МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение

высшего профессионального образования

«Ижевский государственный технический университет

имени М.Т. Калашникова»

Кафедра «Программное обеспечение»

Отчет по лабораторной работе №1

по предмету «Тестирование программного обеспечения»

Номер задания 27

Выполнил

студент гр. Б21-191-1зу Рязанов Д.А.

Принял:

старший преподаватель Е.В. Старыгина

Ижевск

2024

# **Задание**

Дан массив целых чисел . Найти все пары , такие что и . Распечатать их значения и номеpа. Если таких паp нет, то выдать сообщение.

# **Блок-схема алгоритма**

Блок-схема алгоритма представлена на рис. 1.

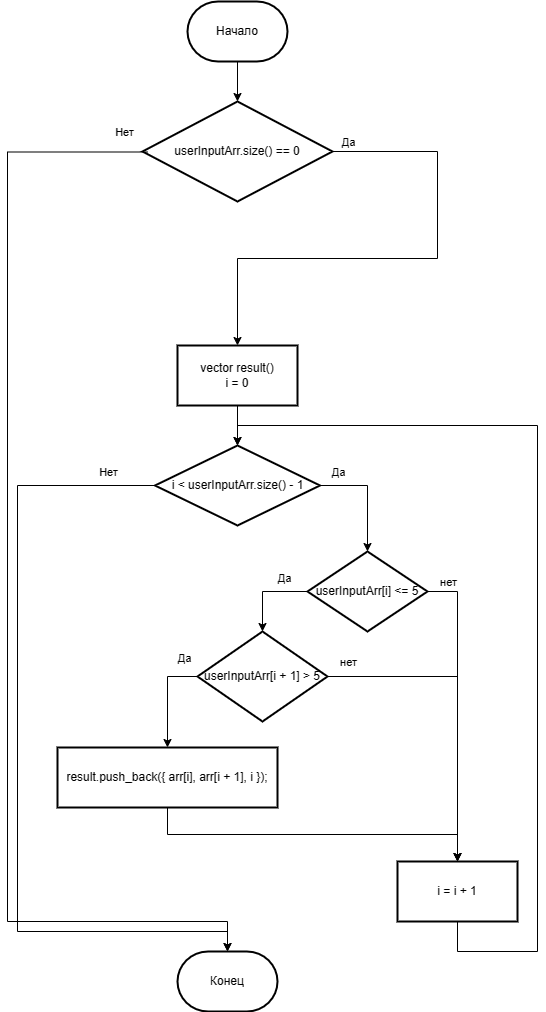


Рис. 1. Блок-схема алгоритма

# **Тестирование базового пути**

## Потоковый граф

Потоковый граф представлен на рис. 2.

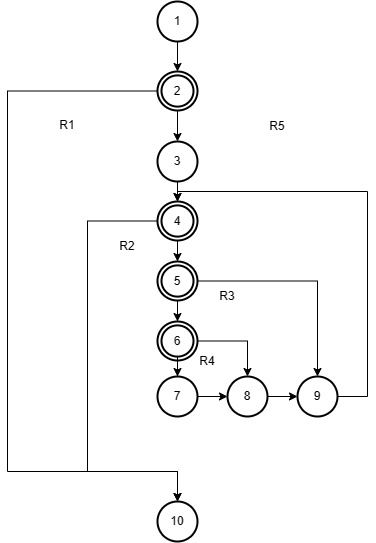


Рис. 2. Потоковый граф

## Цикломатическая сложность

Вычислим цикломатическую сложность потокового графа каждым из трёх способов:

1. потоковый граф имеет 5 регионов;
2. V(G) = 13 дуг – 10 узлов + 2 = 5;
3. V(G) = 4 предикатных узлов + 1 = 5.

