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 19

Выполнил

студент гр.

Принял:

старший преподаватель Е.В. Старыгина

Ижевск

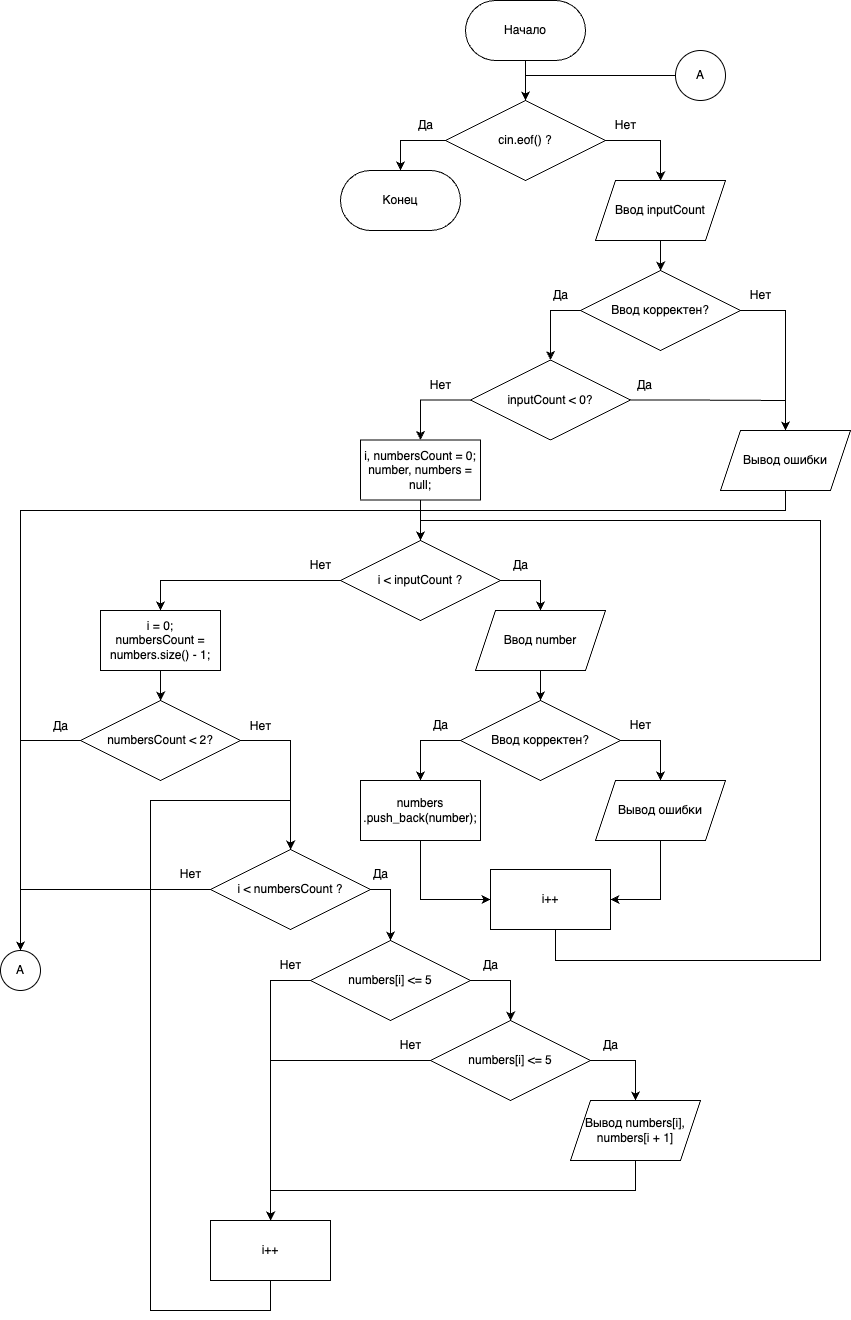
2023

# **Задание**

Дан массив целых чисел а0,...,аn-1. Найти все пары (аi,аi+1), такие что аi<=5 и аi+1>0. Распечатать их значения и номеpа. Если таких паp нет, то выдать сообщение.

