인트아이 정연한

2022-2 인트아이 C++ 수업 자료

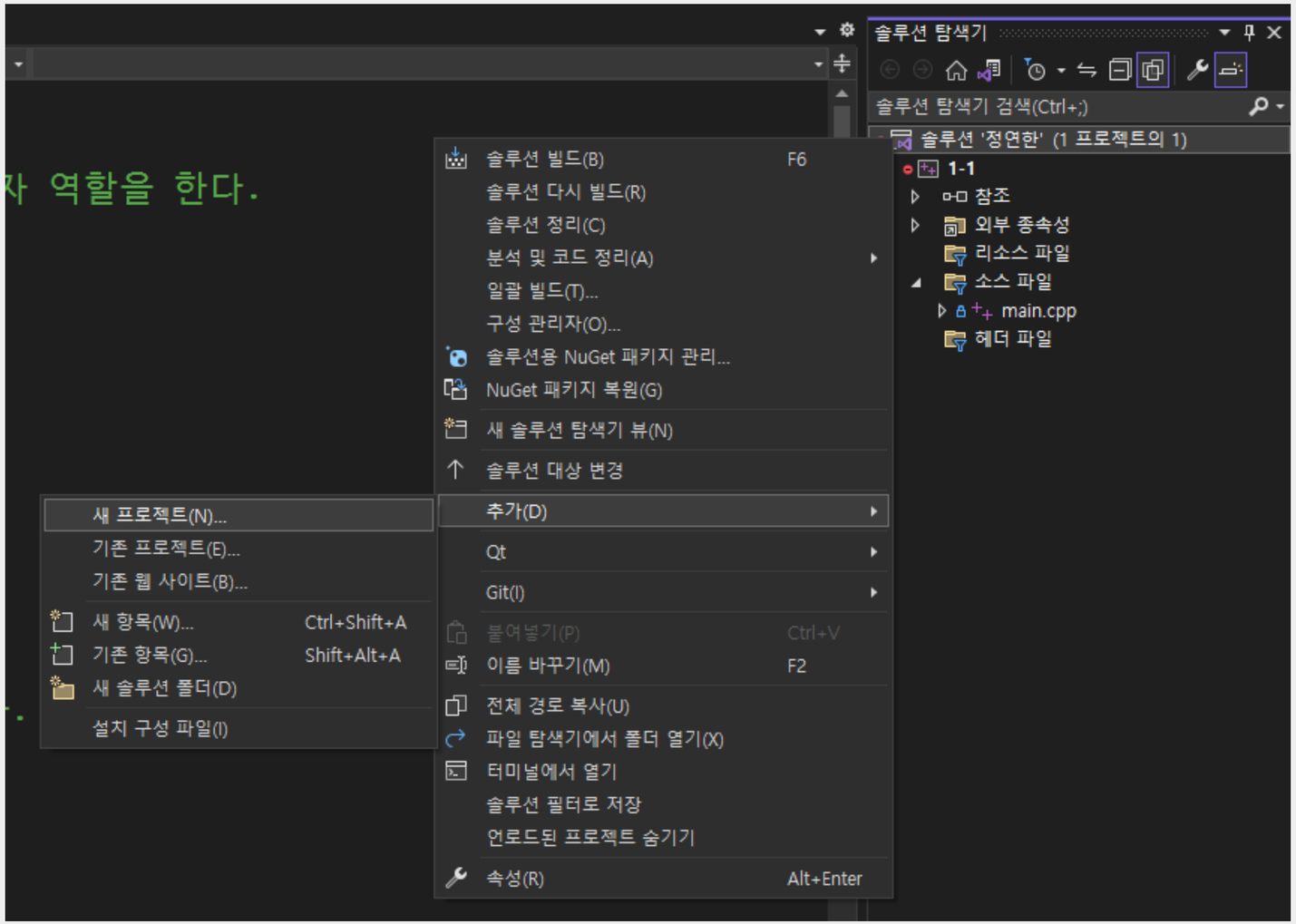
C++ 기초 2강 변수

CONTENTS

새 프로젝트를 구성하는 법

저번 시간에 제대로 깃허브에 있는 리포지토리를 복제했다면,
솔루션 탐색기에 오른쪽과 같이 뜰 것이다.
이때, **솔루션**에 커서를 갖다댄 뒤 **마우스 오른쪽 클릭**을 하고,
추가 -> 새 프로젝트 이 순서대로 새 프로젝트를 구성해주자.

