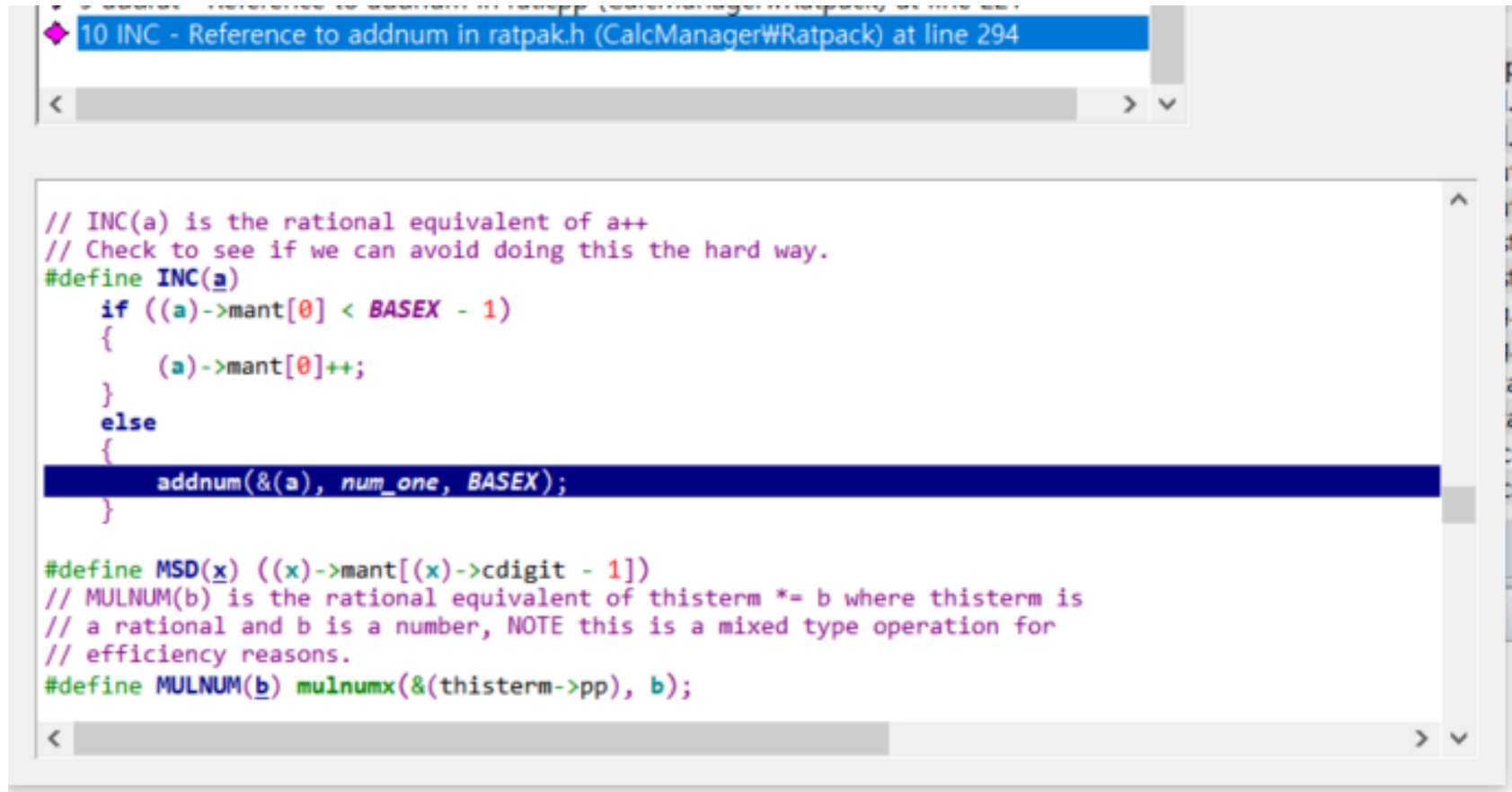
addnum(&pnum, round, radix);

conv.cpp - (1154, 9)

