C++ Features and Data Types

프로그래밍 입문(2)

Topics

- **C++ 프로그램의 기초적인 문법 및 형태**
    - **main()함수, 전처리 지시자, iostream, 문장(statement), 코드 블럭(code block), 주석문(comment) 등**
- C++의 기본 자료형 (Primitive Data Types)
    - 정수(integer), 실수(floating point), 문자(character), 불리언(boolean), void
- C++의 다양한 연산자 (Operators)
    - 대입(assignment), 산술(arithmetic), 관계/비교(relation/comparison), 논리(logic), 비트(bitwise), 복합 대입(compound assignment) 연산자 등.

Hello World

```cpp
#include<iostream>
using namespace std;

int main() {
    cout << "Hello World!\n";
    return 0;
}
```

- 간단한 코드이지만 C++ 문법으로 작성된 코드의 전형적인 모습을 확인할 수 있습니다.
- C의 문법과 거의 동일.
- 이 코드는 크게 main() 함수와 다른 부분들로 구성.
- 그 밖의 다양한 문장부호가 코드에서 사용되고 있습니다.

Hello World

preprocessor directives

using directives

```
#include<iostream>
using namespace std;

int main() {
    cout << "Hello World!\n";
    return 0;
}
```

main function

여러 문장들은 중괄호(curly bracket, {})로 묶임

각각의 문장(statement)들은 semi-colon(;)으로 끝남

전처리 지시자
(Preprocessor Directives)

```
#include<iostream>
```

- 전처리 지시자, 전처리 명령어, 전처리문 등으로 불림.

- 컴파일이 실행되기 전 전처리기(preprocessor)에 의해 실행되는 부분.

- #으로 시작하고 줄바꿈으로 끝남.

- include, define 등 다양한 것들이 있지만 우선 include를 주로 쓰게 됨.

#include

- 컴파일 하기 전, include 나오는 파일을 불러와서 대입하라는 의미.
- #include<iostream>: <> 안의 것은 컴파일러와 함께 제공되는 헤더(header) 파일을 부를 때 사용됨.
- #include “my/data_loader.h”: 특정 경로의 파일을 읽어 올 때.
- 헤더 파일에는 다른 파일들(cpp)에서 볼 수 있는 함수들의 선언문이 포함되어 있음 → 이를 이용해 파일의 이미 구현된 함수들을 다른 파일에서 불러와 사용가능.

#include

- Software 개발에서 가장 중요한 것 중 하나가 재사용(Reuse).
- **DRY**: **D**o not **R**epeat **Y**ourself!
- DB에서 상품정보를 읽고, 쇼핑몰 화면처럼 보여주는 프로그램을 만들었음.
- 만약 같은 상품정보를 가져와 엑셀파일로 저장하는 프로그램을 또 만들어야 한다면?
- 정보를 읽어오는 부분을 따로 하나의 파일로 작성하여 두 프로그램에서 재사용하는 것이 일을 줄일 수 있습니다.

iostream

```
#include<iostream>
```

```
cout << "Hello World!\n";
```

- C의 stdio.h와 유사하다고 생각하면됩니다.

- 실습 등에서 주로 사용하게 되는 것은 `cout`과 `cin`.

- `cout`: console output으로 화면에 무엇인가를 출력해주는 명령. (C: printf)

- cin: console input으로 무엇인가 입력을 받는 명령. (C: scanf)

- 꺽쇠 두 개 (<< 또는 >>)를 사용하여 데이터의 흐름을 표시.

using 지시자
(using directives)

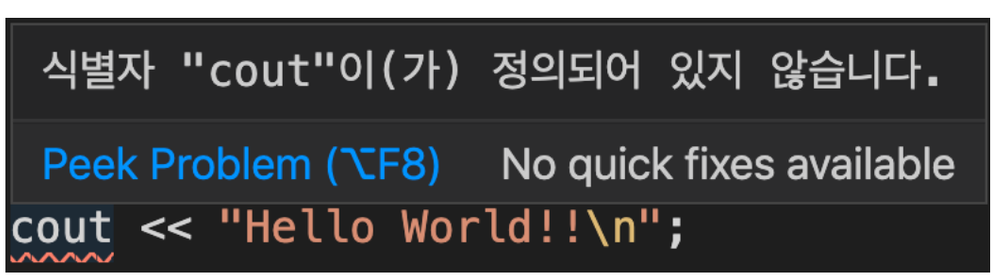
```
using namespace std;
```

- 특정 namespace의 모든 것을 사용하겠다는 의미.

