ПРАКТИЧЕСКАЯ РАБОТА №4 ПРОГРАММИРОВАНИЕ ЦИКЛИЧЕСКИХ АЛГОРИТМОВ

Цель работы: изучить простейшие средства отладки программ в среде Visual Studio. Составить и отладить программу циклического алгоритма.

4.1. Операторы организации циклов

Под циклом понимается многократное выполнение одних и тех же операторов при различных значениях промежуточных данных. Число повторений может быть задано в явной или неявной форме.

К операторам цикла относятся: цикл с предусловием **while**, цикл с постусловием **do while**, цикл с параметром **for** и цикл перебора **foreach.** Рассмотрим некоторые из них.

4.2. Цикл с предусловием while

Оператор цикла **while** организует выполнение одного оператора (простого или составного) неизвестное заранее число раз. Формат цикла **while**:

while (B) S;

где **B** - выражение, истинность которого проверяется (условие завершения цикла); **S** - тело цикла - оператор (простой или составной).

Перед каждым выполнением тела цикла анализируется значение выражения **B**: если оно истинно, то выполняется тело цикла, и управление передается на повторную проверку условия **B**; если значение **B** ложно - цикл завершается и управление передается на оператор, следующий за оператором **S**.

Если результат выражения **B** окажется ложным при первой проверке, то тело цикла не выполнится ни разу. Отметим, что если условие **B** во время работы цикла не будет изменяться, то возможна ситуация зацикливания, то есть невозможность выхода из цикла. Поэтому внутри тела должны находиться операторы, приводящие к изменению значения выражения **B** так, чтобы цикл мог корректно завершиться.

В качестве иллюстрации выполнения цикла **while** рассмотрим программу вывода в консоль целых чисел из интервала от 1 до n.

```
static void Main()
{
 Console.Write("N= ");
 int n=int.Parse(Console.ReadLine());
 int i = 1;
 while (i <= n)          //пока i меньше или равно n
 Console.Write(" "+ i++ ); //выводим i на экран, затем увеличиваем его на
1
}
```

Результаты работы программы:

n ответ
10 1 2 3 4 5 6 7 8 9 10

4.3. Цикл с постусловием do while

Оператор цикла **do while** также организует выполнение одного оператора (простого или составного) неизвестное заранее число раз. Однако в отличие от цикла **while** условие завершения цикла проверяется после выполнения тела цикла. Формат цикла **do while**:

do S while (B);

где **B** - выражение, истинность которого проверяется (условие завершения цикла); **S** - тело цикла - оператор (простой или блок).

Сначала выполняется оператор **S**, а затем анализируется значение выражения **B**: если оно истинно, то управление передается оператору **S**, если ложно - цикл завершается, и управление передается на оператор, следующий за условием **B**. Так как условие **B** проверяется после выполнения тела цикла, то в любом случае тело цикла выполнится хотя бы один раз.

В операторе **do while**, так же как и в операторе **while**, возможна ситуация зацикливания в случае, если условие **B** всегда будет оставаться истинным.

4.4. Цикл с параметром for

Цикл с параметром имеет следующую структуру:

for (<инициализация>; <выражение>; <модификация>) <оператор>;

Инициализация используется для объявления и/или присвоения начальных значений величинам, используемым в цикле в качестве параметров (счетчиков). В этой части можно записать несколько операторов, разделенных запятой. Областью действия переменных, объявленных в части инициализации цикла, является цикл и вложенные блоки. Инициализация выполняется один раз в начале исполнения цикла.

Выражение определяет условие выполнения цикла: если его результат истинен, цикл выполняется. Истинность выражения проверяется перед каждым выполнением тела цикла, таким образом, цикл с параметром реализован как цикл с предусловием. В блоке выражение через запятую можно записать несколько логических выражений, тогда запятая равносильна операции логическое И (&&).

Модификация выполняется после каждой итерации цикла и служит обычно для изменения параметров цикла. В части модификация можно записать несколько операторов через запятую.

Оператор (простой или составной) представляет собой тело цикла.

Любая из частей оператора **for** (инициализация, выражение, модификация, оператор) может отсутствовать, но точку с запятой, определяющую позицию пропускаемой части, надо оставить.

Пример формирования строки состоящей из чисел от 0 до 9 разделенных пробелами:

```
for (var i = 0; i <= 9; i++)
```

```
{
   s += i + " ";
}
```

Данный пример работает следующим образом. Сначала вычисляется начальное значение переменной **i**. Затем, пока значение **i** меньше или равно 9, выполняется тело цикла и затем повторно вычисляется значение i. Когда значение i становится больше 9, условие становится ложным и управление передается за пределы цикла.