호출 계층 구조 오류 목록 출력

기능: Jump to caller

- VS: .h 확장자의 헤더 파일에 포함된 매크로를 찾아내지 못했음

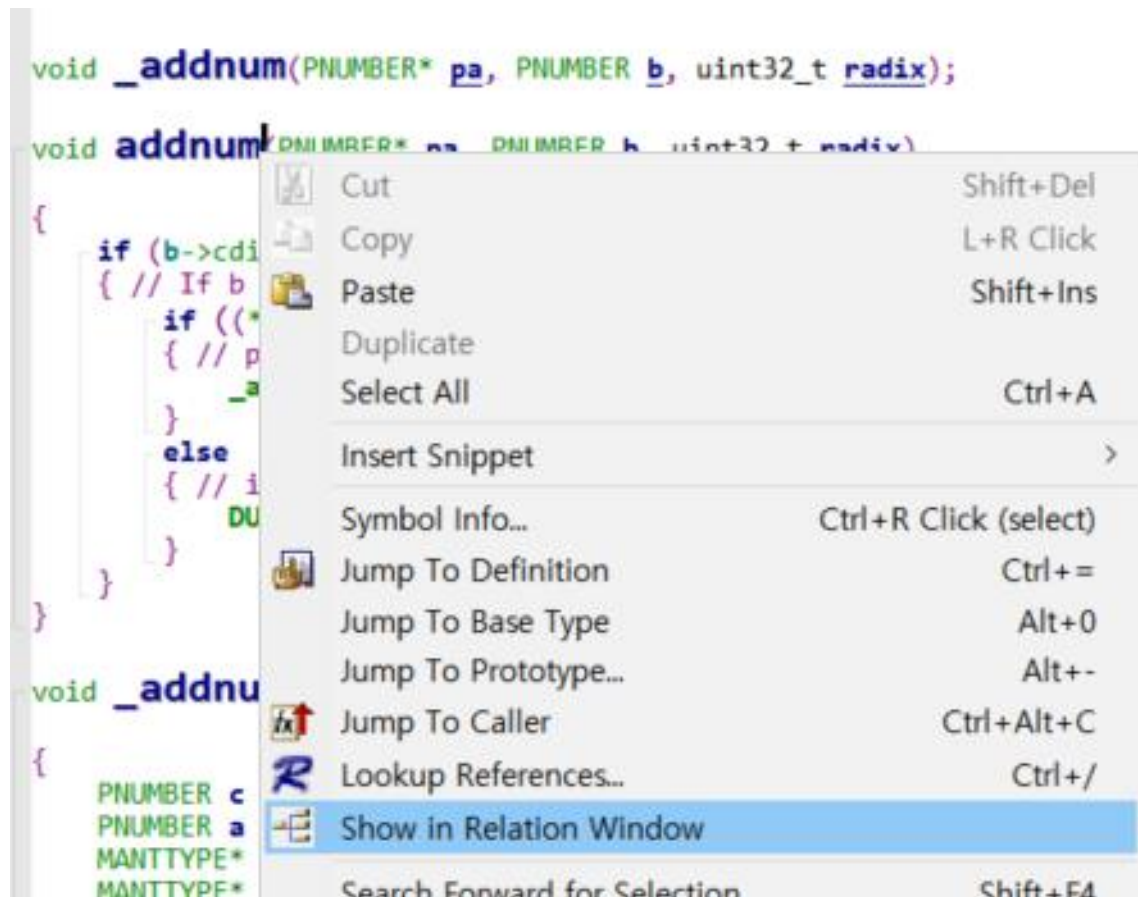


The screenshot shows a Visual Studio error window with a blue header bar containing the text: "10 INC - Reference to addnum in ratpak.h (CalcManagerWRatpack) at line 294". Below the header, the source code is displayed in a monospaced font. The code defines a macro `INC(a)` which checks if the mantissa of a rational number is less than `BASEX - 1`. If so, it increments the mantissa; otherwise, it calls the `addnum` function. Below this, the `MSD` macro is defined to get the most significant digit, and the `MULNUM` macro is defined to multiply a rational number by a constant. The error message indicates that the `addnum` function is referenced but not found, likely because the compiler did not find the definition in the header file.

