Міністерство освіти і науки України
Національний авіаційний університет
Навчально-науковий інститут комп'ютерних інформаційних технологій
Кафедра комп'ютеризованих систем управління

Завдання №1.1
для проходження практики
з дисципліни «Якість програмного забезпечення та тестування»
на тему «Дослідження основних критеріїв вибору тестів»

Виконав:
студент ННІКІТ СП-225
Клокун В. Д.
Перевірив:
Сябрук І. М.

Київ 2018

1 Мета роботи

Ознайомитись з видами тестів та оволодіти навичками вибору тестів.

2 Хід роботи

2.1 Визначення типу трикутника

Створюємо власну версію програми для перевірки типу трикутника (ліст. 2.1). При складанні враховуємо всі тести, що наведені у тексті завдання. Розроблена програма успішно працює з усіма наданим тестовими наборами.

Лістинг 2.1: Програма для перевірки типу трикутника

```
#include <cstdlib>   // EXIT_FAILURE
#include <iostream>  // std::cout, std::cerr
#include <string>    // std::string, std::stod
#include <exception> // std::exception
#include <vector>    // std::vector

enum class TriType {
    INVALID     = -1,
    SCALENE     = 0,
    ISOSCELES   = 1,
    EQUILATERAL = 2,
};

TriType gettritype(const double A, const double B, const double C);
bool isvalidtriangle(const double A, const double B, const double C);

int main(int argc, char* argv[])
{
    // check if a sufficient number of arguments was passed
    if (argc != 4) {
        std::cerr << "Incorrect number of arguments!\n"
            << "Usage: "
            << argv[0]
            << " %lf %lf %lf\n";

        return EXIT_FAILURE;
    }

    std::vector<double> tri;

```

```
31      // start at 1 since relevant arguments start at 1 in argv[]
32      for (std::size_t i = 1; i < 4; i++) {
33          try {
34              std::string strA(argv[i]);
35              tri.push_back(std::stod(strA));
36          } catch (std::exception& e) {
37              std::cerr << "Unable to convert to double: "
38                  << '\"' << argv[i] << "\"\n";
39              return EXIT_FAILURE;
40          };
41      }
42
43      switch (gettritype(tri[0], tri[1], tri[2])) {
44          case TriType::SCALENE:
45              std::cout << "A scalene triangle.\n";
46              break;
47          case TriType::ISOSCELES:
48              std::cout << "An isosceles triangle.\n";
49              break;
50          case TriType::EQUILATERAL:
51              std::cout << "An equilateral triangle.\n";
52              break;
53          default:
54              std::cout << "Not a valid triangle.\n";
55              break;
56      }
57
58      return 0;
59  }
60
61  /*
62   * Checks the type of a given triangle
63   */
64  TriType gettritype(const double A, const double B, const double
    C)
65  {
66      // assume a scalene triangle
67      TriType type = TriType::SCALENE;
68
69      if (!isvalidtriangle(A, B, C))
70          return TriType::INVALID;
71
72      // check if any TWO of the sides are equal
73      if ((A == B) || (A == C) || (B == C)) {
74          type = TriType::ISOSCELES;
75          // check if ALL the sides are equal
```

```
76          if ((A == B) && (B == C)) {
77              type = TriType::EQUILATERAL;
78          }
79      }
80
81      return type;
82  }
83
84  bool isvalidtriangle(const double A, const double B, const
    double C)
85  {
86      bool isvalid = true;
87
88      // check if all triangle sides are positive
89      if (A <= 0 || B <= 0 || C <= 0)
90          isvalid = false;
91
92      // check if given sizes make a valid triangle
93      if (   ((A + B) <= C)
94          || ((A + C) <= B)
95          || ((B + C) <= A) ) {
96          isvalid = false;
97      }
98
99      return isvalid;
100 }
```

2.2 Виведення кожної цифри заданого числа

Створюємо власну версію програми для почергового виведення кожної цифри заданого числа (ліст. 2.2). При складанні враховуємо всі тести, що наведені у тексті завдання. Розроблена програма була успішно протестована.

Лістинг 2.2: Програма для почергового виведення кожної цифри заданого числа

```
1  #include <cstdlib>   // EXIT_FAILURE
2  #include <iostream>  // std::cout, std::cerr
3  #include <exception> // std::exception
4  #include <string>    // std::stoi, std::to_string
5
6  int main(int argc, char* argv[])
7  {
8      int num = 0;
9
10     // check if a correct number of arguments was passed
```

```
11      if (argc != 2) {
12          std::cerr << "Incorrect number of parameters!\n"
13              << "Usage: "
14              << argv[0]
15              << " %d\n";
16          return EXIT_FAILURE;
17      }
18
19      std::string strnum(argv[1]);
        // get string from 1st parameter
20
21      // check if given string is a valid int
22      try {
23          num = std::stoi(strnum);
24      } catch (std::exception& e) {
25          std::cerr << "Unable to convert to int.\n";
26          return EXIT_FAILURE;
27      }
28
29      // print given number (from string) digit by digit
30      for (auto c : std::to_string(num))
31          std::cout << c << '\n';
32
33      return 0;
34  }
```

3 Висновок

Під час виконання даного завдання ми ознайомились з видами тестів та оволоділи навичками вибору тестів. Для першої програми було використано функціональне тестування, яке дозволило перевірити правильність роботи з різними можливими значеннями довжин сторін. Для другої було використано мутаційне тестування, яке дозволило перевірити правильність роботи програми з числами не лише заданого, але й довільного (в межах обраного типу) розміру.