9주차 – 로컬 환경에서 테스트 환경 구축

개발환경 설정

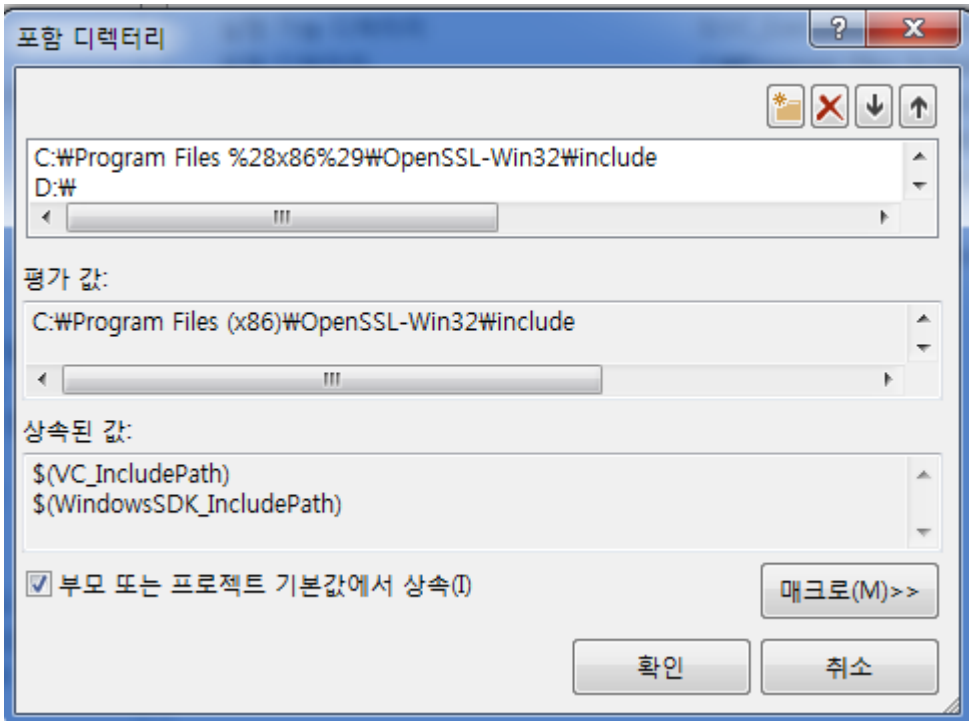
라이브러리 설치

-OpenSSL

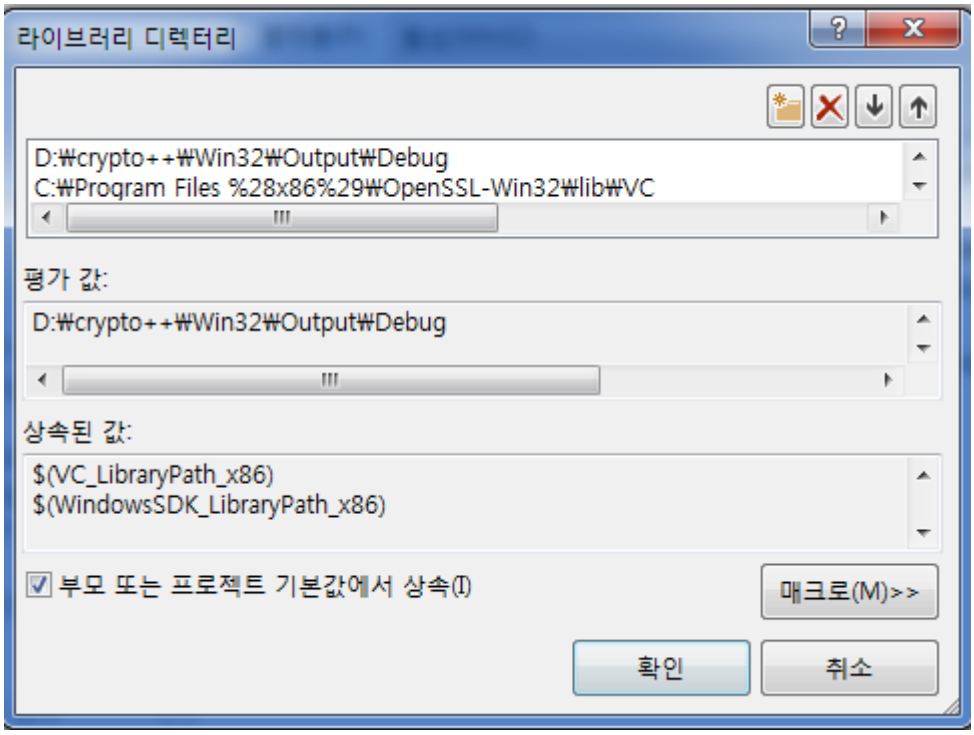
-Crypto++ 설치법은 http://lyb1495.tistory.com/25 참조, D드라이브에 다운로드

