1-2. Data와 Code

창의적소프트웨어프로그래밍
2022년도 여름학기
Racin

프로그래밍 해서 만드는 프로그램

- 우리는 프로그래밍을 해 보기 위해 여기에 앉아 있습니다.

- 프로그래밍이 뭔지는 지금 딱 찝어 말하기는 어렵겠지만,
  아무튼 프로그래밍을 하면... 프로그램이 형성됩니다.

프로그래밍 해서 만드는 프로그램

- '(형성된)프로그램이 무엇인지' 또한 한 줄로 딱 잘라 설명하기는 어렵습니다.
  - 심지어 동네마다 그 실체가 조금씩 다르게 생겼을 수 있어요

- 그렇기는 하지만, 우리는 처음 시작하는 지금 시점에
  (개념적)프로그램을 두 가지 영역의 조합으로 바라보고 진행해 보려 합니다:

프로그래밍 해서 만드는 프로그램

- '(형성된)프로그램이 무엇인지' 또한 한 줄로 딱 잘라 설명하기는 어렵습니다.
  - 심지어 동네마다 그 실체가 조금씩 다르게 생겼을 수 있어요

- 그렇기는 하지만, 우리는 처음 시작하는 지금 시점에
  (개념적)프로그램을 두 가지 영역의 조합으로 바라보고 진행해 보려 합니다:
  - Data
  - Code

- 이번 시간에는 이 두 영역을 나누어, 함께 확인해 보려 해요.

이번 시간에는

- 몇 가지 주요 요소들을 Data와 Code의 두 가지 관점에서 정리해 봅니다.
- '내 첫 C 프로그램'을 함께 구성해 봅니다.
- 새로 나오는 키워드
  - **값**
  - **문장**, **실행**
  - **변수**
  - **수식**, **계산**, **연산자**, **형식**

이번 시간에는

- 몇 가지 주요 요소들을 Data와 Code의 두 가지 관점에서 정리해 봅니다.
- '내 첫 C 프로그램'을 함께 구성해 봅니다.
- 새로 나오는 키워드
  - **값**
  - **문장**, **실행**
  - **변수**
  - **수식**, **계산**, **연산자**, **형식**

수업자료에서 굵은 글씨로 적어 두는 단어들은
전통적인, 그러나 매우 중요한 의미를 함축하고 있어요.
(어떤 단어들은 그냥 적다가 나중에 굵은 글씨가 되기도 해요)

복습할 때는 이 단어들 위주로 봐 두면 좋을 듯해요

이번 시간에는

- 일단 앞 시간에 프로그래밍 환경을 구성해 두었으니,
  이론맛을 보기 전에 '내 첫 C 프로그램'을 함께 만들어 봅시다.

내 첫 C 프로그램

- 표준 환경에서, 새 프로그램을 만들고 싶을 때는 보통 그 프로그램을 위한 새 '프로젝트'를 만들게 돼요.

- 강사와 함께 새 프로젝트를 추가해 봅시다.

내 첫 C 프로그램

- 방금 고른 CSP_Project 항목은
  우리 수업에 적합한 각종 설정들을 미리 해 둔 버전이에요.
  - 다른 곳에서 C 프로그램 만들 때는 그 쪽에 맞게 설정을 고칠 필요가 있을 거예요
  - 뭐 지금은 그냥 필요할 때마다 방금 느낌대로 새 프로젝트를 만들면 돼요
- 다만, 지금 우리는 'C' 프로그램을 만들고 있으니,
  솔루션 탐색기에 보이는 main.cpp의 이름을 main.c 로 고쳐 둡시다.

내 첫 C 프로그램

- 방금 만든 main.c에 아래 내용을 추가해 주세요:

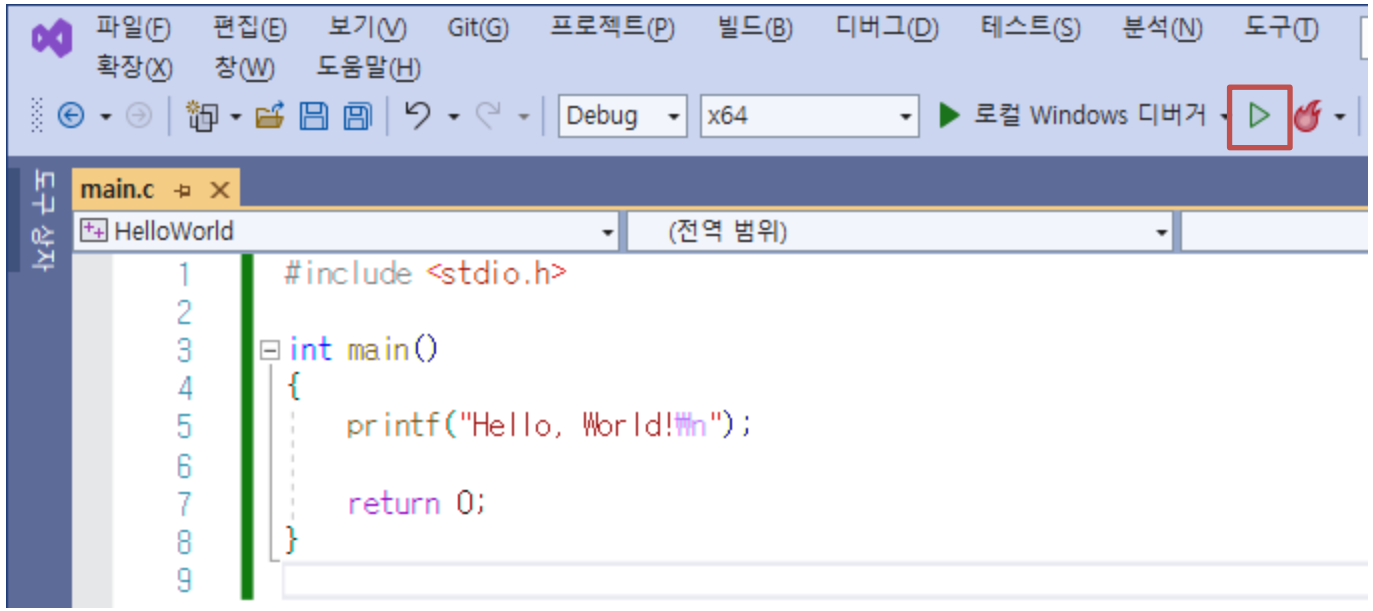
```c
#include <stdio.h>

int main()
{
    printf("Hello, World!₩n");

    return 0;
}
```