## Базовое множество независимых путей

Перечислим независимые пути:

1. 1 – 2 – 10; если массив на входе в функцию был пуст.
2. 1 – 2 – 3 – 4 – 10; если введён только один элемент
3. 1 – 2 – 3 – 4 – 5 – 9 – 4 – …; если первый элемент пары больше 5
4. 1 – 2 – 3 – 4 – 5 – 6 – 8 – 9 – 4 – …; если второй элемент меньше или равен 5
5. 1 -2 – 3 – 4 – 5 – 6 – 7 – 8 – 9 – 4 – …; если первый элемент пары меньше или равен пяти и второй элемент пары больше пяти.

## Тестовые варианты

Тестовый вариант для пути 1:

Исходные данные: вектор userInputArr с count() равным 0.

Ожидаемые результаты: вектор result с count() равным 0.

Тестовый вариант для пути 2:

Исходные данные: вектор userInputArr с count() равным 1.

Ожидаемые результаты: вектор result с count() равным 0.

Тестовый вариант для пути 3:

Исходные данные: вектор userInputArr с count() больше 1 и элементом userInputArr(i) > 5.

Ожидаемые результаты: вектор result без пары начинающейся с индекса i.

Тестовый вариант для пути 3:

Исходные данные: вектор userInputArr с count() больше 1 и элементом userInputArr(i) > 5.

Ожидаемые результаты: вектор result без пары, начинающейся с индекса i.

Тестовый вариант для пути 4:

Исходные данные: вектор userInputArr с count() больше 1 и элементом userInputArr(i) <= 5.

Ожидаемые результаты: вектор result без пары, начинающейся с индекса  1.

Тестовый вариант для пути 5:

Исходные данные: вектор userInputArr с count() больше 1 и элементом userInputArr(i) <= 5 и userInputArr(I + 1) > 5.

Ожидаемые результаты: вектор result со всеми парами, начинающимися с индекса i, где result(i) <= 5 и result(i+1) > 5.

# **Тестирование потоков данных**

## Информационный граф

Информационный граф представлен на рис. 3.

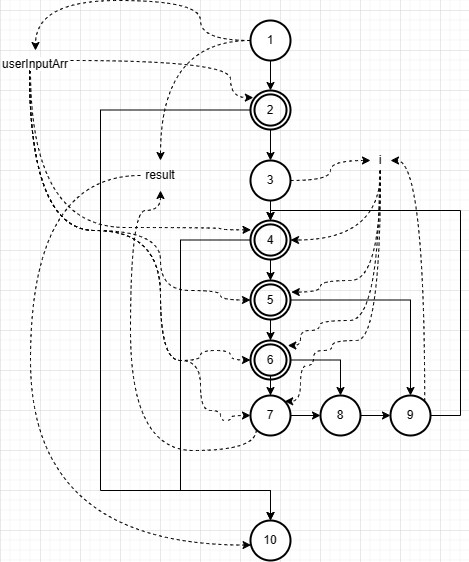


Рис. 3. Информационный граф

## Формирование полного набора du-цепочек

Сформируем список определений переменных:

1. DEF(1) = { userInputArr, 1},
2. DEF(1) = {result, 1},
3. DEF(3) = {i, 3},
4. DEF(7) = {result, 7},
5. DEF(9) = {i, 9}.

Сформируем список использования переменных:

1. USE(2) = { userInputArr, 2 },
2. USE(4) = { i, 4 },
3. USE(4) = { userInputArr, 4 },
4. USE(5) = { i, 5 },
5. USE(5) = { userInputArr, 5 },
6. USE(6) = { i, 6 },
7. USE(6) = { userInputArr, 6 },
8. USE(7) = { i, 7 },
9. USE(7) = { userInputArr, 7 },
10. USE(10) = { result, 10 }.