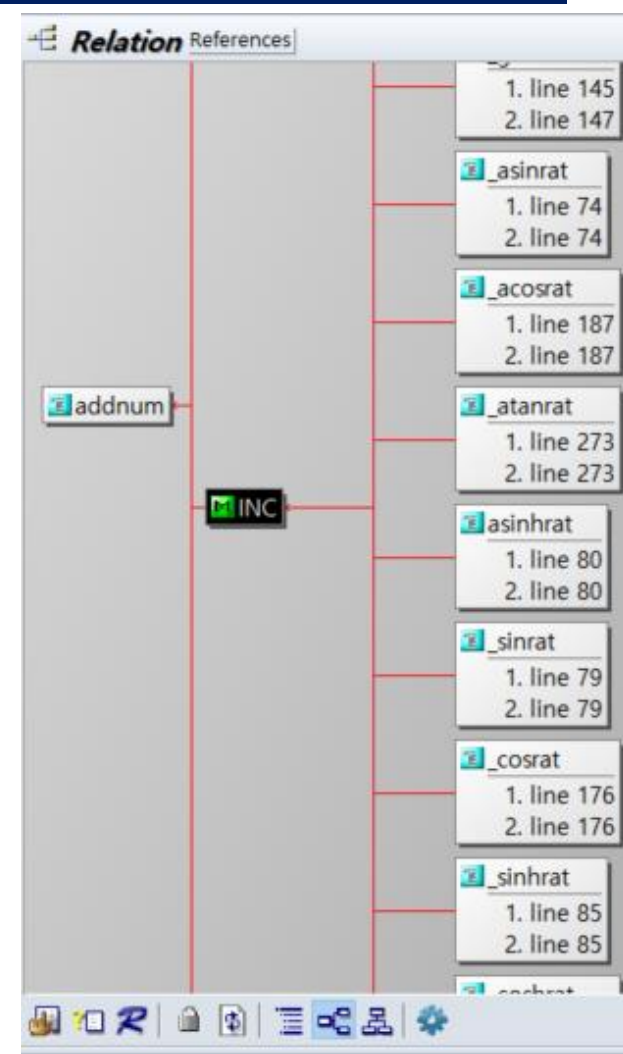
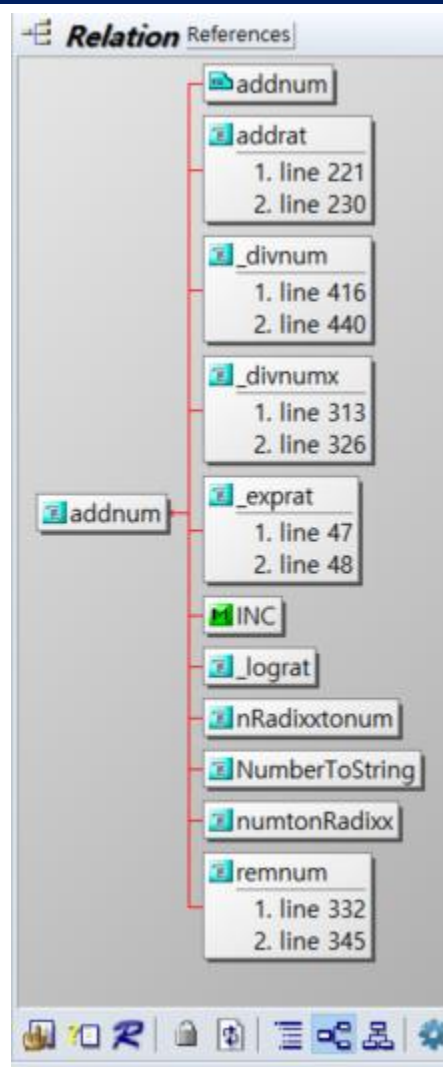
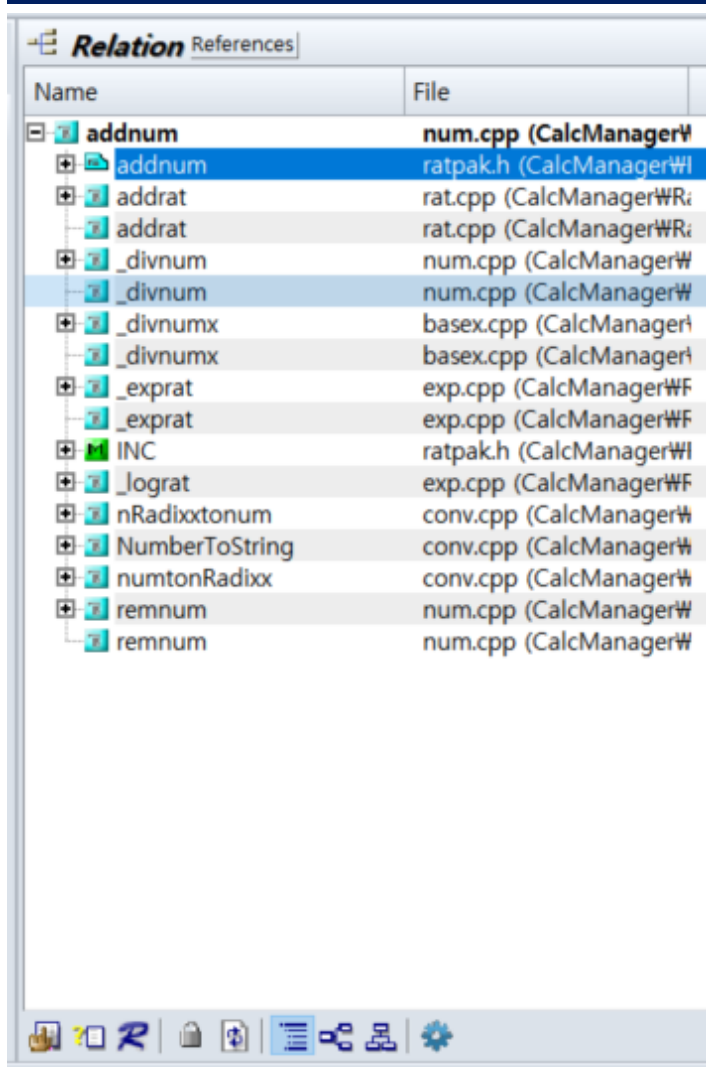
```
// INC(a) is the rational equivalent of a++  
// Check to see if we can avoid doing this the hard way.  
#define INC(a)  
    if ((a)->mant[0] < BASEX - 1)  
    {  
        (a)->mant[0]++;  
    }  
    else  
    {  
        addnum(&(a), num_one, BASEX);  
    }  
  
#define MSD(x) ((x)->mant[(x)->cdigit - 1])  
// MULNUM(b) is the rational equivalent of thisterm *= b where thisterm is  
// a rational and b is a number, NOTE this is a mixed type operation for  
// efficiency reasons.  
#define MULNUM(b) mulnumx(&(thisterm->pp), b);
```