내 첫 C 프로그램

- 내용을 다 적은 다음, 일단 Ctrl + S를 눌러 저장하고 코드 창 위에 보이는 녹색 빈 화살표 모양을 누르면 프로그램을 실행해볼 수 있어요.
  - 검은 창이 뜨고 예상한 출력 결과가 보이면 성공이에요



내 첫 C 프로그램

- 출력 결과를 놓고, 방금 작성한 코드를 한 번 확인해 봅시다.

```c
#include <stdio.h>

int main()
{
    printf("Hello, World!₩n");

    return 0;
}
```

내 첫 C 프로그램

- 출력 결과를 놓고, 방금 작성한 코드를 한 번 확인해 봅시다.

```c
#include <stdio.h>

int main()
{
    printf("Hello, World!₩n");

    return 0;
}
```

프로그램을 실행했더니 화면에 Hello, World! 가 출력되었습니다.

여기서, '왜' 이게 출력되었는지를 묻는다면
뭐라고 답할 수 있을까요?

내 첫 C 프로그램

- 출력 결과를 놓고, 방금 작성한 코드를 한 번 확인해 봅시다.

```c
#include <stdio.h>

int main()
{
    printf("Hello, World!₩n");

    return 0;
}
```

잘은 모르겠지만, 이 동네가 상식적으로 생겼다면
아마도 이 부분을 이렇게 적어놨기 때문일 것 같아요.

(이 한 줄을 지운 다음 다시 한 번 실행해 봐요)

내 첫 C 프로그램

- 출력 결과를 놓고, 방금 작성한 코드를 한 번 확인해 봅시다.

```c
#include <stdio.h>

int main()
{
    printf("Hello, World!₩n");

    return 0;
}
```

구체적으로 상상해본다면,
'왜 하필 Hello, World!냐'고 묻는다면
아마 여기 괄호 안에 큰 따옴표로 그렇게 적어 놓았기 때문인 것 같아요.

내 첫 C 프로그램

- 출력 결과를 놓고, 방금 작성한 코드를 한 번 확인해 봅시다.

```c
#include <stdio.h>

int main()
{
    printf("안녕하세요!₩n");

    return 0;
}
```

만약 다른 메시지를 검은 창에 출력하고 싶다면,
그 다른 메시지에 해당하는 내용으로 괄호 안을 채워주면 될 것 같아요.

(큰따옴표 안 내용을 살짝 바꿔서 한 번 더 실행해 봐요)

내 첫 C 프로그램

- 출력 결과를 놓고, 방금 작성한 코드를 한 번 확인해 봅시다.

그러면... 괄호 왼쪽에 적혀 있는 이 printf는 무얼 의미할까요?

```
#include

int main()
{
    printf("안녕하세요!₩n");

    return 0;
}
```

내 첫 C 프로그램

- 출력 결과를 놓고, 방금 작성한 코드를 한 번 확인해 봅시다.

그러면... 괄호 왼쪽에 적혀 있는 이 printf는 무얼 의미할까요?

맞아요. 아마도, '출력해줘'같은 표현인 것 같아요.

```
#include

int main()
{
    printf("안녕하세요!₩n");

    return 0;
}
```

내 첫 C 프로그램

- 출력 결과를 놓고, 방금 작성한 코드를 한 번 확인해 봅시다.

```
#include

int main()
{
    printf("안녕하세요!₩n");

    return 0;
}
```

결과적으로 이 한 줄은, 프로그램을 실행할 때
프로그래머가 괄호 안에 적어 둔 내용을 바탕으로
검은 창에 적절한 메시지를 출력해 주는 것 같아요.

내 첫 C 프로그램

- 그러면, 이 코드를 살짝 수정해서 새 버전 프로그램을 만들어 봅시다.

```
#include <stdio.h>

int main()
{
    printf("안녕하세요!₩n");

    return 0;
}
```

내 첫 C 프로그램

- 그러면, 이 코드를 살짝 수정해서 새 버전 프로그램을 만들어 봅시다.

키보드나 마우스를 써서 '커서'를 이 한 줄 가장 왼쪽으로 이동시키고...

```
int ma
{
    printf("안녕하세요!₩n");

    return 0;
}
```

내 첫 C 프로그램

- 그러면, 이 코드를 살짝 수정해서 새 버전 프로그램을 만들어 봅시다.

Shift + 아래 화살표 키를 눌러
이 한 줄 전체를 선택해 주세요.

```
int ma
{
    printf("안녕하세요!₩n");

    return 0;
}
```

내 첫 C 프로그램

- 그러면, 이 코드를 살짝 수정해서 새 버전 프로그램을 만들어 봅시다.

그 다음, Ctrl + C를 누르고...

```
int ma
{
    printf("안녕하세요!₩n");

    return 0;
}
```

내 첫 C 프로그램

- 그러면, 이 코드를 살짝 수정해서 새 버전 프로그램을 만들어 봅시다.

