软件测试报告

报告题目：二分法函数测试

学 院： 软件学院

班 级： 1603班

姓 名： 赵 伟

学 号： 201602801553

时 间：2019年6月3日

指导老师： 郭 雪

成 绩：

# 一：编写目的：

1. 掌握软件测试技术中白盒测试方法；

2. 掌握如何进行单元测试。

# 二：背景：

测试程序名称： 二分法查找函数测试

任务提出者：学生本人 赵伟

开发者：学生本人 赵伟

测试环境： Windows10 ；Visual Studio Community 2017；Google-Test测试适配器

说明：由于程序较小，并且只是作为了解白盒测试的小程序，因此不涉及较高的实验环境，并且测试环境和运行环境在理论上是一样的，因此不存在较大的测试差异。

# 三：测试用例：

|  |
| --- |
| ID：dichotomy1 |
| 用例名称：奇数个数是否可以查找成功 |
| 测试项：1、2、3、4、5、8、13、14、16 |
| 环境要求：Windows10 ；Visual Studio Community 2017；Google-Test测试适配器 |
| 参考文档：软件规格说明书 |
| 期望结果：  返回值与预期值相等； |

|  |
| --- |
| ID：dichotomy2 |
| 用例名称：偶数个数是否可以查找成功 |
| 测试项：1、2、3、4、5、8、13、14 |
| 环境要求：Windows10 ；Visual Studio Community 2017；Google-Test测试适配器 |
| 参考文档：软件规格说明书 |
| 期望结果：  返回值与预期值相等； |

|  |
| --- |
| ID：dichotomy3 |
| 用例名称：存在小数是否可以查找成功 |
| 测试项：1、2、3、4、5.1、8、13、14、16 |
| 环境要求：Windows10 ；Visual Studio Community 2017；Google-Test测试适配器 |
| 参考文档：软件规格说明书 |
| 期望结果：  返回值与预期值相等； |

# 四：测试代码：

**pch.h**

#pragma once

#include "gtest/gtest.h"

#include <iostream>

#include <vector>

#include <cstdlib>

#include <algorithm>

#include <string>

#include <ctime>

using namespace std;

class RandomVector

{

public:

RandomVector();

RandomVector(int maxValue, int maxSize);

vector<int> getOrderedRandomArray();

private:

int maxValue;

int maxSize;

vector<int> randomVector;

};

int BinarySearch(std::vector<int> orderedVector, int left, int right, int target);

int compareFunction(const void \*\_left, const void \*\_right);

int comparator(vector<int> orderedVector, int target);

**Pch.cpp**

#include "pch.h"

RandomVector::RandomVector() { }

RandomVector::RandomVector(int maxValue, int maxSize)

{

this->maxValue = maxValue;

this->maxSize = maxSize;

this->randomVector = vector<int>(rand() % maxSize + 5);

}

vector<int> RandomVector::getOrderedRandomArray()

{

for (int i = 0; i < randomVector.size(); i++)

randomVector[i] = rand() % maxValue + 1;

sort(randomVector.begin(), randomVector.end());

return randomVector;

}

int BinarySearch(std::vector<int> ordereVector, int left, int right, int target)

{

while (left <= right)

{

int index = (left + right) / 2;

if (ordereVector[index] > target)

right = index - 1;

else if (ordereVector[index] < target)

left = index + 1;

else

return index;

}

return -1;

}

// for test

int compareFunction(const void \*\_left, const void \*\_right)

{

return (\*(int\*)\_left - \*(int\*)\_right);

}

int comparator(vector<int> orderedVector, int target)

{

int \*orderedArray = new int[orderedVector.size()];

memcpy(orderedArray, &orderedVector[0], orderedVector.size() \* sizeof(int));

int \*item = (int \*)bsearch(&target, orderedArray, orderedVector.size(), sizeof(int), compareFunction);

return item == nullptr ? -1 : item - orderedArray;

}

**TEST.CPP(正确)**

#include "pch.h"

TEST(CorrectResultsTest, TestName) {

srand((unsigned int)time(NULL));

int maxSize = 10;

int maxValue = 10;

int testTime = 5;

bool succeed = true;

