БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Кафедра программного обеспечения информационных технологий

Факультет ФНиДО

Специальность ПОИТ

Индивидуальная практическая работа № 1.2

«Использование стеков для построения различных форм представления выржений»

по дисциплине «Структуры и алгоритмы обработки данных»

Выполнила: Карпеко Н. Г.

Договор № 941 от 20.02.2019 г.

Минск 2020

**Цель работы:** научиться строить префиксную, постфиксную и инфиксную формы выражения, а также преобразовывать выражения из одной формы в другую.

**Задание.**

1. Используя стек, реализовать алгоритм преобразования алгебраического выражения из инфиксной формы записи в постфиксную форму представления.
2. Используя стек, реализовать алгоритм преобразования алгебраического выражения из инфиксной формы записи в префиксную форму представления.

Для обоих алгоритмов предусмотреть вхождение операций с различными приоритетами, а также наличие скобок в инфиксных выражениях.

Листинг программы преобразования алгебраического выражения из инфиксной формы записи в постфиксную форму представления:

**program** IPR1\_2\_1;

**uses**

crt;

**type**

PType = ^TStek;

TStek = **record**

Elem: Char;

next: PType;

**end**;

//---- Добавление к стеку stk элемента data -------------------------

**procedure** AddStek(**var** stk: PType; data: Char);

**var**

X: PType;

**begin**

**new**(X); //Добавляемый элемент

X^.Elem := data;

X^.next := stk; //Стек строим так, что next ссылался на то, что положили раньше

stk := X;

**end**;

//----- Забрать из стека верхний элемент ---------------------------

**function** GetStek(**var** stk: PType): Char;

**begin**

**if** stk <> nil **then** //если стек не пустой

**begin**

GEtStek := stk^.Elem;

stk := stk^.next; //Убираем элемент из стека

**end**

**else**

GetStek := #0; //если стек пустой - возвращаем символ #0

**end**;

//--- Получить стековый приоритет символа с -----------------

**function** StekPriority(c: Char): Integer;

**begin**

**case** c **of** //приоритет элемента, который уже есть в стеке

'+', '-': Result := 2;

'\*', '/': Result := 4;

'^': Result := 5;

'a'..'z', 'A'..'Z': Result := 8;

'(': Result := 0;

**else**

Result := 10; //Для неизвестного символа возвращаем 10

**end**;

**end**;

//--- Получить относительный приоритет символа с ----------------

**function** InpPriority(c: Char): Integer;

**begin**

**case** c **of** //приоритет элемента, который хотим положить в стек

'+', '-': Result := 1;

'\*', '/': Result := 3;

'^': Result := 6;

'a'..'z', 'A'..'Z': Result := 7;

'(': Result := 9;

')': Result := 0;

**else**

Result := 10; //Для неизвестного символа возвращаем 10

**end**;

**end**;

//----- Возвращает ранг символа в польской записи ---------------

**function** CharRang(c: Char): integer;

**begin**

**if** c **in** ['a'..'z', 'A'..'Z'] **then**

Result := 1 //Для операнда ранг 1

**else**

Result := -1; //Для оператора -1

**end**;

**VAR**

stk: PType;

input, output: string; // input – вводимое инфиксное выражение

Rang, i: Integer; // Rang - ранг

Ch: Char;

**BEGIN**

stk := nil; //Стек пустой

Rang := 0; //Обнуляем ранг

output := ''; //Обнуляем выводимую строку (постфиксное выражение)

Write(' Инфиксная форма: '); Readln(input); Writeln;

i := 1;

