**Лабораторная работа №14 «Тестирование интеграции»**

**Цель работы.** Получить практические навыки отладки программ с помощью отладчика среды программирования.

using System;

class Program {

static void Main(string[] args) {

Random rnd = new Random();

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

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

int [,] mas = new int [n,m];

for(int i=0; i<n; i++)

for(int j=0; j<m; j++)

mas[i,j]=rnd.Next(30)-10;

for(int i=0; i<n; i++)

{Console.WriteLine();

for(int j=0; j<m; j++)

Console.Write("{0} ", mas[i,j]);}

Console.WriteLine();

int kol;

int max=0;

int k=0;

for(int i=0; i<mas.GetLength(0);i++)

{

kol=0;

for(int j=0; j<mas.GetLength(1);j++)

if (mas[i, j]>0)

kol++;

if(kol>max)

{

max=kol;

k=i;

}

}

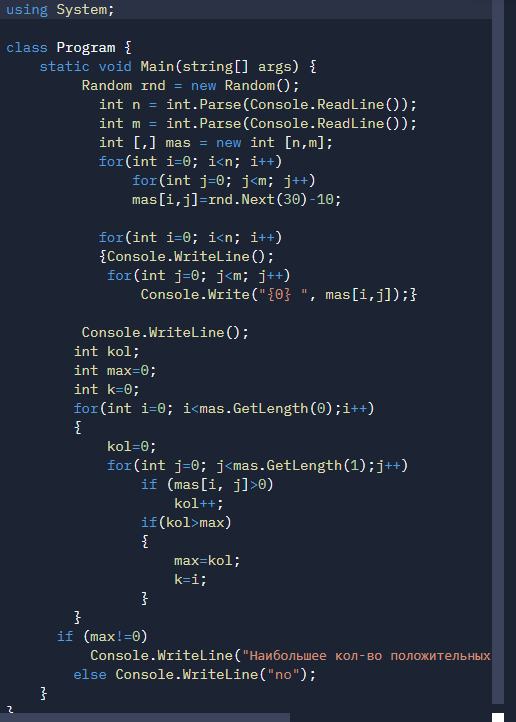
if (max!=0)

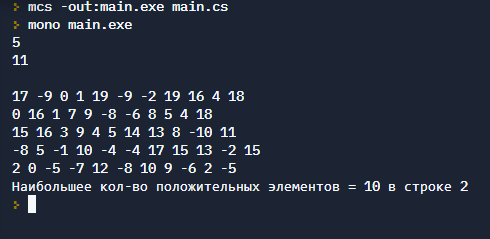
Console.WriteLine("Наибольшее кол-во положительных элементов = {0} в строке {1}", max, k);

else Console.WriteLine("no");

}

}





1) Тестирование программного обеспечения (Software Testing) — проверка соответствия реальных и ожидаемых результатов поведения программы, проводимая на конечном наборе тестов, выбранном определённым образом. Цель тестирования — проверка соответствия ПО предъявляемым требованиям, обеспечение уверенности в качестве ПО, поиск очевидных ошибок в программном обеспечении, которые должны быть выявлены до того, как их обнаружат пользователи программы.

2)

Отладка - это процесс локализации и исправления ошибок в программе. Принципы отладки.

3) Шестой этап – тестирование программы. Различается два вида тестирования: автономное и комплексное. При автономном тестированию подвергаются отдельные программные модули, из которых состоит программный комплекс. Комплексное тестирование заключается в проверке всего программного комплекса.

4) Структурный подход базируется на том, что известна структура и алгоритмы тестируемого ПО, («стеклянный ящик»). Тесты строят так, чтобы проверить правильность реализации заданной логики в коде программы. Функциональный подход основывается на том, что структура ПО не известна («черный ящик»), тесты строят, опираясь на функциональные спецификации.

5) Метод покрытия операторов. Целью этого метода тестирования является выполнение каждого оператора программы хотя бы один раз. Если для тестирования задать значения переменных А = 2, B = 0, Х=3, будет реализован путь асе, т. е. каждый оператор программы выполнится один раз (рис. Л5.1, а). Но если внести в алгоритм ошибки — заменить в первом условии and на or, а во втором Х> 1 на Х< 1

6) Более сильный метод тестирования известен как покрытие решений (покрытие переходов). Согласно данному методу должно быть написано достаточное число тестов, такое, что каждое направление перехода должно быть реализовано по крайней мере один раз. Покрытие решений обычно удовлетворяет критерию покрытия операторов.

7) Покрытие решения или покрытие Ветвления — это метод тестирования, целью которого является обеспечение того, чтобы каждое из возможных ответвлений от каждой точки принятия решения выполнялось хотя бы один раз, и, таким образом, обеспечение выполнения всего достижимого кода. То есть каждое решение принимается по-разному, верно и ложно.

8) Комбинаторные методы построения тестов основаны на разделении каждого тестового воздействия на ряд элементов и построении тестов как всевозможных комбинаций полученных элементов, объединяемых по определенным правилам. Комбинаторные методы дают более высокую полноту покрытия, чем вероятностные, и при этом требуют ненамного больше ресурсов.

9) Эквивалентное разбиение – это метод тестирования «черного ящика». Идея тестирования по методу разбиения классов эквивалентности состоит в том, чтобы исключить набор входных данных, которые заставляют систему вести себя одинаково и давать одинаковый результат при тестировании программы.

10) Анализ граничных значений заключается в получении тестовых вариантов, которые анализируют граничные значения. Данный способ тестирования дополняет способ разбиения на эквивалентности. Основные отличия анализа граничных значений от разбиения на эквивалентности: Тестовые варианты создаются для проверки только границ классов эквивалентностей.

11) Анализ причинно-следственных связей – это структурированный метод, применяемый для определения возможных причин нежелательного события или проблемы.

Вывод: мы получили практические навыки отладки программ с помощью отладчика среды программирования.