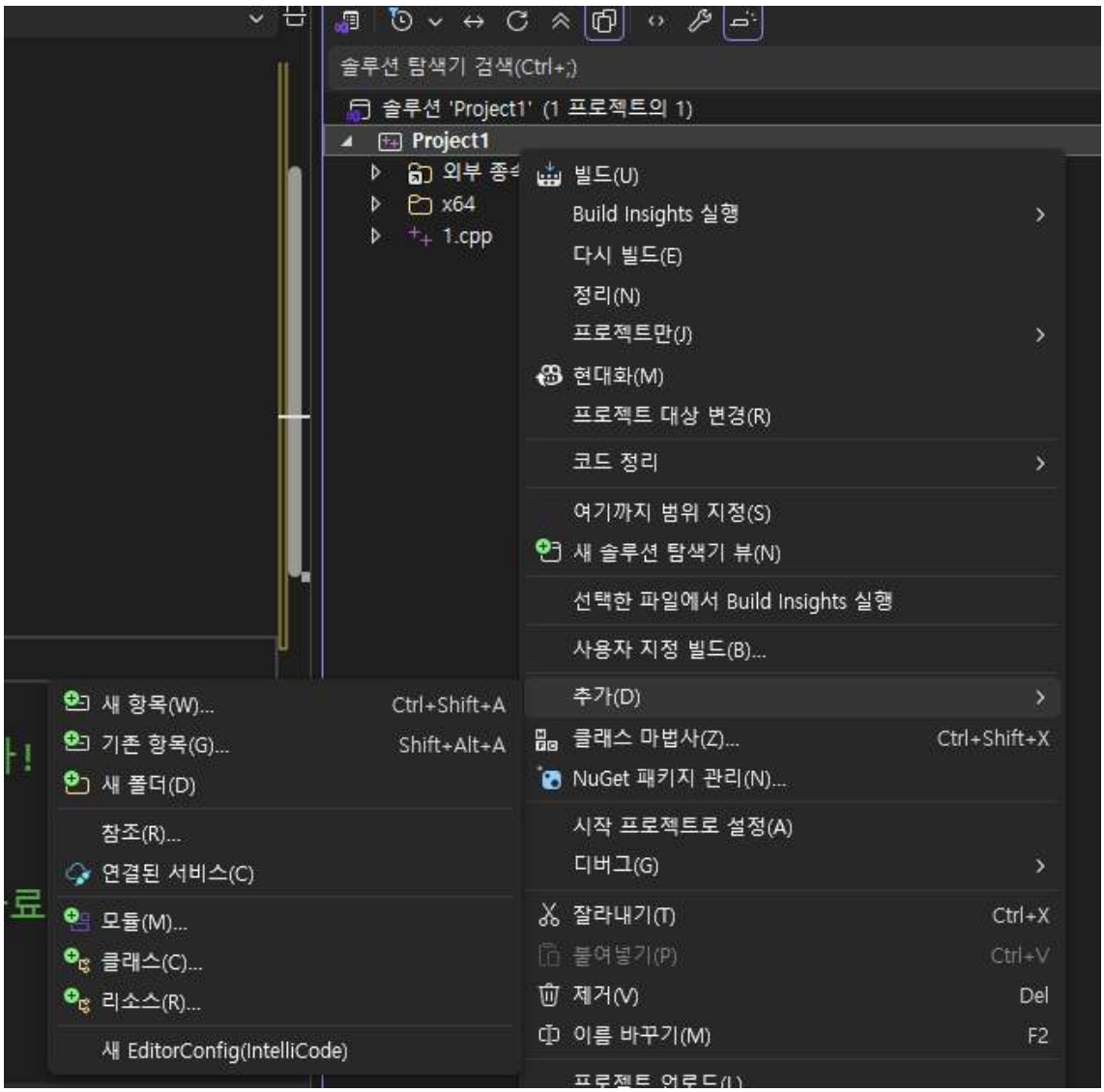
# C언어 스터디
# 4.5주차

<파일 입출력 && 헤더 && malloc은 단순한 동적배열이 아닙니다>

1. 보기 -> 솔루션 탐색기

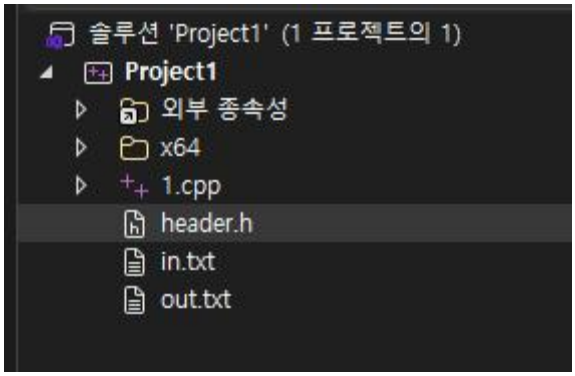

2. 프로젝트 위에 우클릭 -> 추가 -> 새항목



3. header.h, in.txt, out.txt를 만들어 주세요.