기능: Relation window

- 심볼의 호출 관계를 보여주는 윈도우



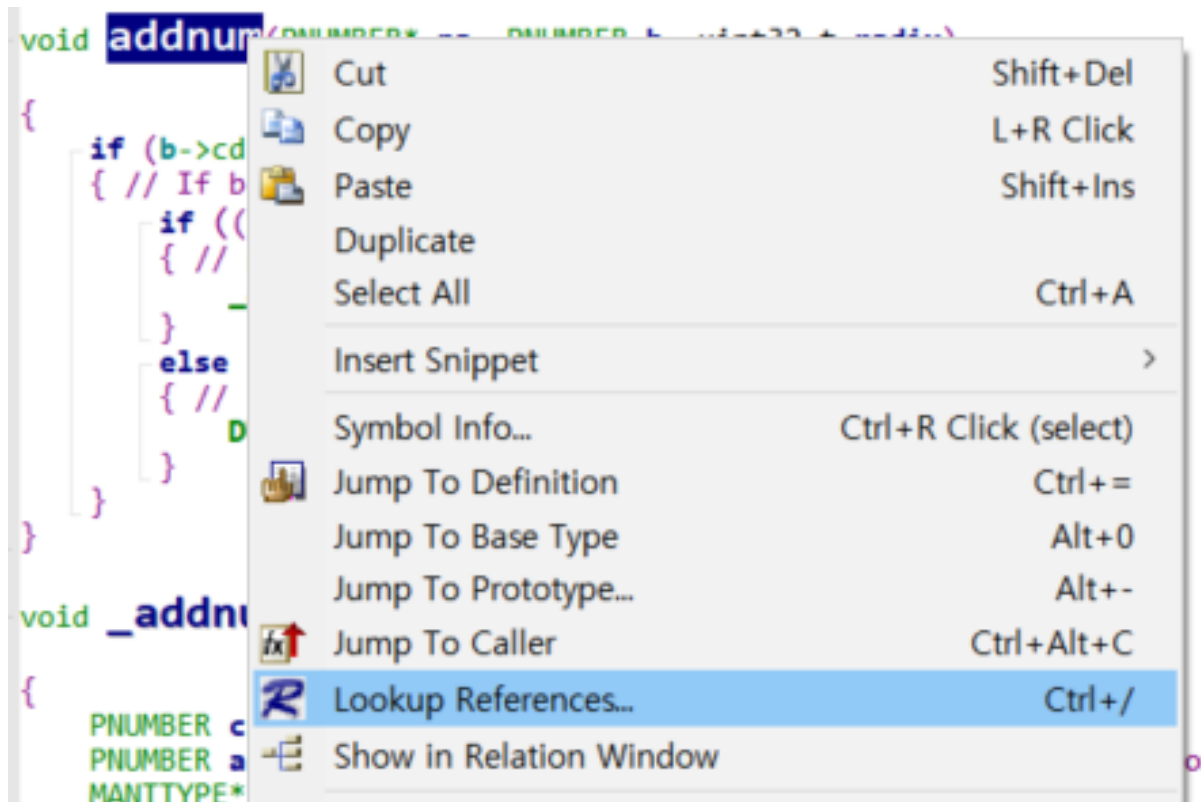
기능: Relation window



* 확장된 Call graph

기능: Lookup References

- 심볼에 대한 모든 참조를 검색함
- 텍스트로 검색이 되어, 주석 등에 포함된 내용도 모두 발견할 수 있음



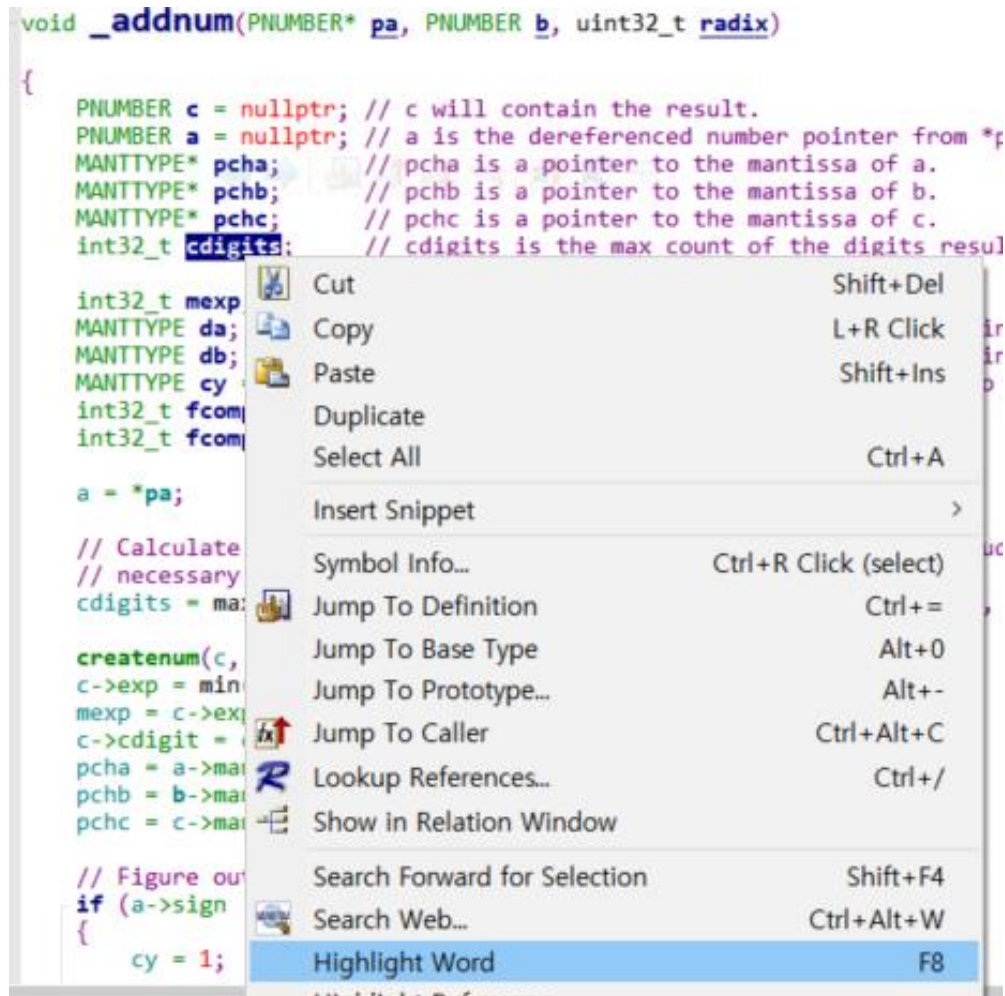
기능: Lookup References

- 결과화면에서 직접 이동하거나, 순차적으로 reference 를 순회할 수 있음
- 기존 caller 들이 발견되고, 각 caller 내에서 호출 위치와 횟수를 바로 확인

```
---- addnum Matches (18 in 6 files) ----
_<divnumx in basex.cpp (CalcManager\Ratpack) :          addnum(&tmp, tmp, BASEX);
_<divnumx in basex.cpp (CalcManager\Ratpack) :          addnum(&rem, tmp, BASEX);
nRadixxtonum in conv.cpp (CalcManager\Ratpack) :          addnum(&sum, sum, radix);
numtonRadixx in conv.cpp (CalcManager\Ratpack) :          addnum(&pnumret, thisdigit, BASEX);
NumberToString in conv.cpp (CalcManager\Ratpack) :          addnum(&pnum, round, radix);
_exprat in exp.cpp (CalcManager\Ratpack) :          addnum(&(pret->pp), num_one, BASEX);
_exprat in exp.cpp (CalcManager\Ratpack) :          addnum(&(pret->pq), num_one, BASEX);
