

Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение
высшего образования

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ
УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

Кафедра комплексной информационной безопасности электронно -
вычислительных систем (КИБЭВС)

ВЕЩЕСТВЕННЫЕ ЧИСЛА. ОШИБКИ ПРИ РАБОТЕ С ВЕЩЕСТВЕННЫ-
МИ ЧИСЛАМИ

Отчет по практической работе №3
по дисциплине «Языки программирования»

Выполнил

Студент гр. 728-2

_____ Д. Р. Геворгян

____.11.2019

Принял

Преподаватель кафедры КИБЭВС

_____ М. А. Кальнеус
оценка подпись

____.11.2019

1 Введение

Цель работы: целью работы является знакомство с основными ошибками, возникающими при обработке вещественных чисел.

Язык программирования C#.

2 Ход работы

2.1 Основные вычисления

Полученное число $x = 18720337282,15112000$. Найдем остальные значения:

$$y = x \cdot 10^{-10} = 1,872033728215112000;$$

$$c = x + y = 18720337284,023153728215112000;$$

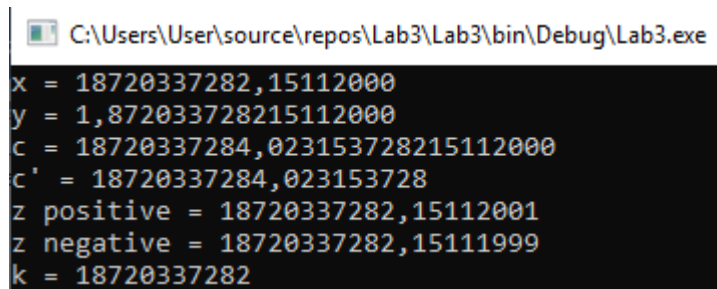
$$c' = 18720337284,023153728;$$

$$z_p = x + 10^{-8} = 18720337282,15112001;$$

$$z_n = x - 10^{-8} = 18720337282,15111999;$$

$$k = 18720337282.$$

При выполнении операций для большей точности использовался тип `decimal`.
Результаты вычислений приведены на рисунке 2.1.



```
C:\Users\User\source\repos\Lab3\Lab3\bin\Debug\Lab3.exe
x = 18720337282,15112000
y = 1,872033728215112000
c = 18720337284,023153728215112000
c' = 18720337284,023153728
z positive = 18720337282,15112001
z negative = 18720337282,15111999
k = 18720337282
```

Рисунок 2.1 - Проведение вычислений

Рассмотрим основные ошибки, возникающие при подсчетах.

2.2 Исчезновение операнда

Операнд может исчезнуть, если он относительно мал по отношению к другому операндом. В данных примерах складываются числа 100500 и 0.0003 в типе float, которые в сумме должны дать 100500.0003, а также 1488322228 и 0.0000000001, дающие в сумме 1488322228.0000000001 в типе double.

Также по формулам $A = |x_1 - x_2|$ и $O = A/x_1$, где x_1 – идеальное значение, x_2 – полученное значение, рассчитаем абсолютную и относительную ошибки:

Для типа float $A = |100500.0003 - 100500| = 0.0003$, $O = 0.0003/100500.0003 = 0,00000000030$.

Для типа double $A = |1488322228.0000000001 - 1488322228| = 0.0000000001$, $O = 0.0000000001/1488322228.0000000001 = 0,00000000000$.

Результаты работы программы представлены на рисунке 2.2. Исходя из результатов можно сделать вывод, что тип double точнее типа float.

```
Исчезновение операнда. Тип float
0,0003 + 100500 = 100500
Абсолютная ошибка = 0,0003
Относительная ошибка = 0,00000000030

Исчезновение операнда. Тип double
0.0000000001 + 1488322228 = 1488322228
Абсолютная ошибка = 0,0000000001
Относительная ошибка = 0,00000000000
```

Рисунок 2.2. - Исчезновение операнда

2.3 Умножение ошибки

Данная ошибка возникает при многократном увеличении абсолютной погрешности операнда, которая может появиться при использовании арифметики с плавающей точкой, даже если относительная ошибка мала.

Результаты работы программы представлены на рисунке 2.3.

```
Умножение ошибки.  
Результат для типа float = 499,9514  
Абсолютная ошибка = 0,0486  
Относительная ошибка = 0,0000972  
  
Результат для типа double = 500,0000000000079  
Абсолютная ошибка = 0,0000000001  
Относительная ошибка = 0,0000000000
```

Рисунок 2.3 - Умножение ошибки

2.4 Потеря значимости

Полная потеря значимости, вызванная вычитанием почти равных чисел. Также возникает, когда результат вычислений невозможно представить в допустимой форме.

Допустим, что мы имеем два числа – 46357,00005 и 46357,00004. Их разность должна быть равна 0,00001, но по факту будет другой.