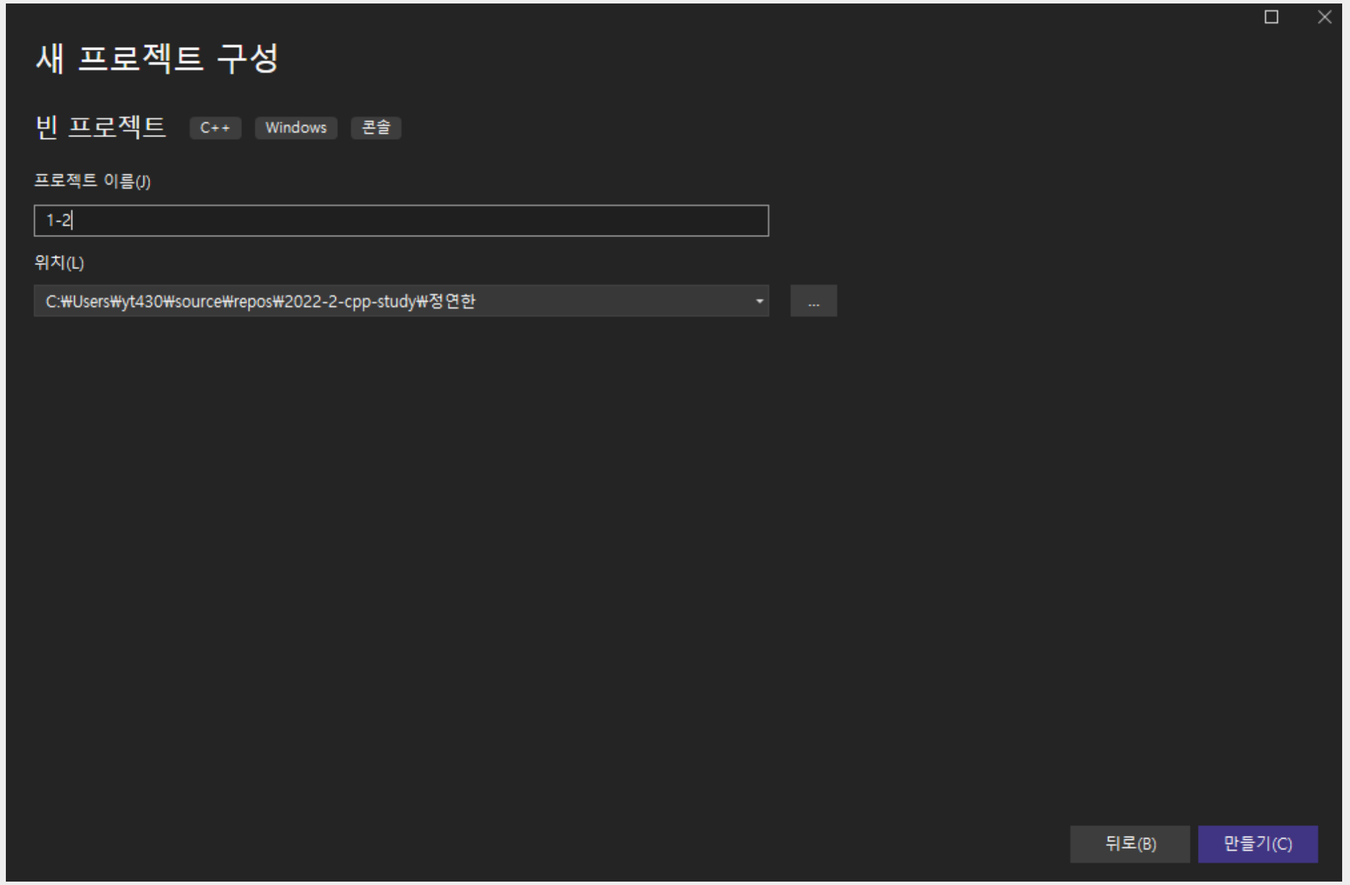
만약 솔루션 탐색기에 오른쪽 사진처럼 나오지 않는다면,
저번 강의노트를 확인하거나 멘토에게 반드시 질문 할 것!!!



