

Algoritmalar ve Programlama I – BLM 103

Hafta 8: Test ve Hata Ayıklama



Fenerbahçe Üniversitesi

8. Hafta İçeriği

- Hata Türleri
 - Sözdizimi (Syntax)
 - Anlamsal (Semantic)
 - Algoritmik
- Hata Ayıklama
- Visual Studio Hata Ayıklama Araçları

Hata Türleri

- Sözdizimsel (Syntactic) Hatalar

- Derlenecek olan kodda programlama dilinde tanımlı olmayan ifadeler veya beklenen bazı şeyler eksiktir.
- Derleyici tarafından yakalanır.
- Derlenme işlemi gerçekleştirilmez.

- Örn.

```
printf("abc);
```

printf'in sonundaki " işareti eksiktir.

Hata Türleri

- Anlamsal (Semantic) Hatalar
 - Kodda sözdizim hatası yoktur.
 - Programlayan kişinin, yazmak istediği kod değildir.

Örn.

```
for(int i=0;i<=100;)
```

Programcı *i*'i bir arttırmayı unutmuştur. Aslında yazmak istemektedir.

Hata Türleri

- Algoritmik Hatalar

- Programın çözüm mantığındaki hatalardır.
- Programcı yazmak istediği kodu doğru yazmıştır ancak
- Yazılan kod çözülmesi gereken problemi çözmemektedir.

Örn.

- Üç sayının en büyüğünü alan kodu yazması beklenirken, en küçüğünü hesaplayan kodun yazılması gibi...

Sözdizim (Syntactic) Hataları

- En sık karşılaşılan hata türüdür.
 - Unutulmuş noktalı virgül veya parantez hataları
 - Yanlış yazılmış veri türleri, flot gibi
- Sadece bir yanlış, derleyicinin birden çok hata vermesine neden olabilir.

```
• main () {  
  int i  
  int j;  
  for (i = 0; i <= 10; i++) {  
    j = i * 7;  
    printf("%d x 7 = %d\n", i, j);  
  }  
}
```

Unutulmuş noktalı virgül



Anlamsal (Semantic) Hatalar

- Genel Hatalar
 - Süslü parantez yazmanın unutulması
 - Eşitlik kontrolü yerine (==) atama işareti konması (=)
 - Operatör önceliklerinin yanlış düşünülmesi
 - For döngüsündeki yanlış konan sınırlar
 - İlk ataması yapılmamış değişkenler

Unutulmuş Süslü parantezler

```
• main () {  
  int i  
  int j;  
  for (i = 0; i <= 10; i++)  
    j = i * 7;  
    printf("%d x 7 = %d\n", i, j);  
}
```

Algoritmik Hatalar

- Tasarım yanlıştır.

Algoritma doğru problemi çözmez

- Tespit edilmesi zordur

- Program, programcının istediği işi yapar.
- Ancak programcı, çözülmesi gereken problemi anlamamıştır.

- Çözmesi zor olabilir.

- Tekrar tasarım ihtiyacı doğrumaktadır.

Hata Ayıklama Teknikleri

- İlkel Yaklaşım
 - Programın akışı esnasında çeşitli yerlere printf yerleştirerek, akışı izlemek
 - Avantajı:
 - Hata ayıklama uygulamasına ihtiyaç yoktur
 - Dezavantaj:
 - Kod ve beklenen değerler hakkında oldukça bilgili olmak gerekir.
 - Kodu yeniden derleme ve çalıştırmak gerekir.
 - Eklenen printf'lerde hata yapılabilir.

