Common Coding Errors
Shomiron Ghose
Thomas Jefferson High School
For Sci. and Tech.

Common Coding Errors

How to Avoid Some of the More Common Errors

1 Introduction

During my adventures into coding I have had the opportunity to experience many error messages. Though this may seem bad, it is actually the opposite as it provides the opportunity to learn how to remedy the mistake and hopefully not commit the same error again. Better yet, it gives you the ability to spot the error over time so that you can identify it in the code of others and help out your teammates!

2 Errors

2.1 The Oh-So-Fun Braces Addition or Omission

This so very common error arises when you have one parenthesis too many or one parentheses missing.

```
#include<stdio.h>

int main()
{
printf("Hello World\n");
printf("%d",add(1,2));
}
int add(int a, int b )
{
    return a+b;
}
```

Figure 1: Working Code

```
#include<stdio.h>

int main()
{
printf("Hello World\n");
printf("%d",add(1,2));
}
int add(int a, int b )
{
return a+b;
```

Figure 2: Missing Parenthesis

```
#include<stdio.h>

int main()
{
printf("Hello World\n");
printf("%d",add(1,2));
}
int add(int a, int b )
{
return a+b;
}
}
```

Figure 3: One Parenthesis Too Many

In Figure 1: Working Code we have a simple add method that adds two values that are passed, not pointers although the same error, at least using GCC 4.4.1. Figure Two is missing a parenthesis and Figure Three could use one less.

Figure 4: Working Code Running

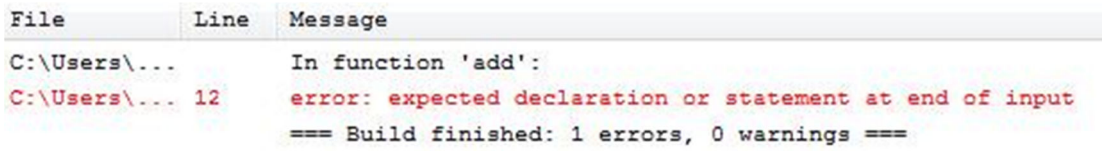

Figure 5: One Parenthesis Missing Error

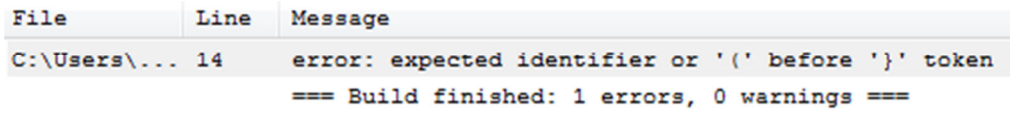

Figure 6: One Parenthesis Too Many Error

In Figure 4 it can be seen that assuming all parenthesis are in there right place, the program runs as it should. If you have one missing though, it expects an end of statement which it never gets (Figure 5) and if you have too many (Figure 6) it expects there to be some identifier which is missing. To alleviate this saving code and compiling helps as you can identify sooner where you may have made a mistake. For large projects, sub versioning can make a world of difference as well.

Another thing that may be of help is coding conventions, choosing either a standard style of where to put parenthesis or using a constant style that you prefer can help you locate where one may have been lost if the line number given is not indicative of the general region. Even better in cases, some IDEs such as CodeBlocks and I assume Visual Studio or even text editors such as Notepad++ will make it such that when you click on a parenthesis, it will show it's 'match' making it easier to find where one may have been lost or inadvertently added.

As a side note, it is useful to apply the same principles to nested parenthesis as trying to find the one parenthesis that is missing among many others is quite an ordeal.

```c
#include<stdio.h>
int main()
{
printf("Hello World\n");}
```

Figure 7

```c
#include<stdio.h>

int main(){
printf("Hello World\n");
}
```

Figure 8

```c
#include<stdio.h>

int main()
{
printf("Hello World\n");
}
```

Figure 8

```c
#include<stdio.h>
int main(){printf("Hello World\n");}
```

Figure 10

Above are different variants of ways to style parenthesis; I believe a style similar to Figure 9 to be the most commonly used. Figure 10 is reminiscent of code squished together, something not typically wanted though sometimes compact parenthesis help such as if you have multiple nested layers it can help with spacing in order to differentiate those. Different ways to style parentheses are detailed in various conventions and style guides and are typically meant to help with clarity for readers as well as functionality for code writers.

2.2 The Semicolon

Semicolon omissions in C may be commonplace for those used to languages that do not use them, an example of which is Python and the more satiric whitespace language composed of a variety of spaces, tabs, newlines and the like., though personally sometimes making a multiple nested hierarchy of loops one space off is frustrating if the spaces are not visible.

```c
#include<stdio.h>

int main()
{
printf("Hello World\n")
printf("Bye Bye");
}
```

Figure 10

```c
#include<stdio.h>

int main()
{
printf("Hello World\n")
}
```

Figure 9

Omitting the semicolon in Figure 11 earns a ‘error: expected ';' before '}’ while in Figure 12 it says it expected a semicolon before the second printf statement. In both of these it conveniently lists line numbers, making it an easy to solve problem.

```
#include<stdio.h>
int main()
{
printf("Hello World\n");;;;;;;;
}
```

Figure 13

```
#include<stdio.h>
int main()
{
1+1;
}
```

Figure 14

```
#include<stdio.h>
int main()
{
    int a = 1;
for(;a<5;a++)
{
    printf("%d\n",a);
}
}
```

Figure 11

It may be interesting to note that Figure 13 and Figure 14 are both valid programs though you cannot really do anything with the later since you do nothing with the value and are calculating the expression and immediately throwing it away. Figure 13 on the other hand is an example of C accepting empty statements. This same concept is used when doing an empty statement in a for loop as in Figure 15 in which the 'initializer' statement is left blank as the current value of integer 'a' is used.

2.3 Forgetting to Assign A Value

```
#include<stdio.h>
int main()
{
int a,b;
printf("%d",a+b);
}
```

Figure 16

```
#include<stdio.h>
int main()
{
int a;
printf("%d",a);
}
```

Figure 17

In these cases (Figures 16 and 17) the variable was 'made' but never given a value which means it keeps the value of the point in memory it was given leading to unexpected results and numbers that make no sense with regard to their intended purpose. The pointer is pointing to an old value which makes for the seemingly-random result. The simple fix here is to remember to assign values to the variables before using them.

3 Summary

In this paper I have discussed some of the more common errors that you may find while learning C or accidentally editing parts of your code. As a result hopefully your code will have fewer bugs and you will be able to spend more time implementing new functionality rather than dealing with small bugs. Of all the things to take away, I think parentheses is the most important as it is tiresome to search for one parenthesis among tens of them especially when they are closely nested as in Figure 18.

```
#include<stdio.h>
int main()
{
printf("%d",sum(1,sum(2,sum(3,sum(4
,sum(5,sum(6,sum(7,sum(8,sum(9,10))
))))))));
}
int sum(int a, int b)
{
    return a+b;
}
```

Figure 12

David B. Stewart. Twenty-FiveMost Common Mistakes with Real-Time Software Development. In *Proc. Embedded Systems Conference San Fransisco (ESC SF) 2001*, April 2001