새 프로젝트를 구성하는 법

위와 같은 방식으로 새 프로젝트를 생성하려 하면 아래와 같이 뜰텐데,
보면 알겠지만 솔루션 이름을 쓰는 곳이 따로 없는 것을 알 수 있다.
이미 있는 솔루션 안에 새 프로젝트를 만드는 것이니 걱정하지 말고 진행해주자.
프로젝트 이름은 1-2로 해주도록 하자.

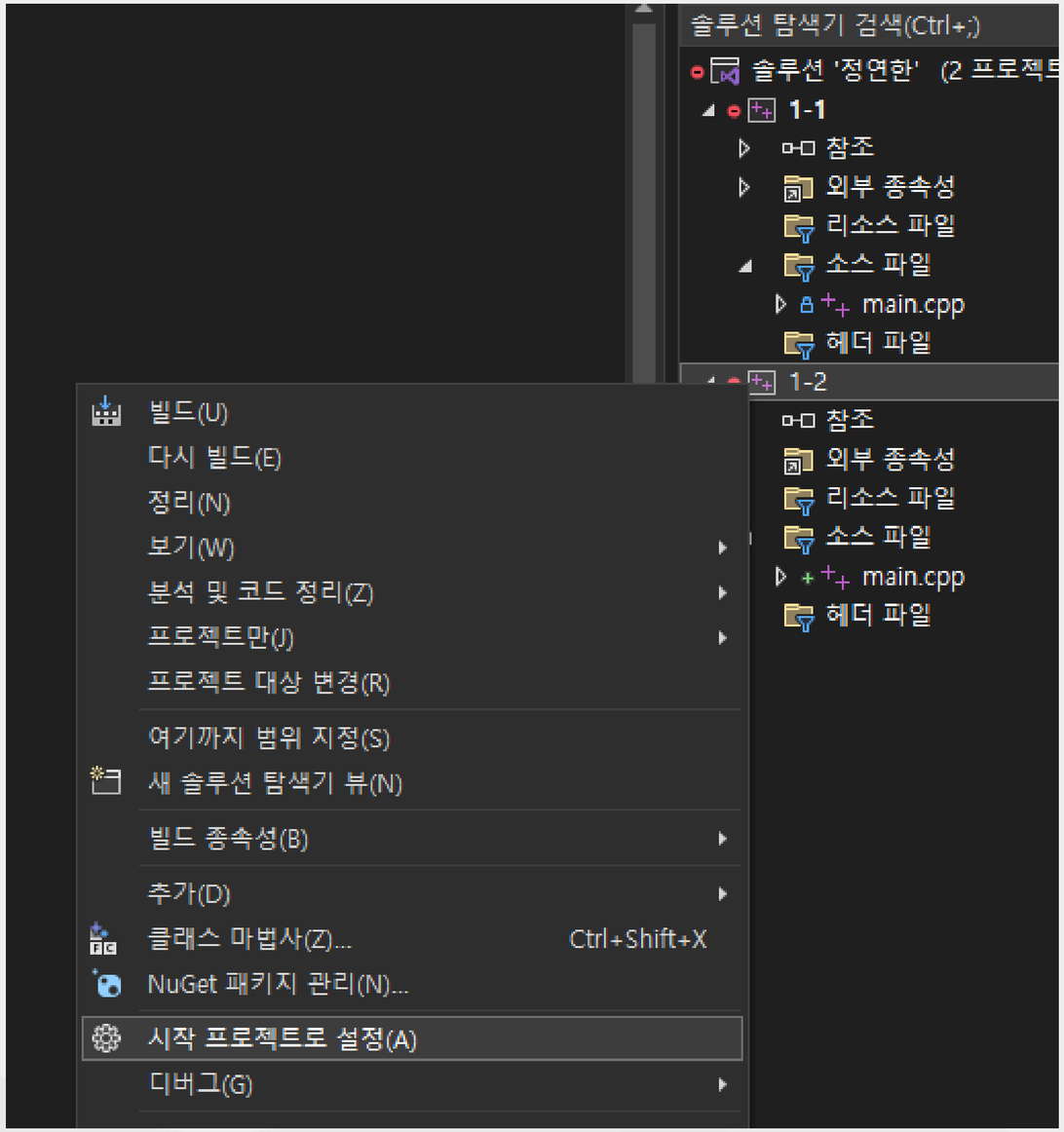
위치는 이미 제대로 된 위치일 것이니 따로 건들지는 말자
건들면 꼬일 수도 있다.
혹시나 잘못건드려서 꼬였다면 오른쪽 사진을 참고해
2022-2-cpp-study₩(자기이름) 위치로 다시 지정해주도록 하자.