# **Блок-схема алгоритма**

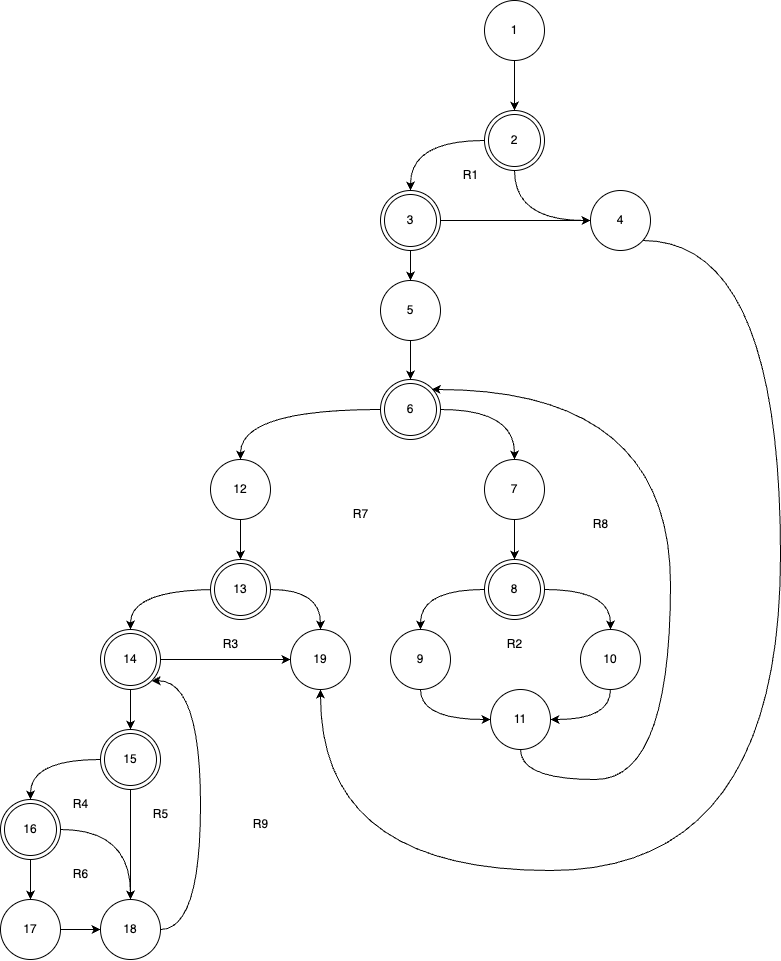
Блок-схема алгоритма представлена на рис. 1.



# **Тестирование базового пути**

* 1. Потоковый граф

Потоковый граф представлен на рис. 2.



* 1. Цикломатическая сложность

Вычислим цикломатическую сложность потокового графа каждым из трёх способов:

1. потоковый граф имеет 9 регионов;
2. V(G) = 26 дуги – 19 узлов + 2 = 9;
3. V(G) = 9 предикатных узлов + 1 = 9.
   1. Базовое множество независимых путей

Перечислим независимые пути:

1. 1 – 2 – 4 – 19; если введено некорректное значение в inputCount.
2. 1 – 2 – 3 – 4 – 19; если введено отрицательное в inputCount.
3. 1 – 2 – 3 – 5 – 6 – 12 – 13 – 19; если введено менее 2 корректных чисел.
4. 1 – 2 – 3 – 5 – 6 – 7 – 8 – 9 – 11 – …; если введено некорректное значение в number(i).
5. 1 – 2 – 3 – 5 – 6 – 7 – 8 – 10 – 11 – …; если едено корректное значение в number(i).
6. 1 – 2 – 3 – 5 – 6 – 12 – 13 – 14 – 19; если введено 2 корректных числа.
7. 1 – 2 – 3 – 5 – 6 – 12 – 13 – 14 – 15 – 18 – 14 – 19; если введено numbers(i) > 5;
8. 1 – 2 – 3 – 5 – 6 – 12 – 13 – 14 – 15 – 16 – 18 – 14 – 19; если введено numbers(i) <= 5 и numbers(i + 1) <= 0;
9. 1 – 2 – 3 – 5 – 6 – 12 – 13 – 14 – 15 – 16 – 17 – 18 – 14 – 19; если введено numbers(i) <= 5 и numbers(i + 1) > 0;
   1. Тестовые варианты

Тестовый вариант для пути 1:

Исходные данные: некорректное значение в inputCount.