프로젝트 설정

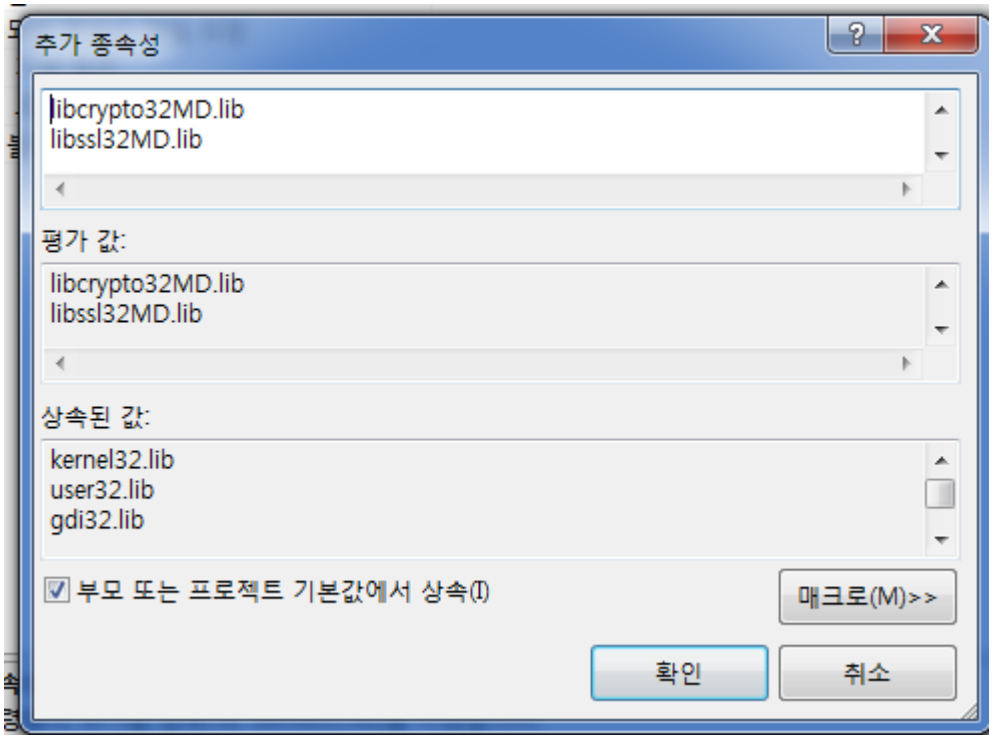
헤더 인클루드 경로 (프로젝트 설정 ->구성속성-> VC++디렉터리 -> 포함 디렉터리)