_lograt in exp.cpp (CalcManager\Ratpack) :          addnum(&((*px)->pp), (*px)->pq, BASEX);
num.cpp (CalcManager\Ratpack) line 28 : //      FUNCTION: addnum
num.cpp (CalcManager\Ratpack) line 46 : void addnum(PNUMBER* pa, PNUMBER b, uint32_t radix)
remnum in num.cpp (CalcManager\Ratpack) :          addnum(&tmp, tmp, radix);
remnum in num.cpp (CalcManager\Ratpack) :          addnum(pa, tmp, radix);
_divnum in num.cpp (CalcManager\Ratpack) :          addnum(&newValue, tmp, radix);
_divnum in num.cpp (CalcManager\Ratpack) :          addnum(&rem, multiple, radix);
addrat in rat.cpp (CalcManager\Ratpack) :          addnum(&((*pa)->pp), b->pp, BASEX);
addrat in rat.cpp (CalcManager\Ratpack) :          addnum(&((*pa)->pp), (*pa)->pq, BASEX);
INC in ratpak.h (CalcManager\Ratpack) :          addnum(&(a), num_one, BASEX);
ratpak.h (CalcManager\Ratpack) line 451 : extern void addnum(_Inout_ PNUMBER* pa, _In_ PNUMBER b, uint32_t radix);
```