Ожидаемые результаты: сообщение о некорректности ввода.

Тестовый вариант для пути 2:

Исходные данные: отрицательное значение в inputCount.

Ожидаемые результаты: сообщение о некорректности ввода.

Тестовый вариант для пути 3:

Исходные данные: допустимое значение в inputCount, введено менее 2 корректных чисел.

Ожидаемые результаты: сообщение об отсутствии чисел для сравнения.

Тестовый вариант для пути 4:

Исходные данные: допустимое значение в inputCount, недопустимое значение в number(i), где i≤ inputCount.

Ожидаемые результаты: сообщение о некорректности ввода, просьба произвести ввод number(i) повторно.

Тестовый вариант для пути 5:

Исходные данные: допустимое значение в inputCount, допустимое значение в number(i), где i≤ inputCount.

Ожидаемые результаты: сообщение о некорректности ввода.

Тестовый вариант для пути 6:

Исходные данные: допустимое значение в inputCount, введено не менее 2 корректных чисел.

Ожидаемые результаты: не найдено ни одной пары.

Тестовый вариант для пути 7:

Исходные данные: допустимое значение в inputCount, введено не менее 2 корректных чисел, допустимое значение numbers(i) > 5, где i < numbersCount.

Ожидаемые результаты: не найдено ни одной пары, включающей number(i).

Тестовый вариант для пути 8:

Исходные данные: допустимое значение в inputCount, введено не менее 2 корректных чисел, допустимое значение numbers(i) <= 5, допустимое зачение numbers(i+1) <= 0, где i < numbersCount.

Ожидаемые результаты: не найдено ни одной пары, включающей number(i).

Тестовый вариант для пути 9:

Исходные данные: допустимое значение в inputCount, введено не менее 2 корректных чисел, допустимое значение numbers(i) <= 5, допустимое зачение numbers(i+1) > 0, где i < numbersCount.

Ожидаемые результаты: вывод найденной пары, включающей числа number(i) и number(i-1).

# **Тестирование потоков данных**

* 1. Информационный граф

Информационный граф представлен на рис. 3.