Hata Ayıklama Teknikleri

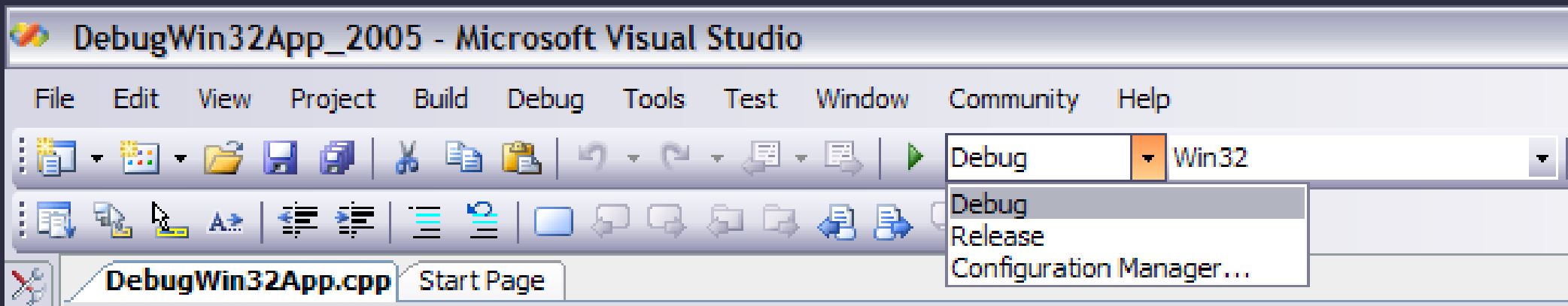
- Hata Ayıklayıcı Kullanmak
 - Visual Studio'nun içindeki hata ayıklama araçları kullanılabilir.

Hata Ayıklama Teknikleri

- Visual Studio iki adet çalışma zamanında hata giderme özelliğine sahiptir.
 - Programın akışını takip etmek
 - Program çalışırken değişkenlerin değerlerini izlemek
- Bu araçlar, program çalışırken belli bir yerde durdurup, değişkenlerin değerlerini okumak hatta onları değiştirmek için çok faydalıdır.