새 프로젝트를 구성하는 법

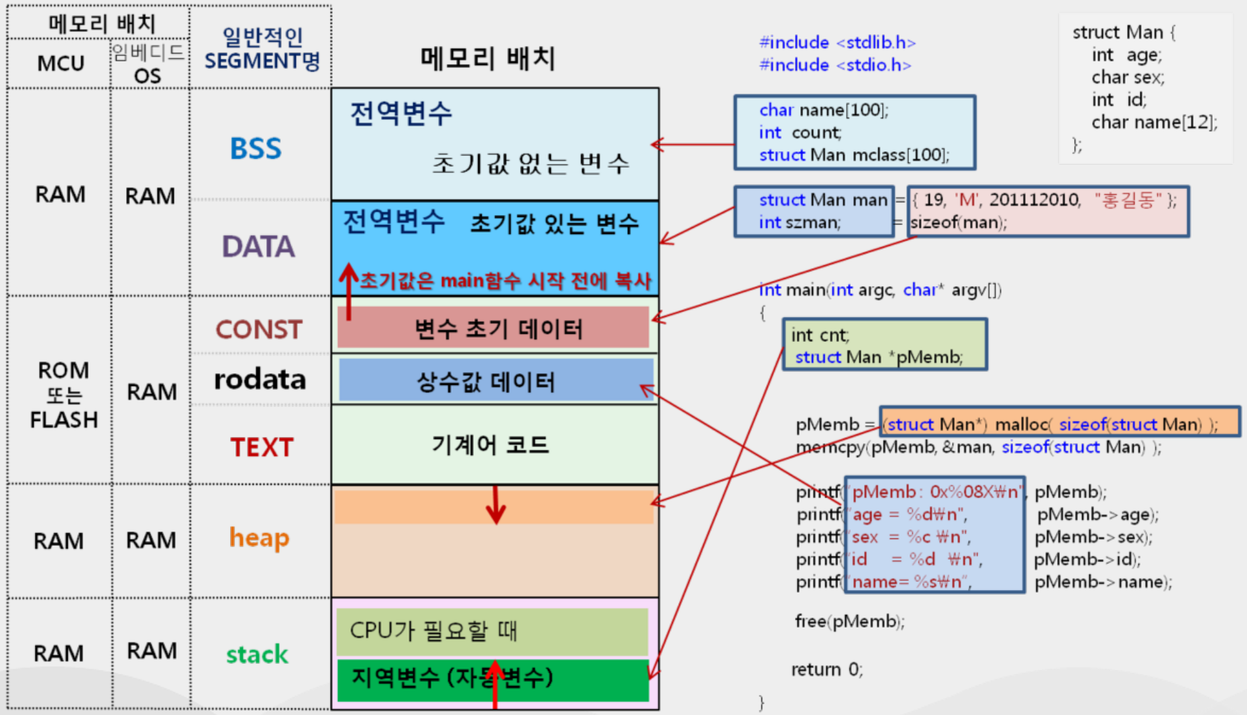
위의 과정을 잘 따라왔다면 1-2 프로젝트가 생겼을 것이다.
해당 프로젝트에 마우스 커서를 갖다댄 뒤 마우스 오른쪽 클릭을 하고,
시작 프로젝트로 설정을 반드시 눌러주자.