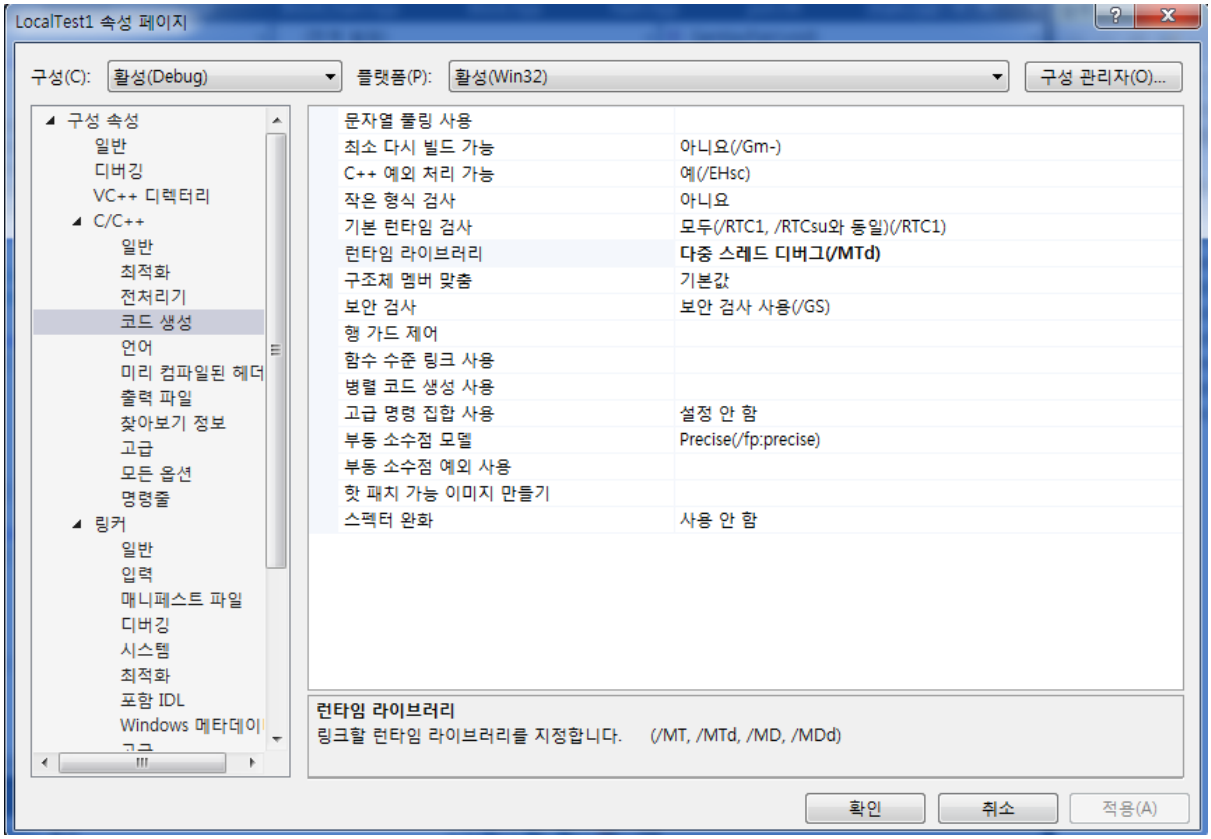
라이브러리 경로 (프로젝트 설정 ->구성속성-> VC++디렉터리 -> 라이브러리 디렉터리)



링크할 파일 (프로젝트 설정 -> 링커 -> 입력 -> 추가 종속성)

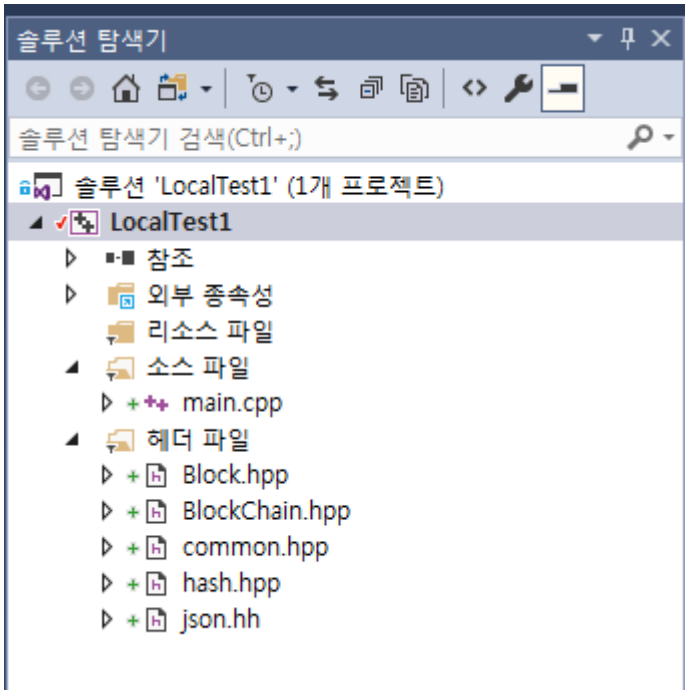


런타임 라이브러리 설정 (구성속성 -> C/C++ -> 코드 생성 -> 런타임 라이브러리)



"다중 스레드 디버그(/MTd)로 설정"