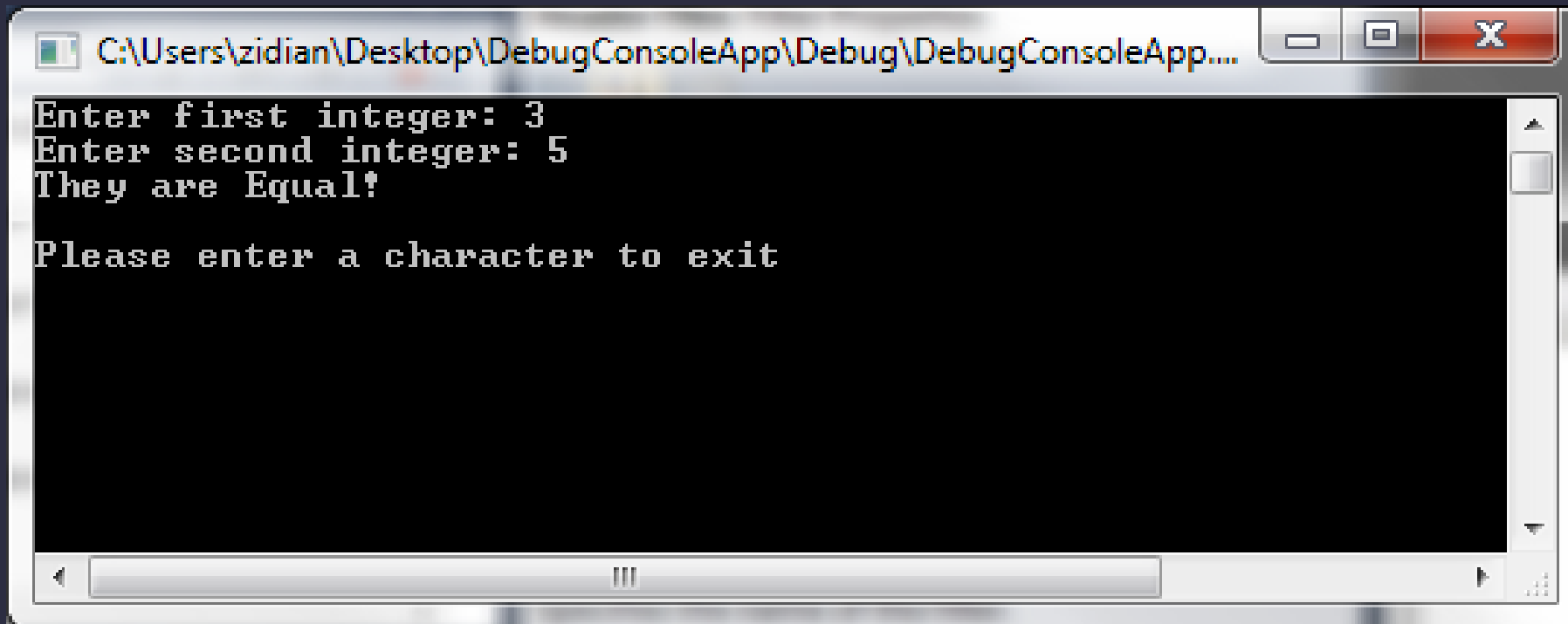
Hata Ayıklama Teknikleri

- Visual Studio'da program çalıştırılırken 2 mod bulunur.
 - Debug: Hata ayıklama için kullanılmaktadır. Program hata ayıklama modunda iken daha yavaş çalışmaktadır.
 - Release: Programın hata ayıklama süreci tamamlandıktan sonra, diğer kullanıcılar ile paylaşmak istendiğinde kullanılmaktadır.



Hata Ayıklama Teknikleri

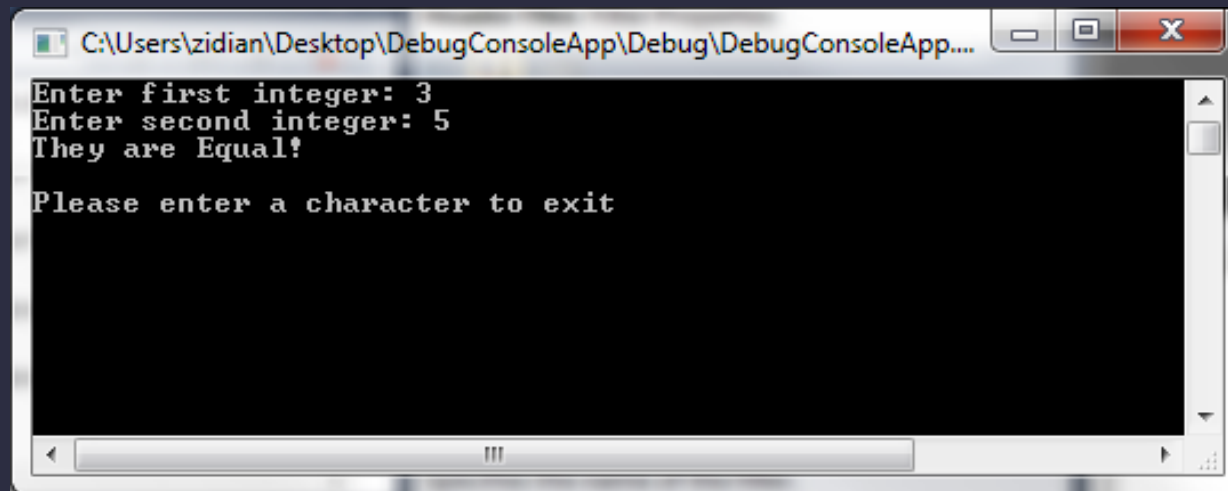
- Örn. Kullanıcıdan iki sayı alın ve eşit olup olmadığını kontrol eden bir program yazınız.



```
C:\Users\zidian\Desktop\DebugConsoleApp\Debug\DebugConsoleApp...  
Enter first integer: 3  
Enter second integer: 5  
They are Equal!  
  
Please enter a character to exit
```

Hata Ayıklama Teknikleri

- Örn. Kullanıcıdan iki sayı alın ve eşit olup olmadığını kontrol eden bir program yazınız.

