



ОТЧЁТ ПО ЛАБРАТОРНОЙ РАБОТЕ №3

Объектно-ориентированное программирование

Подготовил Шкода Глеб Ярославович
Студент 2 курса факультета ИКТ Университета ИТМО
Группа K32211
Преподаватель Иванов Сергей Евгеньевич

Упражнение 1

Задание 1.

Реализуем описанный в тексте задания условный оператор.

```
1  using System;
2
3
4  0 references
class Program {
5      0 references
static void Main() {
6
7          Console.Write("x = ");
8          float x = float.Parse(Console.ReadLine());
9          Console.Write("y = ");
10         float y = float.Parse(Console.ReadLine());
11         if (x * x + y * y < 9 && y > 0)
12             Console.WriteLine("Внутри");
13         else if (x * x + y * y > 9 || y < 0)
14             Console.WriteLine("Вне");
15         else
16             Console.WriteLine("На границе");
17     }
18 }
19
20
```

Протестируем каждый из возможных исходов этой программы.

```
Microsoft Visual Studio Debug Console
x = 2
y = 2
Внутри
```

```
Microsoft Visual Studio Debug Console
x = 10
y = 10
Вне
```

```
Microsoft Visual Studio Debug Console
x = 0
y = 0
На границе
```

Как видно из тестирования, условный оператор работает корректно.

Задание 2.

Напишем программу по образцу из условия.

```
1  using System;
2
3  class Program {
4      static void Main() {
5
6          Console.Write("A = ");
7          double a = double.Parse(Console.ReadLine());
8          Console.Write("OP = ");
9          char op = char.Parse(Console.ReadLine());
10         Console.Write("B = ");
11         double b = double.Parse(Console.ReadLine());
12
13         bool ok = true;
14         double res = 0;
15         switch (op) {
16             case '+':
17                 res = a + b;
18                 break;
19             case '-':
20                 res = a - b;
21                 break;
22             case '*':
23                 res = a * b;
24                 break;
25             case '/':
26                 res = a / b;
27                 break;
28             default:
29                 ok = false;
30                 break;
31         }
32         if (ok)
33             Console.WriteLine("{0} {1} {2} = {3}", a, op, b, res);
34         else
35             Console.WriteLine("Операция не определена");
36     }
37 }
38
39
```

Проверим работу программы на разных входных данных, корректных и некорректных.

```
Microsoft Visual Studio Debug Console
A = 3
OP = +
B = 9
3 + 9 = 12
```

```
Microsoft Visual Studio Debug Console
A = 2
OP = }
B = 8
Операция не определена
```

```
Microsoft Visual Studio Debug Console
A = 23
OP = /
B = 0
23 / 0 = ?
```

```
Microsoft Visual Studio Debug Console
A = 0
OP = /
B = 0
0 / 0 = не число
```

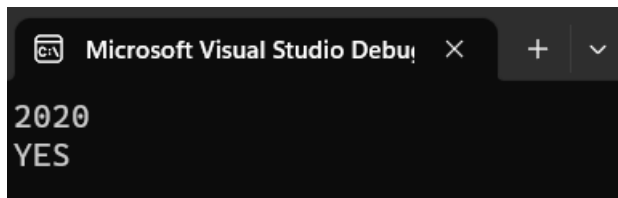
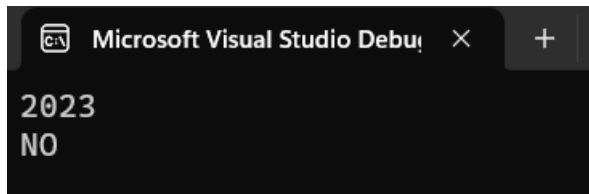
Обратим внимание, что, судя по тестированию в C#, в отличие от многих других языков программирования, существует встроенная защита от деления на 0, то есть программа не будет завершать работу с ошибкой при любой попытке совершить деление на 0, при чём результат при делении 0 на 0 и любого другого числа на 0, как можно заметить из проведённых тестов, отличаются.

Задание 3.

