Министерство образования ХХХ

Государственное бюджетное профессиональное образовательное учреждение ХХХ «Колледж «ХХХ»

09.02.07

ОТЧЕТ

По лабораторным работам

МДК 04.02 Обеспечение качества функционирования компьютерных систем.

ККОО.ПМ.ХХХ.000

Студент ХХХ

Преподаватель ХХХ

Дата защиты\_\_\_\_\_\_\_ Оценка\_\_\_\_\_\_\_\_\_

2022Лабораторная работа №2

«Выявление первичных и вторичных ошибок»

Тема 2.1 Основные методы обеспечения качества функционирования

Цель работы: «Провести тестирование и отладку программного продукта»

Материально-техническое обеспечение: Компьютер, операционная система Windows

Краткие теоретические сведения:

Одной из наиболее трудоемких задач, решаемых на этапе разработки, является тестирование и отладка программ. Под отладкой следует понимать процесс, позволяющий получить программу, функционирующую с заданными характеристиками в заданной области входных данных.

Основным методом отладки является тестирование. Тест – это последовательность исходных данных, подаваемых на вход изделия и соответствующие им наборы эталонных результирующих данных.

Процесс отладки включает:

1. создание совокупности тестовых эталонных заданий и значений, которым должна соответствовать программа.
2. статическую проверку текстов разрабатываемых программ,
3. тестирование и выполнение программ с различным уровнем детализации,
4. комплексную динамическую отладку, при необходимости, в режиме реального времени
5. диагностику и локализацию причин отклонения результатов тестов от эталонных,
6. изменение программы с целью исключения причин отклонений.

Можно выделить три основных стадии тестирования:

1. стадия обнаружения ошибок в программе (на этой стадии выявляются все отклонения результатов функционирования от эталонных)
2. стадия диагностики и локализации причин (на этой стадии необходимо точно определить место, в котором произошло искажение программы или данных и установить причину)
3. стадия контроля выполнения корректировок (после локализации и устранения ошибок выполняется контрольное тестирование, подтверждающее правильность выполненной корректировки и подтверждающее, что в результате корректировки не возникли вторичные ошибки).

Эффективность тестирования определяет стоимость и длительность разработки.

Характеристики ошибок в процессе проектирования ПО помогают:

* оценить реальное состояние проекта, планировать трудоемкость, стоимость, и длительность разработки,
* разрабатывать эффективные средства оперативной защиты от невыявленных первичных ошибок,
* оценивать требуемые ресурсы с учетом затрат на устранение ошибок, и т.д.

Анализ первичных ошибок проводится на двух уровнях детализации:

Во-первых, дифференциированно– с учетом типов ошибок, сложности и степени автоматизации их выявления, затрат на корректировку и этапов наиболее вероятного устранения.

Во-вторых, обобщенно – по суммарным характеристикам их обнаружения в зависимости от продолжительности разработки, эксплуатации и сопровождения ПО.

Существует несколько основных типов ошибок:

1. Технические ошибки документации и фиксирования программы в памяти машины (составляют 5-10% от общего объема ошибок, большинство выявляется автоматизированными формализованными методами).
2. Программные ошибки (по количеству и типу определяются: степенью квалификации разработчика, степенью автоматизации разработки, глубиной формализованного контроля текстов программ, объемом и сложностью разрабатываемого ПО, глубиной логического и информационного взаимодействия модулей и другими факторами).
3. Алгоритмические ошибки – обнаружение таких ошибок методами формализованного контроля практически невозможно. Как правило, эти ошибки выявляются только на этапе эксплуатации. К ним можно отнести ошибки, вызванные некорректной постановкой задачи или ее неверной интерпретации разработчиком.
4. Системные – такие ошибки определяются неполной информацией о реальных процессах, происходящих в источниках и потребителях информации, причем эти процессы не зависят от алгоритмов и не могут быть заранее определены и описаны они выявляются при исследовании функционирования ПО и при обработке результатов его взаимодействия с внешней средой.

Порядок выполнения лабораторной работы:

1. Изучить теоретический материал.
2. Выполнить предлагаемые задания.
3. Ответить на контрольные вопросы и предоставить в тетради в виде отчета. Отчет должен включать:

* номер, наименование лабораторной работы и тему;
* ответы на контрольные вопросы;
* выводы.

1. Выполненную работу и отчет по проделанной работе предъявить преподавателю.

Задания для выполнения лабораторной работы:

1. Провести тестирование разработанного программного продукта и выявить ошибки.