RandomVector randomV(maxValue, maxSize);

std::vector<int> randomOrderedVector = randomV.getOrderedRandomArray();

int res = BinarySearch(randomOrderedVector, 0, randomOrderedVector.size() - 1, 8);

int absoluteRes = comparator(randomOrderedVector, 8);

EXPECT\_EQ(res, absoluteRes);

}

**TEST.CPP(错误)**

#include "pch.h"

TEST(WrongResultsTest, TestName) {

srand((unsigned int)time(NULL));

int maxSize = 10;

int maxValue = 10;

int testTime = 5;

bool succeed = true;

RandomVector randomV(maxValue, maxSize);

std::vector<int> randomOrderedVector = randomV.getOrderedRandomArray();

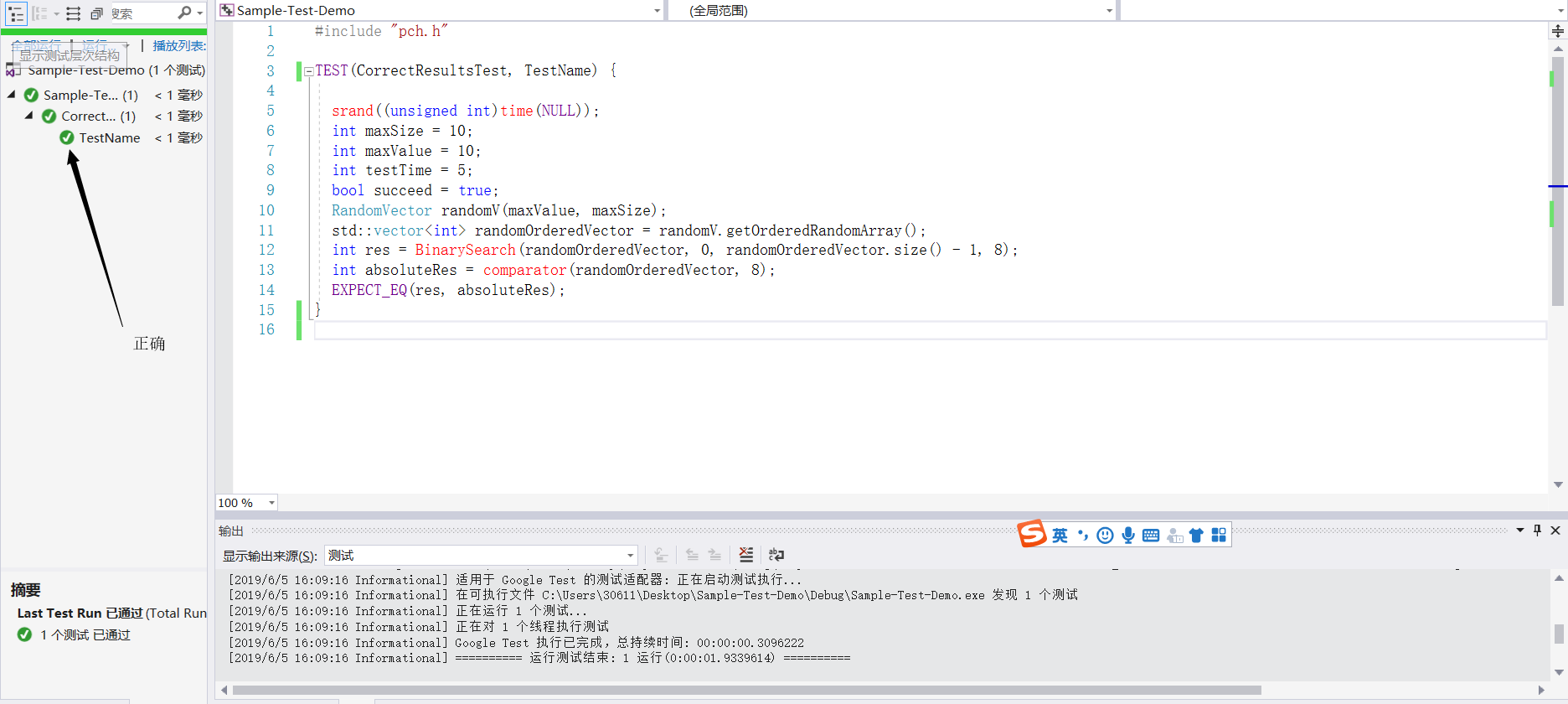
int res = BinarySearch(randomOrderedVector, 0, randomOrderedVector.size() - 1, 8);