Используя определение високосного года из условия, была составлена программа, которая определяет является ли введённый год високосным.

```
1 using System;
2
3 class Program {
4     static void Main() {
5
6         int year = int.Parse(Console.ReadLine());
7         if (year % 400 == 0 || (year % 4 == 0 && year % 100 != 0))
8             Console.WriteLine("YES");
9         else
10            Console.WriteLine("NO");
11    }
12 }
13
```

В целях проверки было произведено тестирование.



Упражнение 2

Задание 1.

Реализуем программу из условия.

```
1  using System;
2
3  class Program {
4      static void Main() {
5
6          Console.Write("n = ");
7          int n = int.Parse(Console.ReadLine());
8
9          // while:
10         Console.Write("\nwhile: \t\t");
11         int i = 1;
12         while (i <= n) {
13             Console.Write(" " + i);
14             i += 2;
15         }
16
17         // do while:
18         Console.Write("\ndo while: \t");
19         i = 1;
20         do {
21             Console.Write(" " + i);
22             i += 2;
23         }
24         while (i <= n);
25
26         // for:
27         Console.Write("\nFor: \t\t");
28         for (i = 1; i <= n; i += 2) {
29             Console.Write(" " + i);
30         }
31
32     }
33 }
34
```

Запустим и протестируем её.

```
Microsoft Visual Studio Debug Console
n = 5
while:      1 3 5
do while:   1 3 5
For:        1 3 5
```

Программа 3 раза подряд выводит все нечётные числа от 1 до n. Таким образом можно сделать вывод, что независимо от вида использованного цикла, результат исполнения каждого из сегментов кода одинаковый.

Реализуем программу вывода значений $\sin(x)$ в определённом диапазоне

```
1  using System;
2
3
4  class Program {
5      static void Main() {
6
7          Console.Write("x1 = ");
8          double x1 = double.Parse(Console.ReadLine());
9          Console.Write("x2 = ");
10         double x2 = double.Parse(Console.ReadLine());
11         double x = x1;
12         Console.WriteLine("x      sin(x)");
13         do {
14             double y = Math.Sin(x);
15             Console.WriteLine("{0:f2} {1}", x, y);
16             x += 0.01;
17         } while (x <= x2);
18     }
19 }
20
```

Пример работы программы:

```
Microsoft Visual Studio Debug Console
x1 = 0
x2 = 0,1
x      sin(x)
0,00 0
0,01 0,009999833334166664
0,02 0,01999866669333308
0,03 0,02999550020249566
0,04 0,03998933418663416
0,05 0,04997916927067833
0,06 0,0599640064794446
0,07 0,06994284733753277
0,08 0,0799146939691727
0,09 0,08987854919801104
0,10 0,09983341664682814
```

Аналогичный алгоритм можно реализовать, используя также и цикл с предусловием.

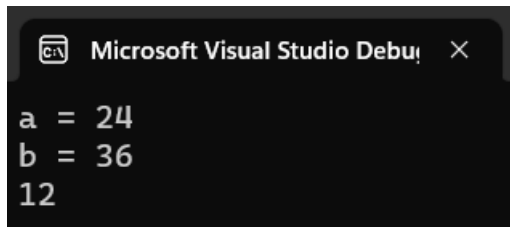
```
1 using System;
2
3
4 class Program {
5     static void Main() {
6
7         Console.Write("x1 = ");
8         double x1 = double.Parse(Console.ReadLine());
9         Console.Write("x2 = ");
10        double x2 = double.Parse(Console.ReadLine());
11        double x = x1;
12        Console.WriteLine("x    sin(x)");
13
14        while (x <= x2) {
15            double y = Math.Sin(x);
16            Console.WriteLine("{0:f2} {1}", x, y);
17            x += 0.01;
18        }
19    }
20 }
21
```

При одинаковых входных данных программы выводят одинаковый результат.