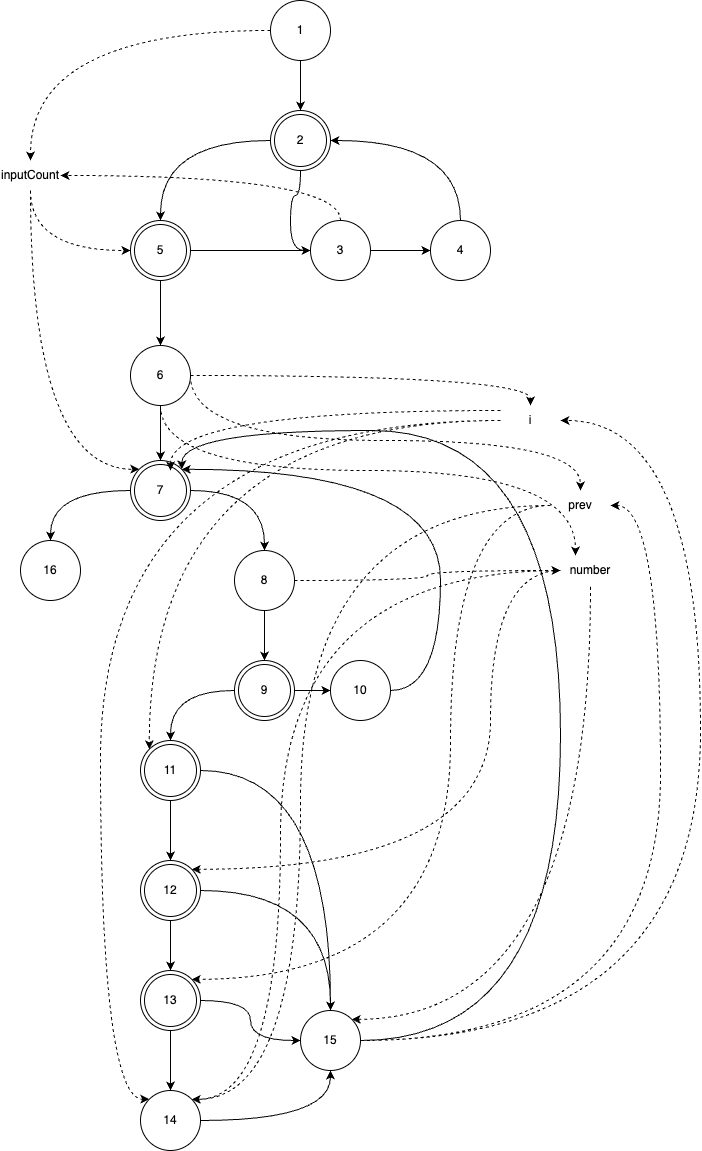


Рисунок 3. Информационный граф

* 1. Формирование полного набора du-цепочек

Сформируем список определений переменных:

1. DEF(1) = {inputCount, 1}
2. DEF(3) = {inputCount, 3}
3. DEF(6) = {i, 6}
4. DEF(6) = {prev, 6}
5. DEF(6) = {number, 6}
6. DEF(8) = {number, 8}
7. DEF(15) = {i, 15}
8. DEF(15) = {prev, 15}

Сформируем список использования переменных:

1. USE(5) = {inputCount, 5}
2. USE(7) = {inputCount, 7}
3. USE(7) = {i, 7}
4. USE(9) = {number, 9}
5. USE(11) = {i, 11}
6. USE(12) = {number, 12}
7. USE(13) = {prev, 13}
8. USE(14) = {prev, 14}
9. USE(14) = {number, 14}
10. USE(14) = {i, 14}
11. USE(15) = {number, 15}

Сформируем список DU-цепочек:

1. DU(inputCount) = {inputCount, 1, 5}
2. DU(inputCount) = {inputCount, 1, 7}
3. DU(inputCount) = {inputCount, 3, 5}
4. DU(inputCount) = {inputCount, 3, 7}
5. DU(i) = {i, 6, 7}
6. DU(i) = {i, 6, 11}
7. DU(i) = {i, 6, 14}
8. DU(i) = {i, 15, 7}
9. DU(i) = {i, 15, 11}
10. DU(i) = {i, 15, 14}
11. DU(prev) = {prev, 6, 13}
12. DU(prev) = {prev, 6, 14}
13. DU(prev) = {prev, 15, 13}
14. DU(prev) = {prev, 15, 14}
15. DU(number) = {number, 6, 9}
16. DU(number) = {number, 6, 12}
17. DU(number) = {number, 6, 14}
18. DU(number) = {number, 6, 15}
19. DU(number) = {number, 8, 9}
20. DU(number) = {number, 8, 12}
21. DU(number) = {number, 8, 14}
22. DU(number) = {number, 8, 15}
    1. Построение маршрутов
23. 1 - inputCount - 5: 1 - 2 - 5 - ...;
24. 1 - inputCount - 7: 1 - 2 - 5 - 6 - 7 - ...;
25. 3 - inputCount - 5: 1 - 2 - 3 - 4 - 2 - 5 - ...;
26. 3 - inputCount - 7: 1 - 2 - 3 - 4 - 2 - 5 - 6 - 7 - ...;
27. 6 - i - 7: ... - 6 - 7 - ...;
28. 6 - i - 11: ... - 6 - 7 - 8 - 9 - 11 - ...;
29. 6 - i - 14: ... - 6 - 7 - 8 - 9 - 11 - 12 - 13 - 14 - ...;
30. 15 - i - 7: ... - 7 - 8 - 9 - 11 - ... - 15 - 7 - ...;
31. 15 - i - 11: ... - 7 - 8 - 9 - 11 - ... - 15 - 7 - ...;
32. 15 - i - 14: ... - 7 - 8 - 9 - 11 - 12 - 13 - 14 - 15 - 7 - ...;
33. 6 - prev - 13: ... - 7 - 8 - 9 - 11 - 12 - 13 - ...;
34. 6 - prev - 14: ... - 7 - 8 - 9 - 11 - 12 - 13 - 14 - ...;
35. 15 - prev - 14: ... - 7 - 8 - 9 - 11 - ... - 15 - 7 - ...;
36. 15 - prev - 14: ... - 7 - 8 - 9 - 11 - 12 - 13 - 14 - 15 - 7 - ...;
37. 6 - number - 9: ... - 6 - 7 - 8 - 9 - ...;
38. 6 - number - 12: ... - 6 - 7 - 8 - 9 - 11 - 12 - ...;
39. 6 - number - 14: ... - 6 - 7 - 8 - 9 - 11 - 12 - 13 - 14 - ...;
40. 6 - number - 15: ... - 6 - 7 - 8 - 9 - 11 - 12 - 13 - 14 - 15 - ...;
41. 8 - number - 9: ... - 8 - 9 - ...;
42. 8 - number - 12: ... - 8 - 9 - 11 - 12 - ...;
43. 8 - number - 14: ... - 8 - 9 - 11 - 12 - 13 - 14 - ...;
44. 8 - number - 15: ... - 8 - 9 - 11 - 12 - 13 - 14 - 15 - ...;

Убрав дублирующиеся маршруты, получаем:

1. 1 - 2 - 5 - 6 - 7 - 16;
2. 1 - 2 - 3 - 4 - 2 - 5 - 6 - 7 - 16;
3. 1 - 2 - 5 - 6 - 7 - 8 - 9 - 11 - 12 - 13 - 14 - 15 - 7 - 16.

# **Области эквивалентности**

Можно выделить области эквивалентности для inputCount на основании типа вводимых данных и их значении (см. табл. 1).

