

Разбор выражений

Programming Taskbook - Электронный задачник по программированию [C#]

РЕКУРСИЯ: РАЗБОР ВЫРАЖЕНИЙ
 Задание: Recur15°
 Выполняет: Иванов Петр
 Результаты (F2) Цвет (F3) Режим (F4)

ДАННОЕ ЗАДАНИЕ В ВАШ ВАРИАНТ НЕ ВКЛЮЧЕНО.

Задание выполнено!

Ввод: 1 из 1 Вывод: 1 из 1 Тесты: 9 из 9

Выход (Esc)

Вывести значение целочисленного выражения, заданного в виде строки S.
 Выражение определяется следующим образом:
 <выражение> ::= <терм> | <выражение>+<терм> | <выражение>-<терм>
 <терм> ::= <цифра> | <терм>*<цифра>

Исходные данные

S = "4-7*7+9"

Полученные результаты

-36

```

static string s;
static int k;

static char NextChar()
{
    if (++k < s.Length)
        return s[k];
    return '\0';
}

static int NextInt()
{
    return NextChar() - '0';
}

static void BackChar()
{
    k--;
}

static int Term()
{
    int i = NextInt();
    if (NextChar() == '*')
        return i * Term();
    BackChar();
    return i;
}

```

```

static int Expr()
{
    int i;
    switch (NextChar())
    {
        case '+':
            i = Term();
            return i + Expr();
        case '-':
            i = Term();
            return -i + Expr();
    }
    return 0;
}

public static int Rec15(string e)
{
    k = -1;
    s = e;
    int i = Term();
    return i + Expr();
}

public static void Solve()
{
    Task("Recur15");
    Put(Rec15(GetString()));
}

```

Programming Taskbook - Электронный задачник по программированию [C#]

РЕКУРСИЯ: РАЗБОР ВЫРАЖЕНИЙ
 Задание: Recur16°
 Выполняет: Иванов Петр
 Результаты (F2) Цвет (F3) Режим (F4)
 ДАННОЕ ЗАДАНИЕ В ВАШ ВАРИАНТ НЕ ВКЛЮЧЕНО.

Задание выполнено! Выход (Esc)

Ввод: 1 из 1 Вывод: 1 из 1 Тесты: 9 из 9

Вывести значение целочисленного выражения, заданного в виде строки S.
 Выражение определяется следующим образом:
 <выражение> ::= <терм> | <выражение>+<терм> | <выражение>-<терм>
 <терм> ::= <элемент> | <терм>*<элемент>
 <элемент> ::= <цифра> | (<выражение>)

Исходные данные

S = "(4-4-3-5)-1*6+3"

Полученные результаты

-11

```

static string s;
static int k;

static int ToInt(char c)
{
    return c - '0';
}

static char NextChar()
{
    if (++k < s.Length)
        return s[k];
    return '\0';
}

static void BackChar()
{
    k--;
}

static int Elem()
{
    char c = NextChar();
    if (char.IsDigit(c))
        return ToInt(c);
    int i = Term();
    return i + Expr();
}

```

```

static int Term()
{
    int i = Elem();
    if (NextChar() == '*')
        return i * Term();
    BackChar();
    return i;
}

static int Expr()
{
    int i;
    switch (NextChar())
    {
        case '+':
            i = Term();
            return i + Expr();
        case '-':
            i = Term();
            return -i + Expr();
    }
    return 0;
}

static int Rec16(string e)
{
    k = -1;
    s = e;
    int i = Term();
    return i + Expr();
}

public static void Solve()
{
    Task("Recur16");
    Put(Rec16(GetString()));
}