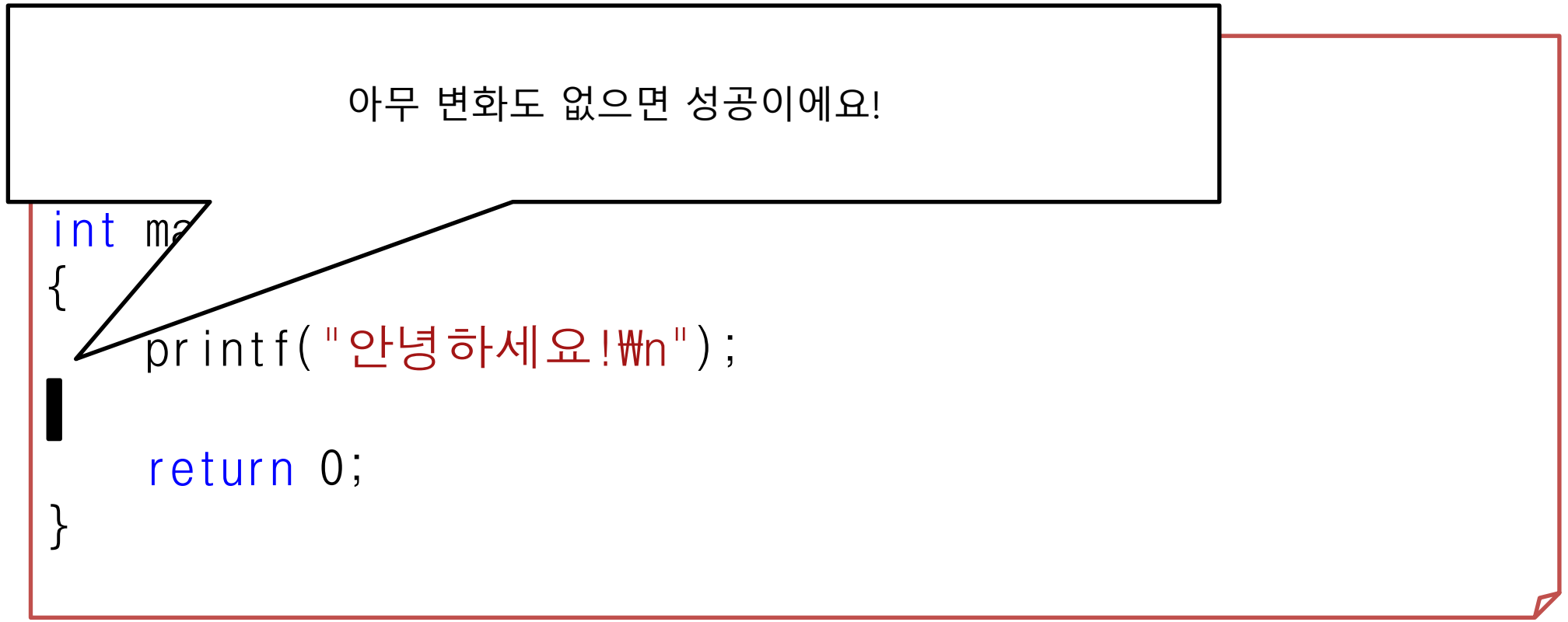
그 다음, Ctrl + C를 누르고...
Ctrl + V를 한 번 눌러보세요.

```
int m
{
    printf("안녕하세요!₩n");

    return 0;
}
```