Сформируем список DU-цепочек:

1. DU(userInputArr) = { userInputArr, 1, 2},
2. DU(userInputArr) = { userInputArr, 1, 4},
3. DU(userInputArr) = { userInputArr, 1, 5},
4. DU(userInputArr) = { userInputArr, 1, 6},
5. DU(userInputArr) = { userInputArr, 1, 7},
6. DU(i) = { i, 3, 4 },
7. DU(i) = { i, 3, 5 },
8. DU(i) = { i, 3, 6 },
9. DU(i) = { i, 3, 7 },
10. DU(i) = { i, 9, 4 },
11. DU(i) = { i, 9, 5 },
12. DU(i) = { i, 9, 6 },
13. DU(i) = { i, 9, 7 },
14. DU(result) = { result, 1, 10 },
15. DU(result) = { result, 7, 10 },

## Построение маршрутов

Построим маршруты:

1. 1 – userInputArr – 2: 1 – 2 - …;
2. 1 – userInputArr – 4: 1 – 2 – 3 – 4 - …;
3. 1 – userInputArr – 5: 1 – 2 – 3 – 4 – 5 - …;
4. 1 – userInputArr – 6: 1 – 2 – 3 – 4 – 5 – 6 - …;
5. 1 – userInputArr – 7: 1 – 2 – 3 – 4 – 5 – 6 – 7 - …;
6. 3 – i – 4: 1 – 2 – 3 – 4 - …;
7. 3 – i – 5: 1 – 2 – 3 – 4 – 5 - …;
8. 3 – i – 6: 1 – 2 – 3 – 4 – 5 – 6 - …;
9. 3 – i – 7: 1 – 2 – 3 – 4 – 5 – 6 – 7 …;
10. 9 – i – 4: … – 4 – 5 – 9 – 4 - …;
11. 9 – i – 5: … – 4 – 5 – 9 – 4 – 5 - …;
12. 9 – i – 6: … – 4 – 5 – 9 – 4 – 5 – 6 - …;
13. 9 – i – 7: … – 4 – 5 – 9 – 4 – 5 – 6 – 7 - …;
14. 3 – result – 10: 1 – 2 – 10;
15. 7 – result – 10: … – 4 – 5 – 6 – 7 – 8 – 9 – 4 – 10;

Убрав дублирующиеся маршруты, получаем:

1. 1 – 2 – 10;
2. 1 – 2 – 3 – 4 – 10;
3. 1 – 2 – 3 – 4 – 5 – 6 – 7 -8 – 9 – 4 – 10;

# **Области эквивалентности**

Можно выделить области эквивалентности для последовательности userInputArr (см. табл. 1). Определим области эквивалентности как:

1. Последовательность не содержит пар , где .
2. Последовательность содержит одну пару , .
3. Последовательность содержит несколько пар , .
4. Последовательность не содержит элементов.
5. Последовательность содержит один элемент.
6. Последовательность содержит чётное количество из нескольких элементов.
7. Последовательность содержит нечётное количество из нескольких элементов.

Для самих элементов , определим область эквивалентности, на основании типа данных, и их отношении к 5 – больше, меньше равно (см. табл. 2).

Таблица 1. Области эквивалентности для userInputArr

|  |  |  |  |
| --- | --- | --- | --- |
| Последовательность | Количество пар | Положение | Пары |
| Не содержит элементов | 0 | – | Нет в последовательности |
| Содержит один элемент | 0 | – | Нет в последовательности |
| Содержит чётное количество из нескольких элементов. | 0 | – | Нет в последовательности |
| Содержит нечётное количество из нескольких элементов. | 0 |  | Нет в последовательности |
| Содержит чётное количество из нескольких элементов. | 1 | С первого элемента последовательности | 1 |
| Содержит нечётное количество из нескольких элементов. | 1 | С первого элемента последовательности | 1 |
| Содержит чётное количество из нескольких элементов. | 1 | С последнего элемента последовательности | 1 |
| Содержит нечётное количество из нескольких элементов. | 1 | С последнего элемента последовательности | 1 |
| Содержит чётное количество из нескольких элементов. | 1 | В середине последовательности | 1 |
| Содержит несколько элементов. | 1 | В середине последовательности | 1 |
| Содержит несколько элементов. |  | С первого, последнего, элемента и в середине последовательности |  |