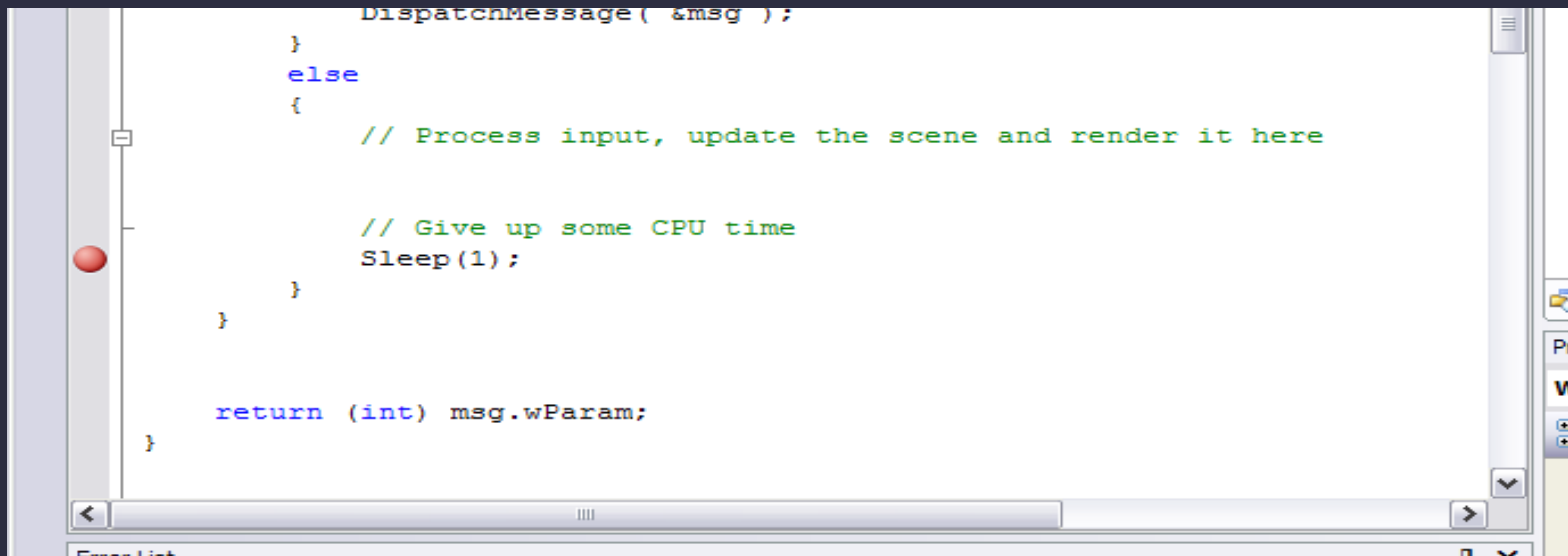


```
C:\Users\zidian\Desktop\DebugConsoleApp\Debug\DebugConsoleApp...  
Enter first integer: 3  
Enter second integer: 5  
They are Equal!  
  
Please enter a character to exit
```

- Derleyici 0 hata, 0 uyarı verdi, ancak program yanlış çalışıyor.

Hata Ayıklama Teknikleri

- Duraklama Noktası (Breakpoint)
 - Program çalışırken uygulamanın duraklamasını istediğiniz satırın seçilmesidir.
 - Duraklamasını istediğiniz satırın yanındaki yere tıkladığınızda kırmızı bir yuvarlak oluşur. Programı debug modunda çalıştırdığınızda o satıra gelip uygulama duracaktır.



```
DispatchMessage( &msg );
}
else
{
    // Process input, update the scene and render it here

    // Give up some CPU time
    Sleep(1);
}

return (int) msg.wParam;
}
```

The screenshot shows a code editor window with a red circular breakpoint marker on the left margin, positioned next to the line containing the comment "// Give up some CPU time". The code is written in a C++ style with blue keywords and green comments. The editor has a standard Windows-style interface with a scrollbar on the right and a status bar at the bottom.

