# 《软件安全技术》综合实验

## 实验一：C/C++代码静态分析

## 作者：谭九鼎

## 学号：522024320137

## 时间：2024年12月

**Flawfinder官方网站**：<https://dwheeler.com/flawfinder/>

**示例C程序**：<https://dwheeler.com/flawfinder/test.c>

**安装：**

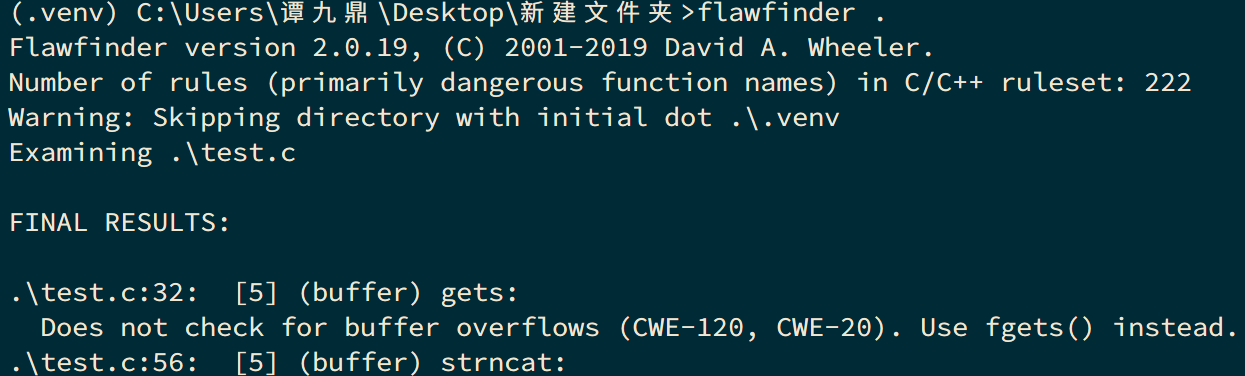
Python –m venv .venv

pip install flawfinder



**运行分析：**

flawfinder .



**完整输出：**

Flawfinder version 2.0.19, (C) 2001-2019 David A. Wheeler.

Number of rules (primarily dangerous function names) in C/C++ ruleset: 222

Warning: Skipping directory with initial dot .\.venv

Examining .\test.c

FINAL RESULTS:

.\test.c:32: [5] (buffer) gets:

Does not check for buffer overflows (CWE-120, CWE-20). Use fgets() instead.

.\test.c:56: [5] (buffer) strncat:

Easily used incorrectly (e.g., incorrectly computing the correct maximum

size to add) [MS-banned] (CWE-120). Consider strcat\_s, strlcat, snprintf,

or automatically resizing strings. Risk is high; the length parameter

appears to be a constant, instead of computing the number of characters

left.

.\test.c:57: [5] (buffer) \_tcsncat:

Easily used incorrectly (e.g., incorrectly computing the correct maximum

size to add) [MS-banned] (CWE-120). Consider strcat\_s, strlcat, or

automatically resizing strings. Risk is high; the length parameter appears

to be a constant, instead of computing the number of characters left.

.\test.c:60: [5] (buffer) MultiByteToWideChar:

Requires maximum length in CHARACTERS, not bytes (CWE-120). Risk is high,

it appears that the size is given as bytes, but the function requires size

as characters.

.\test.c:62: [5] (buffer) MultiByteToWideChar:

Requires maximum length in CHARACTERS, not bytes (CWE-120). Risk is high,

it appears that the size is given as bytes, but the function requires size

as characters.

.\test.c:73: [5] (misc) SetSecurityDescriptorDacl:

Never create NULL ACLs; an attacker can set it to Everyone (Deny All

Access), which would even forbid administrator access (CWE-732).

.\test.c:73: [5] (misc) SetSecurityDescriptorDacl:

Never create NULL ACLs; an attacker can set it to Everyone (Deny All

Access), which would even forbid administrator access (CWE-732).

.\test.c:17: [4] (buffer) strcpy:

Does not check for buffer overflows when copying to destination [MS-banned]

(CWE-120). Consider using snprintf, strcpy\_s, or strlcpy (warning: strncpy

easily misused).

.\test.c:20: [4] (buffer) sprintf:

Does not check for buffer overflows (CWE-120). Use sprintf\_s, snprintf, or

vsnprintf.

.\test.c:21: [4] (buffer) sprintf:

Does not check for buffer overflows (CWE-120). Use sprintf\_s, snprintf, or

vsnprintf.

