Министерство Образования, Культуры,

Исследований Республики Молдова

Технический Университет Молдовы

Департамент Программная Инженерия и Автоматика

**Отчёт**

по лабораторной работе №1

**по дисциплине «TS»**

Выполнил: ст.гр. TI-197

Шарафудинов Николай

Проверил: Ангилоглу Анна

Кишинёв - 2022

Лабораторная работа №1

Тема: Основы тестирования и классификация ошибок. Интуитивное тестирование

Цель работы: Тестирование программы, классификация ошибок и составление отчётов

по ошибкам.

Задание к лабораторной работе:

1. Написание программы и спецификации требований к ней в соответствии с вариантом задания к лабораторной работе. Каждое функциональное требование должно быть описано.
2. Написание тестовых сценариев по спецификации требований; тестирование функциональных требований разработанной программы: провести тестирование программы в соответствии с шагами тестирования, представленными в теоретической части и описание найденных дефектов/недочётов/ошибок. Проект может быть написан на любом языке программирования, должен являться прикладной программой (реализованной с помощью интерфейса, input/output файлов, или связанной с базой данных), также это может быть сайтом и т.д.

Содержание отчета:

1. Постановка задачи.
2. Составление спецификации требований.
3. Алгоритм программы.
4. Шаги тестирования.
5. Результаты работы программы.
6. Приведение классификации найденных ошибок. Составление отчёта на одну ошибку в соответствии с приложениями А и Б.
7. Приложение. Листинг программы.

**Ход работы**

**Вариант 19**

«Умножение комплексных чисел»

**Первый цикл тестирования**

**Шаг 1**. Начало тестирования с простого и наиболее очевидного теста. Для начала с программой нужно познакомиться и посмотреть, достаточно ли она стабильна, чтобы её можно было тестировать. В программах, предоставленных для первого формального тестирования, часто сразу же происходит сбой. В первом тесте будет проведено произведение двух комплексных чисел 1+2i и 3+4i. Последовательность действий и результаты приведены в таблице 1.1.

Таблица 1.1 – Порядок действия и результаты произведения комплексных чисел

|  |  |  |
| --- | --- | --- |
| **№ действия** | **Ввод** | **Вывод** |
| 1 | В меню выберем первый вариант, введя 1 | Появляется первое сообщение «Enter first complex number» вводим число. Появляется сообщение «Real part:». |
| 2 | вводим действительную часть первого числа | После введения данных сообщение «Real part:» появляется сообщение «Image part:» введем число |
| 3 | Вводим мнимую часть первого числа, 2 | Появляется сообщение «Enter Second complex number» вводим число. Появляется сообщение «Real part:». |
| 4 | Вводим действительную часть второго числа, 3 | После введения данных сообщение «Real part:» появляется сообщение «Image part:» введем число |
| 5 | Вводим мнимую часть второго комплексного числа, это 4 | Ниже выводится новое сообщение, ждущее ввода «Result = -5.0 + 10.0i  1. Start  2. Exit» |
| 6 | Вводим 2 для выхода из программы | Выводится сообщение об удачном завершении программы  «Process finished with exit code 0» |

На рисунке 1.1 видно, как выглядит экран по окончанию теста.

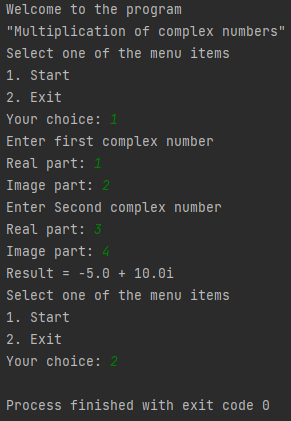


Рисунок 1.1 – Вид экрана по окончанию теста

Отчёт о проблемах, выявленных первым тестом:

“*Программа работает – она приняла числа 1+2i и 3+4i и вернула -5.0+10.0i.*”

В программе не было замечено никаких ошибок проектирования по типу: отсутствие названия, отсутствие инструкций, возможность остановки программы и т.д.

**Шаг 2**. Составление заметок о том, что ещё должно быть протестировано. Выполнив первые, и самые очевидные тесты, следует подумать о том, что ещё следует протестировать.

