4 Comments and whitespace

Students:
Section 1.5 is a part of 2 assignments: CSC108 CH0 CA

This assignment is due now but course is closed and activity recorded but will not count towards this assignment (unless the due date is changed). See this article for more info.

Includes: CA
Due: 01/06/2023, 11:59 PM EST

1.5 Errors and warnings

Syntax errors

People make mistakes. Programmers thus make mistakes—lots of them. One kind of mistake, known as a **syntax error**, is to violate a programming language's rules on how symbols can be combined to create a program. An example is forgetting to end a statement with a semicolon.

A compiler generates a message when encountering a syntax error. The following program is missing a semicolon after the first output statement.

Figure 1.5.1: Compiler reporting a syntax error.

```
$ nano -cl -m -w -p -r 72 -x -S -T 4 -Y "/usr/share/nanorc:/usr/share/nano/*.nanorc" -f /home/zyuser/.nanorc -+ -P /home/zyuser/.nanoprofile -A /home/zyuser/.nanoafterlog -B /home/zyuser/.nanobeforelog -l -k -E -o /home/zyuser/ -O /home/zyuser/ -z -c -- /home/zyuser/main.py
GNU nano 5.4                    /home/zyuser/main.py                    Modified
#!/usr/bin/python3
using namespace std;

int Main() {
    cout << "Hello, world!" << endl;
    cout << "Welcome to CSC108!" << endl;
    return 0;
}

main.py:3: error: syntax error, expected '(' after expression
  cout << "Hello, world!"
                         ^
^G Help    ^O Write Out   ^W Where Is    ^K Cut        ^T Execute    ^C Cursor
^X Exit    ^R Read File   ^\ Replace     ^U Paste      ^J Justify    ^/ Go To Line
```

Feedback?

Above, the 6 refers to the 6th line in the code, and the 27 refers to the 27th column in that line.

INSPECTION ACTIVITY 1.5.1. Syntax errors.

Find the syntax errors. Assume variable numDogs has been declared.

1) cout << numDogs

- Error
- No error

2) cout << "Foggy" << numDogs

- Error
- No error

3) cout << "Everyone wins."

- Error
- No error

4) cout << "Hello friends!" << endl;

- Error
- No error

5) cout << "Amy // Michael" << endl;

- Error
- No error

6) cout << numDogs << endl;

- Error
- No error

7) int numDogs
cout << "Foggy"
cout << numDogs << endl;

- Error
- No error

8) cout << numDogs << endl;

- Error
- No error

Feedback?

Unclear error messages

Compiler error messages are often unclear or even misleading. The message is like the complier's "best guess" of what is really wrong.

Figure 1.5.2: Misleading compiler error message.

```
$ nano -cl -m -w -p -r 72 -x -S -T 4 -Y "/usr/share/nanorc:/usr/share/nano/*.nanorc" -f /home/zyuser/.nanorc -+ -P /home/zyuser/.nanoprofile -A /home/zyuser/.nanoafterlog -B /home/zyuser/.nanobeforelog -l -k -E -o /home/zyuser/ -O /home/zyuser/ -z -c -- /home/zyuser/main.py
GNU nano 5.4                    /home/zyuser/main.py                    Modified
#!/usr/bin/python3
using namespace std;

int Main() {
    cout << "Hello, world!" << endl;
    cout << "Welcome to CSC108!" << endl;
    return 0;
}

main.py:3: error: syntax error, expected '(' after expression
  cout << "Hello, world!"
                         ^
^G Help    ^O Write Out   ^W Where Is    ^K Cut        ^T Execute    ^C Cursor
^X Exit    ^R Read File   ^\ Replace     ^U Paste      ^J Justify    ^/ Go To Line
```

Feedback?

The compiler indicates a missing semicolon ';'. But the real error is the missing << symbols.

Sometimes the compiler error message refers to a line that is actually many lines past where the error actually occurred. Not finding an error at the specified line, the programmer should look to previous lines.

INSPECTION ACTIVITY 1.5.2 The compiler error messages line may be past the line with the actual error.