Результаты работы программы представлены на рисунке 2.4. Поиск абсолютной и относительной погрешности во втором случае сработал некорректно.

```
Потеря значимости
Тип float
Абсолютная ошибка = 0,0000234375
Относительная ошибка = 0,0234375

Тип double
Абсолютная ошибка = 0,9999999999
Относительная ошибка = 0,9999999999
```

Рисунок 2.4 - Потеря значимости

2.5 Явное и неявное преобразование в C#

Для встроенных числовых типов неявное преобразование можно выполнить, если сохраняемое значение может уместиться в переменной без усечения или округления. При использовании целочисленных типов это означает, что диапазон исходного типа является надлежащим подмножеством диапазона для целевого типа. Например, переменная типа `long` (64-разрядное целое число) может хранить любое значение, которое может хранить переменная `int` (32-разрядное целое число).

Пример:

```
float a = 1234567.8f;
```

```
double b = a;
```

Тем не менее если преобразование нельзя выполнить без риска потери данных, компилятор требует выполнения явного преобразования, которое называется приведением. Приведение — это способ явно указать компилятору, что необходимо выполнить преобразование и что вам известно, что может произойти потеря данных.

Пример:

```
decimal d = 98765432.1m;
```

```
double e = Convert.ToDouble(d);
```

3 Заключение

В ходе выполнения практической работы произошло знакомство с основными ошибками, возникающими при обработке вещественных чисел.

Также были получены навыки работы с явным и неявным преобразованием типов.

Полный код программы приведен в приложении А.

Отчёт был написан согласно ОС ТУСУР 2013.

Приложение А

(обязательное)

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Lab3
{
    class Program
    {
        public static decimal absolute_error(decimal x_1, decimal x_2)
        {
            decimal A = Math.Abs((x_1) - (x_2));
            return Math.Round(A, 10);
        }

        public static decimal relative_error(decimal x_3, decimal x_1)
        {
            decimal O = x_3 / x_1;
            return Math.Round(O, 10);
        }

        public static void write(decimal x_3, decimal x_4)
        {
            Console.WriteLine($"Абсолютная ошибка = {x_3}\nОтносительная ошибка = {x_4}\n");
        }

        static void Main(string[] args)
        {
            decimal x = 18720337282.15112000m;
            decimal y = x * (decimal) Math.Pow(10, -10);
            decimal c = x + y;
            decimal c_round = (decimal) Math.Round(c, 9);
            decimal z_positive = x + (decimal) Math.Pow(10, -8);
            decimal z_negative = x - (decimal) Math.Pow(10, -8);
            decimal k = (decimal) Math.Round(x, 0);
            Console.WriteLine($"x = {x}\ny = {y}\nc = {c}\nc' = {c_round}\nz positive = {z_positive}\nz negative = {z_negative}\nk = {k}\n");
            Console.WriteLine("Исчезновение операнда. Тип float");
            float f11 = 0.0003f;
            float f12 = 100500f;
            float result_float = f11 + f12;
            Console.WriteLine($"{f11} + {f12} = " + result_float.ToString());
            decimal flabs = absolute_error(100500.0003m, (decimal)result_float);
            decimal flrel = relative_error(flabs, 100500.0003m);
            write(flabs, flrel);
            Console.WriteLine("Исчезновение операнда. Тип double");
            double d1 = 0.0000000001;
            double d2 = 1488322228;
            double result_double = d1 + d2;
            Console.WriteLine($"0.0000000001 + " + d2.ToString() + " = " + result_double.ToString());
            decimal dabs = absolute_error(1488322228.0000000001m, (decimal)result_double);
            decimal drel = relative_error(dabs, 1488322228.0000000001m);
            write(dabs, drel);
            Console.WriteLine("Умножение ошибки.");
            float afl = 0.0f;
            double dob = 0.0;
        }
    }
}

```

```

decimal cdec = 0.0m;
for (int i = 0; i < 10000; i++)
{
    afl += 0.05f;
    dob += 0.05;
    cdec += 0.05m;
}
Console.WriteLine($"Результат для типа float = {afl}");
decimal fl_mult_abs = absolute_error(cdec, (decimal) afl);
decimal fl_mult_rel = relative_error(fl_mult_abs, cdec);
write(fl_mult_abs, fl_mult_rel);
Console.WriteLine($"Результат для типа double = {dob}");
decimal dmult_abs = absolute_error(cdec, (decimal) dob);
decimal dmult_rel = relative_error(dmult_abs, cdec);
write(dmult_abs, dmult_rel);
Console.WriteLine("Потеря значимости");
Console.WriteLine("Тип float");
float floss1 = 12345.002f;
float floss2 = 12345.001f;
float floss_res = floss1 - floss2;
decimal floss_abs = absolute_error(0.001m, (decimal)floss_res);
decimal floss_rel = relative_error(floss_abs, 0.001m);
write(floss_abs, floss_rel);
Console.WriteLine("Тип double");
double dloss1 = 54321.0000000002;
double dloss2 = 54321.0000000001;
double dloss_res = dloss1 - dloss2;
decimal dloss_abs = absolute_error(0.000000001m, (decimal)dloss_res);
decimal dloss_rel = relative_error(dloss_abs, 0.000000001m);
write(dloss_abs, dloss_rel);
Console.WriteLine("Неявное преобразование");
float a = 10000000.1f;
double b = a;
Console.WriteLine(b);
Console.WriteLine("Явное преобразование");
decimal d = 10000000.1m;
double e = Convert.ToDouble(d);
Console.WriteLine(e);
Console.ReadKey();
}
}
}

```