기본 프로젝트 구성에 필요한 파일들


솔루션 탐색기
솔루션 탐색기 검색(Ctrl+;)
솔루션 'LocalTest1' (1개 프로젝트)
LocalTest1
참조
외부 종속성
리소스 파일
소스 파일
main.cpp
헤더 파일
Block.hpp
BlockChain.hpp
common.hpp
hash.hpp
json.hh

```cpp
#include <iostream>
#include <sstream>
#include <string>
#include <vector>
#include <memory>
#include <stdexcept>

#include <crypto++/rsa.h>
#include <crypto++/osrng.h>
#include <crypto++/base64.h>
#include <crypto++/files.h>
using namespace CryptoPP;


#include "hash.hpp"
#include "Block.hpp"
#include "common.hpp"
#include "BlockChain.hpp"
#include "json.hh"
using json = nlohmann::json;

InvertibleRSAFunction privkey;
RSAFunction pubkey;

void GenKeyPair(void)
{
    //Generate private key and public key
    AutoSeededRandomPool rng;
    privkey.Initialize(rng, 512);
    RSAFunction _pubkey(privkey);
    pubkey = _pubkey;
}

int main()
{
    BlockChain bc;
    bc = BlockChain(0);
    GenKeyPair();

}
```

9주차 계획

* 변동사항

- 파일 암호화로 RSA 대신 AES 사용하기로 변경,

1) 팀원 모두 프로젝트 생성 및 설정 완료

2) 각자 분량을 나눠서 다음의 기능들 구현하기.

1. UserInfoList 클래스 완성 (UIL.hpp)

UserInfoList를 메인함수에서 사용한 모습,

```
int main()
{
    BlockChain bc;
    bc = BlockChain(0);
    GenKeyPair();

    UserInfoList.AddUser("a");
    User& u = UserInfoList.FindUser("a");


    u.AddUserData("DATA1", UserData(1, 1, "127.0.0.1:8080", "127.0.0.1:8080"));


    u = UserInfoList.FindUser("a");
    UserData& b = u.FindData("DATA1");
    cout << b.BackUpFileAddress << b.BlockIndex << b.FileAddress << b.HashIndex << endl;




    return 0;
}
```

2. 사용자 등록 과정

클라이언트 프로그램)

- 처음 사용자 등록을 하는 과정 필요

-AES키를 생성하고 /Key 디렉터리에 파일 형태로 저장하는 코드 작성

이 부분 참고자료

https://www.cryptopp.com/wiki/Advanced_Encryption_Standard#Encrypting_and_Decrypting_Using_AES

-노드에게 자신의 이름을 등록 요청하는 기능 필요

노드 프로그램)

String 형태의 사용자 이름을 받아서 UserInfoList에 추가하는 기능 필요,

3. 클라이언트가 파일을 RSA로 암호화 한 후 블록체인에 저장 요청하는 과정

-클라이언트는 파일을 읽어서 미리 생성해둔 키로 AES 파일 암호화

-암호화 한 후 Base64로 인코딩

-노드에게 공개키, 데이터 이름, AES로 암호화한 파일을 주고 블록체인에 저장요청

-노드는 이를 받아서 파일은 /AESUserFiles에 [사용자이름+데이터이름.dat] 파일명으로 AES로 암호화 한 파일을 저장

-이후 파일을 SHA-256으로 해쉬,

이 부분의 코드 예시는 다음에서 참고 가능

https://stackoverflow.com/questions/7853156/calculate-and-print-sha256-hash-of-a-file-using-openssl

-BlockBuffer에 파일 해쉬 저장 (BlockBuffer 에 파일 해쉬가 5개가 되면 블록 생성 및 전파)

-백업용으로 파일을 복사후 다른 노드 하나에게 전달 (받은 노드는 /AESBackUpFiles 에 [사용자이름+데이터명.dat]로 저장)

-UserInfoList 갱신, 다른 노드에게도 UserInfoList 전달,

*구체적으로 해야할것들

AES 키 생성하고 파일형태로 저장,

파일로부터 AES키를 읽어온 후 임의의 파일을 AES로 암호화 Base64로 인코딩한 후 JSON 형태로 저장.

암호화된 파일을 Base64로 디코딩 한 후 SHA-256으로 해쉬, Base64로 인코딩한후 JSON으로 저장.

참고자료 : RSA 관련 샘플, Crypto++ 키, Base64 인코딩, 디코딩 관련 자료,

https://gist.github.com/TimSC/5251670

Crypto++ 키 파일로 저장, 파일로부터 읽어오는 샘플

https://github.com/dunkyp/crypto-rsa-example