ФИО: Ребдев Павел Александрович
Группа: 5130904/30008
Лабораторная работа: «Обработка последовательностей»

Постановка задачи

Разработать детальные требования и тест план для следующей задачи:

[GRT-LSS]. Реализовать программу, которая рассчитывает характеристику последовательности целых чисел, введённых со стандартного ввода: определить, сколько элементов последовательности меньше предыдущего элемента, но больше следующего

Детальные требования

1. Размер последовательности должен быть корректен:
   1.1. Последовательность должна содержать более двух элементов. Если в последовательности два или менее элемента, то программа завершается с выводом сообщения «0» в консоль и кодом возврата 0
   1.2. Последовательность не должна быть слишком длинная. Если в последовательности слишком большое количество элементов, то программа завершается с выводом сообщения «0» в консоль и кодом возврата 0
2. Элементы последовательности должны быть заданы корректно:
   2.1. Каждый элемент последовательности должен быть целым числом. Если хотя бы один элемент последовательности не является целым числом, то программа должна завершаться с кодом возврата 1 и соответствующим сообщением об ошибке в стандартном потоке ошибок
3. Размер последовательности и элементы заданы корректно:
   3.1. Если размер последовательности и элементы заданы корректно, то программа должна завершится с выводом в консоль сообщения, содержащего количество элементов удовлетворяющих условию задачи и кодом возврата 0

Тест-план

Проверка детальных требований с помощью тест-плана:

#	Описание	Результат

1.1	Последовательность должна содержать более двух элементов. Если в последовательности два или менее элемента, то программа завершается с выводом сообщения «0» в консоль и кодом возврата 0	**Input:**
		`1 0`
		Expected:
		`0`

1.2	Последовательность не должна быть слишком длинная. Если в последовательности слишком большое количество элементов, то программа завершается с выводом сообщения «0» в консоль и кодом возврата 0	**Input:**
		`последовательность состоящая из 100000 случайных элементов`
		Expected:
		`0`

2.1	Каждый элемент последовательности должен быть целым числом. Если хотя бы один элемент последовательности не является целым числом, то программа должна завершаться с кодом возврата 1 и соответствующим сообщением об ошибке в стандартном потоке ошибок	**Input:**
		`12 43 -12 A 21 43 0`
		Expected:
		`Input error!`

		Input:
		`12 4,3 -12 1.32 2<1 43 0`
		Expected:
		`Input error!`

3.1	Если размер последовательности и элементы заданы корректно, то программа должна завершится с выводом в консоль сообщения, содержащего количество элементов удовлетворяющих условию задачи и кодом возврата 0	**Input:**
		`6 5 4 3 2 1 0`
		Expected:
		`4`

		Input: `1 2 3 4 5 6 0`

		Expected: 0

Исходные тексты программы

Файлы с исходными текстами лабораторной работы (полагаем <ROOT> для папки в котором располагаются исходные тексты):

./<ROOT>/main.cpp

```
#include <iostream>
#include "elementCounter.hpp"

int main()
{
  using namespace rebdev;

  int number = 0;

  ElementCounter objectOfElementCounter;

  do
  {
    std::cin >> number;
    if (!std::cin)
    {
      std::cerr << "Input error!\n";
      return 1;
    }
    try
    {
      objectOfElementCounter(number);
    }
    catch (const std::exception & e)
    {
      std::cerr << "Error: " << e.what() << '\n';
      return 2;
    }
  }
  while (number != 0);
  try
  {
    size_t a = objectOfElementCounter();
    std::cout << a << '\n';
    return 0;
  }
  catch (const std::exception & e)
  {
    std::cerr << "Error: " << e.what() << '\n';
    return 2;
  }
}
```

./<ROOT>/elementCounter.cpp

```
#include "elementCounter.hpp"
#include <stdexcept>
#include <limits>

rebdev::ElementCounter::ElementCounter():
  lastElement_(0),
  currentElement_(0),
  counter_(0),
  amountOfElements_(0)
{}

void rebdev::ElementCounter::operator()(int nextElement)
{
  if (nextElement != 0)
  {
    amountOfElements_ += 1;
    if (lastElement_ == 0)
    {
      lastElement_ = nextElement;
    }
    else if (currentElement_ == 0)
    {
      currentElement_ = nextElement;
    }
    else
    {
          if ((lastElement_ > currentElement_) && (currentElement_ >
nextElement))
      {
        if (counter_ < std::numeric_limits< size_t >::max())
        {
          counter_ += 1;
        }
        else
        {
            throw std::logic_error("there are too many elements in the
sequence!");
        }
      }
      lastElement_ = currentElement_;
      currentElement_ = nextElement;
    }
  }
}

size_t rebdev::ElementCounter::operator()() const
{
  if (amountOfElements_ < 3)
  {
     throw std::invalid_argument("there are too little elements in the
sequence!");
  }
  return counter_;
```

```
}
```

./<ROOT>/elementCounter.hpp

```
#ifndef ELEMENTCOUNTER_HPP
#define ELEMENTCOUNTER_HPP

#include <cstddef>

namespace rebdev
{
  class ElementCounter
  {
  public:
    ElementCounter();
    void operator()(int nextElement);
    size_t operator()() const;

  private:
    int lastElement_;
    int currentElement_;
    size_t counter_;
    size_t amountOfElements_;
  };
}
#endif
```