```

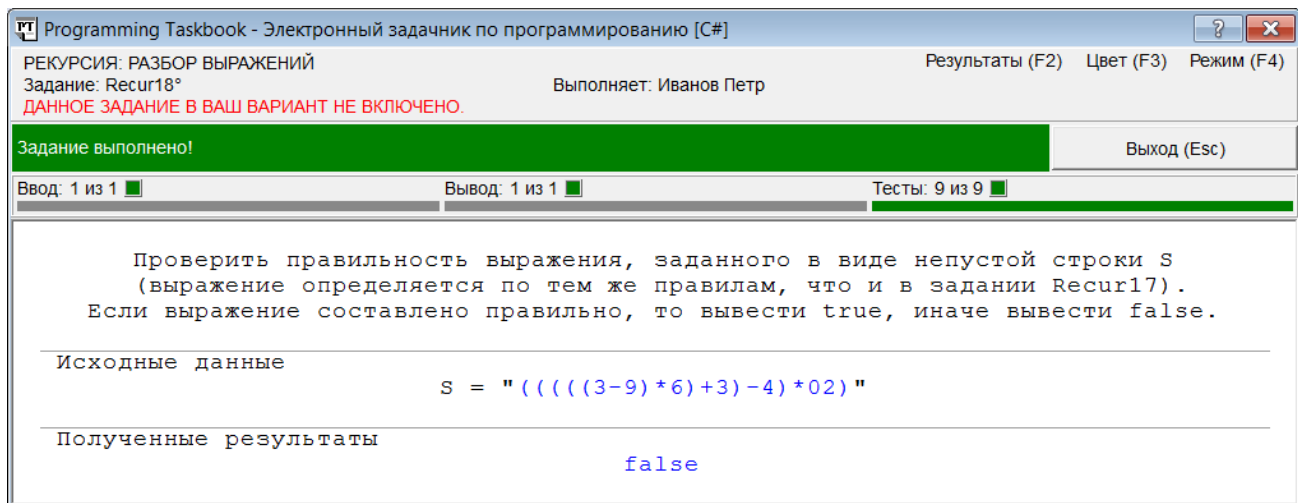
Recur18. Проверить правильность выражения, заданного в виде непустой строки S . Выражение определяется по тем же правилам, что и в задании Recur17:

$\langle \text{выражение} \rangle ::= \langle \text{цифра} \rangle \mid$
 $(\langle \text{выражение} \rangle \langle \text{знак} \rangle \langle \text{выражение} \rangle)$
 $\langle \text{знак} \rangle ::= + \mid - \mid *$

Если выражение составлено правильно, то вывести True, иначе вывести False.

Виды ошибок.

- Ошибки в ситуации, когда выражение начинается с круглой скобки:
 - первый операнд не является допустимым или отсутствует;
 - знак операции не является допустимым или отсутствует;
 - второй операнд не является допустимым или отсутствует;
 - выражение не завершается круглой скобкой.
- Ошибка в ситуации, когда выражение не содержит скобок:
 выражение не является цифрой или отсутствует.
- Строка содержит лишние символы. Например, в строке «(2+4)–9» символы «–9» являются «лишними», так как разбор выражения будет закончен после просмотра текста «(2+4)» (для добавления второй операции в выражение необходимо, в силу приведенного выше определения, заключить ее в скобки: «((2+4)–9)»).



```
static string s;
static int k;

static char NextChar()
{
    if (++k < s.Length)
        return s[k];
    return '\0';
}

static bool Test()
{
    char c = NextChar();
    if (char.IsDigit(c))
        return true;
    if (c == '(')
    {
        if (!Test())
            return false; // (1.1)
        c = NextChar();
        if (c != '+' && c != '-'
            && c != '*')
            return false; // (1.2)
    }

```

```
        if (!Test())
            return false; // (1.3)
        if (NextChar() != ')')
            return false; // (1.4)
        return true;
    }
    return false; // (2)
}

static bool Rec18(string e)
{
    k = -1;
    s = e;
    return Test() &&
        k == s.Length - 1; // (3)
}

public static void Solve()
{
    Task("Recur18");
    Put(Rec18(GetString()));
}

```


Деревья

Programming Taskbook - Электронный задачник по программированию [C#]

АНАЛИЗ БИНАРНОГО ДЕРЕВА
 Задание: Tree12°
 Выполняет: Иванов Петр
 Результаты (F2) Цвет (F3) Режим (F4)

ДАННОЕ ЗАДАНИЕ В ВАШ ВАРИАНТ НЕ ВКЛЮЧЕНО.

Задание выполнено! Выход (Esc)

Ввод: 1 из 1 Вывод: 14 из 14 Тесты: 5 из 5

Дан корень A_1 непустого дерева.
 Вывести значения всех вершин дерева в **инфиксном порядке**
 (вначале выводится содержимое левого поддерева в инфиксном порядке,
 затем выводится значение корня, затем – содержимое
 правого поддерева в инфиксном порядке).