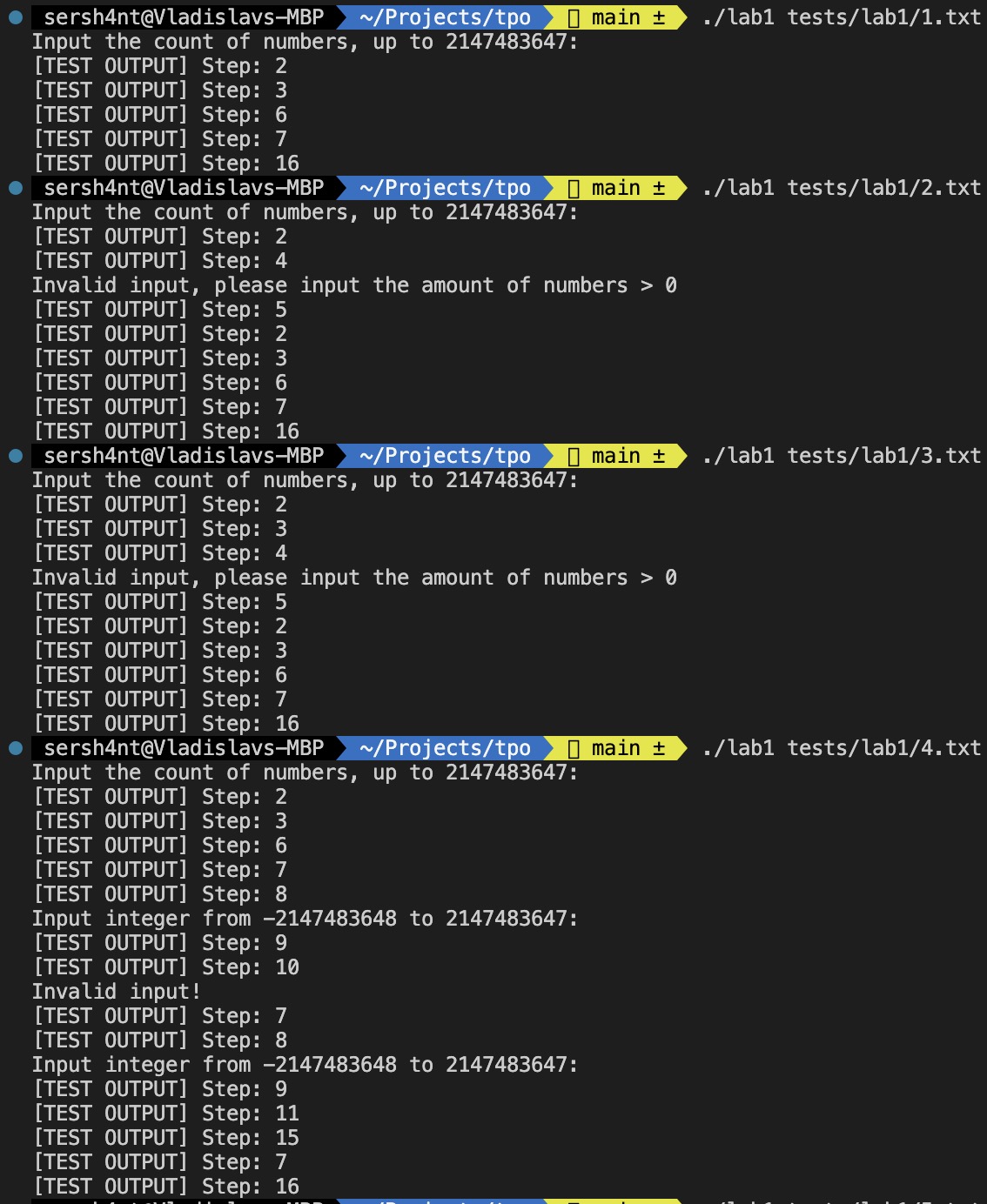
Таблица 1. Области эквивалентности для inputCount

