Common Error – Omitting Semicolons

Common error:

Omitting a semicolon (or two), in this case at the end of the `cout` statement

```
#include <iostream>
using namespace std;
int main()
{
   cout << "Hello, World!" << endl_
   return 0;
}
```

Syntax Errors

Without that semicolon you actually wrote:

```
cout << "Hello, World!" << endl return 0;
```

which thoroughly confuses the compiler with the `endl` immediately followed by the `return`!

This is a *compile-time error* or *syntax error*.

A syntax error is a part of a program that does not conform to the rules of the programming language.

Errors: Misspellings

Suppose you (accidentally of course) wrote:

```
cot << "Hello World!" << endl;
```

- This will cause a compile-time error and the compiler will complain that it has no clue what you mean by `cot`.
- The exact wording of the error message is dependent on the compiler, but it might be something like

“Undefined symbol cot” or

"Unknown identifier".

Errors – How Many Errors?

- The compiler will not stop compiling, and will most likely list lots and lots of errors that are caused by the first one it encountered.
- You should fix only those error messages that make sense to you, starting with the first one, and then recompile (after SAVING, of course!).

Making your Program Readable (by Humans)

C++ has *free-form layout*

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

– *will* compile (but is practically impossible to read)

A good program is readable:

– code spaced across multiple lines, one statement per line
– follows indentation conventions, to be explained later.

Logic Errors

Consider this:

```
cout << "Hollo, World!" << endl;
```

- *Logic errors* or *run-time errors* are errors in a program that compiles (the syntax is correct), but executes without performing the intended action.

- *The programmer must thoroughly inspect and test the program to guard against logic errors.*
  - *Testing and repairing a program usually takes more time than writing it in the first place, but is essential !*

Errors: Run-Time Exceptions

Some kinds of run-time errors are so severe that they generate an *exception*: a signal from the processor that aborts the program with an error message.

For example, if your program includes the statement

```
cout << 1 / 0;
```

your program may terminate with a “divide by zero” exception.

Errors: extra or misspelled `main()` function

- Every C++ program must have one and only one **`main`** function.
- Most C++ programs contain other functions besides **`main`** (more about functions later).

Errors: C++ is Case Sensitive

C++

– is *case sensitive.* Typing:

```
int Main()
```

will compile but will not link.

A link-time error occurs here when the linker cannot find the **`main`** function – because you did not define a function named **`main`**. (**`Main`** is fine as a name but it is not the same as **`main`** and there has to be one **`main`** somewhere.)