.\test.c:22: [4] (format) sprintf:

Potential format string problem (CWE-134). Make format string constant.

.\test.c:23: [4] (format) printf:

If format strings can be influenced by an attacker, they can be exploited

(CWE-134). Use a constant for the format specification.

.\test.c:25: [4] (buffer) scanf:

The scanf() family's %s operation, without a limit specification, permits

buffer overflows (CWE-120, CWE-20). Specify a limit to %s, or use a

different input function.

.\test.c:27: [4] (buffer) scanf:

The scanf() family's %s operation, without a limit specification, permits

buffer overflows (CWE-120, CWE-20). Specify a limit to %s, or use a

different input function.

.\test.c:38: [4] (format) syslog:

If syslog's format strings can be influenced by an attacker, they can be

exploited (CWE-134). Use a constant format string for syslog.

.\test.c:49: [4] (buffer) \_mbscpy:

Does not check for buffer overflows when copying to destination [MS-banned]

(CWE-120). Consider using a function version that stops copying at the end

of the buffer.

.\test.c:52: [4] (buffer) lstrcat:

Does not check for buffer overflows when concatenating to destination

[MS-banned] (CWE-120).

.\test.c:75: [3] (shell) CreateProcess:

This causes a new process to execute and is difficult to use safely

(CWE-78). Specify the application path in the first argument, NOT as part

of the second, or embedded spaces could allow an attacker to force a

different program to run.

.\test.c:75: [3] (shell) CreateProcess:

This causes a new process to execute and is difficult to use safely

(CWE-78). Specify the application path in the first argument, NOT as part

of the second, or embedded spaces could allow an attacker to force a

different program to run.

.\test.c:91: [3] (buffer) getopt\_long:

Some older implementations do not protect against internal buffer overflows

(CWE-120, CWE-20). Check implementation on installation, or limit the size

of all string inputs.

.\test.c:16: [2] (buffer) strcpy:

Does not check for buffer overflows when copying to destination [MS-banned]

(CWE-120). Consider using snprintf, strcpy\_s, or strlcpy (warning: strncpy

easily misused). Risk is low because the source is a constant string.

.\test.c:19: [2] (buffer) sprintf:

Does not check for buffer overflows (CWE-120). Use sprintf\_s, snprintf, or

vsnprintf. Risk is low because the source has a constant maximum length.

.\test.c:45: [2] (buffer) char:

Statically-sized arrays can be improperly restricted, leading to potential

overflows or other issues (CWE-119!/CWE-120). Perform bounds checking, use

functions that limit length, or ensure that the size is larger than the

maximum possible length.

.\test.c:46: [2] (buffer) char:

Statically-sized arrays can be improperly restricted, leading to potential

overflows or other issues (CWE-119!/CWE-120). Perform bounds checking, use

functions that limit length, or ensure that the size is larger than the

maximum possible length.

.\test.c:50: [2] (buffer) memcpy:

Does not check for buffer overflows when copying to destination (CWE-120).

Make sure destination can always hold the source data.

.\test.c:51: [2] (buffer) CopyMemory:

Does not check for buffer overflows when copying to destination (CWE-120).

Make sure destination can always hold the source data.

.\test.c:97: [2] (misc) fopen:

Check when opening files - can an attacker redirect it (via symlinks),

force the opening of special file type (e.g., device files), move things

around to create a race condition, control its ancestors, or change its

contents? (CWE-362).

.\test.c:15: [1] (buffer) strcpy:

Does not check for buffer overflows when copying to destination [MS-banned]

(CWE-120). Consider using snprintf, strcpy\_s, or strlcpy (warning: strncpy

easily misused). Risk is low because the source is a constant character.

.\test.c:18: [1] (buffer) sprintf:

Does not check for buffer overflows (CWE-120). Use sprintf\_s, snprintf, or

vsnprintf. Risk is low because the source is a constant character.

.\test.c:26: [1] (buffer) scanf:

It's unclear if the %s limit in the format string is small enough

(CWE-120). Check that the limit is sufficiently small, or use a different

input function.

.\test.c:53: [1] (buffer) strncpy:

Easily used incorrectly; doesn't always \0-terminate or check for invalid

pointers [MS-banned] (CWE-120).

.\test.c:54: [1] (buffer) \_tcsncpy:

Easily used incorrectly; doesn't always \0-terminate or check for invalid

pointers [MS-banned] (CWE-120).

