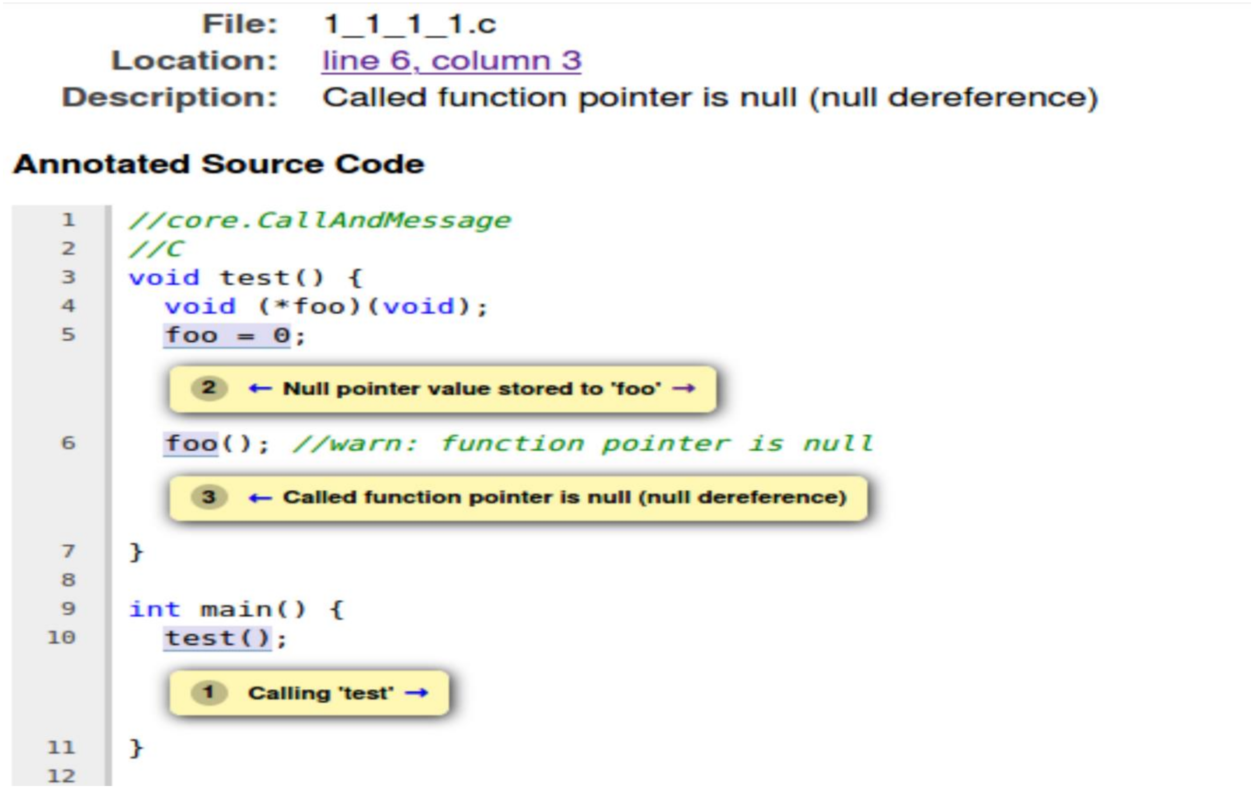
LAB 5: LLVM

Submit a screenshot of the HTML report generated by the scan-build tool, Identified the bugs, submit the corrected code and describe the fix.

Section 1.1.1.1

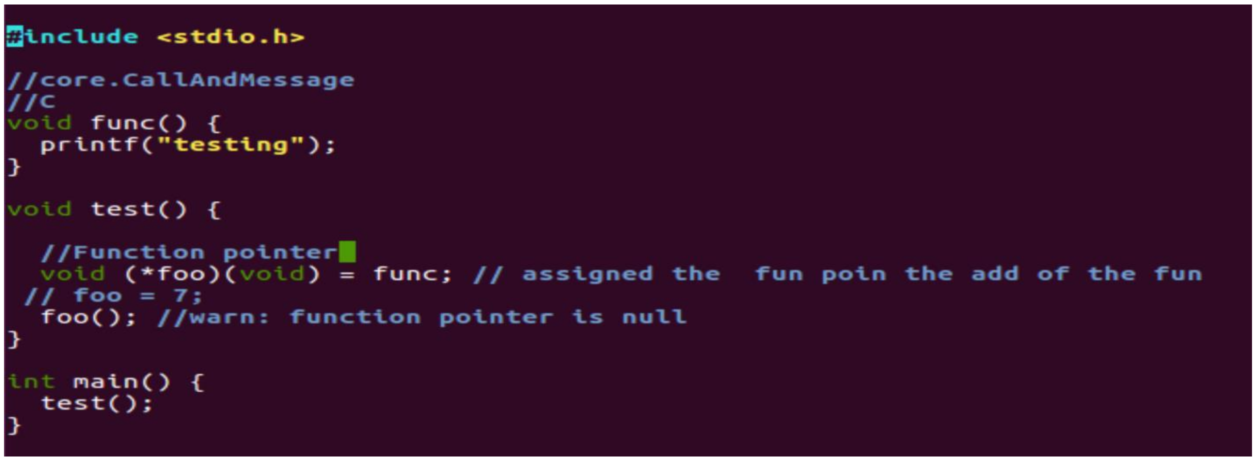


File: 1_1_1_1.c
Location: line 6, column 3
Description: Called function pointer is null (null dereference)

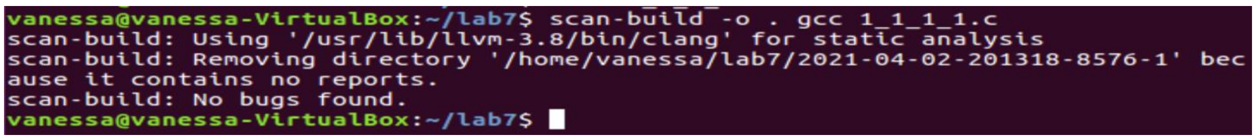
Annotated Source Code

```
//core.CallAndMessage
//C
void test() {
  void (*foo)(void);
  foo = 0;
```
2 ← Null pointer value stored to 'foo' →
```
  foo(); //warn: function pointer is null
```
3 ← Called function pointer is null (null dereference)
```
}

int main() {
  test();
```
1 Calling 'test' →
```
}
```

Solution



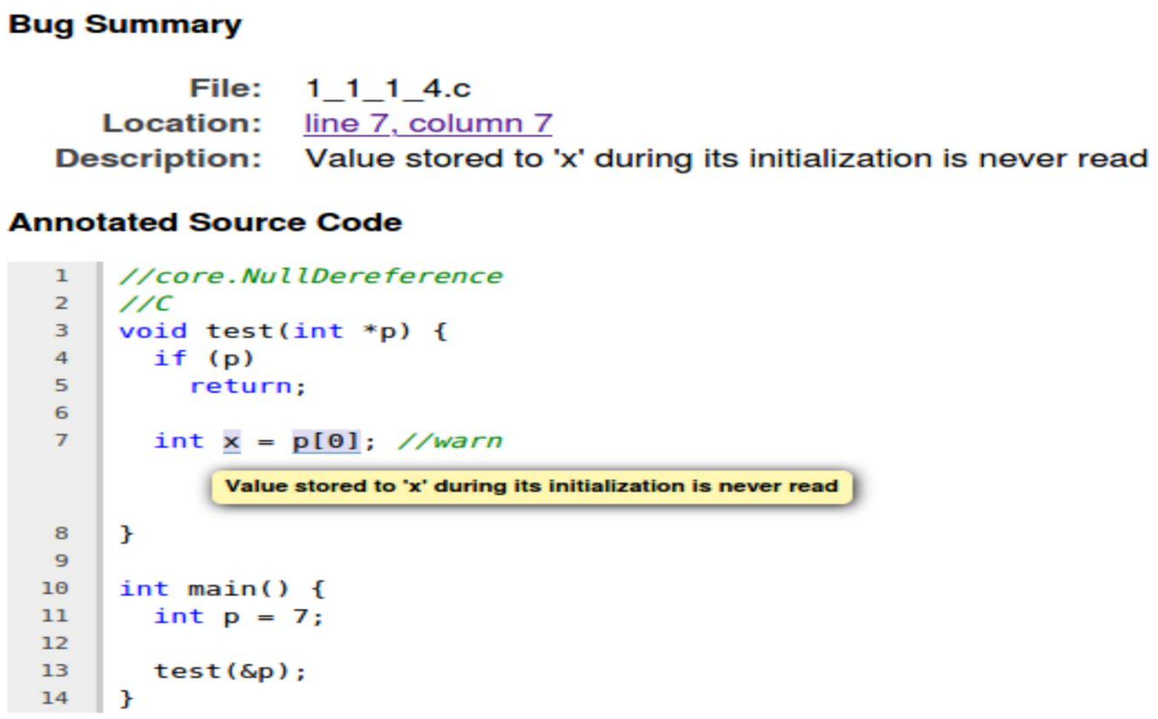
```
#include <stdio.h>

//core.CallAndMessage
//C
void func() {
  printf("testing");
}

void test() {

  //Function pointer
  void (*foo)(void) = func; // assigned the  fun poin the add of the fun
 // foo = 7;
  foo(); //warn: function pointer is null
}

int main() {
  test();
}
```



```
vanessa@vanessa-VirtualBox:~/lab7$ scan-build -o . gcc 1_1_1_1.c
scan-build: Using '/usr/lib/llvm-3.8/bin/clang' for static analysis
scan-build: Removing directory '/home/vanessa/lab7/2021-04-02-201318-8576-1' bec
ause it contains no reports.
scan-build: No bugs found.
vanessa@vanessa-VirtualBox:~/lab7$
```

Explanation:
The function pointer (*foo) was null i.e.. it was not pointing at anything yet. Then, I created a func function and had the function pointer point to the func function to resolve the error.

Section 1.1.1.4



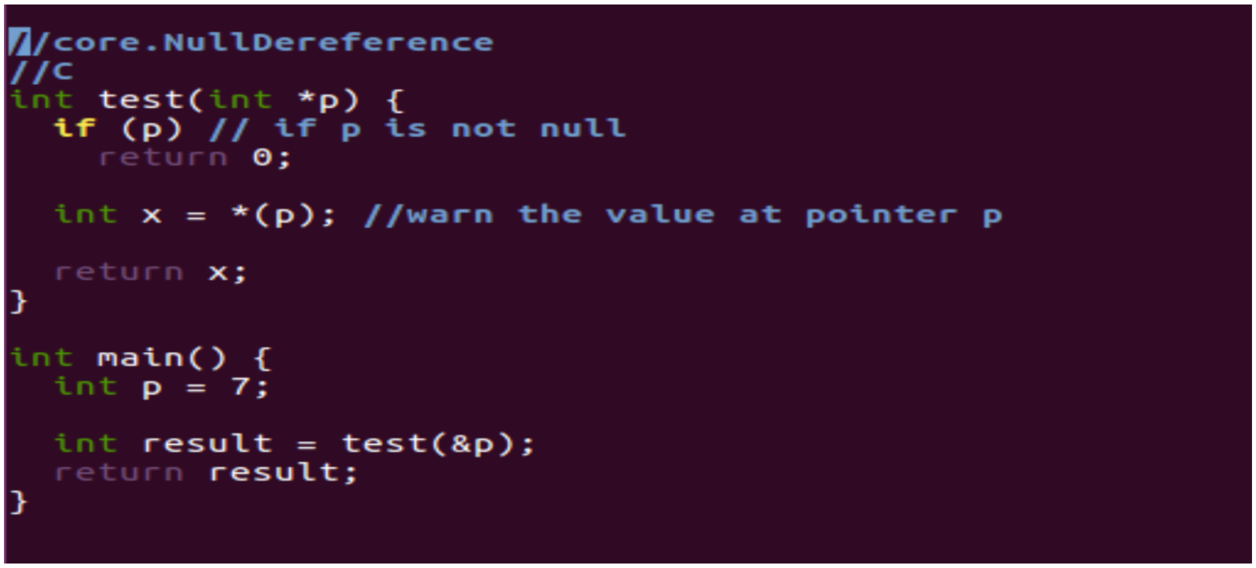
Bug Summary

File:	1_1_1_4.c
Location:	line 7, column 7
Description:	Value stored to 'x' during its initialization is never read