Вариант 11

class Program

{

static double F(double n)

{

return Math.Pow(-1, n) / (n \* Math.Log(n));

}

static void Main(string[] args)

{

double sum = 0;

Console.Write("Введите e:");

double e = double.Parse(Console.ReadLine());

double n = 2;

while (Math.Abs(F(n)) >= e)

{

sum += F(n);

Console.WriteLine($"{F(n)}");

n++;

}

Console.WriteLine($"Сумма = {sum}");

Console.ReadKey();

}

}

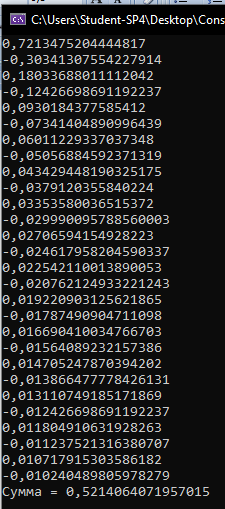


Рисунок 1 Сумма ряда

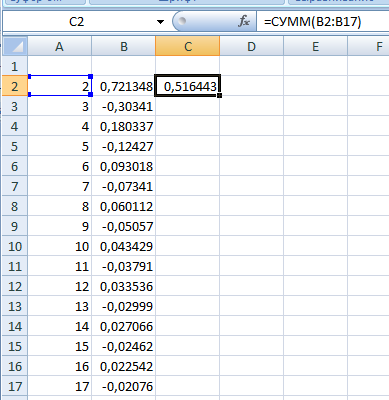


Рисунок 2 Сумма ряда в Excel

1. Используя теоретический материал, проанализировать, классифицировать имеющиеся ошибки.

class Program

{

static double Factorial1(double n)

{

double res = 1;

for (double i = n; i < 1; i++)

res \*= i;

return res;

}

static double Factorial2(double n)

{

double res = 1;

for (double i = 2 \* n; i > 1; i--)

res \*= i;

return res;

}

static double F(double n)

{

retorn(Math.Pow(-1, n) \* Math.Pow(Factorial1(n), 2)) / Factorial2(n);

}

static void Main(string[] args)

{

double sum = 0;

Console.Write("Введите точность ряда: ");

int e = int.Parse(Console.ReadLine());

duble n = 1;

while (Math.Abs(F(n)) <= e)

{

Console.WriteLine($"Элемент номер {n}: {F(n):f4}");

sum += F(n);

n++;

}

Console.WriteLine($"");

Console.WriteLine("Сумма ряда = {sum:f4}");

Console.ReadKey();