Hata Ayıklama Teknikleri

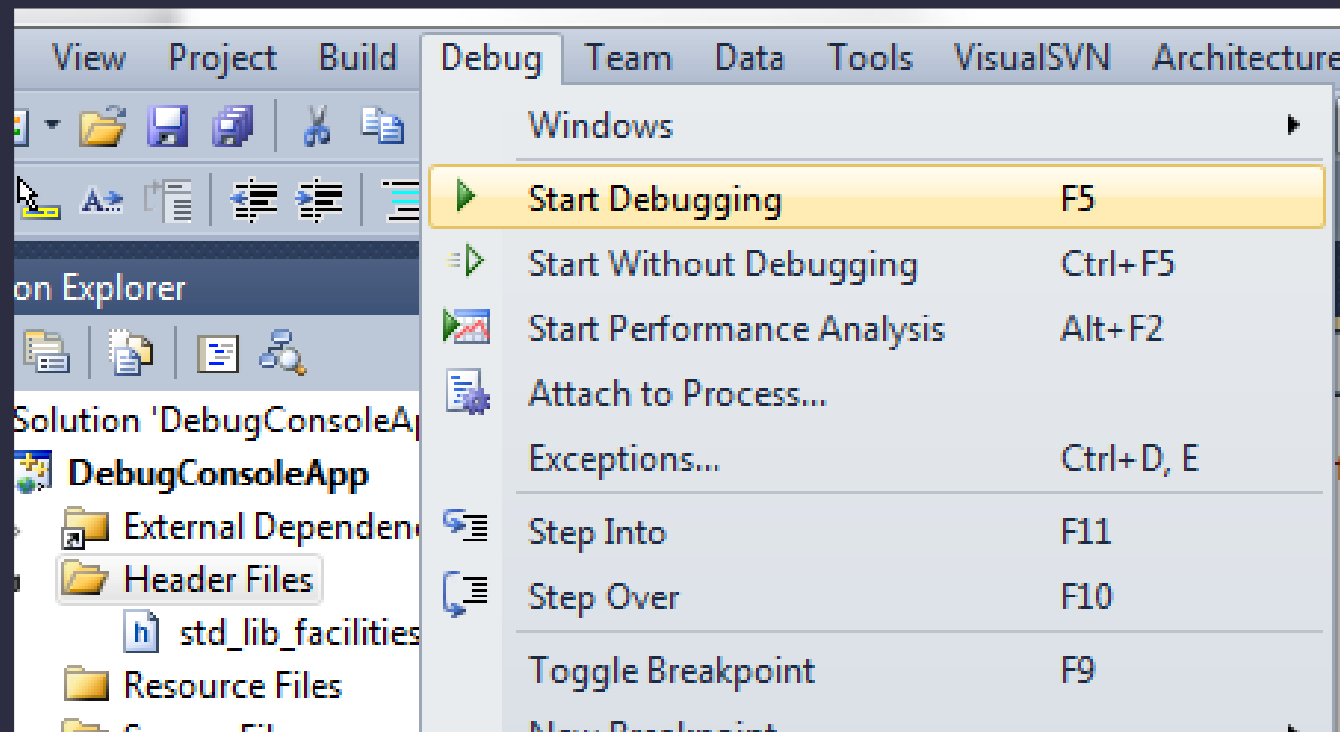
```
(Global Scope) main()
1
2 #include "std_lib_facilities.h"
3
4
5 int main()
6 {
7     int x, y;
8
9     cout << "Enter first integer: ";
10    cin >> x;
11
12    cout << "Enter second integer: ";
13    cin >> y;
14
15    if(x=y)
16        cout << "They are Equal!" << endl;
17    else if(x>y)
18        cout << "The first one is bigger!" << endl;
19    else
20        cout << "The second ont is bigger!" << endl;
21
22    cout << endl;
23
24    keep_window_open();
25
26    return 0;
27 }
```