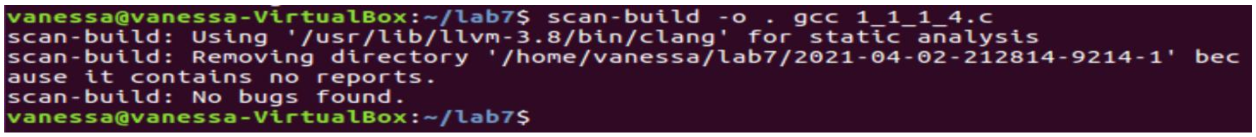
Annotated Source Code

```
//core.NullDereference
//C
void test(int *p) {
  if (p)
    return;

  int x = p[0]; //warn
        Value stored to 'x' during its initialization is never read
}

int main() {
  int p = 7;

  test(&p);
}
```

Solution



```
//core.NullDereference
//C
int test(int *p) {
  if (p) // if p is not null
    return 0;

  int x = *(p); //warn the value at pointer p

  return x;
}

int main() {
  int p = 7;

  int result = test(&p);
  return result;
}
```

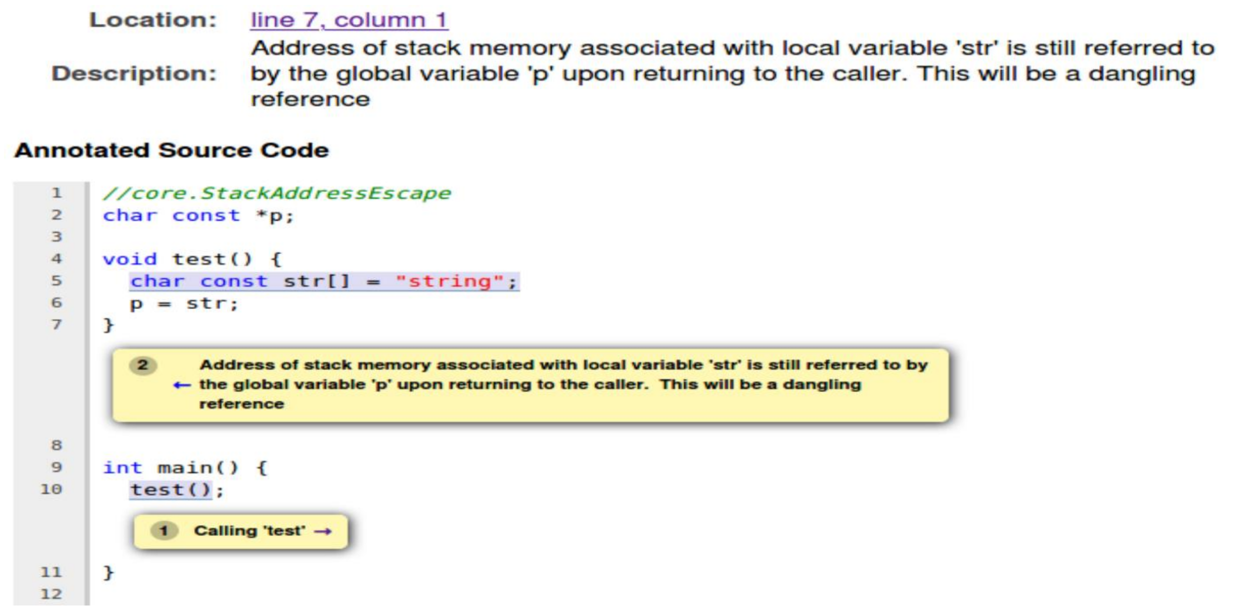


```
vanessa@vanessa-VirtualBox:~/lab7$ scan-build -o . gcc 1_1_1_4.c
scan-build: Using '/usr/lib/llvm-3.8/bin/clang' for static analysis
scan-build: Removing directory '/home/vanessa/lab7/2021-04-02-212814-9214-1' bec
ause it contains no reports.
scan-build: No bugs found.
vanessa@vanessa-VirtualBox:~/lab7$
```

Explanation

The variable x was defined but never used. Thus, I changed the value of x from p[0] to *(p). This means the value of x is now the value at pointer p. Then, I returned the value of x to make sure the variable x is being used.

Section 1.1.1.5



Location:	line 7, column 1
Description:	Address of stack memory associated with local variable 'str' is still referred to by the global variable 'p' upon returning to the caller. This will be a dangling reference

Annotated Source Code

```
//core.StackAddressEscape
char const *p;

void test() {
  char const str[] = "string";
  p = str;
}
```

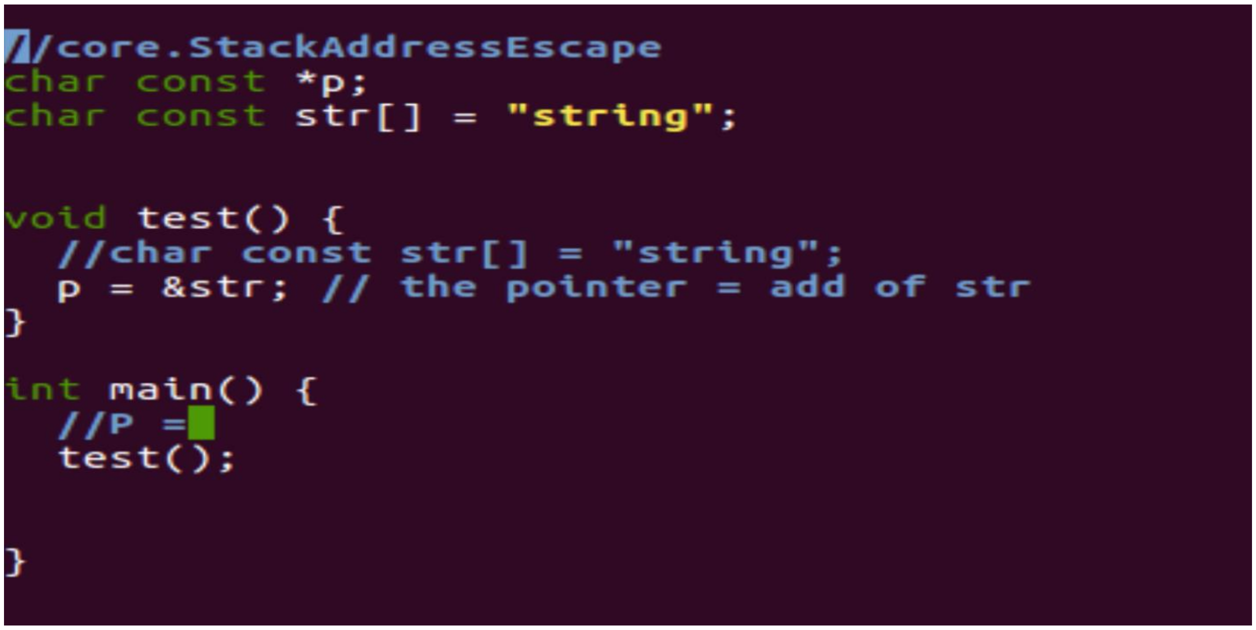
2 ← Address of stack memory associated with local variable 'str' is still referred to by the global variable 'p' upon returning to the caller. This will be a dangling reference

```

int main() {
  test();
```

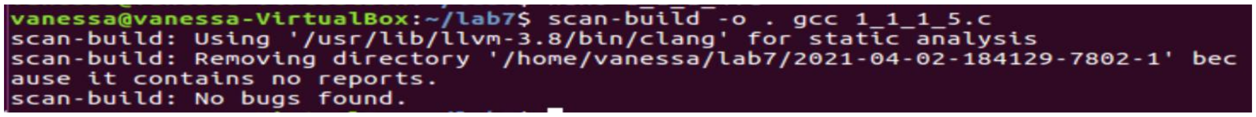
1 Calling 'test' →

```
}
```

Solution



```
//core.StackAddressEscape
char const *p;
char const str[] = "string";


void test() {
  //char const str[] = "string";
  p = &str; // the pointer = add of str
}

int main() {
  //P =
  test();


}
```

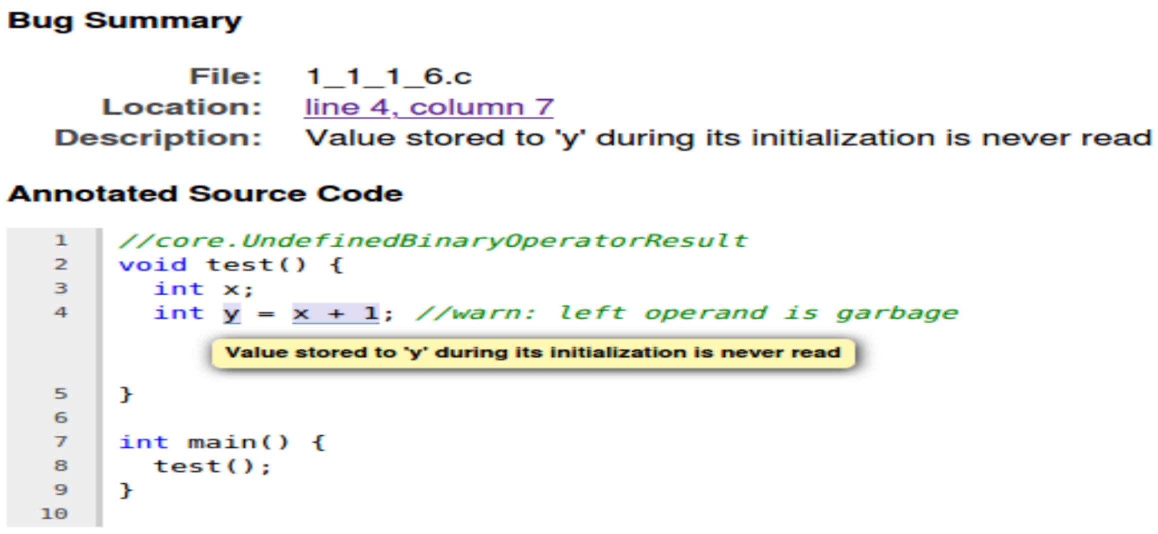


```
vanessa@vanessa-VirtualBox:~/lab7$ scan-build -o . gcc 1_1_1_5.c
scan-build: Using '/usr/lib/llvm-3.8/bin/clang' for static analysis
scan-build: Removing directory '/home/vanessa/lab7/2021-04-02-184129-7802-1' bec
ause it contains no reports.
scan-build: No bugs found.
```

Explanation:

The pointer P that’s pointing to the string (defined in the function) falls out of scope after the function ends because the string is not global. Thus, I made the string global. Now the string no longer falls out of scope when the function ends.