- 코드에서 어떤 이름이 나왔을 때, 그 파일에 정의되지 않았으면 `std` namespace에서 이름을 찾게 됨.

- 출력을 위해 사용한 `cout`이 우리 프로그램에 없지만 사용가능한 이유.

- 만약 using 지시자가 없으면 cout을 사용할 때 다음과 같은 에러가 나게 됨.



using 지시자
(using directives)

```
using std::cout;
```

```
std::cout << "Hello World!!\n";
```

- 같은 문제를 해결하기 위해 위의 두 가지 방식 또한 사용할 수 있습니다.

- 첫번째는 `std` namespace에서 명시적으로 `cout`만을 사용하겠다고 선언하는 방법.

- 두번째는 `cout`을 사용할 때 항상 `std::cout`으로 표시하는 방법.

- 사실 더 나은 방법은 이 두 가지 방법입니다.

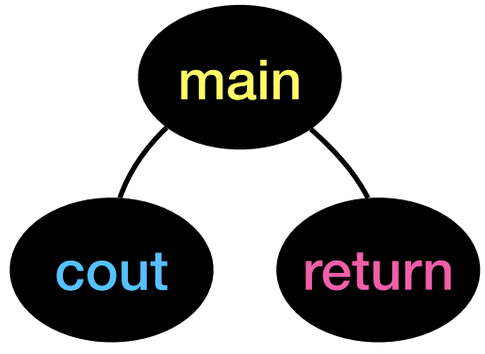
- 전체 namespace를 다 가져오는 것은 비효율적이며 어떤 이름 충돌이 일어날지 추적하기 어렵게 됩니다.

main()

- C++의 모든 프로그램은 진입지점(entry point)이 필요합니다.
- 100개의 cpp파일로 된 프로그램을 컴파일 하여 실행파일로 만들 경우 → 프로그램이 어디서부터 시작해야 하는지?
- 모든 C++ 프로그램의 시작지점은 main함수.
- 라이브러리의 경우 main()이 존재하지 않을 수 있습니다.

main()

```cpp
int main() {
    cout << "Hello World!\n";
    return 0;
}
```



- main(): 반드시 소문자로, 정확히 main이라는 이름으로 지정.
- 중괄호{ }로 둘러싸인 부분이 함수에 해당하는 코드.
- 그 아래에는 세미콜론;으로 끝나는 두 문장(statement)이 있음.
- 프로그램이 실행되면 main() 아래의 문장들이 순차적으로 실행됨.

Statement

- 프로그램 실행의 기본 단위.
- 극단적으로 얘기하면 프로그램은 a sequence of statements.
- 실제로는 보통 각각의 함수가 순차적 문장들이고, 이런 함수들이 연속해서 실행되는 것이 프로그램이 됩니다.
- C++에서 문장들은 줄바꿈이 아닌 세미 콜론;으로 구분됩니다.

Statement

- `cout << “Hello World\n”;` → **OK**

- `cout <<`
  `“Hello World\n”;` → **OK**

- `cout`
  `<<`
  `“Hello World\n”`
  `;` → **OK**

- `cout <<`
  `“Hello`
  `World\n”;` → **Not OK**

- `cout << “Hello ”; cout << “World\n”; return 0;` → **OK**

Back To main()

```cpp
int main() {
    cout << "Hello World!\n";
    return 0;
}
```

- 세미콜론으로 구분되는 문장들을 왜 위의 형식으로 작성하는가?
- 들여쓰기(indentation)는 코드의 가독성을 높이기 위함.
- main() 내부에 있는 문장이라는 의미로 탭(tab) 한 번으로 - 또는 space 여러 번으로 - 들여쓰기를 하여 문장을 작성.
- 보통 한 줄에 한 문장만, 길어지는 경우 여러 줄에 한 문장이 올 수 있음.
- 그 밖에도 다양한 코드 스타일(Code Style) 규칙에 따라야 할 수 있음.

Code Style

- 여러 사람이 코드를 봐야하는 경우 code style을 일치시키는 것이 가독성 향상을 위해 매우 중요함.

- 정해진 Code Style을 따르지 않는 것 → 알아보기 힘든 손글씨

- 파일 이름을 정하는 것에서부터 각각의 문장이나 들여쓰기 규칙까지 매우 광범위한 부분에 대해 정의됨.

- Code Style Guideline을 Coding Convention이라고도 함.