.\test.c:55: [1] (buffer) strncat:

Easily used incorrectly (e.g., incorrectly computing the correct maximum

size to add) [MS-banned] (CWE-120). Consider strcat\_s, strlcat, snprintf,

or automatically resizing strings.

.\test.c:58: [1] (buffer) strlen:

Does not handle strings that are not \0-terminated; if given one it may

perform an over-read (it could cause a crash if unprotected) (CWE-126).

.\test.c:64: [1] (buffer) MultiByteToWideChar:

Requires maximum length in CHARACTERS, not bytes (CWE-120). Risk is very

low, the length appears to be in characters not bytes.

.\test.c:66: [1] (buffer) MultiByteToWideChar:

Requires maximum length in CHARACTERS, not bytes (CWE-120). Risk is very

low, the length appears to be in characters not bytes.

ANALYSIS SUMMARY:

Hits = 36

Lines analyzed = 117 in approximately 0.18 seconds (634 lines/second)

Physical Source Lines of Code (SLOC) = 80

Hits@level = [0] 16 [1] 9 [2] 7 [3] 3 [4] 10 [5] 7

Hits@level+ = [0+] 52 [1+] 36 [2+] 27 [3+] 20 [4+] 17 [5+] 7

Hits/KSLOC@level+ = [0+] 650 [1+] 450 [2+] 337.5 [3+] 250 [4+] 212.5 [5+] 87.5

Dot directories skipped = 1 (--followdotdir overrides)

Suppressed hits = 2 (use --neverignore to show them)

Minimum risk level = 1

Not every hit is necessarily a security vulnerability.

You can inhibit a report by adding a comment in this form:

// flawfinder: ignore

Make \*sure\* it's a false positive!

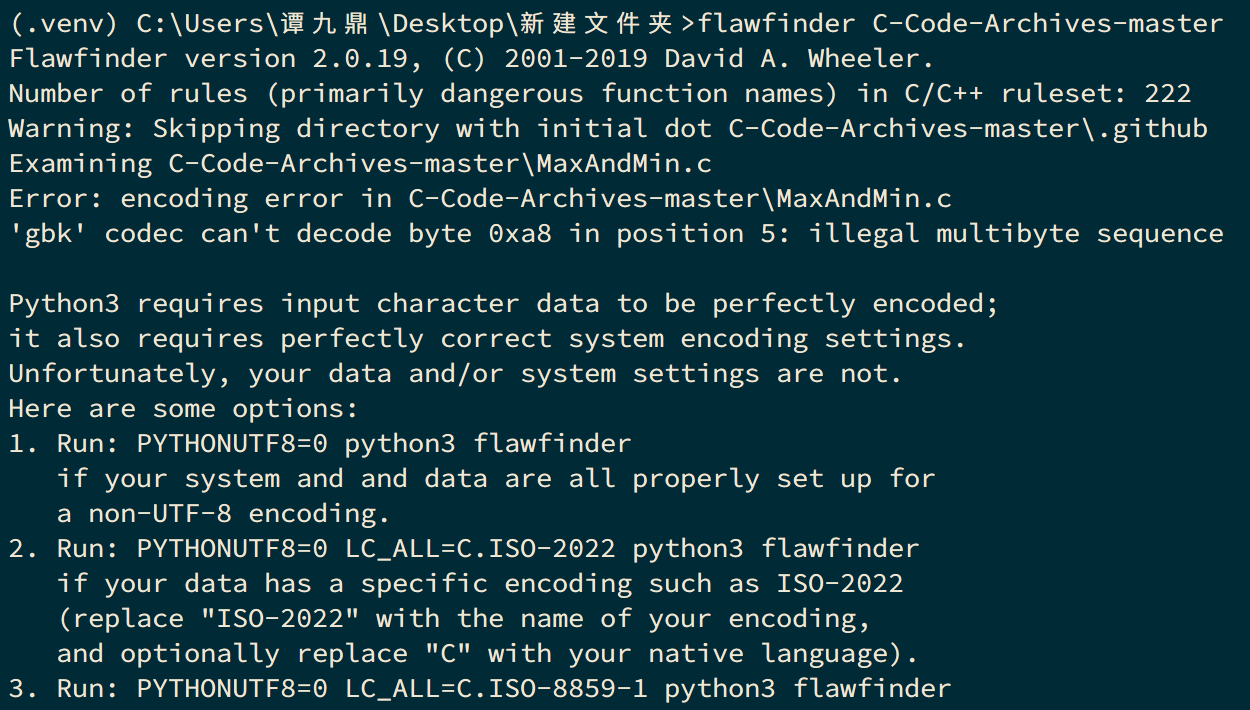
You can use the option --neverignore to show these.