Section 1.1.1.6

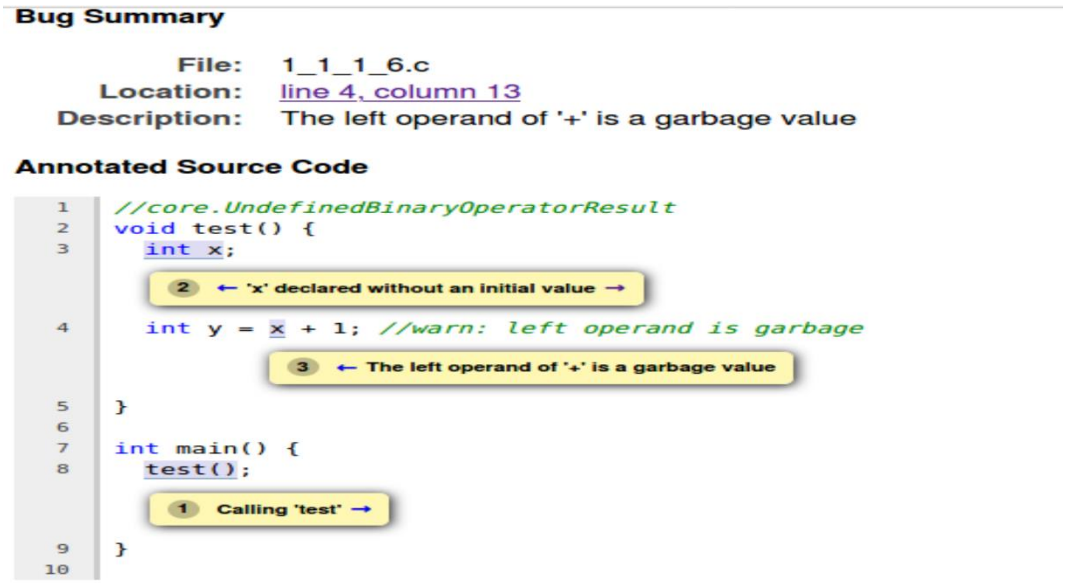


Bug Summary

File:	1_1_1_6.c
Location:	line 4, column 7
Description:	Value stored to 'y' during its initialization is never read

Annotated Source Code

```
//core.UndefinedBinaryOperatorResult
void test() {
  int x;
  int y = x + 1; //warn: left operand is garbage
      Value stored to 'y' during its initialization is never read
}

int main() {
  test();
}
```



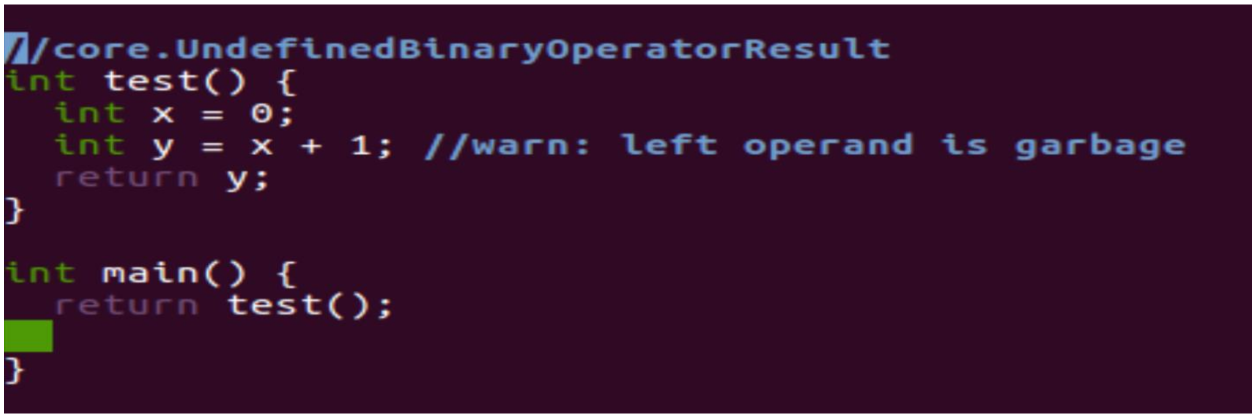
Bug Summary

File:	1_1_1_6.c
Location:	line 4, column 13
Description:	The left operand of '+' is a garbage value

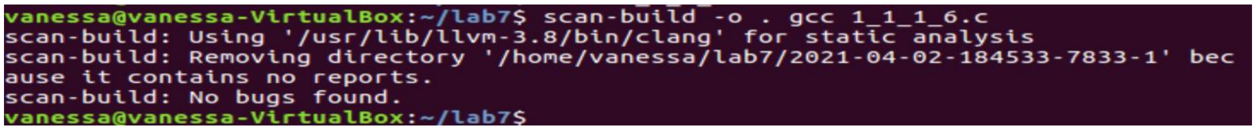
Annotated Source Code

```
//core.UndefinedBinaryOperatorResult
void test() {
  int x;
  2 ← 'x' declared without an initial value →
  int y = x + 1; //warn: left operand is garbage
        3 ← The left operand of '+' is a garbage value
}

int main() {
  test();
  1 Calling 'test' →
}
```

Solution



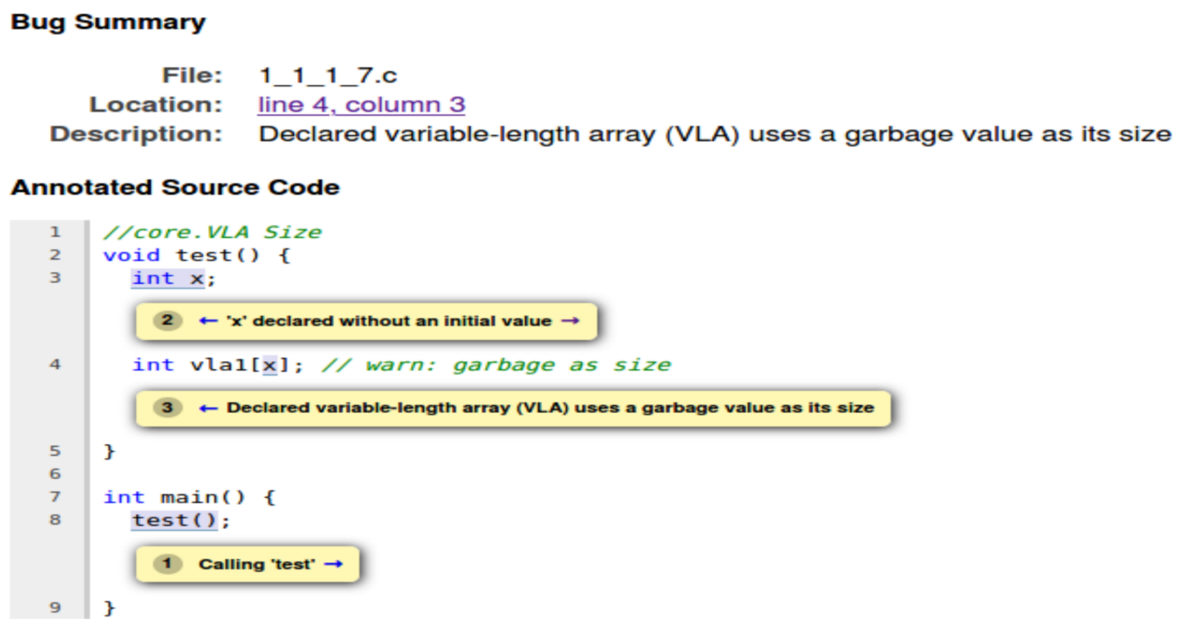
```
//core.UndefinedBinaryOperatorResult
int test() {
   int x = 0;
   int y = x + 1; //warn: left operand is garbage
   return y;
}

int main() {
   return test();

}
```



```
vanessa@vanessa-VirtualBox:~/lab7$ scan-build -o . gcc 1_1_1_6.c
scan-build: Using '/usr/lib/llvm-3.8/bin/clang' for static analysis
scan-build: Removing directory '/home/vanessa/lab7/2021-04-02-184533-7833-1' bec
ause it contains no reports.
scan-build: No bugs found.
vanessa@vanessa-VirtualBox:~/lab7$
```

Explanation:
X was not defined. Thus, I defined X to avoid the left error. Additionally, I changed the function from void to return int so that I can utilize all the variable defined in the test() function.

Section 1.1.1.7



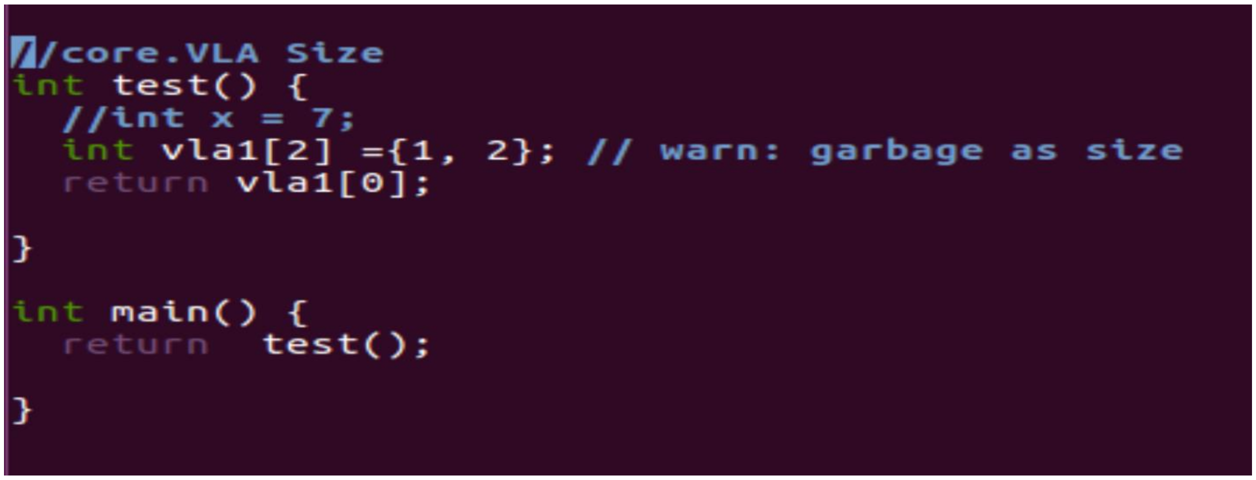
Bug Summary

File:	1_1_1_7.c
Location:	line 4, column 3
Description:	Declared variable-length array (VLA) uses a garbage value as its size

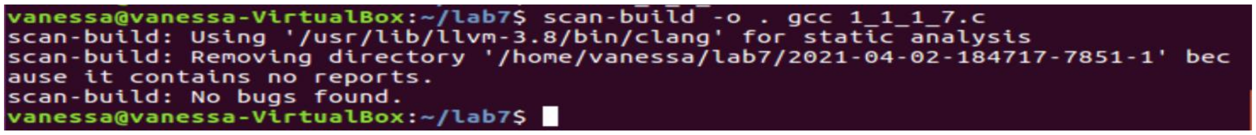
Annotated Source Code

```
//core.VLA Size
void test() {
  int x;
   2 ← 'x' declared without an initial value →
  int vla1[x]; // warn: garbage as size
   3 ← Declared variable-length array (VLA) uses a garbage value as its size
}

int main() {
  test();
   1 Calling 'test' →
}
```

Solution



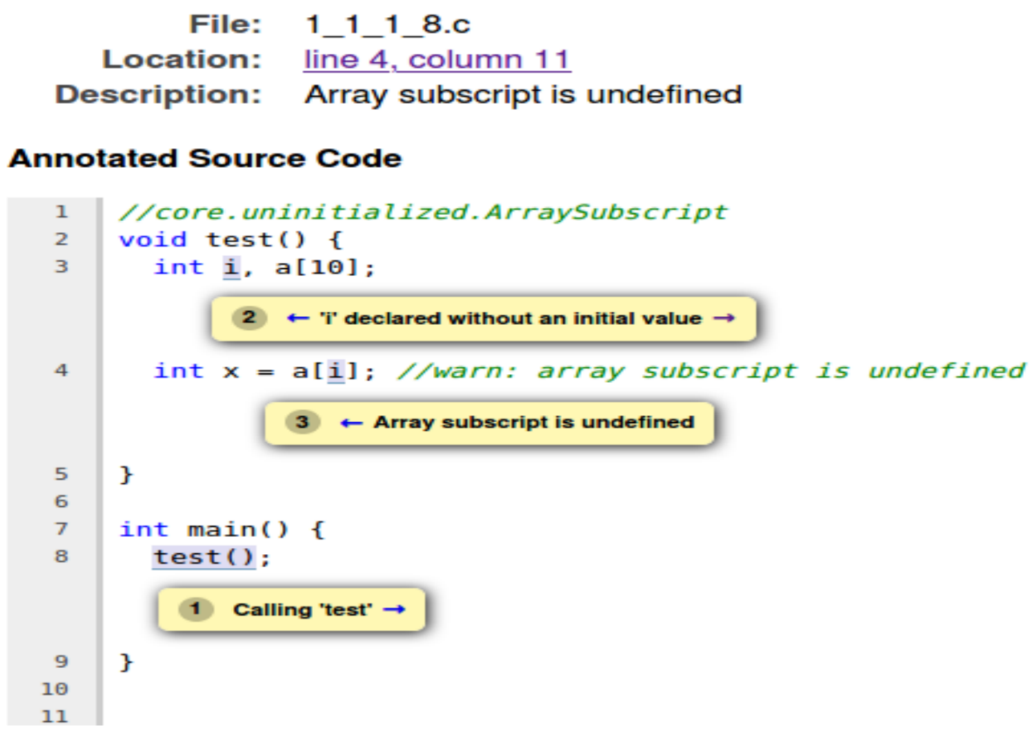
```
//core.VLA Size
int test() {
  //int x = 7;
  int vla1[2] ={1, 2}; // warn: garbage as size
  return vla1[0];

}

int main() {
  return  test();

}
```



```
vanessa@vanessa-VirtualBox:~/lab7$ scan-build -o . gcc 1_1_1_7.c
scan-build: Using '/usr/lib/llvm-3.8/bin/clang' for static analysis
scan-build: Removing directory '/home/vanessa/lab7/2021-04-02-184717-7851-1' bec
ause it contains no reports.
scan-build: No bugs found.
vanessa@vanessa-VirtualBox:~/lab7$
```

Explanation:
X was defined but not initialized ie, it stored a garbage value. Thus, affected vla1 array directly. Additionally, in C we cannot define arrays with variables like what was initially defined on the program. Simply, the program won't run. I could either use malloc to define the array. But, in this case, I simply initialized the array the correct way using a constant value 2.