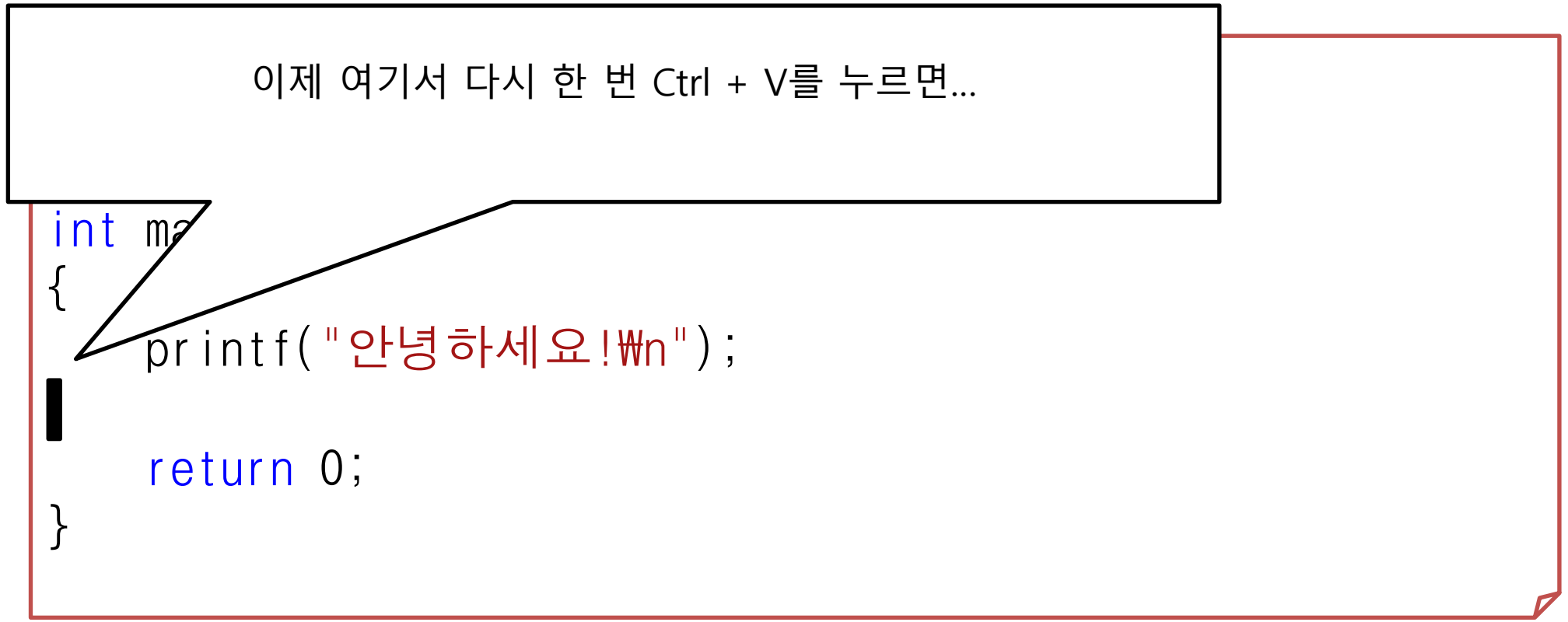
내 첫 C 프로그램

- 그러면, 이 코드를 살짝 수정해서 새 버전 프로그램을 만들어 봅시다.



내 첫 C 프로그램

- 그러면, 이 코드를 살짝 수정해서 새 버전 프로그램을 만들어 봅시다.



내 첫 C 프로그램

- 그러면, 이 코드를 살짝 수정해서 새 버전 프로그램을 만들어 봅시다.

이제 여기서 다시 한 번 Ctrl + V를 누르면...
방금 선택한 그 한 줄을 다시 한 번 적을 수 있어요!

```
int mai
{
    printf("안녕하세요!₩n");
    printf("안녕하세요!₩n");

    return 0;
}
```

내 첫 C 프로그램

- 그러면, 이 코드를 살짝 수정해서 새 버전 프로그램을 만들어 봅시다.

```c
#include <stdio.h>

int main()
{
    printf("안녕하세요!₩n");
    printf("안녕하지 않으세요?₩n");

    return 0;
}
```

새로 적은 줄에 있는 큰따옴표 안 내용을 살짝 고친 다음
다시 한 번 저장-실행해 보세요.

이번에도 역시나, 예상한 대로 출력되면 성공이에요!

'복붙'

- (중요)방금 수행한 작업을 우리 수업에서는 '복붙'이라고 부를게요.
  - 이미 잘 알고 있는 내용을 연거푸 적어야 할 때,
    또는, 여기 말고 다른 곳에 옮겨 적어야 할 때,
    복붙을 사용하면 조금 더 안전하게 진행할 수 있어요

  - 우리 수업을 같이 진행할 때,
    수업자료 .pdf 파일을 다운로드한 다음
    (제가 PowerPoint로 슬라이드를 만든 만큼) Edge 브라우저로 .pdf 파일을 열면
    슬라이드에 적혀 있는 코드를 깔끔하게 복붙해올 수 있을 거예요

  - 그러니, 자주 사용해 주세요

내 첫 C 프로그램

- 다시 코드로 돌아와서…

```c
#include <stdio.h>

int main()
{
    printf("안녕하세요!₩n");
    printf("안녕하지 않으세요?₩n");

    return 0;
}
```

내 첫 C 프로그램

- 다시 한 번 코드를 고쳐서,
  이번에는 '3과 5를 더한 결과를 출력하는 프로그램'을 만들어 봅시다.

```
#include <stdio.h>

int main()
{
    printf("안녕하세요!₩n");
    printf("안녕하지 않으세요?₩n");

    return 0;
}
```

내 첫 C 프로그램

- 아쉽게도 여기는 Python이 아니라서,
  아래처럼 코드를 작성할 수는 없어요. 대신...

```
#include <stdio.h>

int main()
{
    printf(3 + 5);

    return 0;
}
```

내 첫 C 프로그램

- 대신, 큰따옴표 안에 '여기에 덧셈 결과를 담아 출력해줘'와 같은 느낌으로 출력용 '양식'을 지정해 둘 수 있어요.

```c
#include <stdio.h>

int main()
{
    printf("덧셈 결과는 %d입니다.₩n", 3 + 5);

    return 0;
}
```

복붙해서 내 .c 파일에 옮겨 적고,
저장-실행해서 출력 결과를 확인해 봐요.

내 첫 C 프로그램

- 대신, 큰따옴표 안에 '여기에 덧셈 결과를 담아 출력해줘'와 같은 느낌으로 출력용 '양식'을 지정해 둘 수 있어요.

```
#include <stdio.h

int main()
{
    printf("덧셈 결과는 %d입니다.₩n", 3 + 5);

    return 0;
}
```

양식에 %d를 적어 두면...

내 첫 C 프로그램

- 대신, 큰따옴표 안에 '여기에 덧셈 결과를 담아 출력해줘'와 같은 느낌으로 출력용 '양식'을 지정해 둘 수 있어요.

```
#include <stdio.h

int main()
{
    printf("덧셈 결과는 %d입니다.₩n", 3 + 5);

    return 0;
}
```

양식에 %d를 적어 두면...
양식 적어 둔 곳 '뒤'에 , 찍고 적어 둔 내용을 토대로
숫자 **값**의 10진수(decimal) 표현을 %d 자리에 담아 출력해 줘요.