Таблица 2. Области эквивалентности элементов

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | |  | | Пара? |
| Размер | Отношение к 5 | Размер | Отношение к 5 |
| Минимальное значение int32 |  | Минимальное значение int32 |  | Нет |
| Минимальное значение int32 |  | Максимальное значение int32 |  | Да |
| Минимальное значение int32 |  | В пределах int32 |  | Нет |
| Минимальное значение int32 |  | В пределах int32 |  | Да |
| Максимальное значение int32 |  | Минимальное значение int32 |  | Нет |
| Максимальное значение int32 |  | Максимальное значение int32 |  | Нет |
| Максимальное значение int32 |  | В пределах int32 |  | Нет |
| Максимальное значение int32 |  | В пределах int32 |  | Нет |
| В пределах int32 |  | Минимальное значение int32 |  | Нет |
| В пределах int32 |  | Максимальное значение int32 |  | Да |
| В пределах int32 |  | В пределах int32 |  | Нет |
| В пределах int32 |  | В пределах int32 |  | Да |
| В пределах int32 |  | Минимальное значение int32 |  | Нет |
| В пределах int32 |  | Максимальное значение int32 |  | Нет |
| В пределах int32 |  | В пределах int32 |  | Нет |
| В пределах int32 |  | В пределах int32 |  | Нет |

# **Контрольный пример**

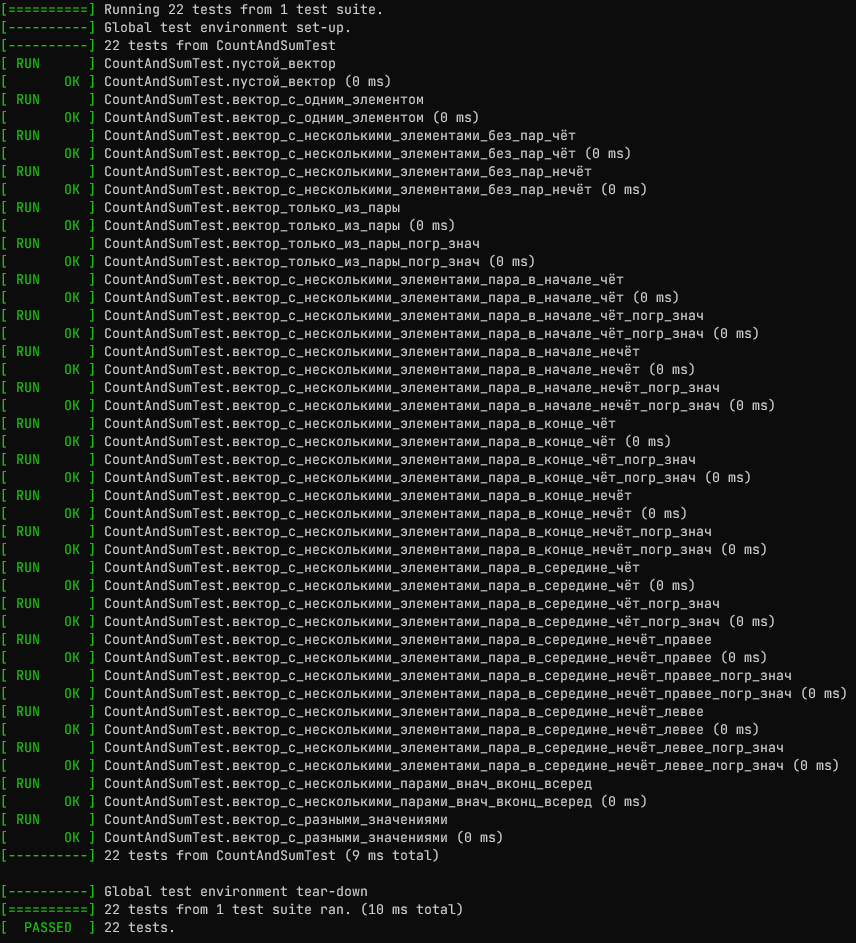


Рис. 4. Выполнение тестов

# **Текст программы**

#pragma once

#include <vector>

using namespace std;

struct PairWithIndex {

int first;

int second;

int index;

};

vector<PairWithIndex> findPairs(const vector<int>& arr);

Листинг 1. Заголовочный файл

#include "lr1.h"

using namespace std;

vector<PairWithIndex> findPairs(const vector<int>& userInputArr) {

vector<PairWithIndex> result;

if (userInputArr.size() == 0)