**while** i <= Length(input) **do** //пока не дойдем до последнего элемента

//Стек пуст или у верхнего элемента стека меньший приоритет

**if** (stk = nil) **or** (InpPriority(Input[i]) > StekPriority(stk^.Elem)) **then**

**begin**

**if** Input[i] <> ')' **then** // ( в стек класть не надо

AddStek(stk, input[i]); //кладем входной символ в стек

i := i + 1; //следующий символ

**end**

**else** //Стек не пустой или приоритет символа стека больше, чем входного

**begin**

Ch := GetStek(stk); //Забираем из стека верхний символ стека

**if** Ch <> '(' **then** // ( Добавлять в результат не надо

**begin**

output := output + Ch; //Дописываем к выходной строке

Rang := Rang + CharRang(Ch); //Изменяем ранг выражения

**end**;

**end**; //переходим к следующему символу входной строки

**while not** (stk = nil) **do** //дописываем то, что осталось в стеке

**begin**

Ch := GetStek(stk); //Забираем из стека верхний символ

**if** Ch <> '(' **then** // ( Добавлять в результат не надо

output := output + Ch; //Дописываем к выходной строке

**end**;

Writeln(' Постфиксная форма: ', output); Writeln;

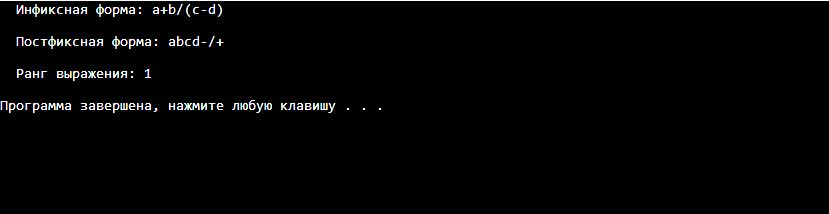
Writeln(' Ранг выражения: ', Rang);

Readln;

**END**.

Результат работы программы:





Листинг программы преобразования алгебраического выражения из инфиксной формы записи в префиксную форму представления:

**program** IPR1\_2\_2;

**uses**

crt;

**type**

PType = ^TStek;

TStek = **record**

Elem: Char;

next: PType;

**end**;

//---- Добавление к стеку stk элемента data -------------------------

**procedure** AddStek(**var** stk: PType; data: Char);

**var**

X: PType;

**begin**

**new**(X); //Добавляемый элемент

X^.Elem := data;

X^.next := stk; //Стек строим так, что next ссылался на то, что положили раньше

stk := X;

**end**;

//----- Забрать из стека верхний элемент ---------------------------

**function** GetStek(**var** stk: PType): Char;

**begin**

**if** stk <> nil **then** //если стек не пустой

**begin**

GEtStek := stk^.Elem;

stk := stk^.next; //Убираем элемент из стека

**end**

**else**

GetStek := #0; //если стек пустой - возвращаем символ #0

**end**;

//--- Получить стековый приоритет символа с -----------------

**function** StekPriority(c: Char): Integer;

**begin**

**case** c **of** //приоритет элемента, который уже есть в стеке

'+', '-': Result := 2;

'\*', '/': Result := 4;

'^': Result := 5;

'a'..'z', 'A'..'Z': Result := 8;

'(': Result := 0;

**else**

Result := 10; //Для неизвестного символа возвращаем 10

**end**;

**end**;

//--- Получить относительный приоритет символа с ----------------

**function** InpPriority(c: Char): Integer;

**begin**

**case** c **of** //приоритет элемента, который хотим положить в стек

'+', '-': Result := 1;

'\*', '/': Result := 3;

'^': Result := 6;

'a'..'z', 'A'..'Z': Result := 7;

'(': Result := 9;

')': Result := 0;

**else**

Result := 10; //Для неизвестного символа возвращаем 10

**end**;

**end**;

//----- Возвращает ранг символа в польской записи ---------------

**function** CharRang(c: Char): integer;

**begin**

**if** c **in** ['a'..'z', 'A'..'Z'] **then**

Result := 1 //Для операнда ранг 1

**else**

Result := -1; //Для оператора -1

**end**;

**VAR**

stk: PType;

input, input1, output: string; // input1 – вводимое инфиксное выражение

Rang, i: Integer; // Rang - ранг

Ch: Char;

**BEGIN**

clrscr;

stk := nil; //Стек пустой

Rang := 0; //Обнуляем ранг

output := ''; //Обнуляем выводимую строку (постфиксное выражение)

input := '';