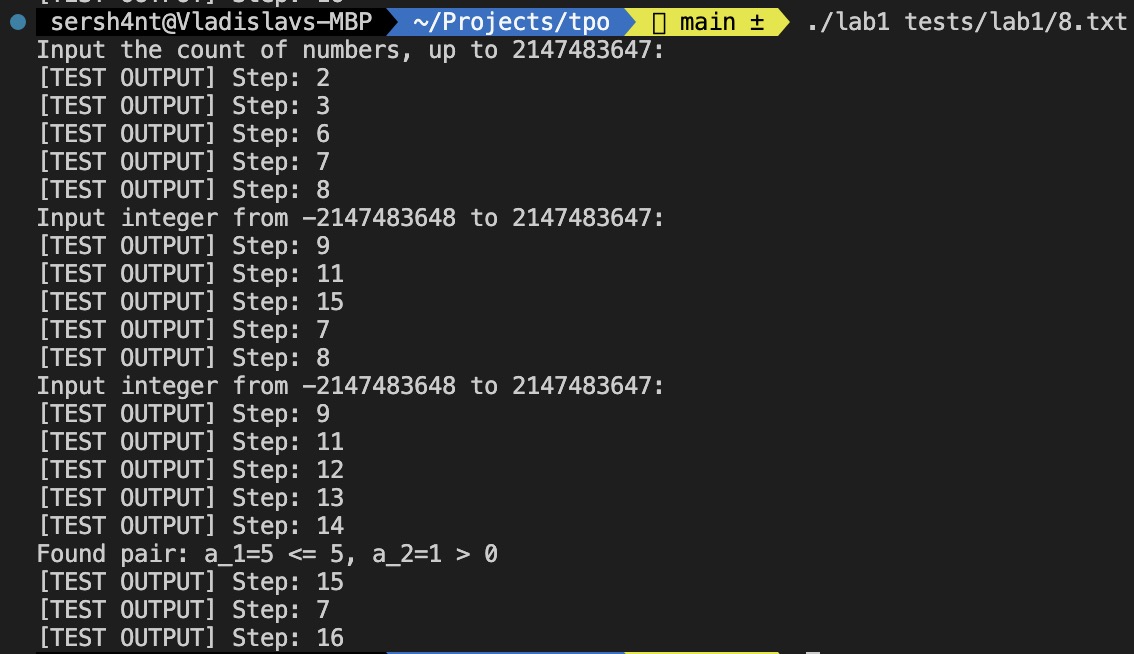
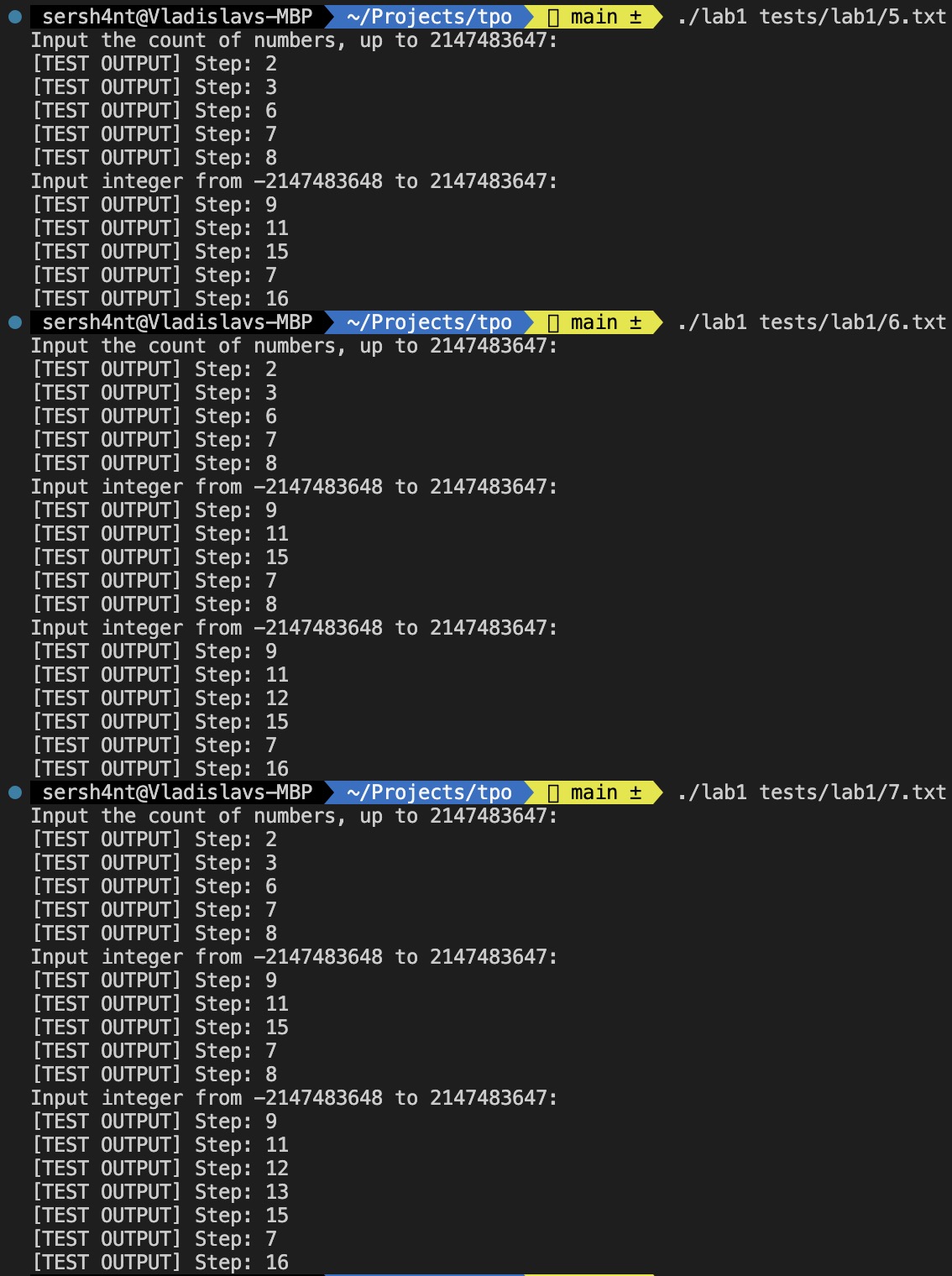
|  |  |
| --- | --- |
| Тип данных | Значение |
| Строка | - |
| Число | Отрицательное |
| Число | Ноль |
| Число | Положительное, но меньше, чем 2^31 – 1, например 10. |
| Число | Максимальное значение int32: 2^31 – 1 = 2147483647 |
| Число | Больше, чем int32: 2^31 = 2147483648 |