이렇게 해야 우리가 새로 만든 프로젝트 안에 있는 소스파일이 컴파일 된다.
다시 1-1 을 컴파일 하고 싶다면 1-1을 시작 프로젝트로 설정 해주면 된다.



변수? 어디서 많이 들어봤는데

자, 변수는 간단하게는 그냥 **변할 수 있는 값**을 의미한다.
수학에서의 미지수 x와 같이 어떤 값이든 넣을 수 있는 그런 값
근데 사실 값을 넣을 수 있는 값이라는건 뭔가 좀 이상하지
프로그래밍에서의 변수는, 좀 더 엄밀히 말해서 **어떤 값을 넣기 위해 할당받은 메모리 공간**이다.
아래 그림은 메모리 구조에 관한 그림인데, 변수를 선언하면 기본적으로 이 메모리 구조의
스택 영역에서 공간을 할당받는다.



비유를 통해 이해해보자

변수 선언: 메모리(RAM)에 내가 사용할 수 있게 일정 크기의 **공간을 할당**받는 것
변수를 창고라고 치면, 땅을 임대받아 **새 창고를 짓는 행위**에 비유할 수 있음

변수의 초기화: 메모리 공간을 **할당 받음과 동시에** 그 곳에 있던 **쓰레기 값을 치우고 새 값을 넣는 것**
창고로 치면, 내가 창고를 **짓고**나서 **창고안에 있던** 건축 자재들을 **치우고**, 내가 넣고 싶은 물건을 **넣는** 행위

변수에 값 대입: **기존에 어떤 값이 들어있는** 메모리 공간에 새로운 값을 집어넣는 것
창고로 치면, 내가 **기존에 지었던** 창고에 있던 물건들을 **빼고 새 물건들을 집어넣는** 행위

자료형(data type): 데이터의 종류
기본적으로 변수를 선언하기 위해선 **변수의 이름 앞**에 반드시
해당 변수에 **들어갈** 데이터의 **자료형**을 적어주어야 한다.

변수를 선언하는 것은 창고를 짓는것에 비유할 수 있다고 했는데,
예를 들어 내가 곰인형을 창고에 넣는다고 치자.
그렇다면 창고의 형태도 **곰인형을 넣을 수 있는 창고**로 만들어야지,
뜬금없이 **냉동 창고를 만들어 버리면 안되는 것**이다.

```cpp
int main() {
    int i = 0; // 4바이트 정수 최대값: 2,147,483,647

    float f = 0.0f; // 4바이트 실수(단정밀도 부동 소수점)

    double d = 0.0; // 8바이트 실수(배정밀도 부동 소수점)

    bool b = true; // 1바이트 논리 자료형 변수에 0이 들어오면 false
    // bool 자료형 변수에 0 이외의 값이 들어오면 true
```

```cpp
// 문자는 반드시 홑따옴표로,
// 문자열은 반드시 쌍따옴표로 선언해주어야 한다.
// 문자는 말 그대로 하나의 문자(charactor), 문자열은 여러 문자들이 배열된 것
char c = 'a'; // 1바이트 문자(실제 메모리에는 해당 문자에 매칭되는 정수값으로 저장 됨)

char str1[] = "abc"; // 기존 C 스타일 문자열
string str2 = "abc"; // C++ 스타일 문자열

// 대괄호: 배열
// "abc" 대신 { 'a', 'b', 'c' } 이렇게도 넣을 수 있다.
```

```cpp
    // 변수에 값 대입
    // 자리수 구분은 홑따옴표(')를 사용해 구분할 수 있다.
    i = 2'147'483'647;

    // int 최대값에 1을 더해보자.
    // 오버플로우가 일어나 -2,147,483,648이 나오게 된다.
    cout << i + 1 << endl;

    return 0;
}
```