기능: Highlight Word

- 특정 문자열에 대해 highlight 표시를 하여, 쉽게 추적할 수 있게 도움
- 변수의 사용을 추적할 때 유용함



기능: Highlight Word

```
void _addnum(PNUMBER* pa, PNUMBER b, uint32_t radix)
{
    PNUMBER c = nullptr; // c will contain the result.
    PNUMBER a = nullptr; // a is the dereferenced number pointer from *pa
    MANTTYPE* pcha; // pcha is a pointer to the mantissa of a.
    MANTTYPE* pchb; // pchb is a pointer to the mantissa of b.
    MANTTYPE* pchc; // pchc is a pointer to the mantissa of c.
    int32_t cdigits; // cdigits is the max count of the digits results
                    // used as a counter.

    int32_t mexp; // mexp is the exponent of the result.
    MANTTYPE da; // da is a single 'digit' after possible padding.
    MANTTYPE db; // db is a single 'digit' after possible padding.
    MANTTYPE cy = 0; // cy is the value of a carry after adding two 'digits'
    int32_t fcompla = 0; // fcompla is a flag to signal a is negative.
    int32_t fcomplb = 0; // fcomplb is a flag to signal b is negative.

    a = *pa;

    // Calculate the overlap of the numbers after alignment, this includes
    // necessary padding 0's
    cdigits = max(a->cdigit + a->exp, b->cdigit + b->exp) - min(a->exp, b->exp);

    createnum(c, cdigits + 1);
    c->exp = min(a->exp, b->exp);