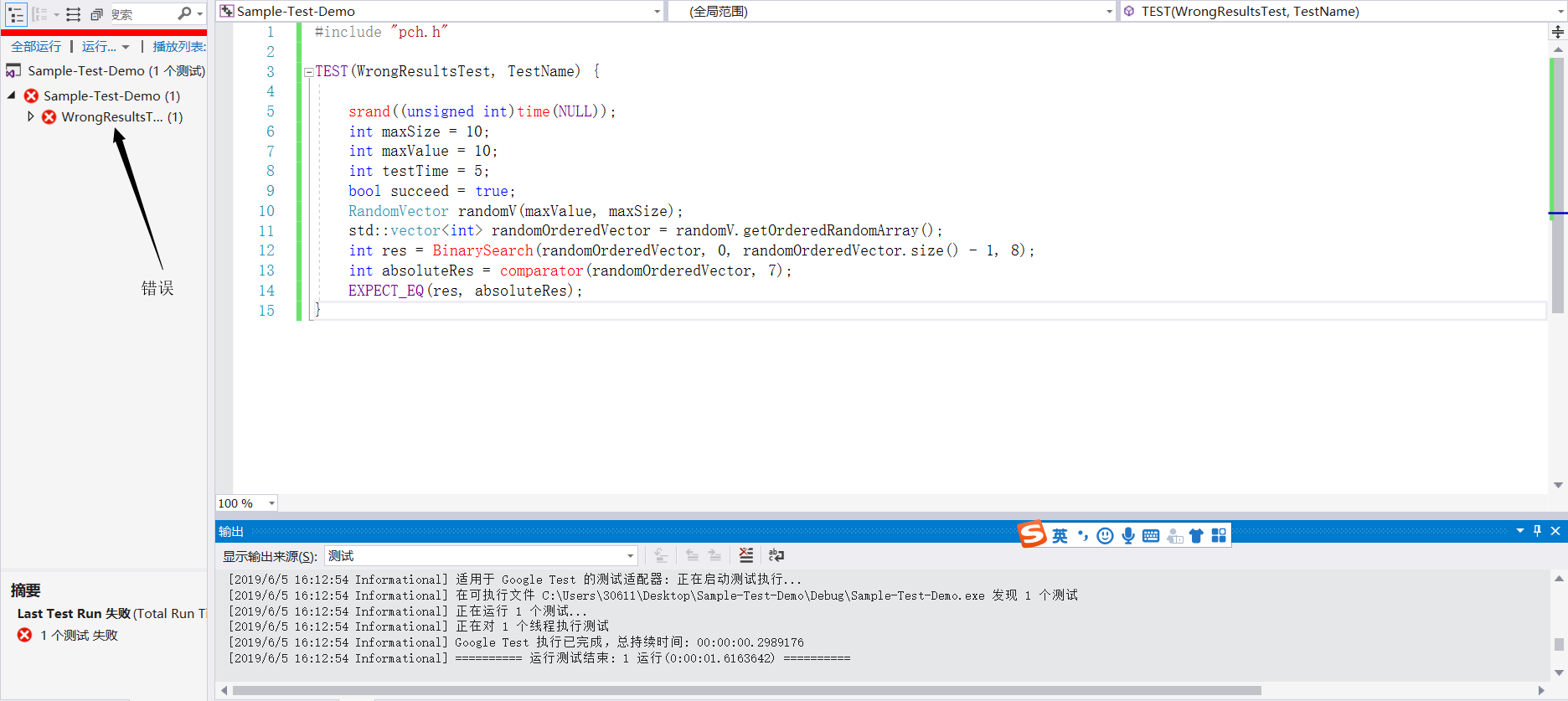
int absoluteRes = comparator(randomOrderedVector, 7);

EXPECT\_EQ(res, absoluteRes);

}

# 五：测试结果：





# 六：总结：

通过对二分法函数的测试，学会了进行测试需要掌握的方法，通过缺陷的映入，判断测试结果与预期值是否相等，从而达到测试目的。