Далее, согласно условию, был реализован алгоритм Евклида с помощью цикла с предусловием.

```
1 using System;
2
3
4 class Program {
5     static void Main() {
6
7         Console.Write("a = ");
8         int a = int.Parse(Console.ReadLine());
9         Console.Write("b = ");
10        int b = int.Parse(Console.ReadLine());
11        int tmp = a;
12        while (tmp != b) {
13            a = tmp;
14            if (a < b) {
15                tmp = a;
16                a = b;
17                b = tmp;
18            }
19            tmp = a - b;
20            a = b;
21        }
22        Console.WriteLine(b);
23    }
24 }
25
```

С помощью нескольких тестов можно убедиться, что алгоритм работает верно.

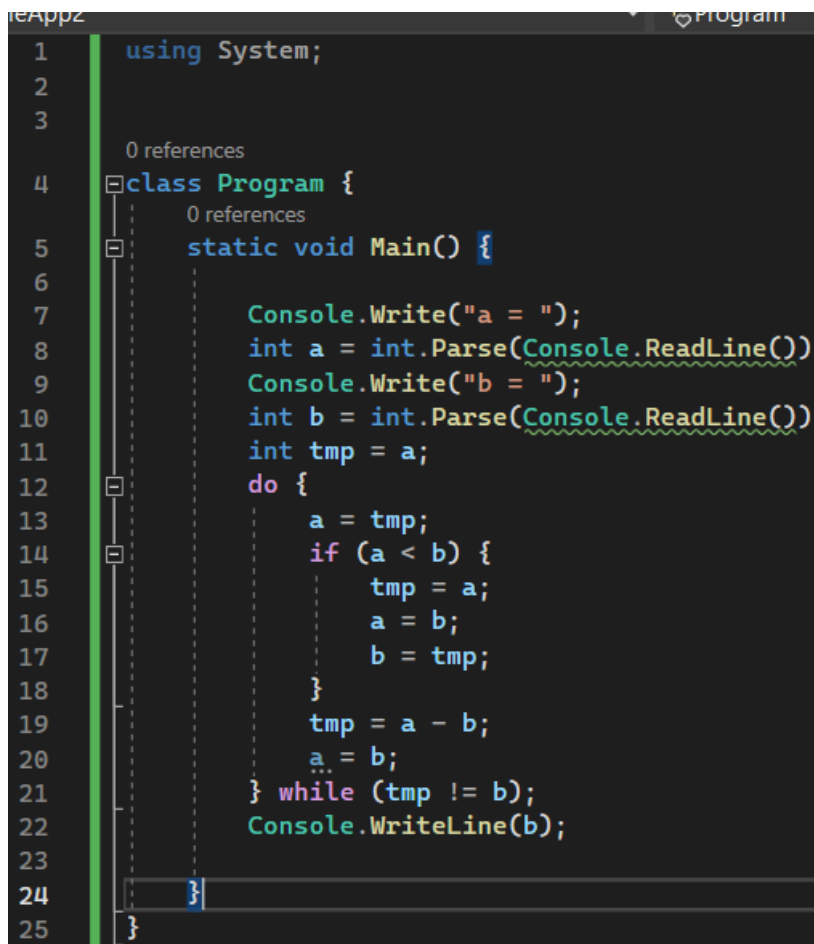


```
Microsoft Visual Studio Debug Console
a = 24
b = 36
12
```



```
Microsoft Visual Studio Debug Console
a = 13
b = 7
1
```

Данный алгоритм можно переписать, используя также и алгоритм с предусловием.



```
1 using System;
2
3
4 class Program {
5     static void Main() {
6
7         Console.Write("a = ");
8         int a = int.Parse(Console.ReadLine());
9         Console.Write("b = ");
10        int b = int.Parse(Console.ReadLine());
11        int tmp = a;
12        do {
13            a = tmp;
14            if (a < b) {
15                tmp = a;
16                a = b;
17                b = tmp;
18            }
19            tmp = a - b;
20            a = b;
21        } while (tmp != b);
22        Console.WriteLine(b);
23
24    }
25 }
```

В данном случае программа будет выводить те же ответы, что и предыдущая, при одинаковых входных данных.