4.2. Средства отладки программ

Практически в каждой вновь написанной программе после запуска обнаруживаются ошибки.

Ошибки первого уровня (ошибки компиляции) связаны с неправильной записью операторов (орфографические, синтаксические). При обнаружении ошибок компилятор формирует список, который отображается по завершению компиляции (рис. 4.1.). При этом возможен только запуск программы, которая была успешно скомпилирована для предыдущей версии программы.

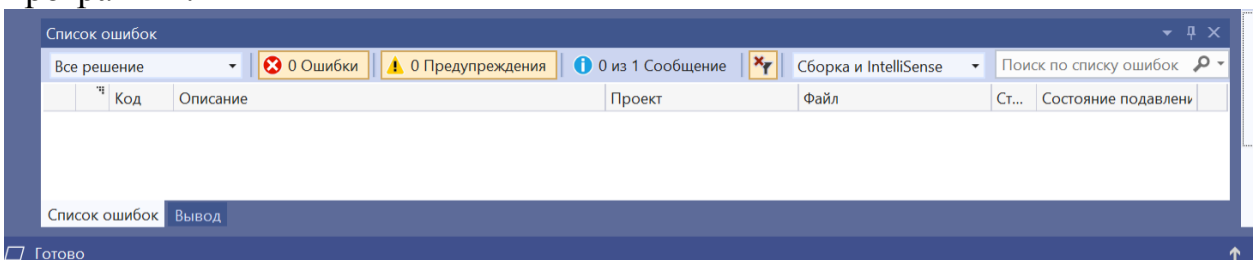

Рис. 4.1. Окно со списком ошибок компиляции.

При выявлении ошибок компиляции в нижней части экрана появляется текстовое окно (рис 4.1.), содержащее сведения обо всех ошибках, найденных в проекте. Каждая строка этого окна содержит имя файла, в котором найдена ошибка, номер строки с ошибкой и характер ошибки. Для быстрого перехода к интересующей ошибке необходимо дважды щелкнуть на строке с ее описанием. Следует обратить внимание на то, что одна ошибка может повлечь за собой другие, которые исчезнут при ее исправлении. Поэтому следует исправлять ошибки последовательно, сверху вниз и, после исправления каждой ошибки компилировать программу снова.

Ошибки второго уровня (ошибки выполнения) связаны с ошибками выбранного алгоритма решения или с неправильной программной реализацией алгоритма. Эти ошибки проявляются в том, что результат расчета оказывается неверным либо происходит переполнение, деление на ноль и др. Поэтому перед использованием отлаженной программы ее надо протестировать, т.е. сделать просчеты при таких комбинациях исходных данных, для которых заранее известен результат. Если тестовые расчеты указывают на ошибку, то для ее поиска следует использовать встроенные средства отладки среды программирования.

В простейшем случае для локализации места ошибки рекомендуется поступать следующим образом. В окне редактирования текста установить точку останова перед подозрительным участком, которая позволит остановить выполнение программы и далее более детально следить за ходом работы операторов и изменением значений переменных. Для этого достаточно в окне редактирования кода щелкнуть слева от нужной строки правой кнопкой мыши и выбрать точка останова/выбрать точку останова. В результате чего данная строка будет выделена красным (рис. 4.2.).