Для number можно выделить области эквивалентности на основании типа вводимых данных, размера вводимых данных и их делимости на 5, на 7 (см. табл. 2).

|  |  |  |  |
| --- | --- | --- | --- |
| Тип данных | Размер | Положение относительно 0 | Положение относительно 5 |
| Строка | - | - | - |
| Число | В пределах int32 | > 0 | <= 5 |
| Число | В пределах int32 | > 0 | > 5 |
| Число | В пределах int32 | <= 0 | - |
| Число | Максимальное значение int32: 2^31 – 1 = 2147483647 | - | - |
| Число | Минимальное значение int32: -2^31= -2147483648 | - | - |
| Число | Больше, чем максимальное значение int32: 2^31 = 2147483648 | - | - |
| Число | Меньше, чем минимальное значение int32: -2^31-1 = -2147483649 | - | - |

# **Контрольный пример**





# **Текст программы**

#include <iostream>

#include <cstdint>

using namespace std;

bool PRINT\_STEP(int step)

{

cout << "[TEST OUTPUT] Step: " << step << endl;

return true;

}

int main(int argc, char \*argv[])

{

freopen(argv[1], "r", stdin);

cout << "Input the count of numbers, up to " << numeric\_limits<int>::max() << ":" << endl; // 1

int32\_t inputCount = 0; // 1

cin >> inputCount; // 1

while ((PRINT\_STEP(2) && cin.fail()) // 2

|| (PRINT\_STEP(3) && inputCount < 0)) // 5

{

PRINT\_STEP(4);

cin.clear();

cin.ignore(); // 3

cout << "Invalid input, please input the amount of numbers > 0" << endl; // 3

cin >> inputCount; // 3

PRINT\_STEP(5);

} // 4

PRINT\_STEP(6); // 6

uint32\_t i = 0; // 6

int32\_t prev, number; // 6

while (PRINT\_STEP(7) && i < inputCount) // 7

{

PRINT\_STEP(8); // 8

cout << "Input integer from " << numeric\_limits<int>::min() << " to " // 8

<< numeric\_limits<int>::max() // 8

<< ":" << endl; // 8

cin >> number; // 8

if (PRINT\_STEP(9) && cin.fail()) // 9

{

PRINT\_STEP(10); // 10

cin.clear(); // 10

cin.ignore(); // 10

cout << "Invalid input!" << std::endl; // 10

}

else

{

if (PRINT\_STEP(11) && i > 0) // 11

{

if (PRINT\_STEP(12) && number > 0) // 12

{

if (PRINT\_STEP(13) && prev <= 5) // 13

{

PRINT\_STEP(14); // 14

cout << "Found pair: a\_" << i << "=" << prev << " <= 5, a\_" << i + 1 << "=" << number << " > 0" << endl; // 14

}

}

}

PRINT\_STEP(15); // 15

i++; // 15

prev = number; // 15

}

}

PRINT\_STEP(16); // 16

return 0;

}