There may be other security vulnerabilities; review your code!

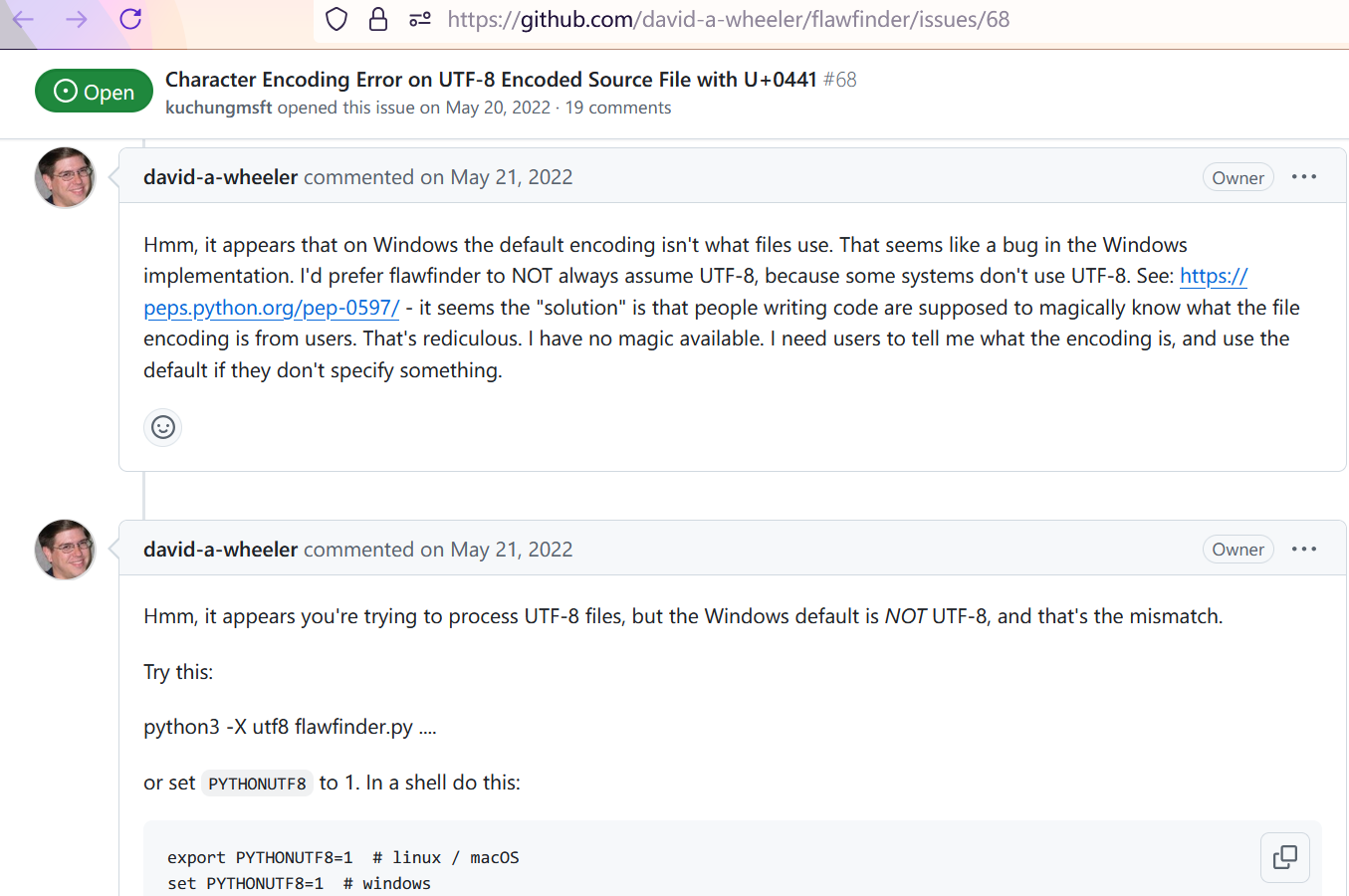
See 'Secure Programming HOWTO'

(https://dwheeler.com/secure-programs) for more information.

**检查我自己写的代码，报编码错误：**



**查找发现issue区已有讨论：**<https://github.com/david-a-wheeler/flawfinder/issues/68>



**设置PYTHONUTF8=1后的结果：**

C-Code-Archives-master\比较字符串关系.c:62: [4] (buffer) scanf:

The scanf() family's %s operation, without a limit specification, permits

buffer overflows (CWE-120, CWE-20). Specify a limit to %s, or use a

different input function.