Исходные данные

$A_1 = \text{Node}$

```

0:      96
   /    \
  33     67
 /  \   /  \
17  96 45  46
 /  \ /  \ /  \
93 21 83 68 40 62

```

Полученные результаты

```

33 17 93 96 96 21 83 45 67 68 46 96 40 62

```

```

static void Infix(Node a)
{
    if (a == null)
        return;
    Infix(a.Left);
    Put(a.Data);
    Infix(a.Right);
}

public static void Solve()
{
    Task("Tree12");
    Infix(GetNode());
}

```

Programming Taskbook - Электронный задачник по программированию [C#]

БИНАРНЫЕ ДЕРЕВЬЯ РАЗБОРА ВЫРАЖЕНИЙ
Задание: Tree74°
Выполняет: Иванов Петр
Результаты (F2) Цвет (F3) Режим (F4)
ДАННОЕ ЗАДАНИЕ В ВАШ ВАРИАНТ НЕ ВКЛЮЧЕНО.

Задание выполнено! Выход (Esc)

Ввод: 1 из 1 Вывод: 1 из 1 Тесты: 7 из 7

Дана строка S , содержащая описание непустого дерева в следующем формате:
 $\langle \text{дерево} \rangle ::= \langle \text{вершина} \rangle \mid \langle \text{вершина} \rangle (\langle \text{левое поддерево} \rangle, \langle \text{правое поддерево} \rangle) \mid$
 $\langle \text{вершина} \rangle (\langle \text{левое поддерево} \rangle) \mid \langle \text{вершина} \rangle (, \langle \text{правое поддерево} \rangle)$
 $\langle \text{вершина} \rangle ::= \langle \text{цифра} \rangle$
 Например, «4(2,6(,7(3)))» (пробелы отсутствуют, вид описания вершины зависит от того, имеет ли вершина непустое левое и/или правое поддерево). Создать дерево по описанию, приведенному в S , и вывести ссылку на его корень.

Исходные данные
 $S = "3(,0(8(1(,0),2(,7)),1(1,8(1)))"$

Полученные результаты

A1 = Node

```
static string s;
static int k;

static char NextChar()
{
    if (++k < s.Length)
        return s[k];
    return '\0';
}

static void BackChar()
{
    k--;
}

static Node CreateNode(char c)
{
    Node a = new Node(c - '0');
    if (NextChar() != '(')
    {
        BackChar();
        return a;
    }
}
```

```
c = NextChar();
if (c != ',')
{
    a.Left = CreateNode(c);
    c = NextChar();
}
if (c == ',')
{
    a.Right = CreateNode(NextChar());
    NextChar();
}
return a;
}

static Node Tree74(string s0)
{
    k = -1;
    s = s0;
    return CreateNode(NextChar());
}

public static void Solve()
{
    Task("Tree74");
    Put(Tree74(GetString()));
}
```

Programming Taskbook - Электронный задачник по программированию [C#]

БИНАРНЫЕ ДЕРЕВЬЯ РАЗБОРА ВЫРАЖЕНИЙ
 Задание: Tree75°
 Выполняет: Иванов Петр

Результаты (F2) Цвет (F3) Режим (F4)

ДАННОЕ ЗАДАНИЕ В ВАШ ВАРИАНТ НЕ ВКЛЮЧЕНО.

Задание выполнено! Выход (Esc)

Ввод: 1 из 1 Вывод: 1 из 1 Тесты: 7 из 7

Дан корень A₁ непустого дерева.
 Вывести строку с описанием исходного дерева
 в формате, приведенном в задании Tree74.

Исходные данные

A₁ = Node

```

0:      2
   /   \
  9     A1
   /   \
  2     4
 /  \  /  \
3    7 2    6
 /  \ /  \ /  \
4: 0  7 2  7 2  6 5  6
  
```

Полученные результаты

"2 (9, 4 (2 (7 (0), 7 (2)), 6 (6 (2), 6 (5))))"

```

static string TreeToStr(Node a)
{
    string s = a.Data + "";
    if (a.Left == null && a.Right == null)
        return s;
    s += "(";
    if (a.Left != null)
        s += TreeToStr(a.Left);
    if (a.Right != null)
        s += "," + TreeToStr(a.Right);
    return s + ")";
}

public static void Solve()
{
    Task("Tree75");
    Put(TreeToStr(GetNode()));
}
  
```