각 파일의 확장자(.h, .txt)이 아닌 이름부분 (header,in)은 기호에 따라 바꾸어도 좋습니다.

.h 는 C언어, .hpp는 C++의 함수를 더 사용할 수 있습니다.



# 파일 입출력

in.txt 파일에 다음과 같이 적어주세요

```
1 2 3
4 5 6
9 9 9
10 10 10
```

1.cpp 파일에 다음과 같이 적어주세요

```
#include<iostream>
#include<fstream> // 이거 추가해 줘야 합니다.
using namespace std;
int arr[4][3];
int main(void) {
	ifstream fin_1; // ifstream은 일종의 자료형. fin_1 부분은 맘대로 지으세요.
	fin_1.open("in.txt"); // 아까 읽을 txt 파일의 경로를 잘 써주고
	for (int i = 0; i < 4; i++) for(int j=0;j<3;j++)
		fin_1>>arr[i][j]; //잘 지은 이름 뒤에 cin 처럼 써주면 됩니다.

	for (int i = 0; i < 4; i++) {
		for(int j=0;j<3;j++)cout<<arr[i][j]<<" ";
		cout<<"\n";
	}// 잘 출력이 되나요?

	ofstream fout_lily; // ofstream은 일종의 자료형. fout_lily 부분은 맘대로 지으세요.
	fout_lily.open("out.txt"); // 써줄 txt 파일의 경로를 잘 써주고
	for (int i = 0; i < 4; i++) {
		int tmp=0;
		for (int j = 0; j < 3; j++)tmp+=arr[i][j];
		fout_lily<<tmp<<"\n"; //잘 지은 이름 뒤에 cout 처럼 써주면 됩니다.
	}
	// out.txt를 열어 봅시다.
}
```

out.txt를 열어주세요. 잘 되어 있죠?

```
6
15
27
30
```

# 헤더

header.h (혹은 .hpp) 파일에 다음과 같이 적어주세요

```
#pragma once // 뭐 이건 나중에 따로 배워 보시죠. 구조가 복잡해지면 알아서 배우실 겁니다.
int add(int a, int b) {
	return a+b;
}
```

1.cpp파일을 사용해 보겠습니다.

```
#include<iostream>
#include"header.h" // 경로를 잘 include 해 주고
using namespace std;
int main(void) {
	int a,b;
	a=10;b=100;
	cout<<add(a,b); // 잘 되나요?
}
```

# malloc? malloc은 단순한 동적 배열이 아닙니다.

```
#include<iostream>
using namespace std;
int* func(void) {
	int a = 10;
	return &a;
}
int main(void) {
	int* n = func();
	cout << *n; // 10이 나올 거 같죠? 네 뭐 잘 나옵니다.
}
```

```
#include<iostream>
using namespace std;
void recurse(int depth) { // 정확히 뭔지는 나중에 배웁시다.
	int x[100];
	if (depth > 0)
		recurse(depth - 1);
}
int* func(void) {
	int a = 10;
	return &a;
}
int main(void) {
	int* n = func();
	recurse(1000); // 그냥 뭐 함수 썼는데,
	cout << *n; // 잘 안나오죠?
}
//중괄호가 끝난 변수는 사라집니다.
```

```
#include<iostream>
#include<stdlib.h>
using namespace std;
void recurse(int depth) {
	int x[100];
	if (depth > 0)
		recurse(depth - 1);
}
int* func(void) {
	int* a = (int*)malloc(sizeof(int)); // 메모리에 강제로 공간과 데이터를 넣습니다.
	(*a)=100;
	return a;
}
int main(void) {
	int* n = func();
	recurse(1000);
	cout << *n; // 이제는 잘 나오네요.
	free(n); // 강제로 쓴 메모리는 이제 안 쓴다고 말 해 줘야 합니다.
}
```

# malloc은 메모리에 직접 데이터를 넣습니다. free, 중요해요