C-Code-Archives-master\语法测试\qsortTest.c:37: [4] (format) printf:

If format strings can be influenced by an attacker, they can be exploited

(CWE-134). Use a constant for the format specification.

C-Code-Archives-master\语法测试\逻辑操作符的阻断特性.c:9: [4] (format) printf:

If format strings can be influenced by an attacker, they can be exploited

(CWE-134). Use a constant for the format specification.

C-Code-Archives-master\洗牌算法.c:52: [3] (random) srand:

This function is not sufficiently random for security-related functions

such as key and nonce creation (CWE-327). Use a more secure technique for

acquiring random values.

C-Code-Archives-master\UTF8StringUnescaper.c:28: [2] (buffer) char:

Statically-sized arrays can be improperly restricted, leading to potential

overflows or other issues (CWE-119!/CWE-120). Perform bounds checking, use

functions that limit length, or ensure that the size is larger than the

maximum possible length.

C-Code-Archives-master\比较字符串关系.c:61: [2] (buffer) char:

Statically-sized arrays can be improperly restricted, leading to potential

overflows or other issues (CWE-119!/CWE-120). Perform bounds checking, use

functions that limit length, or ensure that the size is larger than the

maximum possible length.

C-Code-Archives-master\洗牌算法.c:36: [2] (buffer) memcpy:

Does not check for buffer overflows when copying to destination (CWE-120).

Make sure destination can always hold the source data.

C-Code-Archives-master\语法测试\使用缓冲区.c:5: [2] (buffer) char:

Statically-sized arrays can be improperly restricted, leading to potential

overflows or other issues (CWE-119!/CWE-120). Perform bounds checking, use

functions that limit length, or ensure that the size is larger than the

maximum possible length.

C-Code-Archives-master\输入确认.c:40: [1] (buffer) getchar:

Check buffer boundaries if used in a loop including recursive loops

(CWE-120, CWE-20).

C-Code-Archives-master\进度条 - 横向.c:14: [1] (obsolete) usleep:

This C routine is considered obsolete (as opposed to the shell command by

the same name). The interaction of this function with SIGALRM and other

timer functions such as sleep(), alarm(), setitimer(), and nanosleep() is

unspecified (CWE-676). Use nanosleep(2) or setitimer(2) instead.

**个人评价：**有一些用，但让人改能运行的代码有点让人不爽。

**其他工具：**

Splint: <https://splint.org/> <https://github.com/splintchecker/splint>

虽然最近一次提交时间是5天前，但实际上最新的Release是2007年，没有使用的必要

RATS：<http://www.fortify.com/security-resources/rats.jsp> <https://www.debian.org/security/audit/examples/RATS> <https://github.com/andrew-d/rough-auditing-tool-for-security>

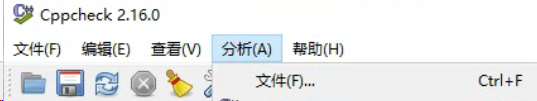
官网404。GitHub上最后一次提交在2014年，再之前是2009年。没有使用的必要

