

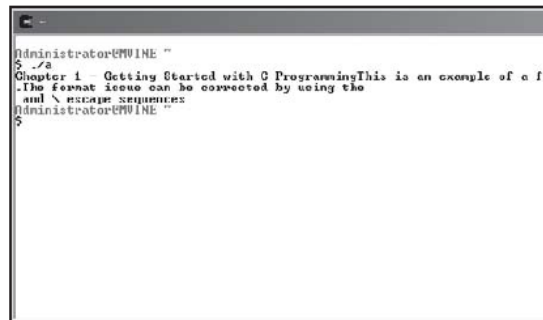
## Lecture 16 How to Debug C Program

If your program compiles, exits, or executes abnormally, there is almost certainly an error (a bug) in your program. A fair amount of your programming time will be spent finding and removing these bugs. This section provides some tips to help you get started. Remember, though, that debugging is as much art as it is computer science and, of course, the more you practice programming the easier debugging will become!

Often a program will compile and execute just fine, but with results you did not expect. For example, the following program and its output shown in Figure 1.11 compiles and executes without error, but the output is unreadable, or in other words, not what I expected.

```
#include <stdio.h>

main()
{
    printf("Chapter 1 - Getting Started with C Programming");
    printf("This is an example of a format bug.");
    printf("The format issue can be corrected by using");
    printf(" the \n and \\ escape sequences");
}
```



```
Administrator: C:\WINDOWS\system32\cmd.exe
$ ./a
Chapter 1 - Getting Started with C ProgrammingThis is an example of a fo
The format issue can be corrected by using the
and \ escape sequences
Administrator: C:\WINDOWS\system32\cmd.exe
$
```

FIGURE 1.11

A sample format bug.

Can you see where the format issue or issues are? What's missing and where should the correction or corrections be placed? The next block of code and its output in Figure 1.12 corrects the format issues with appropriately placed escape sequences.

```
#include <stdio.h>

main()
{
```

```
printf("Chapter 1 - Getting Started with C Programming\n");

printf("This is an example of a format bug.\n");

printf("The format issue can be corrected by using");

printf(" the \\n and \\\\ escape sequences");

}
```

**FIGURE 1.12**  
Correcting format bugs with appropriately placed `\n` and `\\` escape sequences.

Format issues are common in beginning programming and are typically quickly resolved by practicing the `printf()` function and the various escape sequences.

Another common bug type is a logic error, including a loop that doesn't exit when expected, an incorrect mathematical equation, or perhaps a flawed test for equality (condition). The first step in debugging a logic error is to find the first line where the program bug exists. One way of doing this is through print statements, using the `printf()` function, scattered through your code. For example, you might do something like this in your source code:

```
anyFunction(int x, int y)
{
printf("Entering anyFunction()\n"); fflush(stdout);

---- lots of your code here -----

printf("Exiting anyFunction()\n"); fflush(stdout);
}
```

The `fflush()` function ensures that the print statement is sent to your screen immediately, and you should use it if you're using `printf()`'s for debugging purposes. The `stdout` parameter passed to the `fflush()` function is the standard output, generally the computer screen.

After you have narrowed down the line or function where your logic error occurs, the next step is to find out the value of your variables at that time. You can also use the `printf()` function to print variable

values, which will aid you greatly in determining the source of abnormal program behavior. Displaying variable values using the `printf()` function will be discussed in Chapter 2 in detail.

Remember, after you fix any bug, you must recompile your program, run it, and debug it again if necessary.

Beginning programmers will, more often than not, encounter compile errors rather than logic errors, which are generally the result of syntax issues such as missing identifiers and terminators or invalid directives, escape sequences, and comment blocks.

Debugging compile errors can be a daunting task, especially when you see 50 or more errors on the computer screen. One important thing to remember is that a single error at the top of your program can cause cascading errors during compile time. So it goes without saying that the best place to start debugging compile errors is with the first error on the list! In the next few sections, you'll explore some of the more common compile errors beginning C Programmers experience.

## Lecture 16 How to debug C programs (HINDI)

यदि आपका प्रोग्राम संकलित होता है, बाहर निकलता है, या असामान्य रूप से क्रियान्वित होता है, तो आपके प्रोग्राम में लगभग निश्चित रूप से एक त्रुटि (एक बग) है। आपके प्रोग्रामिंग समय का एक उचित हिस्सा इन बग्स को ढूँढने और हटाने में खर्च होगा। यह खंड आपको शुरू करने में मदद के लिए कुछ सुझाव प्रदान करता है। हालांकि, याद रखें कि डिबगिंग उतनी ही कला है जितनी कि कंप्यूटर विज्ञान और, बेशक, जितना अधिक आप प्रोग्रामिंग का अभ्यास करेंगे, डिबगिंग उतनी ही आसान हो जाएगी!