    mexp = c->exp;
    c->cdigit = cdigits;
    pcha = a->mant;
    pchb = b->mant;
    pchc = c->mant;

    // Figure out the sign of the numbers
    if (a->sign != b->sign)
    {
        cy = 1;
        fcompla = (a->sign == -1);
        fcomplb = (b->sign == -1);
    }
}
```

```
// Loop over all the digits, real and 0 padded. Here we
// aligned
for (; cdigits > 0; cdigits--, mexp++)
{
    // Get digit from a, taking padding into account.
    da = (((mexp >= a->exp) && (cdigits + a->exp - c->exp)) ? a->mant[cdigits] : 0);
    // Get digit from b, taking padding into account.
    db = (((mexp >= b->exp) && (cdigits + b->exp - c->exp)) ? b->mant[cdigits] : 0);

    // Handle complementing for a and b digit. Might be
    // haven't found it yet.
    if (fcompla)
    {
        da = (MANTTYPE)(radix)-1 - da;
    }
    if (fcomplb)
    {
        db = (MANTTYPE)(radix)-1 - db;
    }

    // Update carry as necessary
    cy = da + db + cy;
    *pchc++ = (MANTTYPE)(cy % (MANTTYPE)radix);
    cy /= (MANTTYPE)radix;
} // end for ;cdigits>0;cdigits--,...

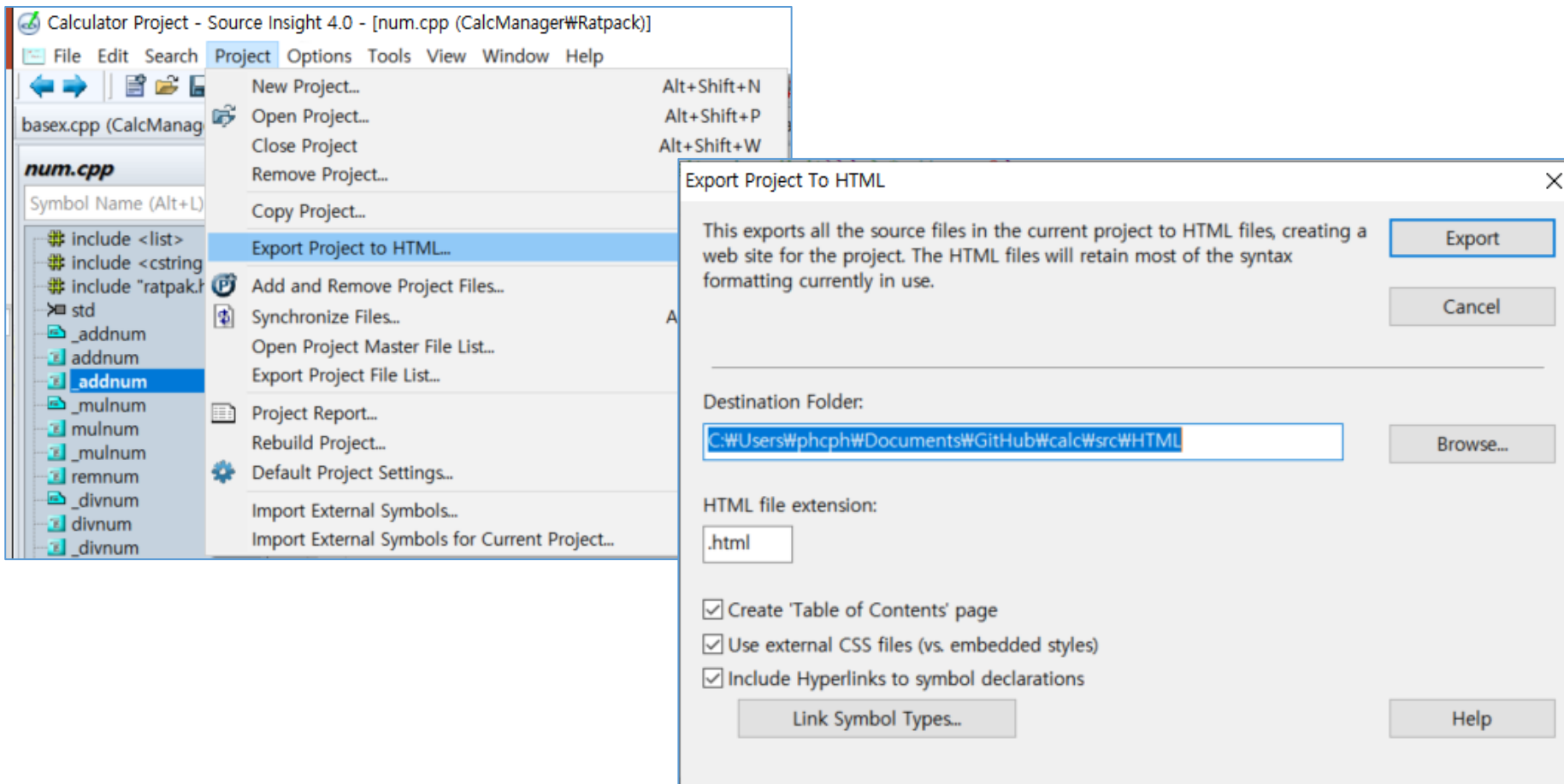
// Handle carry from last sum as extra digit
if (cy && !(fcompla || fcomplb))
{
    *pchc++ = cy;
    c->cdigit++;
}

// Compute sign of result
if (!(fcompla || fcomplb))
{
}
```