내 첫 C 프로그램

- 오... 처음으로 굵은 글씨 단어가 튀어나왔어요.
- 일단 이 정도만 구경해 두고, 잠시 이론 이야기로 넘어가 볼께요.
  - 납득이 안 된다 싶을 때는 꼭 손을 들어 물어봐주세요

프로그램

- 방금 전까지 같이 진행해 보면서 우리는 몇 가지 스킬을 습득했어요:
  - 글자들을 출력하고 싶을 때 어떻게 적을 수 있나
  - 숫자 **값**에 대한 10진수 표현을 출력에 섞고 싶을 때 양식을 어떻게 꾸밀 수 있나

프로그램

- 방금 전까지 같이 진행해 보면서 우리는 몇 가지 스킬을 습득했어요:
  - 글자들을 출력하고 싶을 때 어떻게 적을 수 있나
  - 숫자 **값**에 대한 10진수 표현을 출력에 섞고 싶을 때 양식을 어떻게 꾸밀 수 있나

- 지금 가진 스킬만으로도 꽤나 다양한 프로그램을 만들 수 있을 거예요.
  - 실행하면 프랑스어 인사를 하는 프로그램
  - 3과 5를 곱한 결과를 출력하는 프로그램

프로그램

- 지금 여러분이 보고 있는 이 수업자료도,
  PowerPoint, PDF reader같은 프로그램이 글자들을 출력해 주고 있습니다.
  - 뭐 당연하겠지만 얘네는 우리 것 보다는 많이 복잡할 거예요

프로그램

- 지금 여러분이 보고 있는 이 수업자료도,
  PowerPoint, PDF reader같은 프로그램이 글자들을 출력해 주고 있습니다.
  - 뭐 당연하겠지만 얘네는 우리 것 보다는 많이 복잡할 거예요

- 그럼에도 불구하고,
  모든 프로그램들은 사실 동일한 재료를 써서 만들었다고 볼 수 있습니다.
  - PowerPoint는 우리 프로그램보다 훨씬 크긴 하지만 아무튼 그래요

- 좀 더 이론적으로 접근하면,
  우리가 봤던, 우리가 만들 모든 프로그램들은
  오늘 우리가 살펴 보는 두 가지 영역의 조합이라고 말할 수 있어요.

프로그램

- 프로그램 = Data + Code

프로그램

- 프로그램 = Data + Code
  - Data

프로그램

- 프로그램 = Data + Code
  - Data: 여러분이 떠올릴 수 있는 바로 그것들
    - 그냥 숫자
    - 한국어 인사말
    - 웹 브라우저 주소창에 치는 것
    - ID와 비밀번호
    - 오늘 밤에 게시될 우리 수업 영상

프로그램

- 방금 전에 우리는, main.c의 내용을 조금씩 고쳐 가며
  프로그램이 이전과는 다른 Data를 사용하여 실행되도록 만들 수 있었어요.
  - 아직 우리는 '우리가 직접 정한' '글자들'이나 '숫자 **값**'만 다루어 봤지만,
    앞으로 더 다양한 Data를 더 다양한 방식으로 다룰 수 있게 될 거예요

프로그램

- 프로그램 = Data + Code
  - Code

프로그램

- 프로그램 = Data + Code
  - Code: **실행**(execute)되는 것
    - 검은 창에 내가 원하는 것을 출력하기 위한 printf()
    - printf()를 사용하도록 우리가 구성해 둔 그 한 줄…

프로그램

- 프로그램 = Data + Code
  - Code: **실행**(execute)되는 것
    - 검은 창에 내가 원하는 것을 출력하기 위한 printf()
    - printf()를 사용하도록 우리가 구성해 둔 그 한 줄…을 **문장**(statement)이라 불러요
    - 잠시 코드로 돌아가서 구경해 보고 올께요

프로그램

- 방금 적어 본 C 코드예요:

```c
#include <stdio.h>

int main()
{
    printf("덧셈 결과는 %d입니다.₩n", 3 + 5);

    return 0;
}
```

프로그램

- 방금 적어 본 C 코드예요:

```c
#include <stdio.h

int main()
{
    printf("덧셈 결과는 %d입니다.₩n", 3 + 5);

    return 0;
}
```

이 한 줄은 C **문장**이에요.
프로그램 실행(run) 도중 이 **문장**을 **실행**(execute)하면
'덧셈 결과는 8입니다'라고 출력하게 돼요.

프로그램

- 방금 적어 본 C 코드예요:

```c
#include <stdio.h

int main()
{
    printf("덧셈 결과는 %d입니다.₩n", 3 + 5);
    printf("곱셈 결과는 %d입니다.₩n", 3 * 5);
    return 0;
}
```

지금보다 더 복잡한 프로그램을 만들 때는,
지금보다 더 많은, 더 다양한 **문장**들을 적게 될 거예요!

프로그램

- 프로그램 = Data + Code
  - Code: **실행**(execute)되는 것
    - 검은 창에 내가 원하는 것을 출력하기 위한 printf()
    - printf()를 사용하도록 우리가 구성해 둔 **문장**(statement)
      - 살짝 중요한 이름이니 다음 슬라이드에서 정리해 볼께요

프로그램

- **문장**(statement)
  - **실행**(execute)의 대상이 되는 친구예요
    - 무언가가 **문장**이 아니라면 그건 기본적으로 **실행**의 대상이 아니에요
  - 방금 이리저리 고치고 복붙하고 했던 한 줄 한 줄을 C **문장**이라 부를 수 있어요
    - 한 **문장**을 여러 줄에 걸쳐 적는 것도 가능해요
  - 프로그램의 Code 부분을 구성하는 기본 단위라 볼 수 있어요
  - 이 두 단어는 매우 중요하므로,
    '**문장**은 **실행**되는 것' 정도로 묶어 외워 두면 좋을 것 같아요

프로그램

- 프로그램 = Data + Code
  - Code: **실행**(execute)되는 것
    - 검은 창에 내가 원하는 것을 출력하기 위한 printf()
    - printf()를 사용하도록 우리가 구성해 둔 **문장**(statement)

프로그램

- 프로그램 = Data + Code
  - Code: **실행**(execute)되는 것
    - 검은 창에 내가 원하는 것을 출력하기 위한 printf()
    - printf()를 사용하도록 우리가 구성해 둔 **문장**(statement)

    - 아직 지금 시점에는 **실행**이 정확히 무엇인지는 몰라도 괜찮아요
      - '**문장** 가지고 뭔가 하는 것' 정도로만 두어도 무방해요

    - 앞으로 다양한 **문장**들을 적어 보면서
      지금보다 더 다채로운 **실행**을 야기할 수 있을 거예요
      - Code가 더 복잡한 프로그램을 만들 수 있을 거예요

프로그램

- Code 부분에 대한 설명은 이 정도 해 두면 끝날 것 같아요.