अक्सर एक प्रोग्राम संकलित होता है और ठीक से क्रियान्वित भी होता है, लेकिन परिणाम वैसे नहीं होते जैसे आपने अपेक्षा की थी। उदाहरण के लिए, निम्नलिखित प्रोग्राम और इसका आउटपुट जो चित्र 1.11 में दिखाया गया है, बिना किसी त्रुटि के संकलित होता है और क्रियान्वित भी होता है, लेकिन आउटपुट पढ़ने योग्य नहीं है, या दूसरे शब्दों में, वह नहीं है जो मैंने अपेक्षा की थी।

```
#include <stdio.h>

int main()
{
    printf("Chapter 1 - Getting Started with C Programming");
    printf("This is an example of a format bug.");
    printf("The format issue can be corrected by using");
    printf("the \\n and \\\\\\ escape sequences");
}
```

FIGURE 1.11

A sample format bug.

```
Administrator@WIN10:~$ ./a
Chapter 1 - Getting Started with C ProgrammingThis is an example of a fo
.The format issue can be corrected by using the
and \ escape sequences
Administrator@WIN10:~$
```

संकलन त्रुटियों की डिबगिंग एक भयानक कार्य हो सकता है, खासकर जब आपको कंप्यूटर स्क्रीन पर 50 या उससे अधिक त्रुटियां दिखाई दें। एक महत्वपूर्ण बात याद रखनी चाहिए कि आपके प्रोग्राम के शीर्ष पर एक अकेली त्रुटि संकलन समय के दौरान श्रृंखलाबद्ध त्रुटियों का कारण बन सकती है। इसलिए यह कहना बिना कहे चलता है कि संकलन त्रुटियों की डिबगिंग शुरू करने का सबसे अच्छा स्थान सूची में पहली त्रुटि के साथ है! अगले कुछ खंडों में, आप सी प्रोग्रामर्स द्वारा अनुभव की जाने वाली कुछ अधिक सामान्य संकलन त्रुटियों का पता लगाएंगे।

```
Administrator@WIN10:~$ ./a
Chapter 1 - Getting Started with C Programming
This is an example of a format bug.
.The format issue can be corrected by using the \n and \\ escape sequences
Administrator@WIN10:~$
```

FIGURE 1.12

Correcting format bugs with appropriately placed `\n` and `\\` escape sequences.

प्रोग्रामिंग में शुरुआत करते समय फॉर्मेट की समस्याएं आम होती हैं और आमतौर पर `printf()` फंक्शन और विभिन्न एस्केप सीक्वेंसों का अभ्यास करके जल्दी हल की जा सकती हैं।

एक और आम बग प्रकार लॉजिक एरर है, जिसमें एक ऐसा लूप शामिल होता है जो उम्मीद के मुताबिक बाहर नहीं निकलता, एक गलत गणितीय समीकरण, या शायद बराबरी (कंडीशन) के लिए एक दोषपूर्ण परीक्षण। लॉजिक एरर की डिबगिंग का पहला कदम यह पता लगाना है कि प्रोग्राम में बग कहाँ मौजूद है। इसे करने का एक तरीका आपके कोड में बिखरे हुए `printf()` फंक्शन का उपयोग करके प्रिंट स्टेटमेंट्स के माध्यम से है। उदाहरण के लिए, आप अपने सोर्स कोड में कुछ इस तरह कर सकते हैं:

```

void                anyFunction(int                x,                int                y)
{
    printf("Entering                anyFunction()\n");                fflush(stdout);
    //    ----    आपके    कोड    का    बहुत    सारा    हिस्सा    यहाँ    होगा    -----
    printf("Exiting                anyFunction()\n");                fflush(stdout);
}

```

`fflush()` फंक्शन सुनिश्चित करता है कि प्रिंट स्टेटमेंट तुरंत आपकी स्क्रीन पर भेजा जाए, और यदि आप डिबगिंग के उद्देश्य से `printf()` का उपयोग कर रहे हैं तो आपको इसका उपयोग करना चाहिए। `fflush()` फंक्शन को पास किया गया `stdout` पैरामीटर स्टैंडर्ड आउटपुट होता है, आमतौर पर कंप्यूटर स्क्रीन।

जब आप उस लाइन या फंक्शन को संकीर्ण कर लेते हैं जहाँ आपकी लॉजिक एरर होती है, तो अगला कदम उस समय आपके वेरिएबल्स के मूल्यों का पता लगाना होता है। आप `printf()` फंक्शन का उपयोग करके वेरिएबल मूल्यों को प्रिंट कर सकते हैं, जो असामान्य प्रोग्राम व्यवहार के स्रोत का पता लगाने में आपकी बहुत मदद करेगा। `printf()` फंक्शन का उपयोग करके वेरिएबल मूल्यों को प्रदर्शित करना अध्याय 2 में विस्तार से चर्चा किया जाएगा।

याद रखें, जब आप किसी भी बग को ठीक कर लेते हैं, तो आपको अपने प्रोग्राम को फिर से संकलित करना होगा, इसे चलाना होगा, और यदि आवश्यक हो तो इसे फिर से डिबग करना होगा।

शुरुआती प्रोग्रामर अक्सर लॉजिक एरर की तुलना में संकलन एरर का सामना करते हैं, जो आमतौर पर सिंटैक्स समस्याओं का परिणाम होते हैं जैसे कि गायब पहचानकर्ता और समाप्तिकर्ता या अमान्य निर्देश, एस्केप सीक्वेंस, और टिप्पणी ब्लॉक।

संकलन त्रुटियों की डिबगिंग एक भयानक कार्य हो सकता है, खासकर जब आपको कंप्यूटर स्क्रीन पर 50 या उससे अधिक त्रुटियां दिखाई दें। एक महत्वपूर्ण बात याद रखनी चाहिए कि आपके प्रोग्राम के शीर्ष पर एक अकेली त्रुटि संकलन समय के दौरान श्रृंखलाबद्ध त्रुटियों का कारण बन सकती है। इसलिए यह कहना बिना कहे चलता है कि संकलन त्रुटियों की डिबगिंग शुरू करने का सबसे अच्छा स्थान सूची में पहली त्रुटि के साथ है! अगले कुछ खंडों में, आप सी प्रोग्रामर्स द्वारा अनुभव की जाने वाली कुछ अधिक सामान्य संकलन त्रुटियों का पता लगाएंगे।