100 %

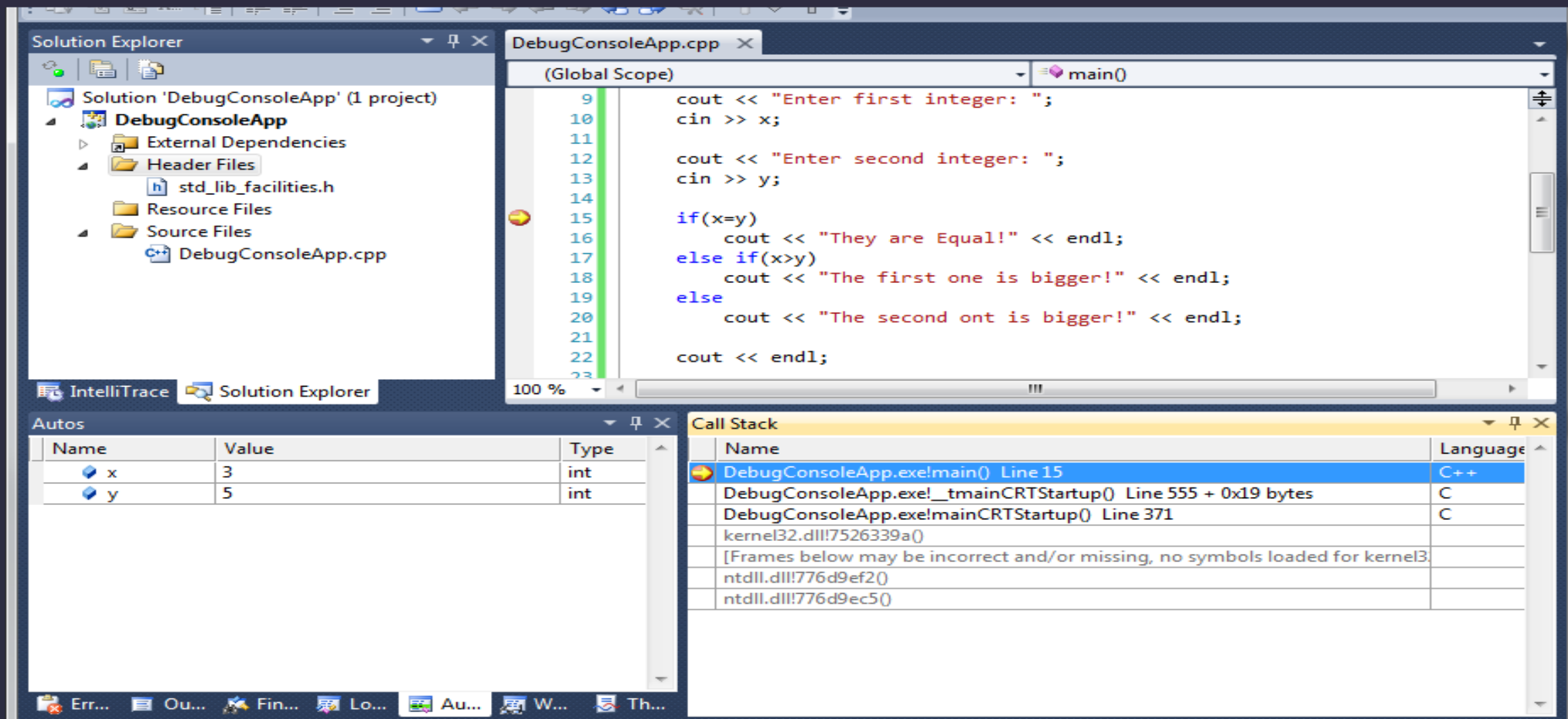
Metrics Results Error List

Ln 15 Col 1 Ch 1 INS

Hata Ayıklama Teknikleri



Hata Ayıklama Teknikleri



Solution Explorer

DebugConsoleApp.cpp

```
(Global Scope) main()
9      cout << "Enter first integer: ";
10     cin >> x;
11
12     cout << "Enter second integer: ";
13     cin >> y;
14
15     if(x=y)
16         cout << "They are Equal!" << endl;
17     else if(x>y)
18         cout << "The first one is bigger!" << endl;
19     else
20         cout << "The second ont is bigger!" << endl;
21
22     cout << endl;
23
```