Start (x open)

```
1 #!/usr/bin/python3
2 using namespace std;
3
4 int Main() {
5     cout << "Hello, world!"
6     cout << "Welcome to CSC108!"
7     return 0;
8 }
```

Programmer should examine line 5 and earlier lines

4 Compiler processing

Compiler message: Error at line 3

Captions

1. The compiler hasn't yet detected the error.
2. Now the compiler is confused, so it generates a message. But the reported line number is past the actual syntax error.
3. Upon not finding an error at line 3, the programmer should look at earlier lines.

Feedback?

INSPECTION ACTIVITY 1.5.3. Unclear error messages.

1) When a compiler says that an error exists on line 5, that line must have an error.

- True
- False

2) If a compiler says that an error exists on line 90, the actual error may be on line 87, 85, etc.

- True
- False

3) If a compiler generates a specific message like "missing semicolon", then a semicolon must be missing somewhere, though maybe from an earlier line.

- True
- False

Feedback?

Fixing the first error

Some errors create an overwhelmingly long list of error messages. Good practice is to focus on fixing just the first error reported by the compiler and then recompiling. The remaining error messages may be not but are more commonly due to the compiler's confusion caused by the first error and are thus irrelevant.

Good practice for fixing errors reported by the compiler.

1. Focus on FIRST error message, ignoring the rest
2. Look at reported line of first error message. If error found, fix. Else, look at previous few lines.
3. Compile, repeat

zyDE 1.5.1: Fixing syntax errors.

Click run to compile, and note the long error list. Fix only the first error, then recompile. Repeat this process (fix first error, recompile) until the program compiles and runs. Expect to see misleading error messages as well as errors that occur before the reported line number.

Load default template... Run

```
#!/usr/bin/python3
using namespace std;

int main() {
    int numFoxes;
    int numGeese;
    int totalNeeds;

    numFoxes = 100;
    numGeese = 75;

    cout << numFoxes << " foxes\n" << endl;
    cout << numGeese << " geese\n" << endl;
    totalNeeds = numFoxes + numGeese;
    cout << totalNeeds << " total" endl;

    return 0;
}
```

Feedback?

INSPECTION ACTIVITY 1.5.4. Fixing the first error.

A compiler generates the following error messages:
Line 7: Missing semicolon
Line 9: numXmas not defined
Line 10: Expected ')'

1) The programmer should start by examining line _____.

- 7
- 9
- 10

2) If the programmer corrects an error on line 7, the programmer should _____.

- check earlier lines too
- compile
- check line 9

3) If the programmer does NOT find an error on line 7, the programmer should check line _____.

- 8
- 9
- 10

Feedback?

CHALLENGE ACTIVITY 1.5.1. Basic syntax errors.

Type the statements below, correcting the one syntax error in each statement. Hints: Statements end in semicolons, and string literals use double quotes.

```
cout << "Interesting is taxes." << endl
cout << "Particularly" "
cout << "of the fiscal." << endl;
cout << "Fact one is:" << daxfee << endl;
```

See How to take zyBooks for info on how our automated program grader works.

Lab 1 Write a program

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

int main() {
    int daxfee;

    daxfee = 1;

    /* Your solution goes here */

    return 0;
}
```

Run

View your last submission

Feedback?

CHALLENGE ACTIVITY 1.5.2 More syntax errors.

Each cout statement has a syntax error. Type the first cout statement, and press Run to observe the error message. Fix the error, and run again. Repeat for the second, then third, cout statement.

```
cout << "Pluto: " << sourgum << endl;
cout << int sourgum << endl;
cout << sourgum " scorp" << endl;
```

See How to take zyBooks for info on how our automated program grader works.

Lab 1 Write a program

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

int main() {
    int sourgum;

    sourgum = 5;

    /* Your solution goes here */

    return 0;
}
```

Run

View your last submission

Feedback?

Logic errors

Because a syntax error is detected by the compiler, a syntax error is known as a type of **compile-time error**.