- https://google.github.io/styleguide/cppguide.html

Unresolved Debates

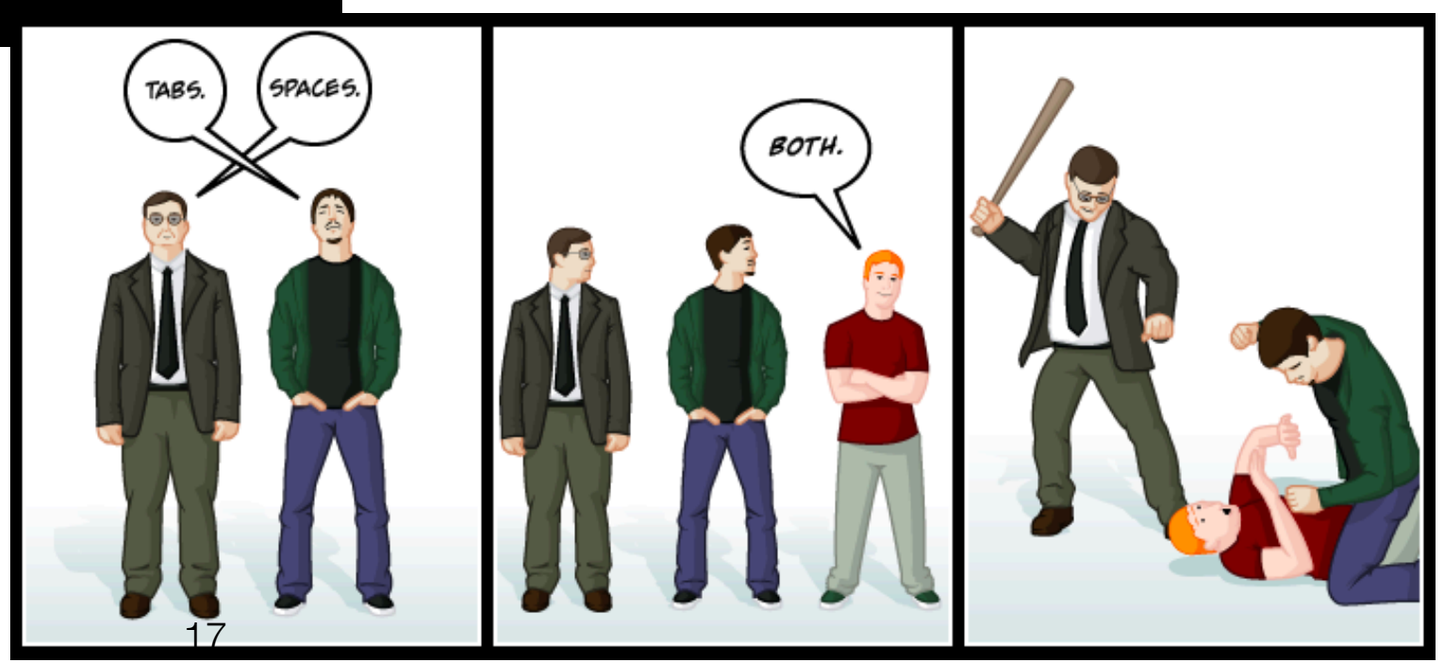
There are two types of people:

```
if (Condition) {
  Statement
  /* ....
   */
}
```

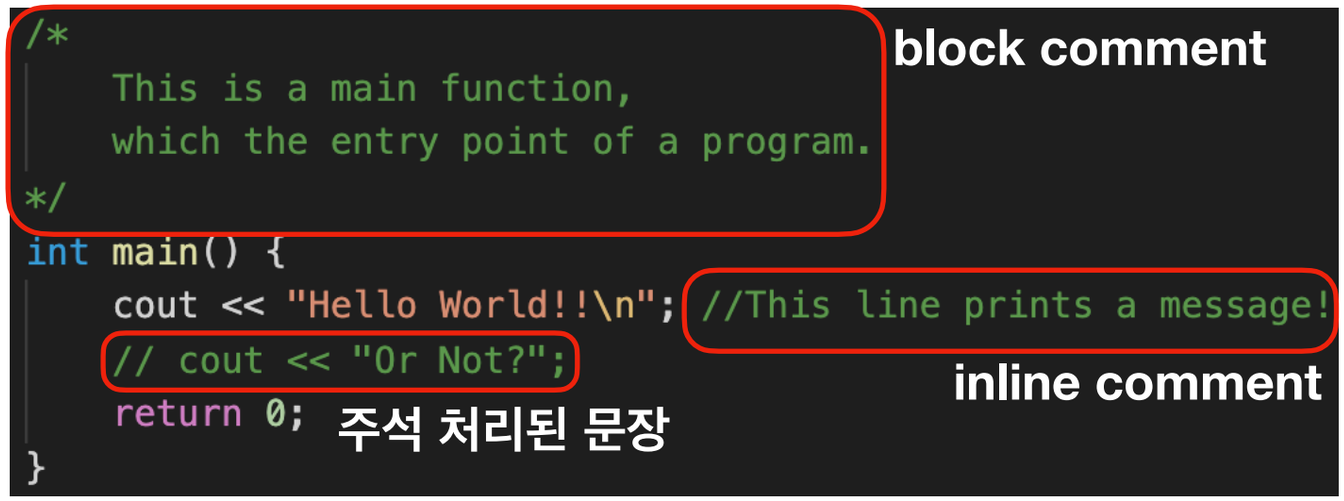
```
if (Condition)
{
  Statement
  /* ....
   */
}
```