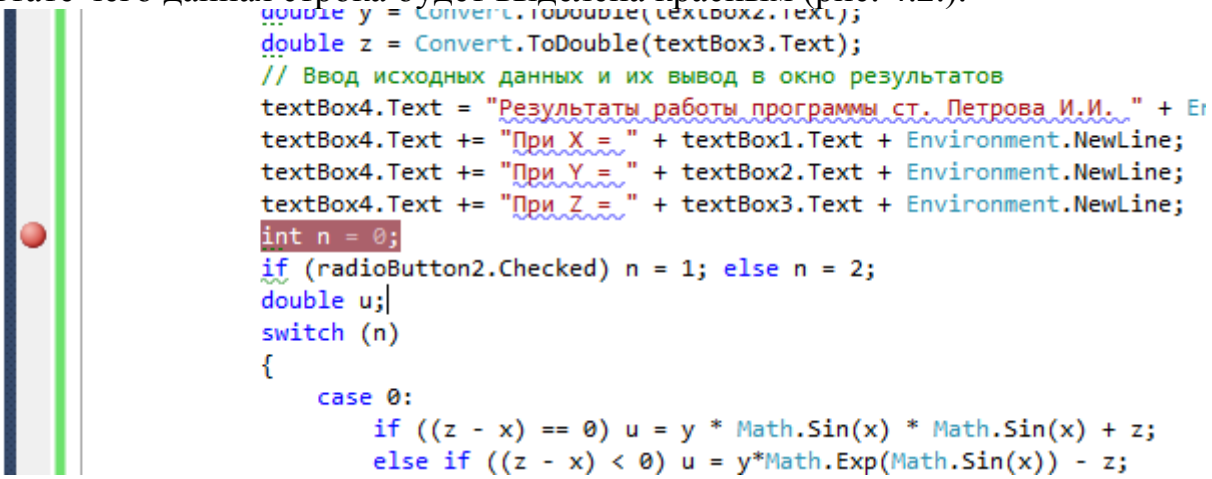

Рис. 4.2. Фрагмент кода с точкой останова

При выполнении программы и достижения установленной точки, программа будет остановлена и далее можно будет выполнять код по операторно с помощью кнопок F10 (без захода в подпрограммы) или F11 (с заходом в подпрограммы) (рис 4.3.).

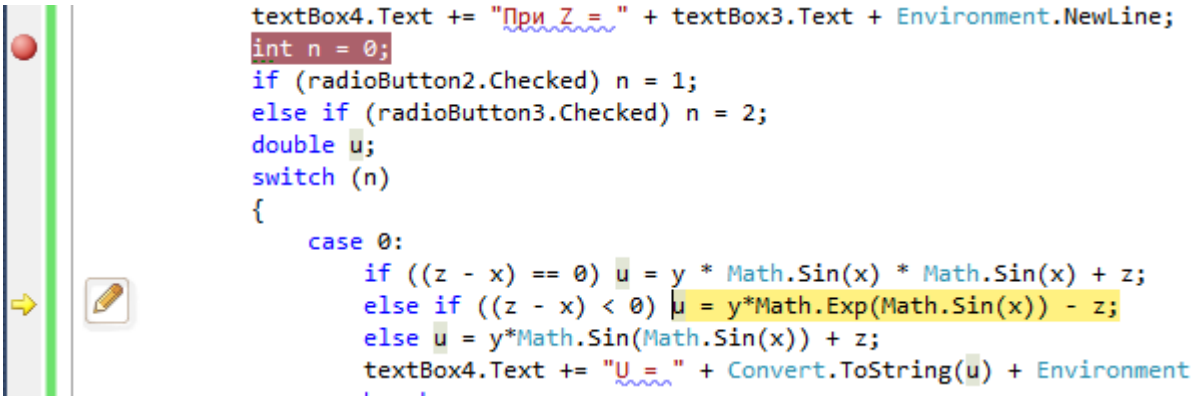

Рис. 4.3. Отладка программы

Желтым выделяется оператор, который будет выполнен. Значение переменных во время выполнения можно увидеть наведя на них курсор. Для снятия программы с выполнения необходимо нажать Shift F5.

Для поиска алгоритмических ошибок контролируются значения промежуточных переменных на каждом шаге выполнения подозрительного кода и сравниваются с результатами, полученными вручную.

4.3. Порядок выполнения задания

Задание: Вычислим и выведем на экран значения суммы ряда $S = \sum_{i=1}^{\infty} 1/i$ и произведения ряда $P = \prod_{i=1}^{m} i$.

Панель диалога представлена на рис 4.4.