IntelliTrace Solution Explorer

Autos

Name	Value	Type
x	3	int
y	5	int

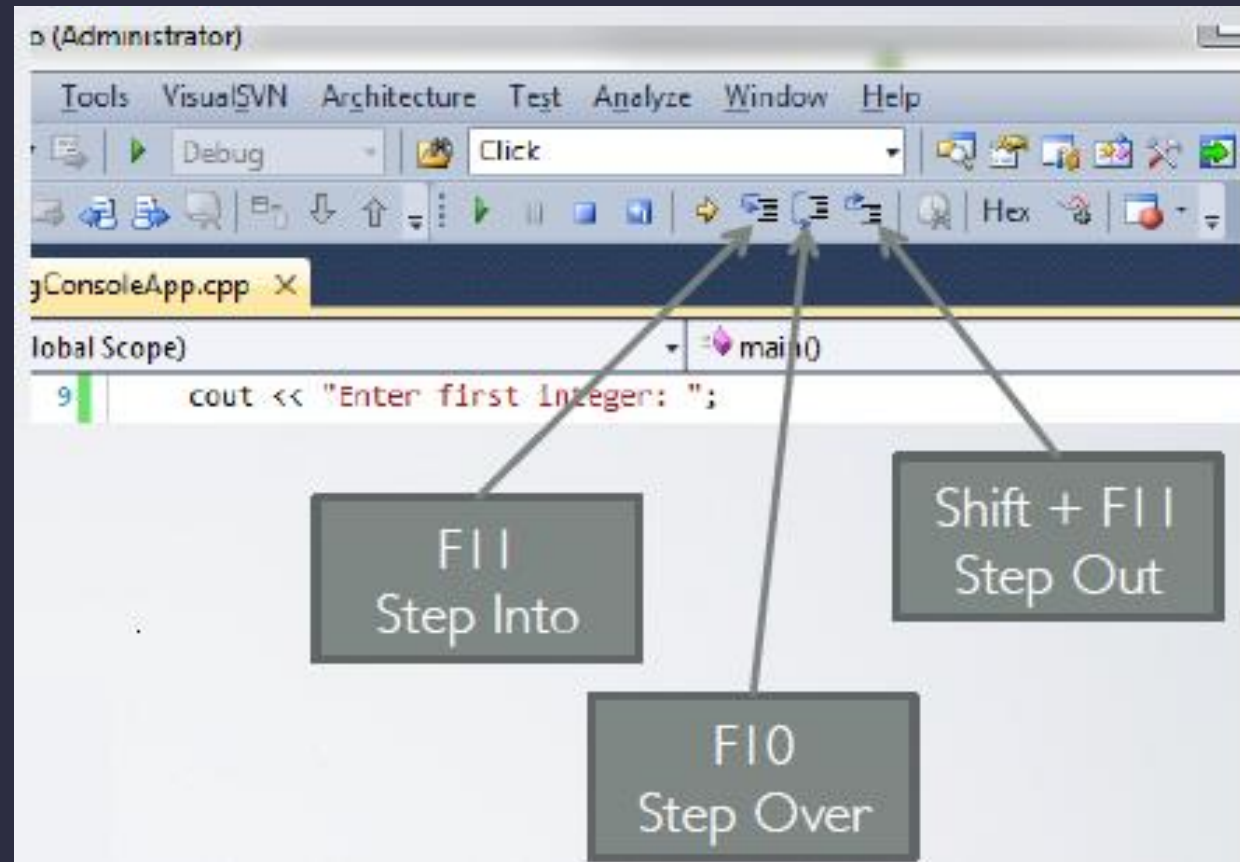
Call Stack

Name	Language
DebugConsoleApp.exe!main() Line 15	C++
DebugConsoleApp.exe!_tmainCRTStartup() Line 555 + 0x19 bytes	C
DebugConsoleApp.exe!mainCRTStartup() Line 371	C
kernel32.dll!7526339a()	
[Frames below may be incorrect and/or missing, no symbols loaded for kernel32.dll]	
ntdll.dll!776d9ef2()	
ntdll.dll!776d9ec5()	

Err... Ou... Fin... Lo... Au... W... Th...