Таким образом, можно сделать вывод, что циклы с постусловием и предусловием как правило, аналогичны, и переход от одного к другому осуществить несложно. Цикл с предусловием используется реже, однако его

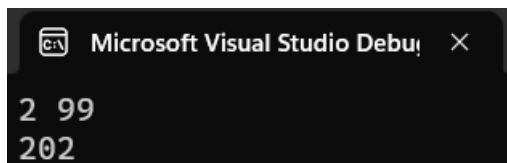
целесообразно применять, например, в случае, если первый проход по циклу необходимо совершить в любом случае, независимо от условия.

Задание 2.

В этом задании на вход даются числа k и m необходимо рассчитать сумму от 1 до k и от m до 100. Пример кода дан в условии.

```
1  using System;
2
3
4  class Program {
5      static void Main() {
6
7          var input = Console.ReadLine().Split();
8          int k = int.Parse(input[0]), m = int.Parse(input[1]);
9          int ans = 0;
10         for (int i = 1; i <= 100; i++) {
11             if (i > k && i < m)
12                 continue;
13             ans += i;
14         }
15         Console.WriteLine(ans);
16     }
17 }
18
```

Пример работы программы:



```
Microsoft Visual Studio Debug Console
2 99
202
```

Можно заметить, что в случае, если $(m - k)$ – достаточно большое число, эффективнее было бы реализовать этот алгоритм просто с помощью 2-ух циклов, потому что в текущей реализации большинство шагов цикла просто пропускаются и не вносят никакой вклад в ответ.

Задание 3.

В этом задании необходимо было на основе координат центра мишени и координат попадания начислять баллы за стрельбу. Перед началом ввода информации с помощью класса Random создаются случайные координаты центра мишени. Далее запускается бесконечный цикл, который будет читать 2 вещественных числа, до тех пор, пока их подадут на вход. В нём считываются координаты попадания, в них вносится искажение, и затем проверяется, зафиксировано ли попадание в зону, за которую начисляются

баллы, также учитывается, что существует 2 варианта мишеней, и в зависимости от номера выстрела, баллы начисляются по-разному.

```
1 using System;
2
3
4 class Program {
5     static void Main() {
6
7         Random rnd = new Random();
8         double x_center = rnd.NextDouble() * (double) rnd.Next(-3, 3);
9         double y_center = rnd.NextDouble() * (double) rnd.Next(-3, 3);
10        int cur = 1;
11        while (true) {
12            double x, y;
13            try {
14                var input = Console.ReadLine().Split();
15                x = double.Parse(input[0]);
16                y = double.Parse(input[1]);
17            }
18            catch {
19                Console.WriteLine("Завершение работы");
20                break;
21            }
22            int key = rnd.Next(0, 1);
23            if (key == 0)
24                x += rnd.NextDouble();
25            else
26                x -= rnd.NextDouble();
27            key = rnd.Next(0, 2);
28            if (key == 0)
29                y += rnd.NextDouble();
30            else
31                y -= rnd.NextDouble();
32            int ans = 0;
33            if ((x - x_center) * (x - x_center) + (y - y_center) * (y - y_center) <= 1)
34                ans = 10;
35            else if ((x - x_center) * (x - x_center) + (y - y_center) * (y - y_center) <= 4)
36                ans = 5;
37            else if ((x - x_center) * (x - x_center) + (y - y_center) * (y - y_center) <= 9 && cur % 2 == 0)
38                ans = 1;
39            Console.WriteLine(ans);
40            cur++;
41        }
42    }
43 }
```

Протестируем написанную программу:

```
Microsoft Visual Studio Debug Console
0 0
5
1 1
5
0,5 0,5
5
0,5 0
1
0,7 0,7
5
0,8 0,8
5
1,2 1,2
5
1 1,2
1
1,5 1,5
0
1,5 1,5
1
1,3 1,1
0
0 0
1
exit
Завершение работы
```

Можно заметить, что из-за того, что координаты центра неизвестны, а входные данные вносятся случайные искажения, “нащупать” центр мишени очень сложно, поэтому часто программа начисляет неполные баллы или вообще 0.