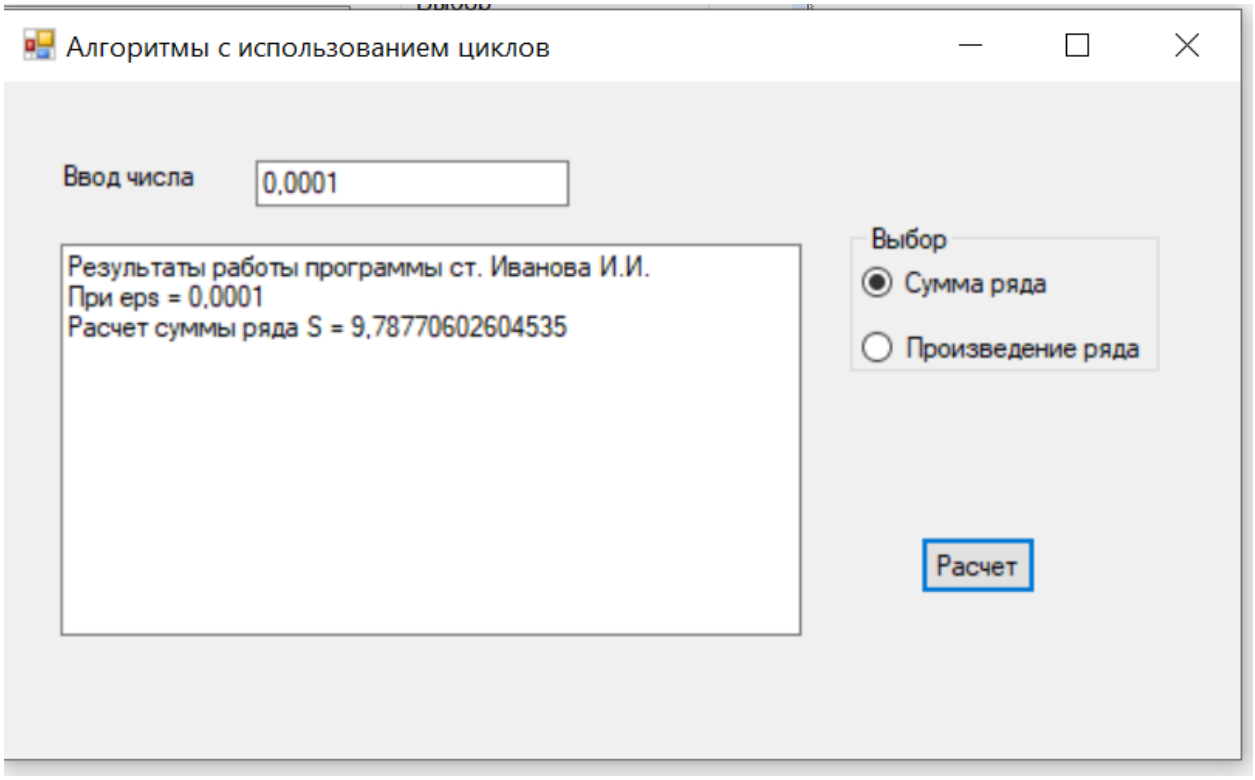

Рис. 4.4. Окно программы для расчета рядов.

Текст обработчика нажатия кнопки **Расчет** приведен ниже.

```
private void button1_Click(object sender, EventArgs e)
      {
         Double n = Convert.ToDouble(textBox1.Text);
         Double eps = Convert.ToDouble(textBox1.Text);
         textBox2.Text = "Результаты работы программы ст. Иванова И.И. " + Environment.NewLine;
         int m = 0;
         if (radioButton2.Checked) m = 1;
        double s=0,p=1,ch;
        double i=1;
         switch (m)
         {
            case 0:
               ch = 1 / i;
               while (ch>=eps)
               {
                  ch = 1 / i;
                  s += ch;
                  i++;
               }
               textBox2.Text += "При eps = " + textBox1.Text + Environment.NewLine;
               textBox2.Text += "Расчет суммы ряда S = " + Convert.ToString(s) + Environment.NewLine;
               break;
            case 1:
               for (i=1; i<= n; i++)
               {
                  ch = i;
                  p *= ch;
                              }
```

```
            textBox2.Text += "При m = " + textBox1.Text + Environment.NewLine;
            textBox2.Text += "Расчет произведения ряда P = " + Convert.ToString(p) + Environment.NewLine;
            break;
        }
    }
```

Запустите программу в отладочном режиме (F5). После попадания на точку остановки, нажимая клавишу F10, выполните пошагово программу и проследите, как меняются все переменные в процессе выполнения.

4.4. Выполнение индивидуального задания

По указанию преподавателя выберите нужный вариант задачи из нижеприведенного списка. Откорректируйте текст программы.

Индивидуальные задания

Составить программу для

1. $\sum_{n=1}^{\infty} \frac{n+1}{2n!};$
2. $\sum_{n=1}^{\infty} \frac{1}{2n+n!};$
3. $\sum_{n=5}^{\infty} \frac{\sqrt{n+2}}{1+\sqrt{(n+1)!}};$
4. $\sum_{n=1}^{\infty} \frac{5+n}{3n!-1};$
5. $\sum_{n=5}^{\infty} \frac{\sqrt{n}}{(n-3)!};$
6. $\sum_{n=0}^{\infty} \frac{2n+1}{(3n+1)!};$
7. $\sum_{n=1}^{\infty} \frac{2}{\sqrt{n!+4}};$
8. $\sum_{n=2}^{\infty} \frac{n-1}{2n!-2};$
9. $\prod_{n=1}^{m} \frac{\sqrt{n!}+8n}{3n-2};$
10. $\prod_{n=1}^{m} 0{,}5(2+0{,}1!);$
11. $\prod_{n=0}^{m} \frac{(2n+1)!}{\sqrt{n+2}};$
12. $\prod_{n=2}^{m} \frac{0{,}5n!}{0.1n+1};$
13. $\prod_{n=0}^{m} \frac{(n+2)!}{n+1};$
14. $\prod_{n=0}^{m} \sqrt{(3n+1)!};$
15. $\prod_{n=1}^{m} \frac{3n!-1}{n+1};$
16. $\prod_{n=1}^{m} \frac{\sqrt{n!+10n}}{3n-2};$