코드를 통해 이해해 보자

자 아래 코드를 보면 int i = 0; 이런식으로 코드가 작성되어있다.
여기서 **int**는 위에 설명 되었듯 **자료형**이다.
변수에 **들어갈 수 있는** 데이터의 **형태**를 지정해주는 역할을 하며, 여기서 **int는 정수**를 의미한다.
int의 경우 -2,147,483,648 ~ 2,147,483,647 사이의 값이 들어갈 수 있다.
그리고 자료형 옆에 있는 **i**는 **변수의 이름**을 뜻한다.
여기서 변수의 이름이 이 변수를 고유하게 식별하게 하는 **식별자**의 역할을 한다.
그리고 그 옆 = 은 변수를 **초기화** 하는 역할을 하며, 여기서는 0이란 값을 넣어 변수를 초기화 하고 있다.

```c
int main() {
    int i = 0; // 4바이트 정수 최대값: 2,147,483,647

    float f = 0.0f; // 4바이트 실수(단정밀도 부동 소수점)

    double d = 0.0; // 8바이트 실수(배정밀도 부동 소수점)

    bool b = true; // 1바이트 논리 자료형 변수에 0이 들어오면 false
    // bool 자료형 변수에 0 이외의 값이 들어오면 true
```

코드를 통해 이해해 보자

이 코드를 보면 자료형이 여러가지가 있는데, **int, float, double, bool**이 보인다.
int는 integer(정수)의 약자로, 말 그대로 **정수값**을 저장할 수 있다.
float는 floating point(부동 소수점)의 약자로 이름만 보자면 소수점이 둥둥 떠다닌다 라는 뜻인데,
실제로 float는 **실수값**을 저장하기 위해 정확도를 조금 희생하여 값에따라 소수점의 위치를 지정한다.
자세한건 https://youtu.be/ZQDsWySjY6g 참조
double는 double precision floating point(배정밀도 부동 소수점)의 약자로 float보다 좀 더 **정밀하게 실수를 표현**할 수 있다.
마지막으로 **bool**은 **논리 연산**을 위한 자료형으로 boolean algebra(불 대수)의 약자이다.
0(false)과 1(true)을 통해 논리 연산을 하기 위해 사용 되어진다.

```
int main() {
    int i = 0; // 4바이트 정수 최대값: 2,147,483,647

    float f = 0.0f; // 4바이트 실수(단정밀도 부동 소수점)

    double d = 0.0; // 8바이트 실수(배정밀도 부동 소수점)

    bool b = true; // 1바이트 논리 자료형 변수에 0이 들어오면 false
    // bool 자료형 변수에 0 이외의 값이 들어오면 true
```

문자는 어떻게 표현하지?

위에서 말한 4가지 자료형 외에도 자주 쓰이는 자료형이 하나 더 있는데, 바로 **char**이다.
char은 **하나의 문자**를 저장할 수 있는 자료형으로, 기본적으로 **알파벳과 몇몇 특수문자**를 저장할 수 있다.
아니 그럼 한글은 어떻게 저장하냐? 할 수 있는데 **한글은 문자열로 저장**할 수 있다.
문자열이란 문자들이 모인 것으로, 한글 한 글자는 **1바이트로 표현 할 수 없으며**, UTF-8 인코딩 기준 3바이트를 차지한다.
visual studio는 기본적으로 EUC-KR 인코딩을 사용하여 한글을 저장하는데, 이 인코딩 기준 한글은 2바이트를 차지한다.
UTF-8이니, EUC-KR이니 뭔소린지 모르겠다면 일단 한글은 문자열을 사용해 저장한다는 것만 알아두면 된다.
인코딩 관련한 설명은 나중에 번외편으로 따로 올릴 예정이다.

```
// 문자는 반드시 홑따옴표로,
// 문자열은 반드시 쌍따옴표로 선언해주어야 한다.
// 문자는 말 그대로 하나의 문자(charactor), 문자열은 여러 문자들이 배열된 것
char c = 'a'; // 1바이트 문자(실제 메모리에는 해당 문자에 매칭되는 정수값으로 저장 됨)

char str1[] = "abc"; // 기존 C 스타일 문자열
string str2 = "abc"; // C++ 스타일 문자열

// 대괄호: 배열
// "abc" 대신 { 'a', 'b', 'c' } 이렇게도 넣을 수 있다.
```