New programmers commonly complain "The program compiled perfectly but isn't working." Successfully compiling means the program doesn't have compile-time errors, but the program may have other kinds of errors. A **logic error** (also called a **bug**) is an error that occurs while a program runs. For example, a programmer might mean to type `numBeans = numJars` but accidentally types `numBeans = numJars / 10`. Without 10, The program would compile but would not run as intended.

Figure 1.5.3: Logic errors.

```
#!/usr/bin/python3
using namespace std;

int main() {
    int numBeans;
    int numJars;
    int totalBeans;

    numBeans = 100;
    numJars = 5;

    cout << numBeans << " beans in a\n" << endl;
    cout << numJars << " jars\n" << endl;
    totalBeans = numBeans + numJars;
    cout << totalBeans << " total" << endl;

    return 0;
}
```

Feedback?

zyDE 1.5.2: Fix the bug.

Click run to compile and execute and then note the incorrect program output. Fix the bug in the program.

Load default template... Run

```
#!/usr/bin/python3
using namespace std;

int main() {
    // This program has a bug that causes a logic error
    // Can you find the bug?
    int numBeans;
    int numJars;
    int totalBeans;

    numBeans = 100;
    numJars = 5;

    cout << numBeans << " beans in a\n" << endl;
    cout << numJars << " jars\n" << endl;
    totalBeans = numBeans + numJars;
    cout << totalBeans << " total" << endl;

    return 0;
}
```

Feedback?

Bugs

This term bug to describe a runtime error was popularized when in 1947 engineers discovered their program on a Harvard University Mark II computer was not working because a moth was stuck in one of the relays (a type of mechanical switch). They taped the bug into their engineering log book, still preserved today. Click to see it.

Credit: National Geographic Education

Compiling frequently

Good practice, especially for new programmers, is to compile after writing only a few lines of code, rather than writing tens of lines and then compiling. New programmers commonly write tens of lines before compiling, which may result in an overwhelming number of compilation errors and warnings and logic errors that are hard to debug and correct.

INSPECTION ACTIVITY 1.5.5. Compile and run after writing just a few statements.

Start (x open)

```
1 #!/usr/bin/python3
2 using namespace std;
3
4 int Main() {
5     cout << "Hello, world!"
6     cout << "Welcome to CSC108!"
7     return 0;
8 }
```

Compile & run

Compile & run

Captions

1. Writing many lines of code without compiling and running is bad practice.
2. New programmers should compile and run programs after every few lines. Even experienced programmers compile and run frequently.

Feedback?

INSPECTION ACTIVITY 1.5.6. Compiling and running frequently.

1) A new programmer writes 5 lines of code, compiles and runs, writes 5 more lines, and compiles and runs again. The programmer is _____.

- wasting time
- following good practice

2) An experienced programmer writes 80 lines of code and then compiles and runs. The programmer is probably _____.

- programming dangerously
- following good practice

Feedback?

Compiler warnings

A compiler will sometimes report a **warning**, which doesn't stop the compiler from creating an executable program but indicates a possible logic error. Ex: Some compilers will report a warning like "Warning, dividing by 0 is not defined" if encountering code like `totalBeans = numJars / 0`. Dividing that program does result in a runtime error. Even though the compiler may create an executable program, good practice is to write programs that produce no compiler warnings. In fact, many programmers recommend the good practice of configuring compilers to print even more warnings. For example, the gcc compiler can be run as `gcc -Wall prog1file.cpp`, where `-Wall` indicates that the compiler should display all warnings.

INSPECTION ACTIVITY 1.5.7. Compiler warnings.

1) A compiler warning by default will prevent a program from being created.

- True
- False

2) Generally, a programmer should not ignore warnings.

- True
- False

3) A compiler's default settings cause most warnings to be reported during compilation.

- True
- False

Feedback?

How was this section? Provide section feedback

Activity summary for assignment: CSC108 CH0 CA

Due: 01/06/2023, 11:59 PM EST
This assignment is due now but course is closed and activity recorded, but will not count towards this assignment (unless the due date is changed). See this article for more info.

Completion details

18 / 18 points
10 / 10 points submitted in Blackboard Learn