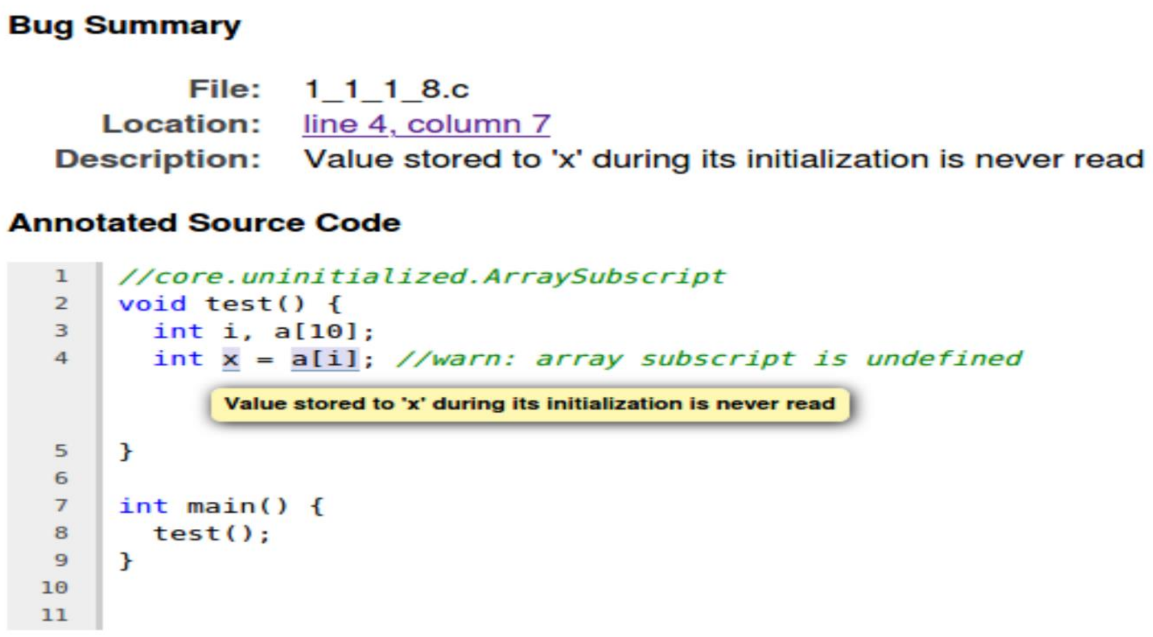
Section 1.1.1.8



File:	1_1_1_8.c
Location:	line 4, column 11
Description:	Array subscript is undefined

Annotated Source Code

```
//core.uninitialized.ArraySubscript
void test() {
  int i, a[10];
      2 ← 'i' declared without an initial value →
  int x = a[i]; //warn: array subscript is undefined
        3 ← Array subscript is undefined
}

int main() {
  test();
     1 Calling 'test' →
}
```

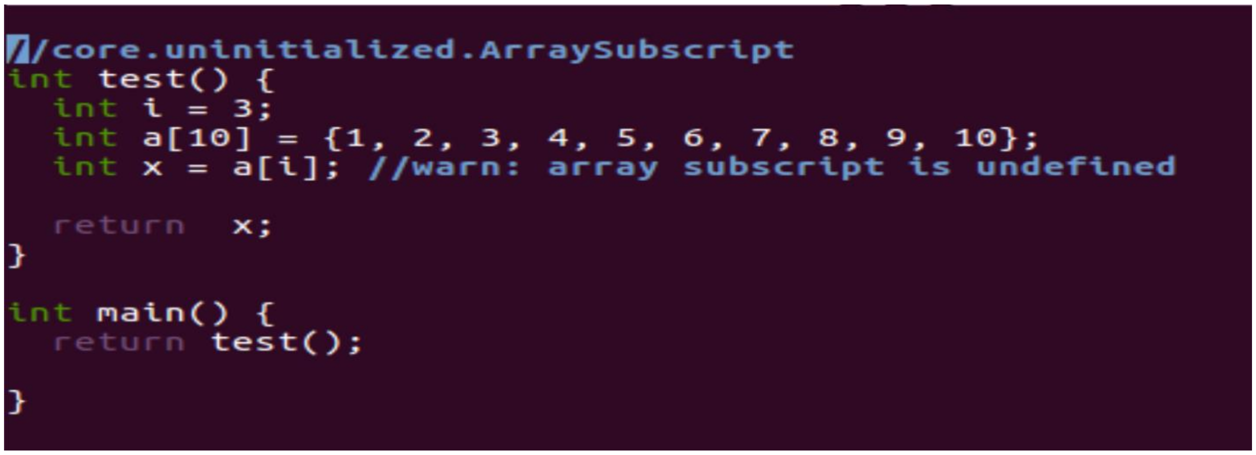


Bug Summary

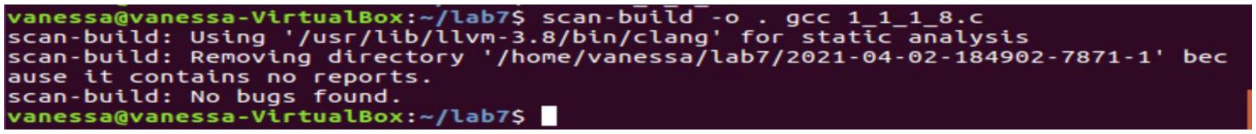
File: 1_1_1_8.c
Location: line 4, column 7
Description: Value stored to 'x' during its initialization is never read

Annotated Source Code

```
//core.uninitialized.ArraySubscript
void test() {
  int i, a[10];
  int x = a[i]; //warn: array subscript is undefined
        Value stored to 'x' during its initialization is never read

}

int main() {
  test();
}
```



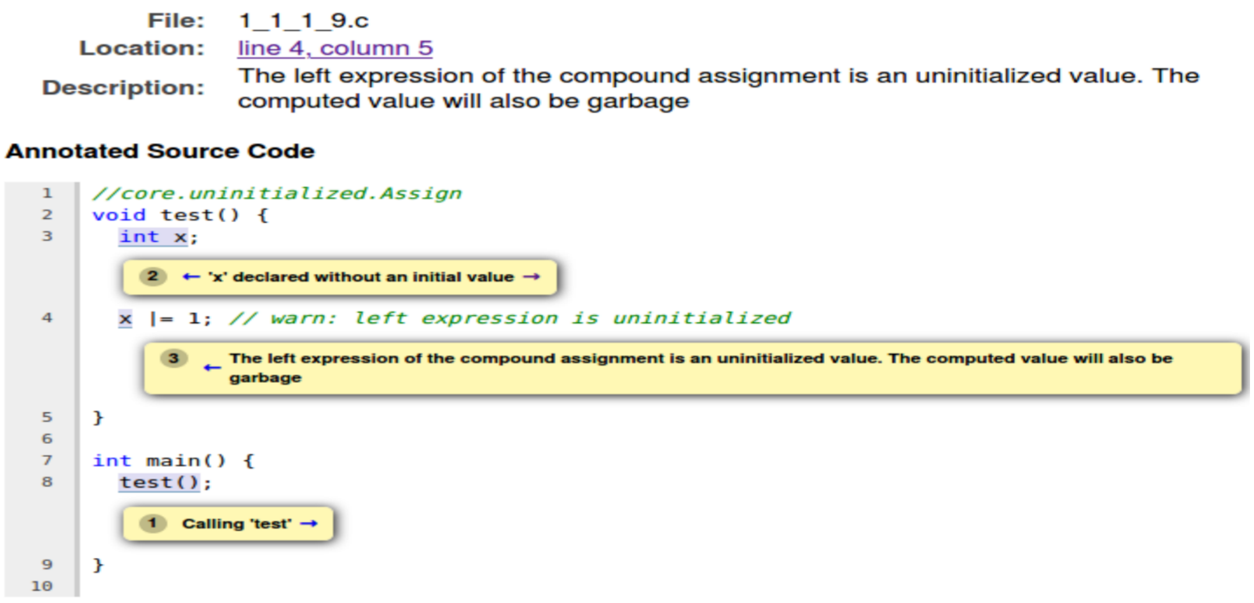
```
//core.uninitialized.ArraySubscript
int test() {
  int i = 3;
  int a[10] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
  int x = a[i]; //warn: array subscript is undefined

  return  x;
}

int main() {
  return test();

}
```



```
vanessa@vanessa-VirtualBox:~/lab7$ scan-build -o . gcc 1_1_1_8.c
scan-build: Using '/usr/lib/llvm-3.8/bin/clang' for static analysis
scan-build: Removing directory '/home/vanessa/lab7/2021-04-02-184902-7871-1' bec
ause it contains no reports.
scan-build: No bugs found.
vanessa@vanessa-VirtualBox:~/lab7$
```

Explanation:
The subscript of the array a[i] was not intialized i.e. I in a[i] was not initialized to provide the correct value to x. To fix this, I defined the subscript of the array i.
Additionally, x was not read i.e. it was not used. I then changed the function from void to int to return the value of x

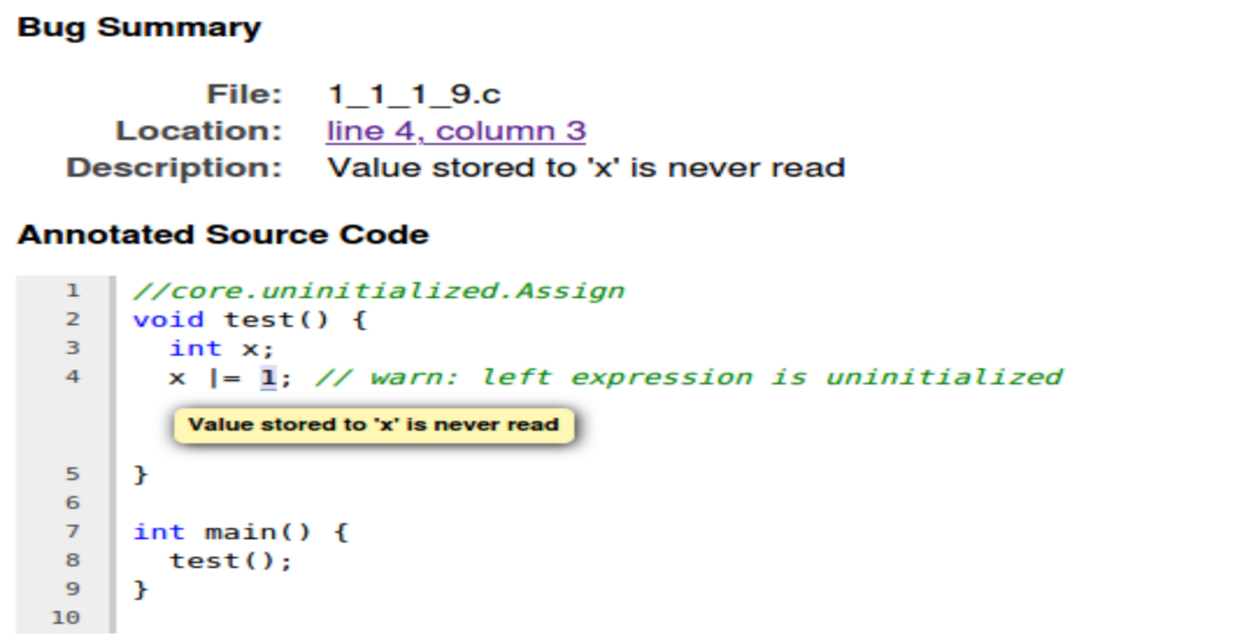
Section 1.1.1.9



File: 1_1_1_9.c
Location: line 4, column 5
Description: The left expression of the compound assignment is an uninitialized value. The computed value will also be garbage