그렇다면 문자열은?

아무튼 그래서 문자는 char을 이용해 저장 할 수 있다는 건 알겠는데, 문자열은 어떻게 표현하느냐,
바로, **char 배열** 또는 C++의 문자열을 저장하는, 자료구조의 일종인 **string**을 통해 저장할 수 있다.
우선 char 배열부터 보자, 코드를 보면 char str1[] = "abc"라고 되어있는데,
딱 보아하니 문자열은 쌍따옴표를 통해서 선언할 수 있고, 이렇게 선언된 문자열을 저 str1[]라는 놈한테 넣는 것 같다.

그런데 여길 보면 처음 보는 기호(**[]**)가 하나 등장하는데, 이 기호는 배열을 선언하는 **선언자**로,
변수를 선언 할 때, 이 배열 선언자를 붙이면 일반적인 변수가 아닌 **배열을 선언**하게 된다.

```
// 문자는 반드시 홑따옴표로,
// 문자열은 반드시 쌍따옴표로 선언해주어야 한다.
// 문자는 말 그대로 하나의 문자(charactor), 문자열은 여러 문자들이 배열된 것
char c = 'a'; // 1바이트 문자(실제 메모리에는 해당 문자에 매칭되는 정수값으로 저장 됨)

char str1[] = "abc"; // 기존 C 스타일 문자열
string str2 = "abc"; // C++ 스타일 문자열

// 대괄호: 배열
// "abc" 대신 { 'a', 'b', 'c' } 이렇게도 넣을 수 있다.
```

그렇다면 문자열은?

이때, 문자열은 문자들이 **정해진 순서대로 나열**된 것, 즉 **배열 된 문자들**을 뜻한다.
따라서 문자열을 저장하기 위해서, 문자 배열(char array)을 사용할 수 있는데,
여기선 char str1[]을 통해 문자 배열을 선언하였다.
이렇게 선언된 문자 배열에 "abc"를 넣게 되면, str1의 크기가 정해지는데,
이때, **문자열의 끝**에는 반드시 **₩0**이라는 **제어 문자**가 들어가게 된다.
이 문자는 문자열의 끝을 나타내는 문자로, 이 문자로 인해 str1은 **abc₩0 이렇게 4개의 문자가 저장**되게 된다.
이 때문에 str1의 **크기**는 이 초기화 시점에 **4**로 지정되게 된다.(문자 하나가 1바이트이므로)

```
// 문자는 반드시 홑따옴표로,
// 문자열은 반드시 쌍따옴표로 선언해주어야 한다.
// 문자는 말 그대로 하나의 문자(charactor), 문자열은 여러 문자들이 배열된 것
char c = 'a'; // 1바이트 문자(실제 메모리에는 해당 문자에 매칭되는 정수값으로 저장 됨)

char str1[] = "abc"; // 기존 C 스타일 문자열
string str2 = "abc"; // C++ 스타일 문자열

// 대괄호: 배열
// "abc" 대신 { 'a', 'b', 'c' } 이렇게도 넣을 수 있다.
```

그렇다면 문자열은?

