

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»

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

Київ 2018

1 Мета роботи

Ознайомитись з видами тестів, 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

```
1  #include <cstdlib>    // EXIT_FAILURE
2  #include <iostream>  // std::cout, std::cerr
3  #include <string>    // std::string, std::stoi
4  #include <exception> // std::exception
5  #include <vector>    // std::vector
6
7  enum class TriType {
8      INVALID      = -1,
9      SCALENE      = 0,
10     ISOSCELES     = 1,
11     EQUILATERAL   = 2,
12 };
13
14 TriType gettritype(const double A, const double B, const double
15 C);
16
17 bool isvalidtriangle(const double A, const double B, const
18 double C);
19
20 int main(int argc, char* argv[])
21 {
22     // check if a sufficient number of arguments was passed
23     if (argc != 4) {
24         std::cerr << "Incorrect number of arguments!\n"
25         << "Usage: "
26         << argv[0]
27         << " %lf %lf %lf\n";
28
29         return EXIT_FAILURE;
30     }
31
32     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             << '\n' << 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
65 C)
66 {
67     // assume a scalene triangle
68     TriType type = TriType::SCALENE;
69
70     if (!isvalidtriangle(A, B, C))
71         return TriType::INVALID;
72
73     // check if any TWO of the sides are equal
74     if ((A == B) || (A == C) || (B == C)) {
75         type = TriType::ISOSCELES;
76         // 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 invalid = true;
87
88     // check if all triangle sides are positive
89     if (A <= 0 || B <= 0 || C <= 0)
90         invalid = 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         invalid = false;
97     }
98
99     return invalid;
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]);
20     // get string from 1st parameter
21
22     // check if given string is a valid int
23     try {
24         num = std::stoi(strnum);
25     } catch (std::exception& e) {
26         std::cerr << "Unable to convert to int.\n";
27         return EXIT_FAILURE;
28     }
29
30     // print given number (from string) digit by digit
31     for (auto c : std::to_string(num))
32         std::cout << c << '\n';
33
34     return 0;
35 }

```

3 Висновок

Під час виконання даного завдання ми ознайомились з видами тестів та оволоділи навичками вибору тестів.