Annotated Source Code

```
//core.uninitialized.Assign
void test() {
  int x;
    2 ← 'x' declared without an initial value →
  x |= 1; // warn: left expression is uninitialized
    3 ← The left expression of the compound assignment is an uninitialized value. The computed value will also be garbage
}

int main() {
  test();
    1 Calling 'test' →
}
```

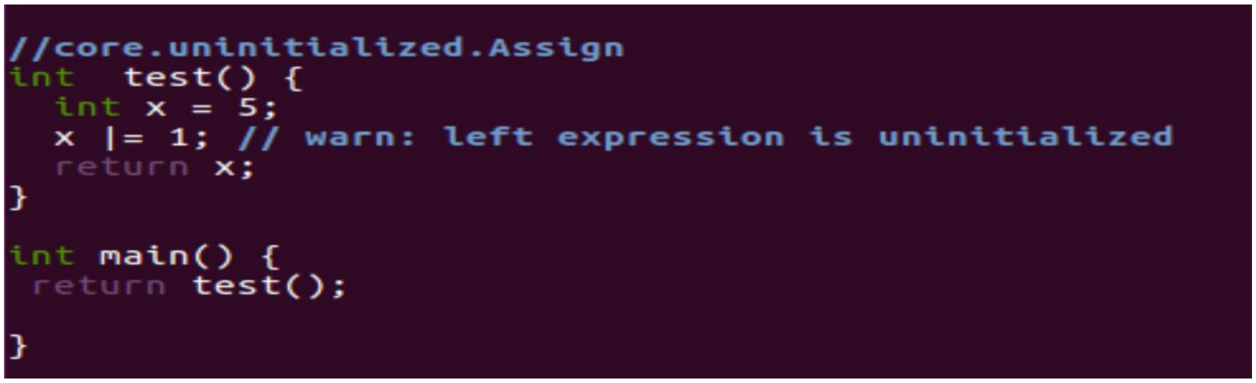


Bug Summary

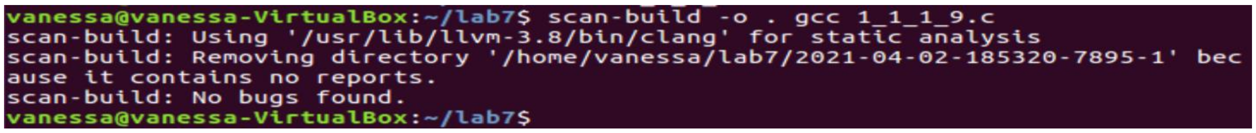
File: 1_1_1_9.c
Location: line 4, column 3
Description: Value stored to 'x' is never read

Annotated Source Code

```
//core.uninitialized.Assign
void test() {
  int x;
  x |= 1; // warn: left expression is uninitialized
    Value stored to 'x' is never read
}

int main() {
  test();
}
```



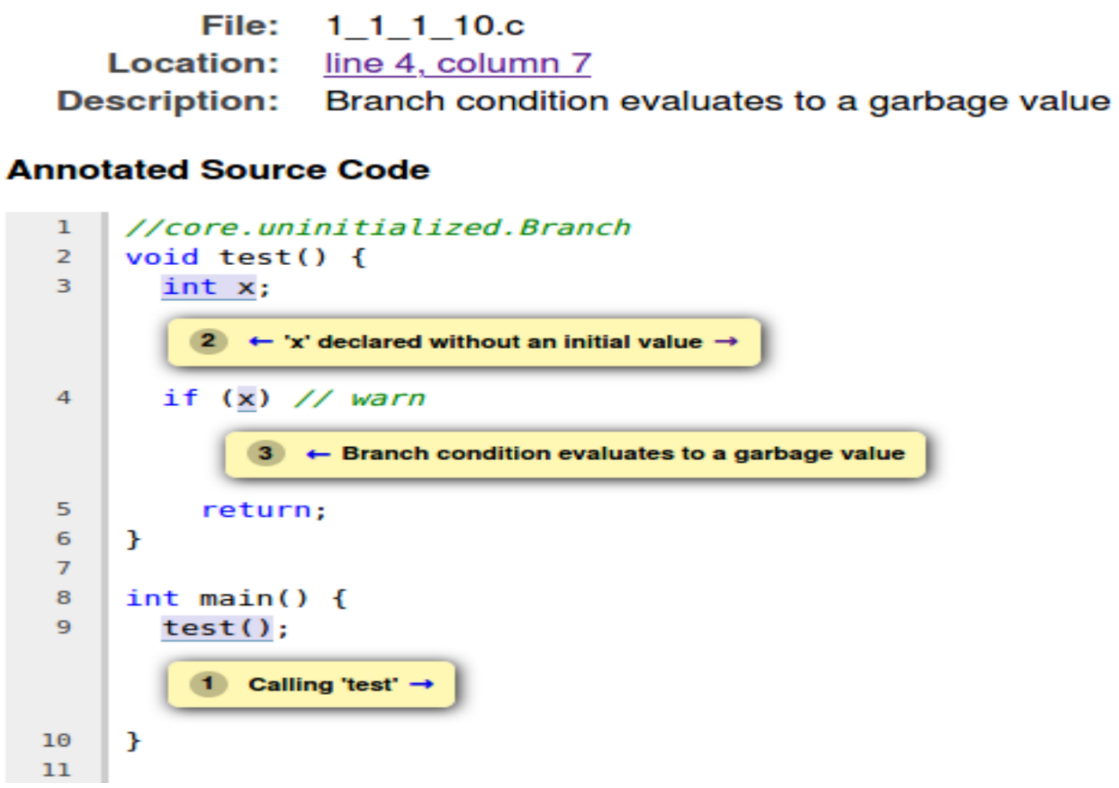
```
//core.uninitialized.Assign
int  test() {
   int x = 5;
   x |= 1; // warn: left expression is uninitialized
   return x;
}

int main() {
 return test();

}
```



```
vanessa@vanessa-VirtualBox:~/lab7$ scan-build -o . gcc 1_1_1_9.c
scan-build: Using '/usr/lib/llvm-3.8/bin/clang' for static analysis
scan-build: Removing directory '/home/vanessa/lab7/2021-04-02-185320-7895-1' bec
ause it contains no reports.
scan-build: No bugs found.
vanessa@vanessa-VirtualBox:~/lab7$
```

Explanation:
X was declared without an initial value. Thus, I initialized x with 5. Changed the function to return int so that I could return x.

Section 1.1.1.10

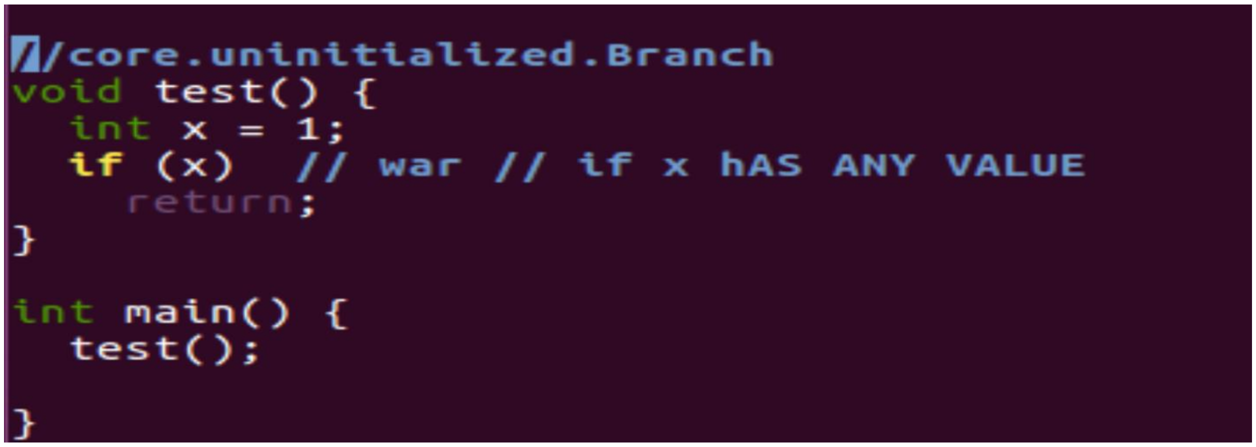


File:	1_1_1_10.c
Location:	line 4, column 7
Description:	Branch condition evaluates to a garbage value

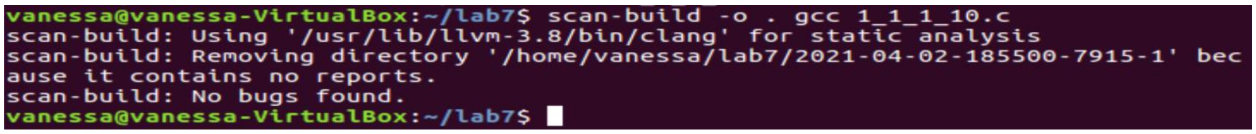
Annotated Source Code

```
//core.uninitialized.Branch
void test() {
  int x;
    2  ← 'x' declared without an initial value →
  if (x) // warn
      3  ← Branch condition evaluates to a garbage value
    return;
}

int main() {
  test();
    1  Calling 'test' →
}
```

Solution



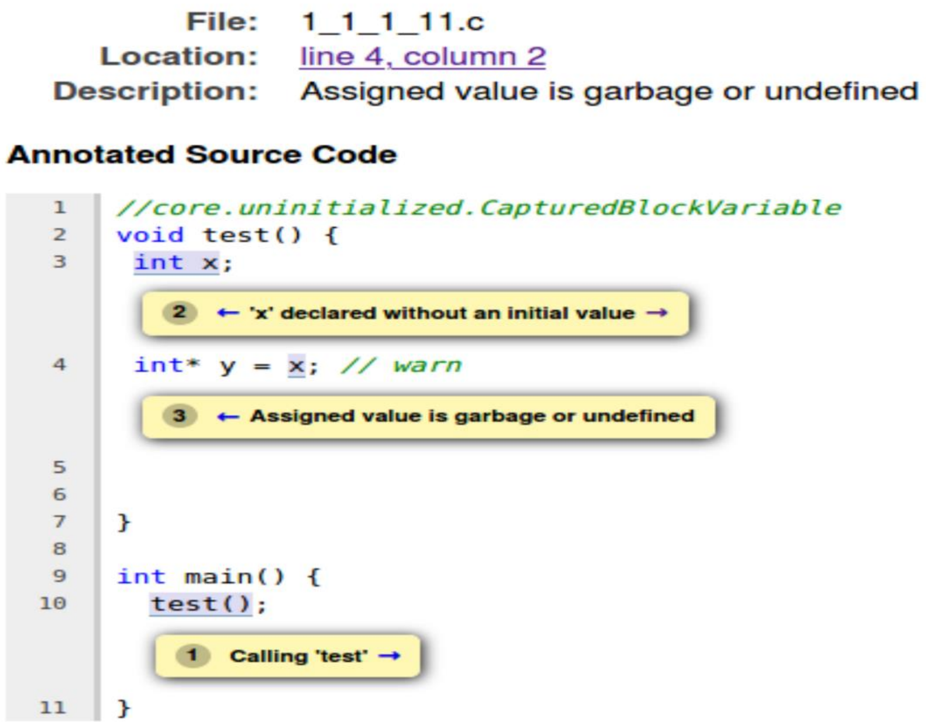
```
//core.uninitialized.Branch
void test() {
  int x = 1;
  if (x)  // war // if x hAS ANY VALUE
    return;
}

int main() {
  test();

}
```



```
vanessa@vanessa-VirtualBox:~/lab7$ scan-build -o . gcc 1_1_1_10.c
scan-build: Using '/usr/lib/llvm-3.8/bin/clang' for static analysis
scan-build: Removing directory '/home/vanessa/lab7/2021-04-02-185500-7915-1' bec
ause it contains no reports.
scan-build: No bugs found.
vanessa@vanessa-VirtualBox:~/lab7$
```

Explanation:
The function had an if function if x has any value, return. But, the variable x itself was not initialized. Thus, I fixed it by initializing it with a constant one.