Hata Ayıklama Teknikleri

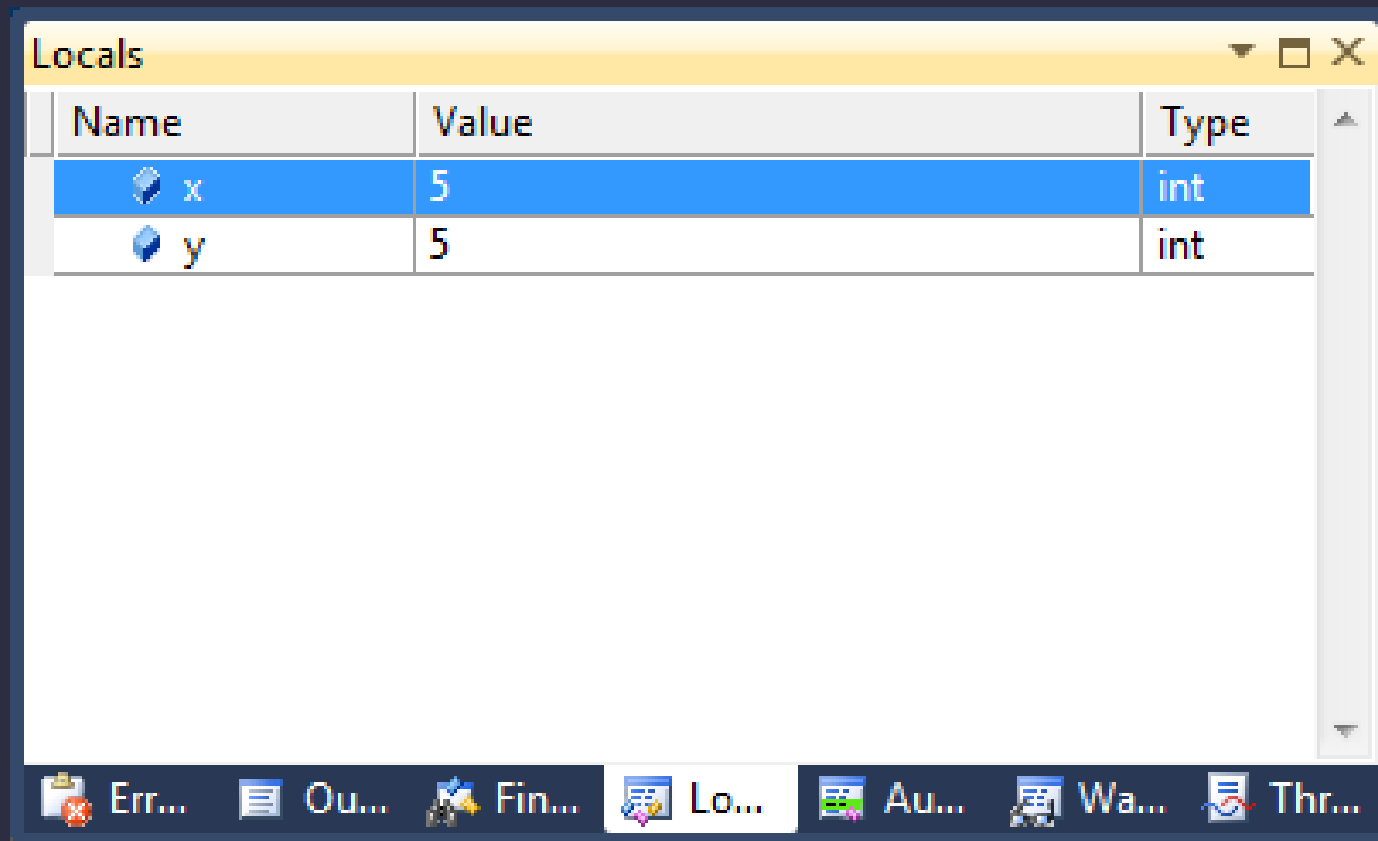
- Yukarıdaki Step Into ve Step Over tuşları ile kodun içinde hata ayıklamaya devam edebilirsiniz.
- Step Into, debug yaparken satırda bir fonksiyon varsa, o fonksiyonun içine girerek debug yapmaya devam etmeyi sağlar
- Step Over, debug yaparken fonksiyon varsa, içerisine girmeden devam etmeyi sağlar.



Hata Ayıklama Teknikleri

- Locals Ekranı

- Değişken ismi
- Değişken değeri
- Değişken türünü gösterir.



Name	Value	Type
x	5	int
y	5	int