- 이제 main.c를 조금 더 고쳐가면서,
  Data 부분 이야기를 좀 더 구경해 봅시다.

사용자가 입력한 수에 3을 더해 출력하는 프로그램

- 이번에는 조금 복잡한 프로그램을 만들어 봅시다.

- 프로그램을 실행하면...
  - 사용자에게 수 하나를 요청하고(요청하는 메시지를 출력하고)
  - 키보드로 숫자 **값** 하나를 입력받은 다음
  - 그 **값**에 3을 더한 **값**이 얼마인지 출력해요

- 이번에는 강사와 함께 천천히 진행해 봅시다.

복습을 위한 슬라이드

- scanf()
  - 사용자(의 키보드)를 통해 Data를 가져오기 위해 사용할 수 있어요
  - 얘도 우리가 '양식'을 적어 둠으로써
    사용자 입력을 어떻게 간주하여 가져올지 정할 수 있어요
    - printf()와 scanf()의 'f'가 사실 formatted의 약자예요
  - 조금 이상하긴 하지만,
    숫자 **값**을 가져오고 싶을 때는 &(ampersand)을 곁들여 적어야 하는 것 같아요

사용자가 입력한 수에 3을 더해 출력하는 프로그램

- 이전까지 우리는 '미리 정해 둔 Data'만 사용해서 프로그램을 구성했는데,
  이번에 처음으로 '실행 도중' 프로그램 바깥에서 Data를 가져올 수 있었어요.
  - 가져온 Data를 일단 number에 담아 두었다가,
    그 다음 **문장**에서 3을 더해 출력해 보았어요

사용자가 입력한 수에 3을 더해 출력하는 프로그램

- 이전까지 우리는 '미리 정해 둔 Data'만 사용해서 프로그램을 구성했는데,
  이번에 처음으로 '실행 도중' 프로그램 바깥에서 Data를 가져올 수 있었어요.
  - 가져온 Data를 일단 number에 담아 두었다가,
    그 다음 **문장**에서 3을 더해 출력해 보았어요

- 여기까지 오면 이제 프로그램의 Data 부분을 정리해볼 수 있을 것 같아요.

프로그램

- 프로그램 = Data + Code
  - Data: 여러분이 떠올릴 수 있는 바로 그것들
    - 그냥 숫자
    - 한국어 인사말
    - 웹 브라우저 주소창에 치는 것
    - ID와 비밀번호
    - 오늘 밤에 게시될 우리 수업 영상

프로그램

- 프로그램 = Data + Code
  - Data: 여러분이 떠올릴 수 있는 바로 그것들

  - 어떤 Data는 프로그래머가 미리 정해 둬요.
    어떤 Data는 프로그램 실행 도중 바깥에서 가져와요.
    어떤 Data는 프로그램 실행 도중 바깥으로 내보내요('검은 창에' 출력 등)

  - 필요하다면, 나중에 사용하기 위해 Data를 담아 둘 수도 있어요
    - 이 때 우리는 **변수**(variable)를 사용할 수 있어요

프로그램

- **변수**(variable)

프로그램

- **변수**(variable)
  - 어쩌면 수학 시간에 이미 들어본 적이 있을 거예요

  - 그 수학 분야에서의 변수가 정확히 무엇인지는 설명하기 어려울 것 같아요.
    다행히도, 프로그래밍 분야에서 말하는 (굵은 글씨) **변수**는 쉬워요!

프로그램

- **변수**(variable)
  - '**값**(value)을 하나 담을 수 있는 친구'를 보통 **변수**라고 말해요
    - 방금 전 프로그램에서는 사용자가 입력한 숫자 **값**을 `number`에 담을 수 있었어요

프로그램

- **변수**(variable)
  - '**값**(value)을 하나 담을 수 있는 친구'를 보통 **변수**라고 말해요

  - 앞에서 본 **문장**과 달리 **변수**는 동네에 따라 조금 추상적일 수 있어요
    - C에서는 '얘는 변수야 / 아니야'라고 딱 잘라 말할 수 있어요
    - Python의 경우 조금 오묘해요

  - 아무튼 C에서는, 새 **변수**를 내 프로그램에 도입해서 사용하고 싶을 때 미리 적당한 곳에 선언을 적어 두어야 해요
    - 선언 이야기는 내일 좀 더 진득하게 다루어 볼께요

프로그램

- **변수**(variable)
  - '**값**(value)을 하나 담을 수 있는 친구'를 보통 **변수**라고 말해요
  - C에서는, 새 **변수**를 내 프로그램에 도입해서 사용하고 싶을 때 미리 적당한 곳에 선언을 적어 두어야 해요
  - (중요)어떤 **변수**에 담겨 있는 **값**을 가져와 사용하고 싶을 때, 우리는 그 **변수** 이름을 적어 가며 **문장**을 구성해야 했어요
    - 나중에 **값**을 여럿 담기 위해 **변수** 여러 개를 도입해 사용할 때도, 그 때 그 때 무슨 **값**이 필요한지 잘 확인한 다음 그 **값**이 담겨 있을 **변수** 이름을 잘 골라 적어 주어야 할 거예요