}

}

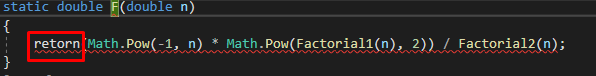


Рисунок 3 Неправильная запись оператора return

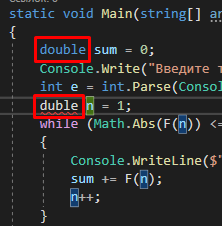


Рисунок 4 Неправильная запись типа данных double

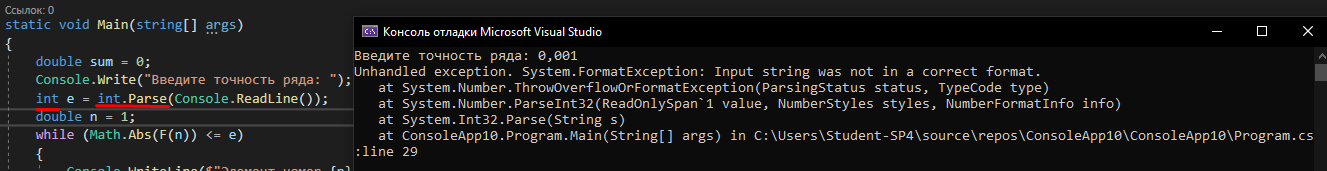


Рисунок 5 Неправильный преобразование типа введенных данных

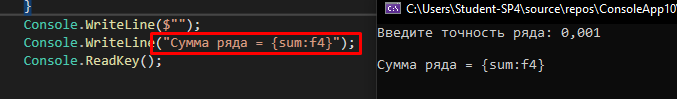


Рисунок 6 Неверная интерполяция строк

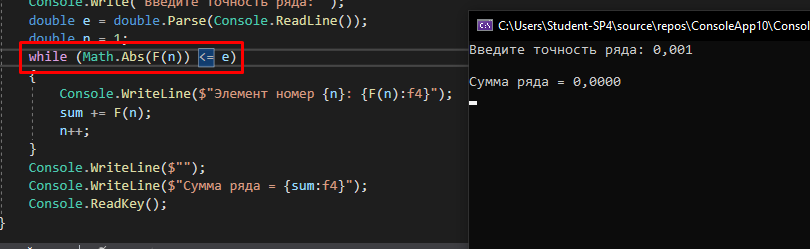


Рисунок 7 Неверное определение условия точности вычислений

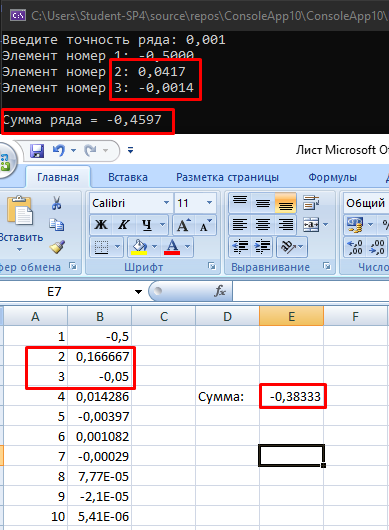


Рисунок 8 Ошибка в вычислениях

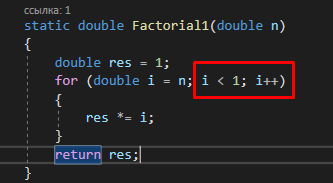


Рисунок 9 Найденная ошибка в методе подсчета факторила

1. Осуществить корректировку выявленных ошибок.

Осуществленная корректировка выявленных ошибок представлена в коде ниже:

class Program

{

static double Factorial1(double n)

{

double res = 1;

for (double i = n; i > 1; i--)

{

res \*= i;

}

return res;

}

static double Factorial2(double n)

{

double res = 1;

for (double i = 2 \* n; i > 1; i--)

{

res \*= i;

}

return res;

}

static double F(double n)

{

return (Math.Pow(-1, n) \* Math.Pow(Factorial1(n), 2)) / Factorial2(n);

}

static void Main(string[] args)

{

double sum = 0;

Console.Write("Введите точность ряда: ");

double e = double.Parse(Console.ReadLine());

double n = 1;

while (Math.Abs(F(n)) >= e)

{

Console.WriteLine($"Элемент номер {n}: {F(n):f4}");

sum += F(n);

n++;

}

Console.WriteLine($"");

Console.WriteLine($"Сумма ряда = {sum:f4}");

Console.ReadKey();

}

}

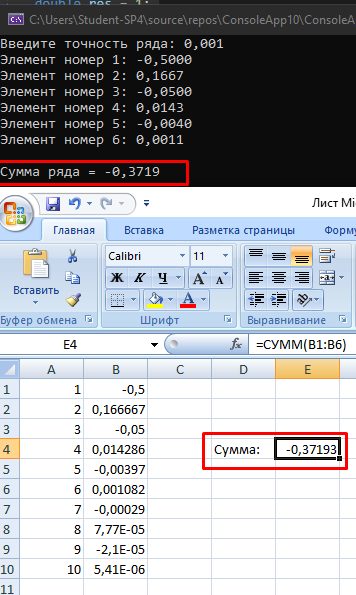


Рисунок 10 Исправленная программа

1. Проверить программу на наличие вторичных ошибок.

Сбоев, катастрофических и ординарных отказов в данной программе не было обнаружено.

Содержание отчета:

Программа без ошибок, готовая к эксплуатации, представленная на электронном носителе

Контрольные вопросы:

* + - 1. Для чего необходимо проводить тестирование ПО?

Для оценки реального состояние проекта, планирования трудоемкости, стоимости, и длительности разработки. А также для разработки эффективных средств оперативной защиты от не выявленных первичных ошибок.

* + - 1. Перечислите основные типы ошибок при тестировании?
* Технические ошибки документации и фиксирования программы в памяти машины (составляют 5-10% от общего объема ошибок, большинство выявляется автоматизированными формализованными методами).
* Программные ошибки (по количеству и типу определяются: степенью квалификации разработчика, степенью автоматизации разработки, глубиной формализованного контроля текстов программ, объемом и сложностью разрабатываемого ПО, глубиной логического и информационного взаимодействия модулей и другими факторами).
* Алгоритмические ошибки – обнаружение таких ошибок методами формализованного контроля практически невозможно. Как правило, эти ошибки выявляются только на этапе эксплуатации. К ним можно отнести ошибки, вызванные некорректной постановкой задачи или ее неверной интерпретации разработчиком.
* Системные – такие ошибки определяются неполной информацией о реальных процессах, происходящих в источниках и потребителях информации, причем эти процессы не зависят от алгоритмов и не могут быть заранее определены и описаны они выявляются при исследовании функционирования ПО и при обработке результатов его взаимодействия с внешней средой.