문자 배열은 대충 알아봤으니, **string**에 대해 살펴보자.
string은 다른 자료형들과는 다르게, 하이라이트가 파란색이 아닌 초록색으로 들어가있다.
이는 다른 자료형들과 달리 class로 선언 된 **자료구조**이기 때문인데,
string은 **C++에서 기본 지원**하는 자료 구조 중 하나로, **문자를 요소로 가지는 자료구조**이다.
따라서 문자 배열처럼, 문자열을 집어넣을 수 있는 **컨테이너**인데,
내부에 여러가지 **메서드**를 가지고 있어, 각종 **연산**이 쉽게 가능하다는 장점이 있다.
또한 보기에도 훨씬 **직관적**이기 때문에 앞으로 **문자열을 사용할때는 string을 사용**해주도록 하자.

```
// 문자는 반드시 홑따옴표로,
// 문자열은 반드시 쌍따옴표로 선언해주어야 한다.
// 문자는 말 그대로 하나의 문자(charactor), 문자열은 여러 문자들이 배열된 것
char c = 'a'; // 1바이트 문자(실제 메모리에는 해당 문자에 매칭되는 정수값으로 저장 됨)

char str1[] = "abc"; // 기존 C 스타일 문자열
string str2 = "abc"; // C++ 스타일 문자열

// 대괄호: 배열
// "abc" 대신 { 'a', 'b', 'c' } 이렇게도 넣을 수 있다.
```

대입 그리고 오버플로우

아래 코드를 보면 아까 초기화 할 때 처럼 변수 i에 **= 연산자를 사용**해 값을 집어넣고 있다.
하지만 이건 **초기화가 아니라 대입**이다.
변수가 **선언될 때 값을 넣어주는 것**만이 **초기화**이고 **그 이외의 상황에 값을 넣어주는 것**은 전부 **대입**이다.
자 그런데 여기서 int 최대값을 i에 넣어주었는데, 여기에 1을 더한 값을 출력하게 되면,
오버플로우가 일어나 int 최소값인 -2,147,483,648이 나오게 된다.

```
    // 변수에 값 대입
    // 자리수 구분은 홑따옴표(')를 사용해 구분할 수 있다.
    i = 2'147'483'647;

    // int 최대값에 1을 더해보자.
    // 오버플로우가 일어나 -2,147,483,648이 나오게 된다.
    cout << i + 1 << endl;

    return 0;
}
```

대입 그리고 오버플로우

오버플로우가 뭔지, 오버플로우가 나면 왜 음수가 나오는지에 대한 **자세한 설명은 GitHub에 올린 1-2 코드**에 자세히 올려두었다.
여기서는 오버플로우가 뭔지에 대해서만 간략하게 설명하도록 하겠다.
기본적으로 오버플로우란 **나타낼 수 있는 값의 상한 또는 하한을 넘어서 값이 증가/감소** 하게 되면,
해당 자료형이 나타낼 수 있는 **가장 작은 값 또는 가장 큰 값**을 나타내는 것을 말한다.
아래 예시에선 3비트를 예시로 들었는데, 최상위 비트를 부호비트로 사용하였기 때문에 나타낼 수 있는 값의 범위가 -4~3이다.
예시를 잘 보면 011 까지는 양수 3이었는데 거기에 1을 증가시켜보니 100 이 되면서, 부호비트를 침범해, 음수인 -4가 되었다.

```
// 3비트로 줄여서 보면
// 3비트로 나타낼 수 있는 값의 범위 = -4~3
// 2^3 == 8개
// 0을 양수 취급하기 때문에 양수의 범위는
// 해당 변수가 나타낼 수 있는 최대값 / 2 - 1 이 되는 것이고
// 음수는 그런 거 없으니까 그냥 최대값 / 2가 되는 것이고
// 000 == 0
// 001 == 1
// 010 == 2
// 011 == 3
// 100 == -4
// 101 == -3
// 110 == -2
// 111 == -1
// 000 == 0
```

백준 1000번 A + B

문제

두 정수 A와 B를 입력받은 다음, A+B를 출력하는 프로그램을 작성하시오.

입력

첫째 줄에 A와 B가 주어진다. (0 < A, B < 10)

출력

첫째 줄에 A+B를 출력한다.

예제 입력 1 복사	예제 출력 1 복사
1 2	3