프로그램

- 프로그램 = Data + Code
  - Data: 여러분이 떠올릴 수 있는 바로 그것들

  - 어떤 Data는 프로그래머가 미리 정해 둬요.
    어떤 Data는 프로그램 실행 도중 바깥에서 가져와요.
    어떤 Data는 프로그램 실행 도중 바깥으로 내보내요('검은 창에' 출력 등)

  - 필요하다면, 나중에 사용하기 위해 Data를 담아 둘 수도 있어요
    - 이 때 우리는 **변수**(variable)를 사용할 수 있어요

프로그램

- 프로그램 = Data + Code
  - Data: 여러분이 떠올릴 수 있는 바로 그것들

  - 우리 프로그램은 실행 도중
    다양한 Data들을 여기저기서 가져와서,
    덧셈 등을 통해 새로운 Data를 만들고,
    그 결과를 어딘가에 담아 가면서 목표를 달성하게 돼요
    - '출력'도 '검은 창에 담기'라 볼 수 있어요

프로그램

- 프로그램 = Data + Code
  - 여기까지 오면 이제 프로그램을 구성하는 두 부분에 대해 어느 정도 둘러본 셈이 돼요

  - 그런데, 아직 남은 부분이 있어요

프로그램

- 프로그램 = Data + Code
  - +

프로그램

- 프로그램 = Data + Code
  - +: ?
  - 잠시 코드로 돌아가 봅시다

프로그램

- 아까 적어 본 코드의 일부예요:

```
printf("덧셈 결과는 %d입니다.₩n", number + 3);
```

프로그램

- 아까 적어 본 코드의 일부예요:

재탕해서 잘 써먹은 유효한 C **문장**이에요.

```
printf("덧셈 결과는 %d입니다.₩n", number + 3);
```

프로그램

- 아까 적어 본 코드의 일부예요:

재탕해서 잘 써먹은 유효한 C **문장**이에요.
사용자가 프로그램 실행 도중 5를 입력할 예정이라 가정했을 때,
'왜 이 **문장**을 **실행**하면 8이 출력되나요?'라고 물어보면
뭐라고 답할 수 있을까요?

```
printf("덧셈 결과는 %d입니다.₩n", number + 3);
```

프로그램

- 아까 적어 본 코드의 일부예요:

맞아요. 이 **문장**을 구성하는 이 부분이 이렇게 적혀 있고,

```
printf("덧셈 결과는 %d입니다.₩n", number + 3);
```

프로그램

- 아까 적어 본 코드의 일부예요:

맞아요. 이 **문장**을 구성하는 이 부분이 이렇게 적혀 있고...

```
printf("덧셈 결과는 %d입니다.₩n", number + 3);
```

프로그램

- 아까 적어 본 코드의 일부예요:

맞아요. 이 **문장**을 구성하는 이 부분이 이렇게 적혀 있고...
이전 **문장**에서 **변수** number 자리에 5를 담아 두었기 때문이라고 말하면
딱 알맞은 답이 될 것 같아요.

```
printf("덧셈 결과는 %d입니다.₩n", number + 3);
```

프로그램

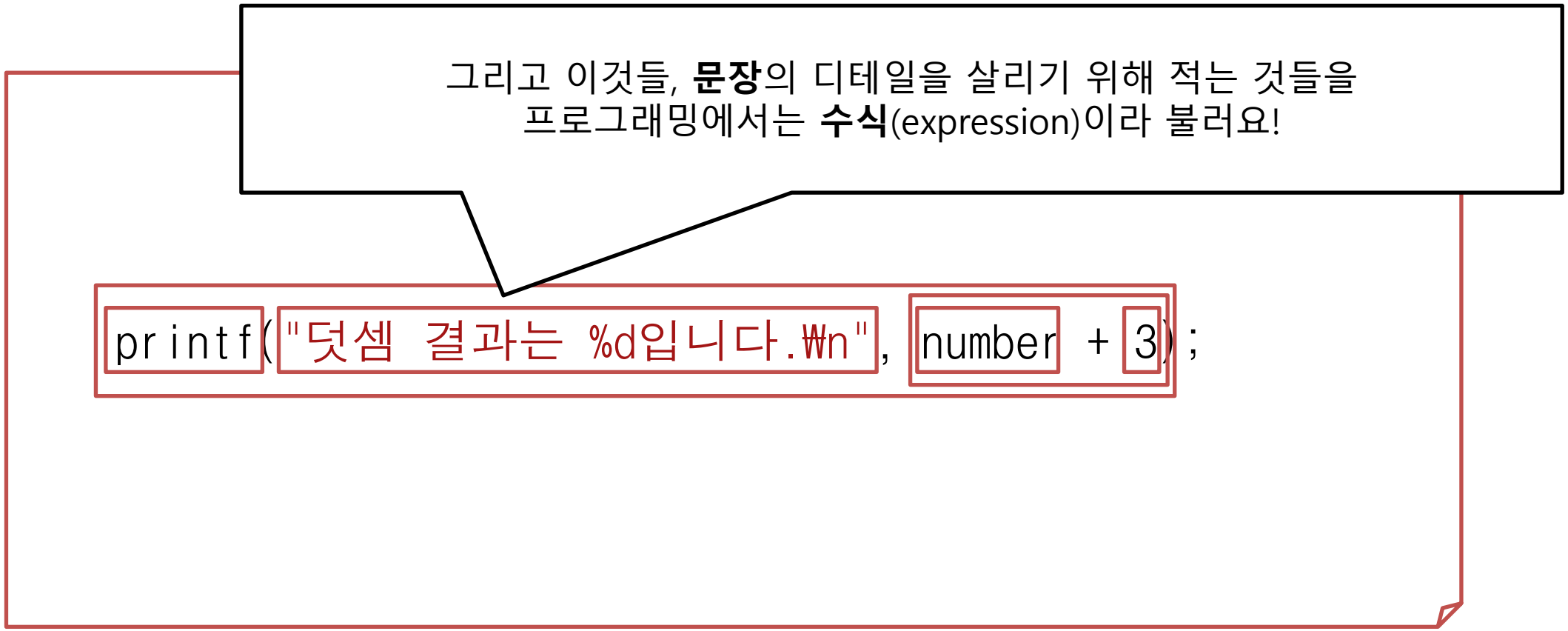
- 아까 적어 본 코드의 일부예요:

조금 더 세밀하게 본다면, 이 짧은 **문장**에는
각각 적당한 곳에 적혀 있음으로써 이 **문장**의 **실행**을 커스터마이즈하는
다양한 부분들이 포함되어 있어요.
(저 부분을 바꾸면 **실행** 양상이 어떻게 달라질지 상상해 봐요)

```
printf("덧셈 결과는 %d입니다.₩n", number + 3);
```

프로그램

- 아까 적어 본 코드의 일부예요:



프로그램

- **수식**(expression, '표현식'이라고도 불러요)
  - 이 단어는 수학에서 쓰는 의미랑 매우 비슷하게 쓰여요.
    역시나, 복잡하게 생각하지는 않아도 괜찮아요

프로그램

- **수식**(expression, '표현식'이라고도 불러요)
  - 프로그래밍에서 **수식**은,
    '**계산**(evaluate: **계산**하다)하면 결과**값**이 하나 나오는 친구'예요
    - **수식** 3을 계산하면, **값** 3이 나와요
    - **수식** number를 계산하면 number에 담겨 있는 **값**이 나와요
    - + **연산자**(operator)로 묶은 큰 **수식** 3 + 5를 계산하면, **값** 8이 나와요
    - 이건 사실 몹시 중요하기 때문에
      '**수식**을 **계산**한 결과**값**' 아홉 글자로 딱 외워 두면 좋을 것 같아요

프로그램

- **수식**(expression, '표현식'이라고도 불러요)
  - 프로그래밍에서 **수식**은,
    '**계산**(evaluate)하면 결과**값**이 하나 나오는 친구'예요

  - C에서 모든 **수식**은,
    계산했을 때 무슨 **형식**(type) **값**이 나오는지가 딱 정해져 있어요
    - **수식** 3을 **계산**하면 (선언 적을 때 슬쩍 본) int **형식 값** 3이 나와요

    - **수식** 3.5를 **계산**하면 (아직은 써본 적 없는) double **형식 값** 3.5가 나와요

프로그램

- **수식**(expression, '표현식'이라고도 불러요)
  - 프로그래밍에서 **수식**은,
    '**계산**(evaluate)하면 결과**값**이 하나 나오는 친구'예요

  - C에서 모든 **수식**은,
    계산했을 때 무슨 **형식**(type) **값**이 나오는지가 딱 정해져 있어요
    - 다시 말하면, C에서는 '**수식**의 **형식**'을 중요하게 다루어요
      - 그렇다 보니 VS는 우리가 적어 둔 **수식**에 마우스 포인터를 갖다 대면
        그 수식이 어떤 **형식 수식**인지 파악해서 툴팁으로 보여줘요

프로그램

- **수식**(expression, '표현식'이라고도 불러요)
  - 프로그래밍에서 **수식**은,
    '**계산**(evaluate)하면 결과**값**이 하나 나오는 친구'예요
  - C에서 '**수식**의 **형식**'은 매우 중요해요
  - 우리는 **문장**의 **실행**을 커스터마이즈하기 위해 그 **문장** 안에 **수식**을 적을 수 있어요
    - **수식**이 **문장**에 포함되어 있다면 그 **수식**의 **계산**은 그 **문장**을 **실행**할 때 수행돼요
    - 이렇게 놓고 본다면 **수식**은 **문장**보다 작은 개념이라 말할 수 있을 것 같아요

프로그램

- 프로그램 = Data + Code
  - +: ?

프로그램

- 프로그램 = Data + Code
  - +: **수식**
    - 어떤 Code를 **실행**할 때 어떤 Data를 사용할 것인지를 특정(specify)하기 위해 적게 돼요
    - Data를 보다 복잡하게 다루어야 한다면, **연산자**를 적어 가며 좀 더 큰 **수식**을 구성하게 돼요

프로그램

- 프로그램 = Data + Code
  - +: **수식**
    - 어떤 Code를 **실행**할 때 어떤 Data를 사용할 것인지를 특정(specify)하기 위해 적을 수 있어요
      - Data를 가져오기
      - Data를 어딘가에 담기
    - Data를 보다 복잡하게 다루어야 한다면, **연산자**를 적어 가며 좀 더 큰 **수식**을 구성하게 돼요
    - 그 **수식**의 **계산**은, 그 **문장**을 **실행**할 때 수행돼요
      - Data가 '언제' **변수**에 담기는지를 묻는다면,
        그 **변수**에 Data를 담는 **문장**이 main.c의 어디에 적혀 있는지 찾으면 돼요

정리

- 여기까지,
  프로그램을 구성하는 가장 기본적인 요소들을 주욱 구경해 보았어요.

- 굵은 글씨 단어들은 앞으로도 꾸준히 등장할 예정이니,
  일단 지금은 '들어는 보았다' 정도로 두고 넘어가도 좋을 것 같아요.

마무리

- 여기까지,
  프로그램을 구성하는 가장 기본적인 요소들을 주욱 구경해 보았어요.

- 굵은 글씨 단어들은 앞으로도 꾸준히 등장할 예정이니,
  일단 지금은 '들어는 보았다' 정도로 두고 넘어가도 좋을 것 같아요.

- 잠시 쉬었다가 오늘의 실습 목표로 넘어가 보도록 합시다.