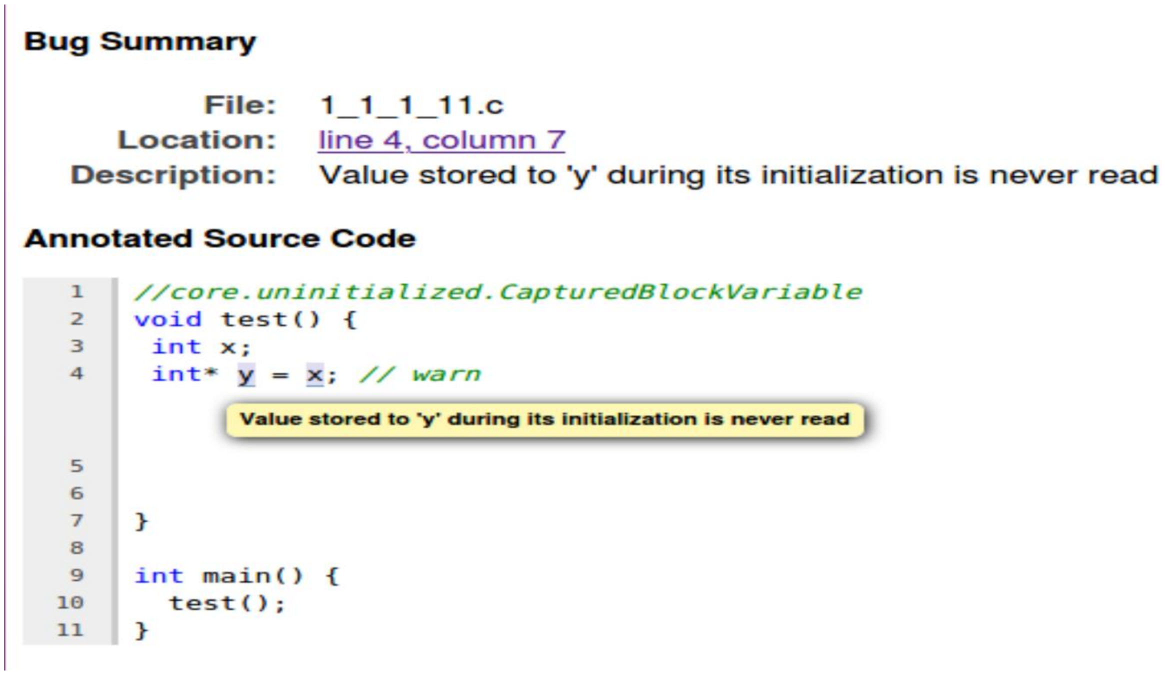
Section 1.1.1.11



File: 1_1_1_11.c
Location: line 4, column 2
Description: Assigned value is garbage or undefined

Annotated Source Code

```
//core.uninitialized.CapturedBlockVariable
void test() {
 int x;
   2 ← 'x' declared without an initial value →
 int* y = x; // warn
   3 ← Assigned value is garbage or undefined



}

int main() {
  test();
    1 Calling 'test' →
}
```



Bug Summary

File: 1_1_1_11.c
Location: line 4, column 7
Description: Value stored to 'y' during its initialization is never read

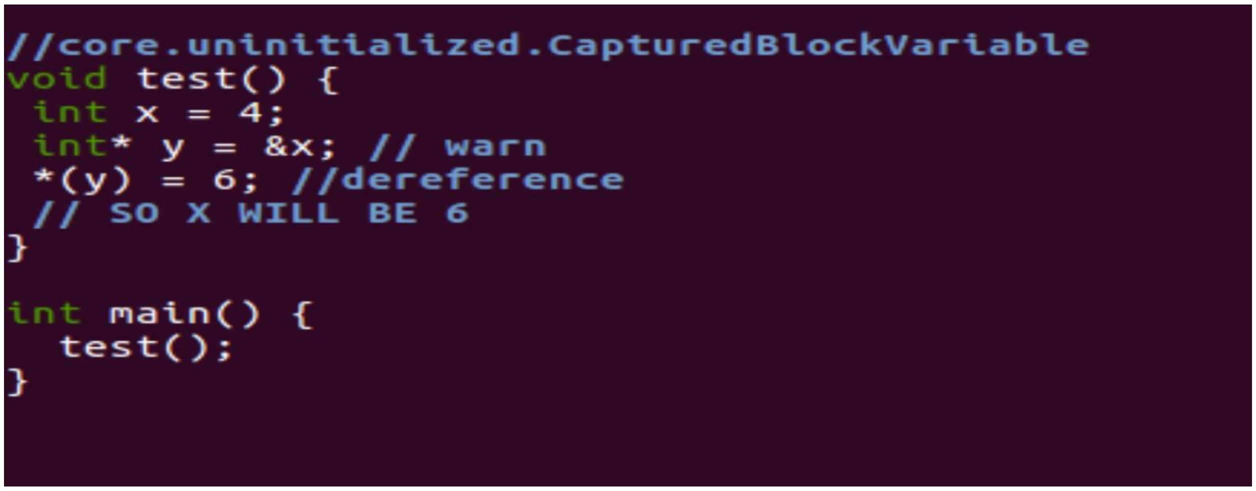
Annotated Source Code

```
//core.uninitialized.CapturedBlockVariable
void test() {
 int x;
 int* y = x; // warn
```

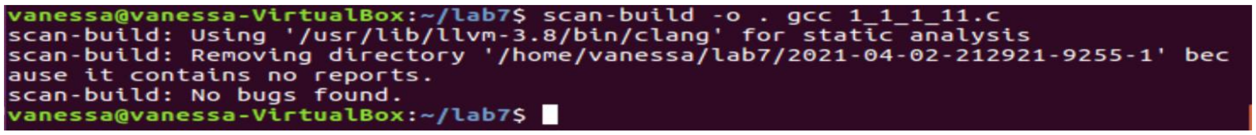
Value stored to 'y' during its initialization is never read

```


}

int main() {
  test();
}
```

Solution:



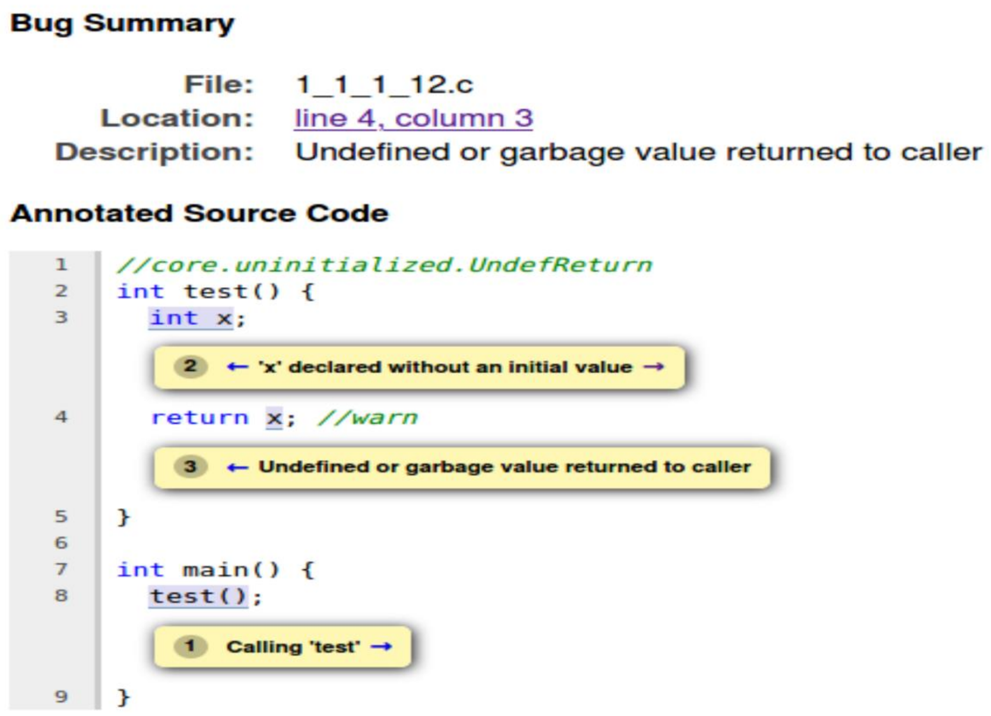
```
//core.uninitialized.CapturedBlockVariable
void test() {
 int x = 4;
 int* y = &x; // warn
 *(y) = 6; //dereference
 // SO X WILL BE 6
}

int main() {
  test();
}
```



```
vanessa@vanessa-VirtualBox:~/lab7$ scan-build -o . gcc 1_1_1_11.c
scan-build: Using '/usr/lib/llvm-3.8/bin/clang' for static analysis
scan-build: Removing directory '/home/vanessa/lab7/2021-04-02-212921-9255-1' bec
ause it contains no reports.
scan-build: No bugs found.
vanessa@vanessa-VirtualBox:~/lab7$
```

Explanation:
X was declared without an initial value. Thus I made x = 4. Then made pointer y be the address of x. Finally dereferenced y to 6 i.e. by the end of function test(), the value of x will be 6.

Section 1.1.1.12



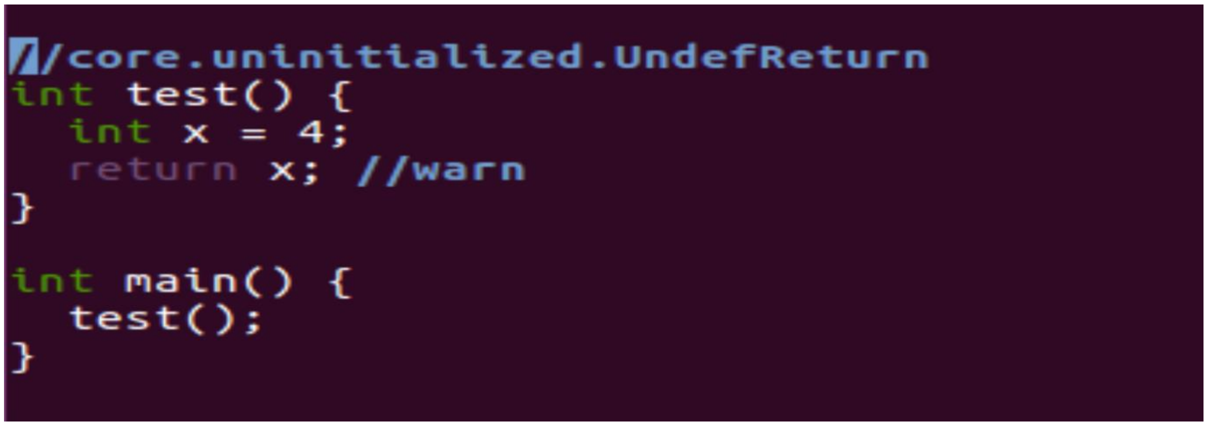
Bug Summary

File: 1_1_1_12.c
Location: line 4, column 3
Description: Undefined or garbage value returned to caller

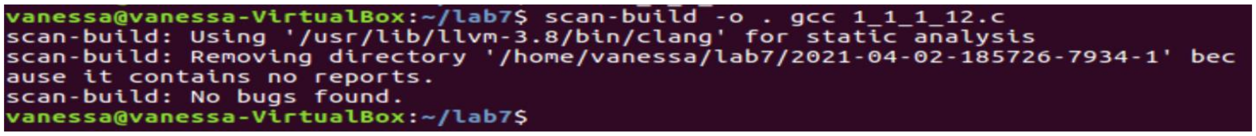
Annotated Source Code

```
//core.uninitialized.UndefReturn
int test() {
  int x;
    2 ← 'x' declared without an initial value →
  return x; //warn
    3 ← Undefined or garbage value returned to caller
}

int main() {
  test();
    1 Calling 'test' →
}
```