Таблица 1.2 – Серия тестов для программы сложения двух чисел

|  |  |  |
| --- | --- | --- |
| **Входные данные** | **Ожидаемый результат** | **Замечания** |
| START | Начало работы с программой и вывод сообщения «Enter first complex number  Real part:» | Нигде не указано каким способом можно вводить параметр начала программы. |
| start | Начало работы с программой и вывод сообщения «Enter first complex number  Real part:» | Нигде не указано каким способом можно вводить параметр начала программы. |
| STATS | Вывод ошибки на экран и предоставление возможности попробовать снова выбрать пункт меню | Нигде не указано что произойдет в случае не правильного ввода в начале программы |
| 1+2i -1+2i | Result = -5.0 + 0.0i | Сложение отрицательного числа с положительным числом. |
| 1+2i 1000+10000i | Result = -190000.0 + 120000.0i | Проверка на то, влияет ли слишком большое второе число на интерпретацию первого. |
| -1-2i -1-2i | Result = -5.0 + 10.0i | Все числа отрицательные |
| Пустой ввод | Все пустые вводы заменятся нулями и ответ  Result = 0.0 + 0.0i | Программы часто дают сбои при вводе пустоты |

**Шаг 3**. Проверка допустимых значений и наблюдение за реакцией программы.

Недопустимыми значениями ввода являются границы типа данных, используемые в данной лабораторной работе, а именно от 1. 7е-308 до 1. 7е+ 308. Все же остальные ошибки обрабатываются программой такие как: не введение данных, нули, отрицательные числа, введение символов вместо числовых значений и т.д.

**Шаг 4**. Немного тестирования в режиме «свободного полёта»

Таблица 1.3 – Неформальные тесты для программы сложения двух чисел

|  |  |  |
| --- | --- | --- |
| **Тест** | **Особенности теста** | **Замечания** |
| 1.8e-308 + 1.8e+308i  1.8e+308 + 1.8e-308i | Граничное условие – числа  больше максимального допустимого значения | Программа приняла введенные значения и выдало результат  Result = NaN + Infinityi.  Данный результат является ошибкой |
| <*Enter*> + <*Enter*> | Проверка реакции программы на отсутствие числовых данных. | Когда нажали <*Enter*>, программа напечатала что пустой ввод был заменен нулём Если ввести все пустые вводы, то они все заменятся нулями и ответ  Result = 0.0 + 0.0i |
| * 1. + 2.2i   2. + 2.3i | Числа с десятичной частью. | Так как формат данных для работы double то и проблем с десятичными числами нет. |
| A + b  B + C | Недопустимые символы. | Когда нажали <*Enter*> после <*А*> программа выводит сообщение об ошибке и указывает какая ошибка была совершена. Для продолжения тестирования требуется выбрать пункт меню для начала работы. |
| <*Ctrl* + *A*>, <*Ctrl* + *B*>, <*Ctrl* + *C*>, <*Ctrl* + *D*>, <*F1*>, <*Esc*> | Управляющие символы и функциональные клавиши часто являются источниками проблем. | Для всех комбинаций клавиш, кроме <*Ctrl* + *D*>, прошли успешно. В случае <*Ctrl* + *D*>, происходит ошибка основного потока, что заставляет программу завершится с кодом ошибки 1 |

**Шаг 5**. Подведение итогов о программе и выясненных недостатках.

По итогу данная программа уязвима на числовых границах от 1. 7е-308 до 1. 7е+ 308, а также для функциональных клавиш <*Ctrl* + *D*>.

**Итоги первого цикла тестирования**

Начали с простейшего из возможных тестов. Поскольку программа его прошла, разработали серию формальных тестов, чтобы проверить, как она работает с допустимыми данными. После чего провели несколько неформальных экспериментов и выясним, что программа достаточно стабильна. Записали несколько замечаний, к которым обратимся при тестировании следующей версии программы. Если бы программа успешно прошла первую серию тестов, разрабатывается вторая, более обстоятельная. Если бы программа снова показала себя надежной, продолжается ее тестирование, пока не исчерпались бы идеи или отведенное время. Напоследок провели несколько беглых тестов, не входивших в ранее разработанные серии, и записали замечания на будущее.

**Второй цикл тестирования**

Поговорили с программистом, и он сказал, что скорость работы программы исключительно важна, а вот объём кода не имеет никакого значения. Резолюции программиста на отчётах тестировщика:

1. Ошибка кодирования: сбой при нажатии функциональных клавиш. Резолюция: не проблема. Примечание: не используйте функциональные клавиши

**Шаг 1.** Ознакомление с резолюцией программиста. Так ознакомившись с резолюцией программиста, узнаете, что нужно делать, а чего не нужно.

Из резолюций на отчетах видно, какие тесты больше проводить не нужно, а какие нужно заменить новыми.

**Шаг 2.** Анализ комментариев к ошибкам, которые не будут исправлены.

Вот вероятный список таких клавиш:

a) цифры;

b) знак минус (‘-’);

c) знак плюс (‘+’);

d) пробел до числа;

e) пробел после числа;

f) операции умножения и деления без остатка (‘\*’, ‘/’);

g) знак доллара (‘$’);

h) знак процента (‘%’);

i) скобки;

j) клавиша <*Backspace*>;

m) клавиши управления курсором

**Шаг 3**. Просмотр записей, которые были сделаны в прошлый раз. Необходимо добавить к предыдущим замечаниям новые и приступить к тестированию.