Curly Brackets:
Are you left or right?

Tabs vs. Spaces:
Are you a tab guy or a space guy?



Comments



- 주석(Comment)에는 Block Comment와 Inline Comment의 두 가지 방식이 있음.
- Block Comment (/* … */): 여러 줄의 주석에 사용.
- Inline Comment (// …): 한 줄의 주석을 문장 끝에 붙일 때 사용.
- Inline comment는 //에서 줄바꿈까지 모든 내용을 주석으로 간주하게 함.
- 이를 이용해 특정부분의 코드를 잠시 무효화할 때, ‘주석 처리’(Toggle Comment)하는 경우가 있습니다.

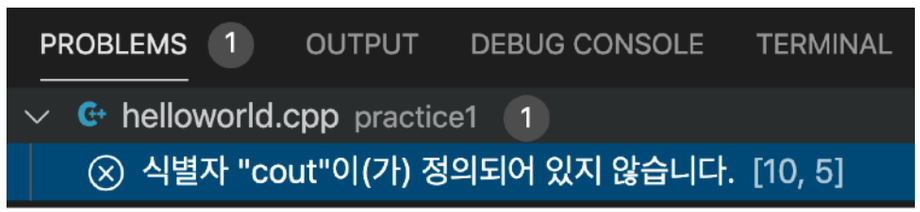
Return Statement

```cpp
int main() {
    cout << "Hello World!\n";
    return 0;
}
```

- 반환문(return statement)는 무엇인가를 반환하는 함수의 마지막 문장.
- 함수에서 반환문 뒤에 다른 문장이 올 경우 컴파일 에러.
- 예시의 경우 실행을 끝내고 숫자 0을 반환.
- main() 함수의 경우 이 값이 error code로 활용됨.
- 0은 정상 종료, 그 외 0이상의 값들은 각각 다른 종류의 에러를 표시.

Compile Errors

```
cout << "Hello World!!\n";
```




```
/Users/jindae/vscode-workspace/cpp-2020/practice1/helloworld.cpp:10:5:
error:
      use of undeclared identifier 'cout'; did you mean 'std::cout'?
    cout << "Hello World!!\n";
    ^~~~
    std::cout
/Library/Developer/CommandLineTools/usr/include/c++/v1/iostream:54:33:
note:
      'std::cout' declared here
extern _LIBCPP_FUNC_VIS ostream cout;
                                ^
1 error generated.
```

- 문법에 맞지 않는 코드나 모호한 표현을 사용한 경우 컴파일 단계에서 에러가 발생함.
- IDE에서는 컴파일 에러를 미리 체크하여 코드 작성 단계에 표시를 해주는 기능이 있습니다.
- 에러 메시지는 에러의 위치, 에러에 대한 설명, 간혹 해결을 위한 힌트를 포함합니다.

Common Mistakes

```
cout << "Hello World!!\n" //No semicolon(;)
return 0;
```

- 문장 끝에 세미콜론을 빠뜨리는 경우.

```
    if(a > 0) {
    }else{
    //missing }
    return 0;
}
```

- 괄호(‘{}’, ‘()’)를 열고 닫는 것이 쌍이 맞지 않는 경우.

- 따옴표가 쌍이 맞지 않는 경우.

```
cout << "Hello World!!\n;
return 0;
```

- 대부분 IDE를 쓰면 편집시점에 확인 가능함.

- 쌍을 맞추는 것은 여는 괄호나 따옴표를 쓰면 자동으로 닫는 괄호와 따옴표가 써지므로 지우지 않도록 주의하면 됨.

Summary

- C++ 프로그램의 기본적인 구조
- C++ 프로그램을 구성하는 요소들
- 컴파일 에러 및 자주하는 실수들