{

return result;

}

for (int i = 0; i < userInputArr.size() - 1; ++i) {

if (userInputArr[i] <= 5 && userInputArr[i + 1] > 5) {

result.push\_back({ userInputArr[i], userInputArr[i + 1], i });

}

}

return result;

}

Листинг 2. Файл с реализацией

#include <lib/lr1.h>

#include <gtest/gtest.h>

#include <vector>

#include <utility>

struct borderline\_value {

vector<int> input;

bool expectPair;

};

class CountAndSumTest : public ::testing::Test {

protected:

void SetUp() override {

borderlineValues =

{

{ { INT\_MIN, INT\_MIN }, false },

{ { INT\_MIN, INT\_MAX }, true },

{ { INT\_MIN, 5 }, false }, { { INT\_MIN, 4 }, false },

{ { INT\_MIN, 6 }, true },

{ { INT\_MAX, INT\_MIN }, false },

{ { INT\_MAX, INT\_MAX }, false },

{ { INT\_MAX, 5 }, false }, { { INT\_MAX, 4 }, false },

{ { INT\_MAX, 6 }, false },

{ { 5, INT\_MIN }, false }, { { 4, INT\_MIN }, false },

{ { 5, INT\_MAX }, true }, { { 4, INT\_MAX }, true },

{ { 5, 4 }, false }, { { 4, 4 }, false }, { { 4, 5 }, false }, { { 5, 4 }, false },

{ { 5, 6 }, true }, { { 4, 6 }, true },

{ { 6, INT\_MIN, }, false },

{ { 6, INT\_MAX, }, false },

{ { 6, 4, }, false }, { { 6, 5 }, false },

{ { 6, 6, }, false },

};

}

std::vector<borderline\_value> borderlineValues;

};

TEST\_F(CountAndSumTest, пустой\_вектор) {

vector<int> emptyVector;

vector<PairWithIndex> res = findPairs(emptyVector);

EXPECT\_EQ(res.size(), 0);

}

TEST\_F(CountAndSumTest, вектор\_с\_одним\_элементом) {

vector<int> vectorWithOneElement{ 0 };

vector<PairWithIndex> res = findPairs(vectorWithOneElement);

EXPECT\_EQ(res.size(), 0);

}

TEST\_F(CountAndSumTest, вектор\_с\_несколькими\_элементами\_без\_пар\_чёт) {

vector<int> userInput{ 1, 2, 3, 4 };

vector<PairWithIndex> res = findPairs(userInput);

EXPECT\_EQ(res.size(), 0);

}

TEST\_F(CountAndSumTest, вектор\_с\_несколькими\_элементами\_без\_пар\_нечёт) {

vector<int> userInput{ 1, 2, 3 };

vector<PairWithIndex> res = findPairs(userInput);

EXPECT\_EQ(res.size(), 0);

}

#define CHECK\_PAIR(p, f, s, i) \

EXPECT\_EQ(p.first, f); \

EXPECT\_EQ(p.second, s); \

EXPECT\_EQ(p.index, i);

TEST\_F(CountAndSumTest, вектор\_только\_из\_пары) {

vector<int> userInput{ 5, 6 };

vector<PairWithIndex> res = findPairs(userInput);

EXPECT\_EQ(res.size(), 1);

CHECK\_PAIR(res[0], 5, 6, 0);

}

TEST\_F(CountAndSumTest, вектор\_только\_из\_пары\_погр\_знач) {

for each (borderline\_value b in borderlineValues)

{

int first = b.input[0];

int second = b.input[1];

vector<PairWithIndex> res = findPairs(b.input);

if (b.expectPair)

{

EXPECT\_EQ(res.size(), 1);

CHECK\_PAIR(res[0], first, second, 0);

}

else

{

EXPECT\_EQ(res.size(), 0);

}

}

}

TEST\_F(CountAndSumTest, вектор\_с\_несколькими\_элементами\_пара\_в\_начале\_чёт) {

vector<int> userInput{ 5, 6, 0, 0 };

vector<PairWithIndex> res = findPairs(userInput);