**Cppcheck：**<https://sourceforge.net/projects/cppcheck/>

检查内存泄漏、分配-释放 不匹配、缓冲区溢出

目标是0误报

下载msi安装包，安装。它支持分析项目，但我们测试用的是单个文件，不是项目。

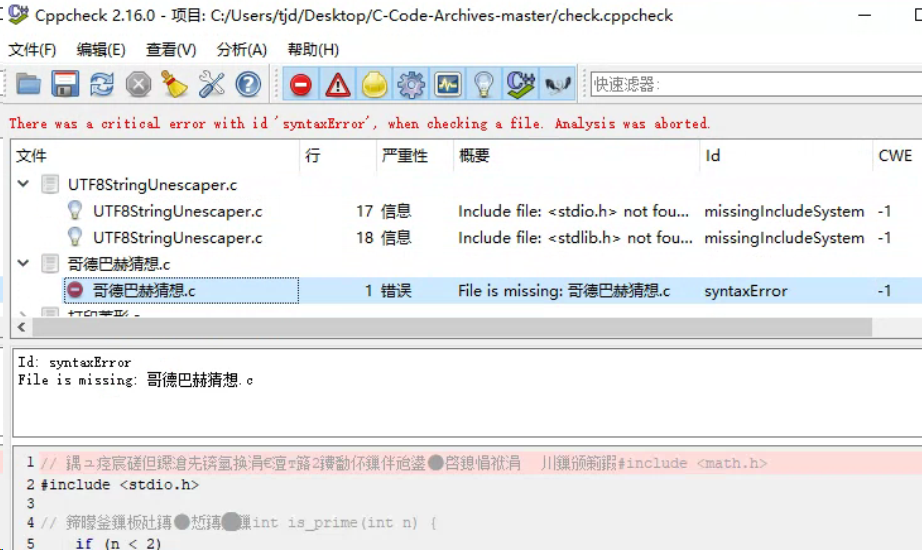


**分析flawfinder提供的test.c：**



可以看出对于无法编译的程序，它不会进一步分析。

**测试自己的代码：**



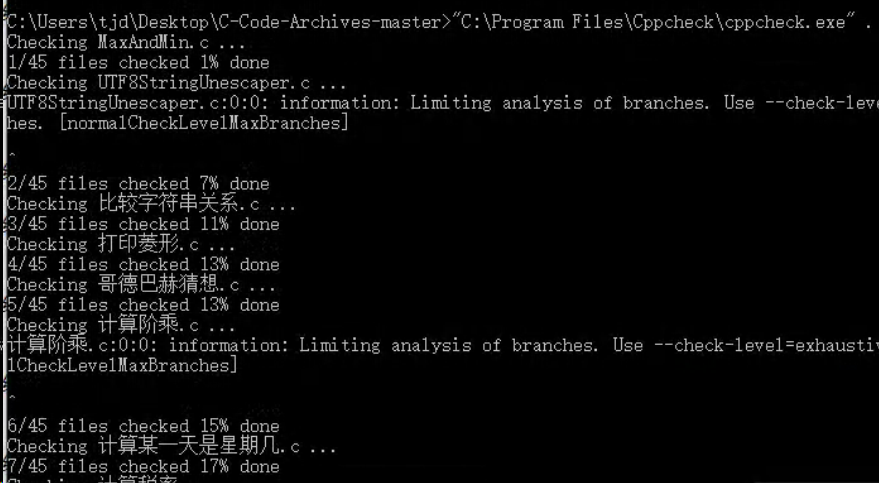
**分析结果：**

1. 虽然系统里有能正常使用的gcc(mingw)，但它还是说找不到头文件。
2. 文件名有中文，就说找不到文件，但Id又是syntaxError。
3. 文件预览窗口可以看见乱码。

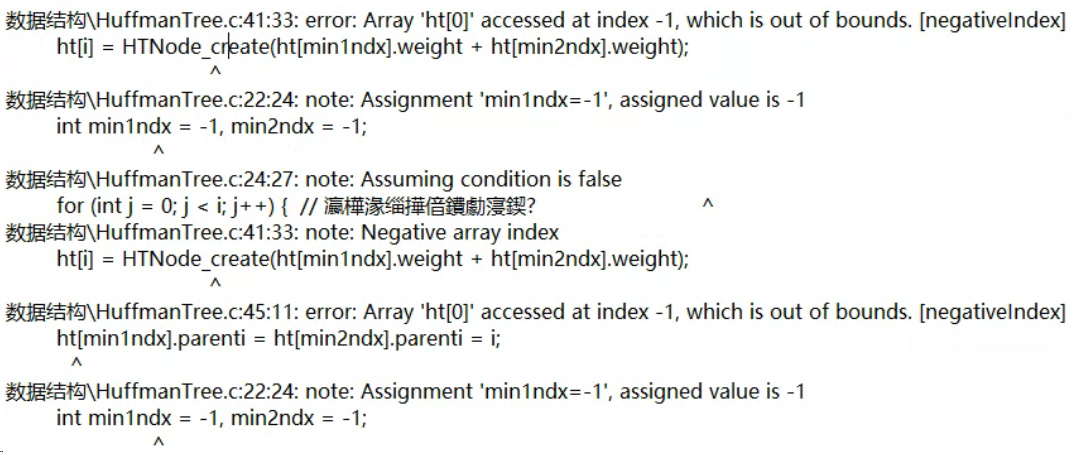
总结：无法使用。

**使用命令行：**

"C:\Program Files\Cppcheck\cppcheck.exe" . 2>result.txt



**结果：**



仍然存在乱码，不过好歹能跑了。

对于其输出结果，看了一下源代码，实际并不成立；不过单从函数本身来说，没有检查边界，如果调用者传了错误的参数，是可以成立的。算是成功检测出了吧。