* vs 등 다른 도구에서도 주로 찾기 기능에 의해 하이라이트가 되기는 하지만, 다른 기능을 사용하면 없어지는 경우가 많음

기능: HTML 형태로 Project Export

- Syntax Highlighting 된 결과물을 HTML 형태로 export
- 웹사이트에 올려 소스 코드를 쉽게 브라우징 할 수 있음



기능: HTML 형태로 Project Export

Calculator Project

Root

- [CMakeLists.txt](#)

build

- [appversion.rc](#)

CalcManager

CalculatorHistory.cpp	CalculatorHistory.h
CalculatorManager.cpp	CalculatorManager.h
CalculatorResource.h	CalculatorVector.h
CMakeLists.txt	Command.h
ExpressionCommand.cpp	ExpressionCommand.h
ExpressionCommandInterface.h	NumberFormattingUtils.cpp
NumberFormattingUtils.h	pch.cpp
pch.h	sal_cross_platform.h
UnitConverter.cpp	UnitConverter.h
winerror_cross_platform.h	

CalcManager/ARM/Debug

MultipleQualifiersPerDimensionFound.txt	qualifiers.txt
---	--------------------------------

정리

- 많은 양의 소스 코드를 정확하고 빠르게 분석하는데 활용
 - 각종 도구의 제공으로 작업 효율성이 높음
- 작업의 단계에 따라 적절한 도구를 선택해서 사용할 것
 - SI 는 코드 분석과 수정에 활용
 - Build 및 수행은 몇 가지 설정이 필요하고, 복잡한 프로젝트에는 x
 - VS 는 Debug와 성능 분석 등이 강점이므로 build & debug에 활용