EXPECT\_EQ(res.size(), 1);

CHECK\_PAIR(res[0], 5, 6, 0);

}

TEST\_F(CountAndSumTest, вектор\_с\_несколькими\_элементами\_пара\_в\_начале\_чёт\_погр\_знач) {

for each (borderline\_value b in borderlineValues)

{

int first = b.input[0];

int second = b.input[1];

vector<int> paddedInput = { first, second, 0, 0 };

vector<PairWithIndex> res = findPairs(paddedInput);

if (b.expectPair)

{

EXPECT\_EQ(res.size(), 1);

CHECK\_PAIR(res[0], first, second, 0);

}

else

{

EXPECT\_EQ(res.size(), 0);

}

}

}

TEST\_F(CountAndSumTest, вектор\_с\_несколькими\_элементами\_пара\_в\_начале\_нечёт) {

vector<int> userInput{ 5, 6, 0 };

vector<PairWithIndex> res = findPairs(userInput);

EXPECT\_EQ(res.size(), 1);

CHECK\_PAIR(res[0], 5, 6, 0);

}

TEST\_F(CountAndSumTest, вектор\_с\_несколькими\_элементами\_пара\_в\_начале\_нечёт\_погр\_знач) {

for each (borderline\_value b in borderlineValues)

{

int first = b.input[0];

int second = b.input[1];

vector<int> paddedInput = { first, second, 0 };

vector<PairWithIndex> res = findPairs(paddedInput);

if (b.expectPair)

{

EXPECT\_EQ(res.size(), 1);

CHECK\_PAIR(res[0], first, second, 0);

}

else

{

EXPECT\_EQ(res.size(), 0);

}

}

}

TEST\_F(CountAndSumTest, вектор\_с\_несколькими\_элементами\_пара\_в\_конце\_чёт) {

vector<int> userInput{ 0, 0, 5, 6 };

vector<PairWithIndex> res = findPairs(userInput);

EXPECT\_EQ(res.size(), 1);

CHECK\_PAIR(res[0], 5, 6, 2);

}

TEST\_F(CountAndSumTest, вектор\_с\_несколькими\_элементами\_пара\_в\_конце\_чёт\_погр\_знач) {

for each (borderline\_value b in borderlineValues)

{

int first = b.input[0];

int second = b.input[1];

vector<int> paddedInput = { INT\_MAX, INT\_MAX, first, second };

vector<PairWithIndex> res = findPairs(paddedInput);

if (b.expectPair)

{

EXPECT\_EQ(res.size(), 1);

CHECK\_PAIR(res[0], first, second, 2);

}

else

{

EXPECT\_EQ(res.size(), 0);

}

}

}

TEST\_F(CountAndSumTest, вектор\_с\_несколькими\_элементами\_пара\_в\_конце\_нечёт) {

vector<int> userInput{ 0, 5, 6 };

vector<PairWithIndex> res = findPairs(userInput);

EXPECT\_EQ(res.size(), 1);

CHECK\_PAIR(res[0], 5, 6, 1);

}

TEST\_F(CountAndSumTest, вектор\_с\_несколькими\_элементами\_пара\_в\_конце\_нечёт\_погр\_знач) {

for each (borderline\_value b in borderlineValues)

{

int first = b.input[0];

int second = b.input[1];

vector<int> paddedInput = { INT\_MAX, first, second };

vector<PairWithIndex> res = findPairs(paddedInput);

if (b.expectPair)

{

EXPECT\_EQ(res.size(), 1);

CHECK\_PAIR(res[0], first, second, 1);

}

else

{

EXPECT\_EQ(res.size(), 0);

}

}

}

TEST\_F(CountAndSumTest, вектор\_с\_несколькими\_элементами\_пара\_в\_середине\_чёт) {

vector<int> userInput{ 0, 0, 0, 5, 6, 0, 0, 0 };

vector<PairWithIndex> res = findPairs(userInput);

EXPECT\_EQ(res.size(), 1);