Solution



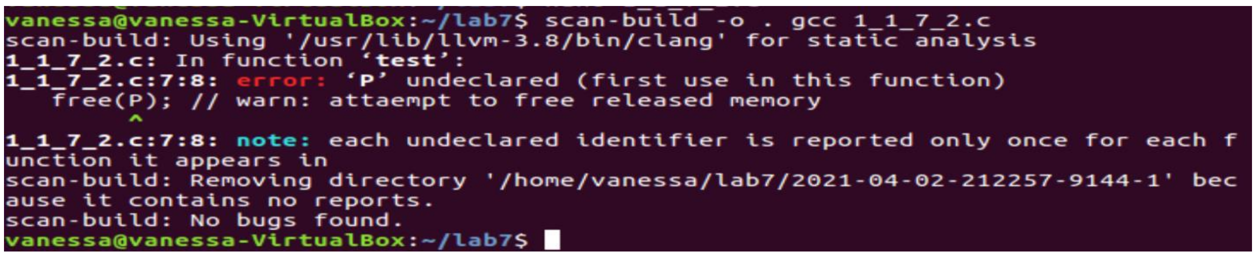
```
//core.uninitialized.UndefReturn
int test() {
  int x = 4;
  return x; //warn
}

int main() {
  test();
}
```



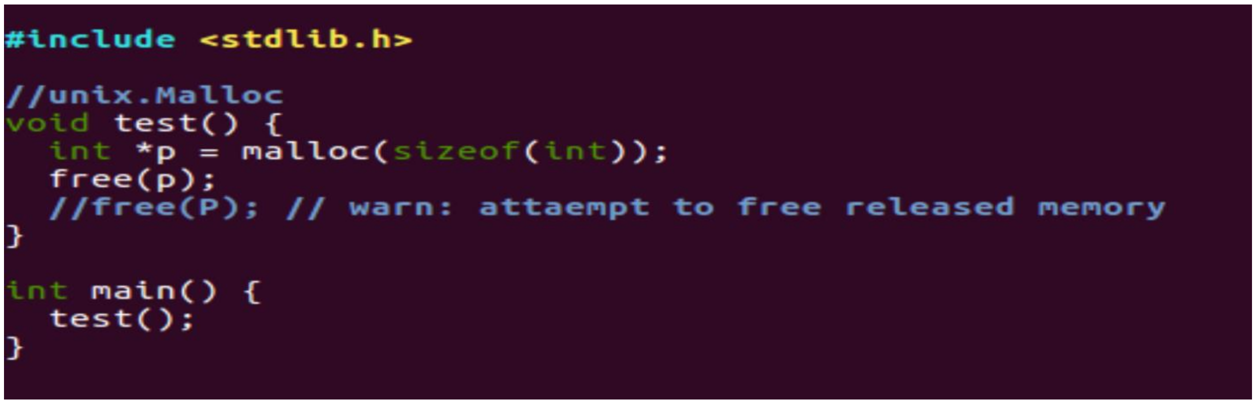
```
vanessa@vanessa-VirtualBox:~/lab7$ scan-build -o . gcc 1_1_1_12.c
scan-build: Using '/usr/lib/llvm-3.8/bin/clang' for static analysis
scan-build: Removing directory '/home/vanessa/lab7/2021-04-02-185726-7934-1' bec
ause it contains no reports.
scan-build: No bugs found.
vanessa@vanessa-VirtualBox:~/lab7$
```

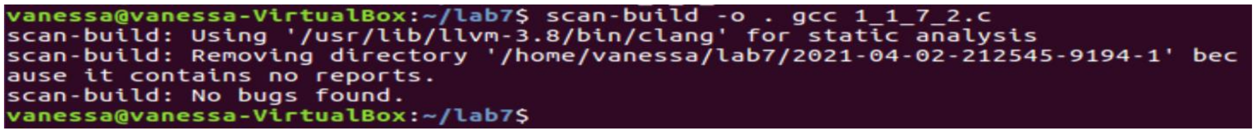
Explanation:
Int x was declared without an initial value. Thus, I initialized x with 4. Then changed the function from void to int in order to return x.

Section 1.1.7.2


```
vanessa@vanessa-VirtualBox:~/lab7$ scan-build -o . gcc 1_1_7_2.c
scan-build: Using '/usr/lib/llvm-3.8/bin/clang' for static analysis
1_1_7_2.c: In function 'test':
1_1_7_2.c:7:8: error: 'P' undeclared (first use in this function)
   free(P); // warn: attaempt to free released memory
        ^
1_1_7_2.c:7:8: note: each undeclared identifier is reported only once for each f
unction it appears in
scan-build: Removing directory '/home/vanessa/lab7/2021-04-02-212257-9144-1' bec
ause it contains no reports.
scan-build: No bugs found.
vanessa@vanessa-VirtualBox:~/lab7$
```

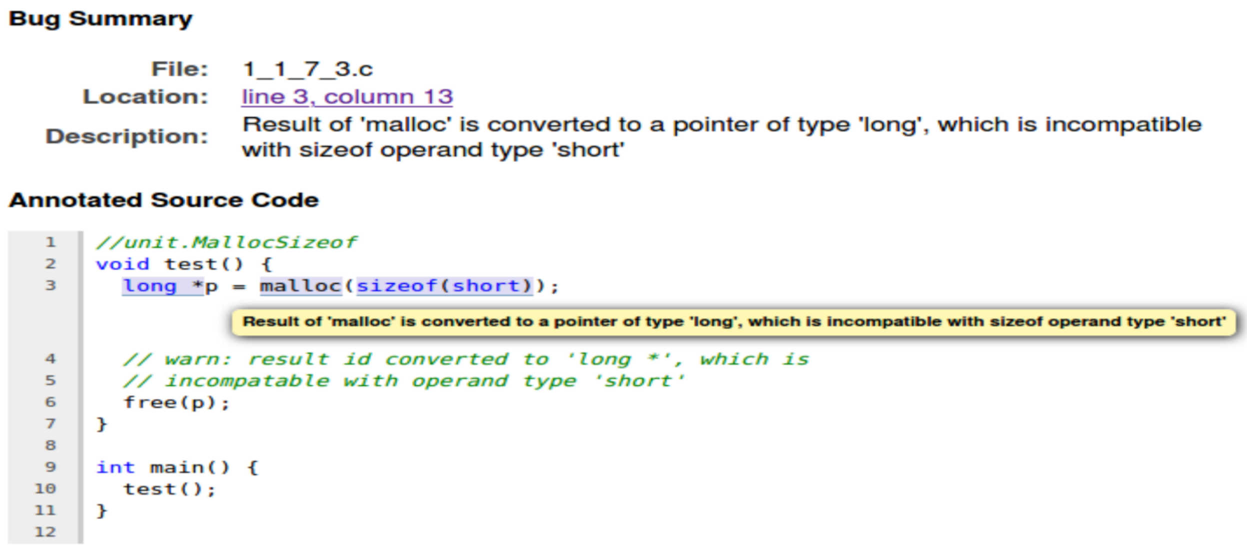
There was no html file created as the error was detected by the gcc compiler.


```
#include <stdlib.h>

//unix.Malloc
void test() {
  int *p = malloc(sizeof(int));
  free(p);
  //free(P); // warn: attaempt to free released memory
}

int main() {
  test();
}
```


```
vanessa@vanessa-VirtualBox:~/lab7$ scan-build -o . gcc 1_1_7_2.c
scan-build: Using '/usr/lib/llvm-3.8/bin/clang' for static analysis
scan-build: Removing directory '/home/vanessa/lab7/2021-04-02-212545-9194-1' bec
ause it contains no reports.
scan-build: No bugs found.
vanessa@vanessa-VirtualBox:~/lab7$
```

Explanation
Deleted the extra line that was freeing memory of the pointer p. This line was trying to delete something that was already deleted.

Section 1.1.7.3

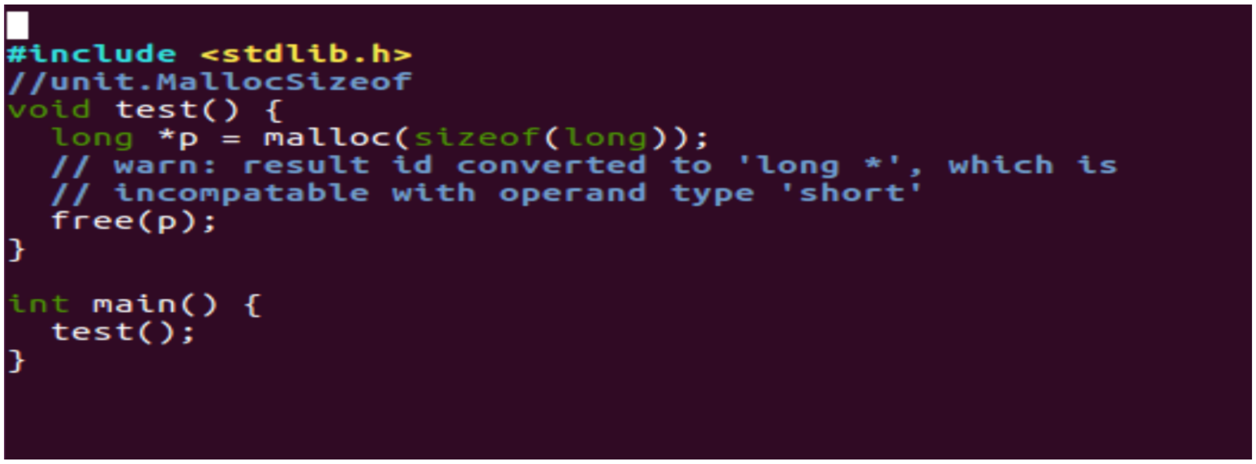

Bug Summary

File:	1_1_7_3.c
Location:	line 3, column 13
Description:	Result of 'malloc' is converted to a pointer of type 'long', which is incompatible with sizeof operand type 'short'

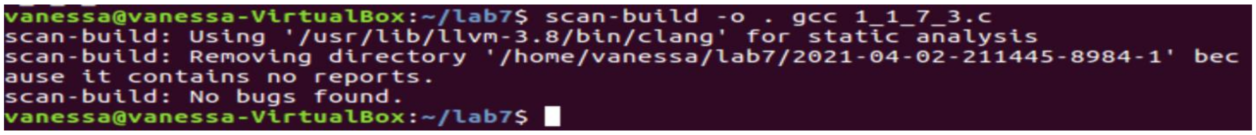
Annotated Source Code

```
//unit.MallocSizeof
void test() {
  long *p = malloc(sizeof(short));
            Result of 'malloc' is converted to a pointer of type 'long', which is incompatible with sizeof operand type 'short'
  // warn: result id converted to 'long *', which is
  // incompatable with operand type 'short'
  free(p);
}

int main() {
  test();
}
```