Составляете отчёты по следующим ошибкам:

1. *Ошибка проектирования*. Для вывода на экран подсказки тратится лишнее компьютерное время. Поскольку одной из задач разработки является создание очень быстрой программы, это ошибка.

**Результат работы программы**

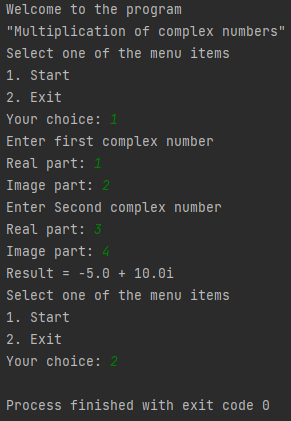


Рисунок 2 – результат работы программы.

**Приложение**

Класс комплексного числа

import java.nio.charset.StandardCharsets;  
  
public class Znumber {  
 private double realPart;  
 private double imagePart;  
  
 public Znumber(double realPart, double imagePart) {  
 this.realPart = realPart;  
 this.imagePart = imagePart;  
 }  
  
 public double getRealPart() {  
 return realPart;  
 }  
  
 public void setRealPart(double realPart) {  
 this.realPart = realPart;  
 }  
  
 public double getImagePart() {  
 return imagePart;  
 }  
  
 public void setImagePart(double imagePart) {  
 this.imagePart = imagePart;  
 }  
  
 Znumber mul (Znumber a) {// умножить комплексные числа  
 double real2 = a.getRealPart();  
 double image2 = a.getImagePart();  
 double newReal = realPart \* real2 - imagePart \* image2;  
 double newImage = imagePart \* real2 + realPart \* image2;  
 Znumber result = new Znumber(newReal, newImage);  
 return result;  
 }  
  
}

основной класс, где происходит общение с пользователем

import java.util.Locale;  
import java.util.Scanner;  
  
public class main {  
 public static void main(String[] args) {  
  
 double real = 0, image = 0; String choose;  
 Scanner scanner = new Scanner(System.*in*);  
 Znumber firstComplexNumber = null, secondComplexNumber = null;  
  
 System.*out*.println("Welcome to the program\n" +  
 "\"Multiplication of complex numbers\"");  
 while(true) {  
 System.*out*.println("Select one of the menu items");  
 System.*out*.println("1. Start");  
 System.*out*.println("2. Exit");  
 System.*out*.print("Your choice: ");  
 choose = scanner.nextLine();  
 if (choose.equals("1") || choose.toLowerCase(Locale.*ROOT*).equals("start")) {  
 try {  
 System.*out*.println("Enter first complex number");  
 String[] str = *dialog*().split(":");  
  
 real = Double.*parseDouble*(str[0]);  
 image = Double.*parseDouble*(str[1]);  
 firstComplexNumber = new Znumber(real, image);  
  
 System.*out*.println("Enter Second complex number");  
 String[] str1 = *dialog*().split(":");  
 real = Double.*parseDouble*(str1[0]);  
 image = Double.*parseDouble*(str1[1]);  
  
 secondComplexNumber = new Znumber(real, image);  
  
 } catch (Exception e) {  
 System.*out*.println("\n"+e.getMessage());  
 System.*out*.println("INCORRECTLY ENTERED DATA!!!!\n");  
 continue;  
 }  
  
 Znumber res = secondComplexNumber.mul(firstComplexNumber);  
 if (res.getImagePart() >= 0)  
 System.*out*.println("Result = " + res.getRealPart() + " + " + res.getImagePart() + "i");  
 else if (res.getRealPart() == 0 && res.getImagePart() == 0)  
 System.*out*.println("Result = 0");  
 else  
 System.*out*.println("Result = " + res.getRealPart() + "" + res.getImagePart() + "i");  
 }  
 else if (!choose.toLowerCase(Locale.*ROOT*).equals("exit")){  
 System.*out*.println("INCORRECTLY ENTERED DATA!!!!\n");  
 }  
 else  
 break;  
 }  
 }  
  
 static String dialog(){  
 Scanner scanner = new Scanner(System.*in*);  
  
 System.*out*.print("Real part: ");  
 String strReal = scanner.nextLine();  
 if (strReal == null || strReal.equals("")) {strReal = "0";  
 System.*out*.println("\t replaced by zero");}  
  
 System.*out*.print("Image part: ");  
 String strImage = scanner.nextLine();  
 if (strImage == null || strImage.equals("")) {strImage = "0";  
 System.*out*.println("\t replaced by zero");}  
  
  
 return strReal+":"+strImage;  
 }  
  
}