CHECK\_PAIR(res[0], 5, 6, 3);

}

TEST\_F(CountAndSumTest, вектор\_с\_несколькими\_элементами\_пара\_в\_середине\_чёт\_погр\_знач) {

for each (borderline\_value b in borderlineValues)

{

int first = b.input[0];

int second = b.input[1];

vector<int> paddedInput = { INT\_MAX, INT\_MAX, INT\_MAX, first, second, INT\_MIN, INT\_MIN, INT\_MIN };

vector<PairWithIndex> res = findPairs(paddedInput);

if (b.expectPair)

{

EXPECT\_EQ(res.size(), 1);

CHECK\_PAIR(res[0], first, second, 3);

}

else

{

EXPECT\_EQ(res.size(), 0);

}

}

}

TEST\_F(CountAndSumTest, вектор\_с\_несколькими\_элементами\_пара\_в\_середине\_нечёт\_правее) {

vector<int> userInput{ 0, 0, 5, 6, 0 };

vector<PairWithIndex> res = findPairs(userInput);

EXPECT\_EQ(res.size(), 1);

CHECK\_PAIR(res[0], 5, 6, 2);

}

TEST\_F(CountAndSumTest, вектор\_с\_несколькими\_элементами\_пара\_в\_середине\_нечёт\_правее\_погр\_знач) {

for each (borderline\_value b in borderlineValues)

{

int first = b.input[0];

int second = b.input[1];

vector<int> paddedInput = { INT\_MAX, INT\_MAX, first, second, INT\_MIN };

vector<PairWithIndex> res = findPairs(paddedInput);

if (b.expectPair)

{

EXPECT\_EQ(res.size(), 1);

CHECK\_PAIR(res[0], first, second, 2);

}

else

{

EXPECT\_EQ(res.size(), 0);

}

}

}

TEST\_F(CountAndSumTest, вектор\_с\_несколькими\_элементами\_пара\_в\_середине\_нечёт\_левее) {

vector<int> userInput{ 0, 5, 6, 0, 0 };

vector<PairWithIndex> res = findPairs(userInput);

EXPECT\_EQ(res.size(), 1);

CHECK\_PAIR(res[0], 5, 6, 1);

}

TEST\_F(CountAndSumTest, вектор\_с\_несколькими\_элементами\_пара\_в\_середине\_нечёт\_левее\_погр\_знач) {

for each (borderline\_value b in borderlineValues)

{

int first = b.input[0];

int second = b.input[1];

vector<int> paddedInput = { INT\_MAX, first, second, INT\_MIN, INT\_MIN };

vector<PairWithIndex> res = findPairs(paddedInput);

if (b.expectPair)

{

EXPECT\_EQ(res.size(), 1);

CHECK\_PAIR(res[0], first, second, 1);

}

else

{

EXPECT\_EQ(res.size(), 0);

}

}

}

TEST\_F(CountAndSumTest, вектор\_с\_несколькими\_парами\_внач\_вконц\_всеред) {

vector<int> userInput{ 4, 6, 0, 3, 7, 0, 1, 9 };

vector<PairWithIndex> res = findPairs(userInput);

EXPECT\_EQ(res.size(), 3);

CHECK\_PAIR(res[0], 4, 6, 0);

CHECK\_PAIR(res[1], 3, 7, 3);

CHECK\_PAIR(res[2], 1, 9, 6);

}

TEST\_F(CountAndSumTest, вектор\_с\_разными\_значениями) {

vector<int> userInput{ -89, -96, -87, 77, 50, -40, 3, 64, 61, 32, -96, -78, -64, 54, -21, 58, -71, -52, 25, -45 };

vector<PairWithIndex> res = findPairs(userInput);

EXPECT\_EQ(res.size(), 5);

CHECK\_PAIR(res[0],-87, 77, 2);

CHECK\_PAIR(res[1], 3, 64, 6);

CHECK\_PAIR(res[2], -64, 54, 12);

CHECK\_PAIR(res[3], -21, 58, 14);

CHECK\_PAIR(res[4], -52, 25, 17);

}