solution



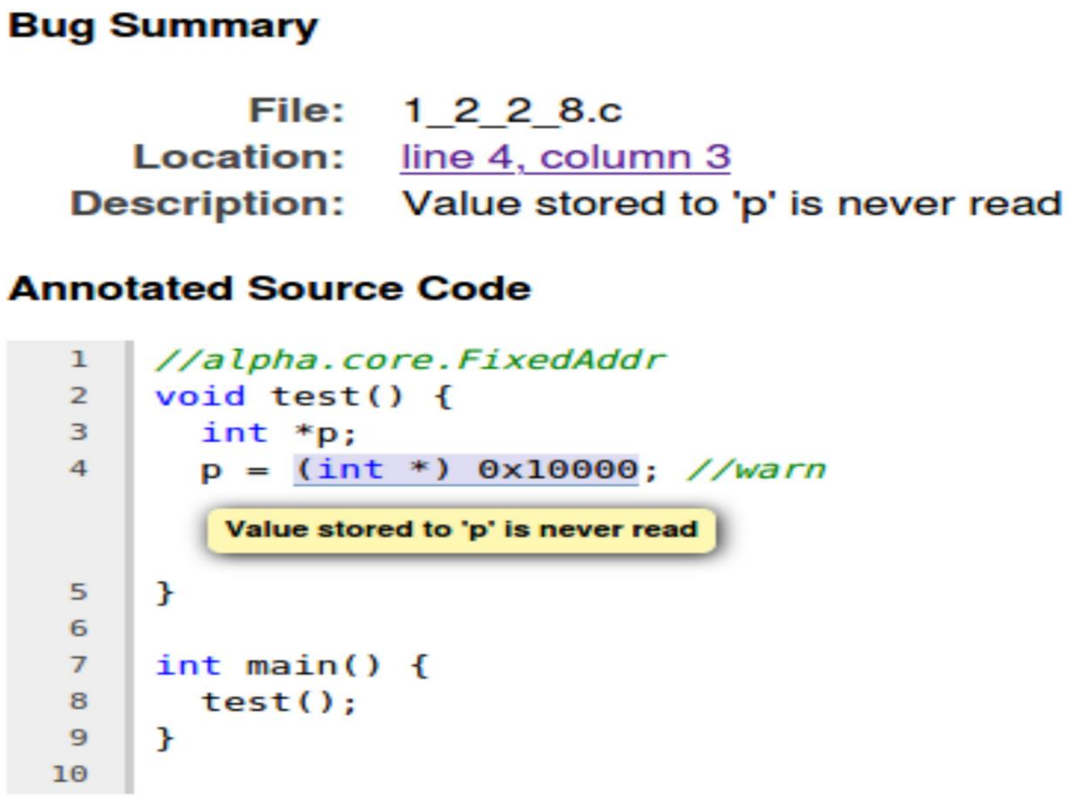
```
#include <stdlib.h>
//unit.MallocSizeof
void test() {
  long *p = malloc(sizeof(long));
  // warn: result id converted to 'long *', which is
  // incompatable with operand type 'short'
  free(p);
}

int main() {
  test();
}
```



```
vanessa@vanessa-VirtualBox:~/lab7$ scan-build -o . gcc 1_1_7_3.c
scan-build: Using '/usr/lib/llvm-3.8/bin/clang' for static analysis
scan-build: Removing directory '/home/vanessa/lab7/2021-04-02-211445-8984-1' bec
ause it contains no reports.
scan-build: No bugs found.
vanessa@vanessa-VirtualBox:~/lab7$
```

Explanation:
Initially, the type of the pointer was long but the assigned size was set to short. This was a mismatch. Thus, I matched the type of the pointer p to the type of its assigned size.

Section 1.2.2.8



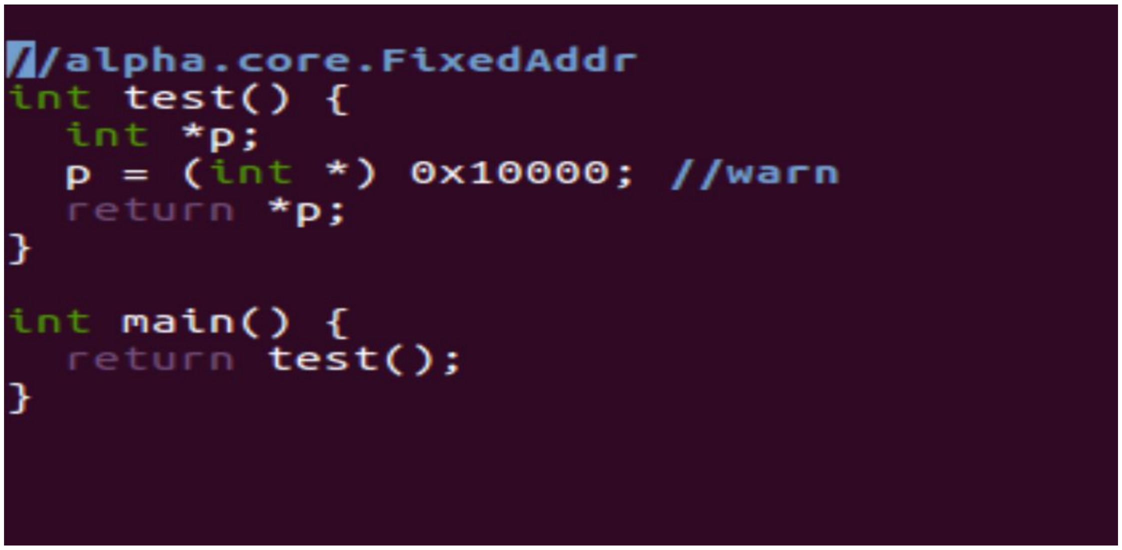
Bug Summary

File:	1_2_2_8.c
Location:	line 4, column 3
Description:	Value stored to 'p' is never read

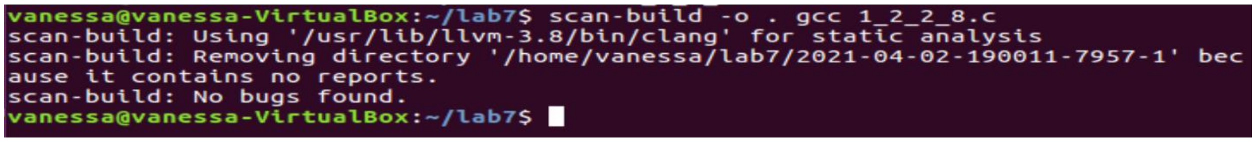
Annotated Source Code

```
1   //alpha.core.FixedAddr
2   void test() {
3     int *p;
4     p = (int *) 0x10000; //warn
        Value stored to 'p' is never read
5   }
6
7   int main() {
8     test();
9   }
10
```

Solution



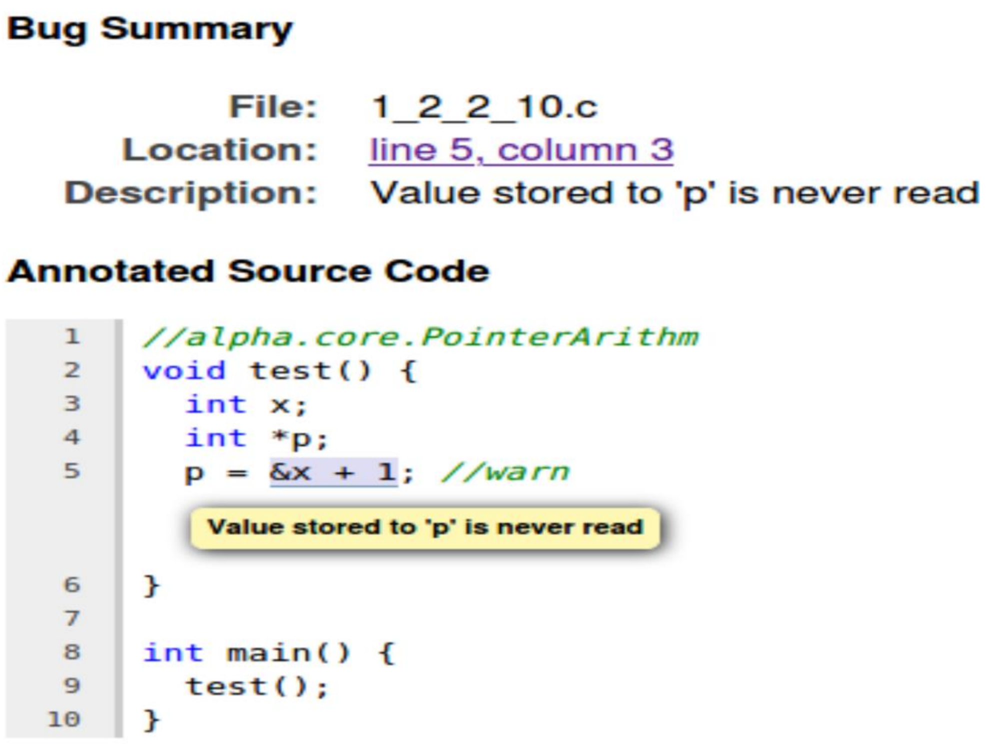
```
//alpha.core.FixedAddr
int test() {
  int *p;
  p = (int *) 0x10000; //warn
  return *p;
}

int main() {
  return test();
}
```



```
vanessa@vanessa-VirtualBox:~/lab7$ scan-build -o . gcc 1_2_2_8.c
scan-build: Using '/usr/lib/llvm-3.8/bin/clang' for static analysis
scan-build: Removing directory '/home/vanessa/lab7/2021-04-02-190011-7957-1' bec
ause it contains no reports.
scan-build: No bugs found.
vanessa@vanessa-VirtualBox:~/lab7$
```

Solution
The error was the pointer p was never used. I returned pointer p to fix the issue.

Section 1.2.2.10



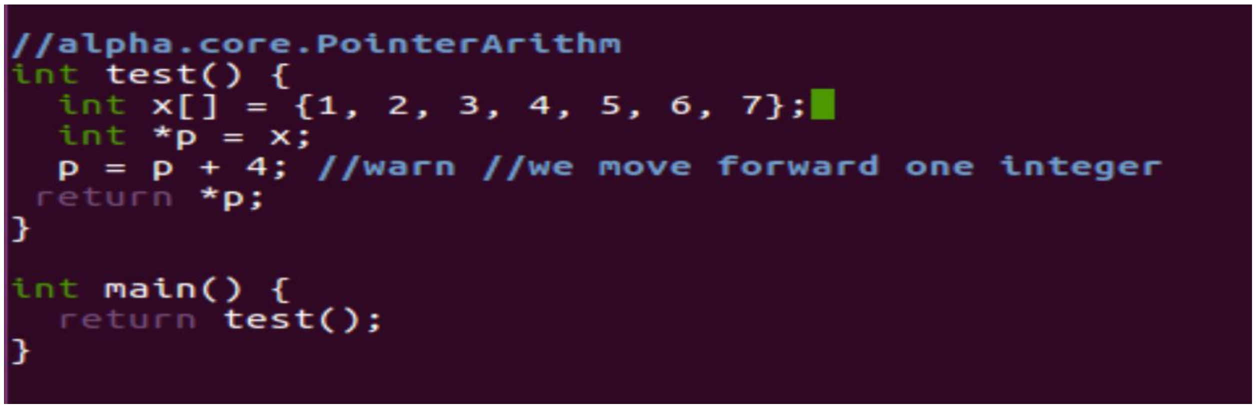
Bug Summary

File: 1_2_2_10.c
Location: line 5, column 3
Description: Value stored to 'p' is never read

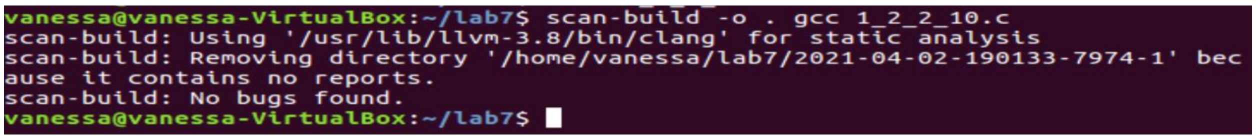
Annotated Source Code

```
//alpha.core.PointerArithm
void test() {
  int x;
  int *p;
  p = &x + 1; //warn

     Value stored to 'p' is never read

}

int main() {
  test();
}
```

Solution



```
//alpha.core.PointerArithm
int test() {
  int x[] = {1, 2, 3, 4, 5, 6, 7};
  int *p = x;
  p = p + 4; //warn //we move forward one integer
 return *p;
}

int main() {
  return test();
}
```



```
vanessa@vanessa-VirtualBox:~/lab7$ scan-build -o . gcc 1_2_2_10.c
scan-build: Using '/usr/lib/llvm-3.8/bin/clang' for static analysis
scan-build: Removing directory '/home/vanessa/lab7/2021-04-02-190133-7974-1' bec
ause it contains no reports.
scan-build: No bugs found.
vanessa@vanessa-VirtualBox:~/lab7$
```

Explanation
The error was pointer was never read but x was not initialized as well. Thus, I initialized x and returned pointer p to fix the issue.