Write(' Инфиксная форма: ');Readln(input1); writeln;

**for** i := length(input1) **downto** 1 **do** //перепишем справа налево

input := input + input1[i];

writeln(' (Промежуточная Обратная форма: ', input, ' )');

writeln; writeln(' ------------'); writeln;

i := 1;

**while** i <= Length(input1) **do** //пока не дойдем до последнего элемента

//Стек не пустой или приоритет символа стека больше, чем входного

**if** (stk = nil) **or** (InpPriority(Input[i]) < StekPriority(stk^.Elem)) **then**

**begin**

**if** Input[i] <> ')' **then** // ( в стек класть не надо

AddStek(stk, input[i]); //кладем входной символ в стек

i := i + 1; //следующий символ

**end**

**else** //Стек не пустой и приоритет входного символа меньше

**begin**

Ch := GetStek(stk); //Забираем из стека верхний символ

**if** Ch <> '(' **then** // ( Добавлять в результат не надо

**begin**

output := output + Ch; //Дописываем к выходной строке

Rang := Rang + CharRang(Ch); //Изменяем ранг выражения

**end**;

**end**; //переходим к следующему символу входной строки

**while not** (stk = nil) **do** //Дописываем всё что осталось в стеке

**begin**

Ch := GetStek(stk); //Забираем из стека верхний символ

**if** Ch <> '(' **then** // ( Добавлять в результат не надо

output := output + Ch; //Дописываем к выходной строке

**end**;

Writeln(' Префиксная форма: ', output);

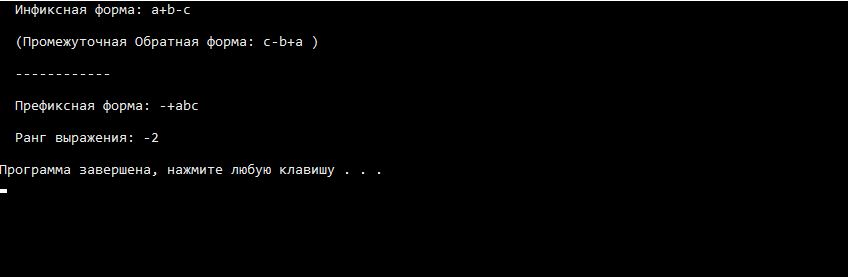
writeln;

Writeln(' Ранг выражения: ', Rang);

Readln;

**END**.

Результат работы программы:



**Контрольные вопросы:**

1. Дайте определение абстрактному типу данных «стек».

Стек – это специальный тип списка, в котором все вставки и удаления выполняются только на одном конце, называемом вершиной (*top*).

1. Перечислите основные операторы, которые определены для работы со стеком.

MAKENULL(S). Делает стек пустым.

TOP(S). Возвращает элемент из вершины стека S. Обычно вершина стека идентифицируется позицией 1.

POP(S). Удаляет элемент из вершины стека (выталкивает из стека).

PUSH(x, S). Вставляет элемент х в вершину стека S (заталкивает элемент в стек). Элемент, ранее находившийся в вершине стека, становится элементом, следующим за вершиной и т.д.

EMPTY (S). Эта функция возвращает значение true (истина), если список  S пустой и значение false (ложь) –  в противном случае.

1. Назовите основное преимущество обратной польской записи перед обычной записью выражений со скобками.

Основное преимущество обратной польской записи перед обычной записью выражений со скобками: выражение можно вычислить в процессе однократного просмотра слева направо.

1. Каким образом используется стек для преобразования выражений из одной формы записи в другую?

Алгоритм преобразования:

1. В стек помещается признак пустого стека.

2. Значение приоритета очередного входного символа сравнивается с приоритетом верхнего элемента стека.

3. Если приоритет символа больше приоритета верхнего элемента стека, то символ помещается в стек,  выбирается следующий входной символ.

4. Если приоритет входного символа меньше или равен приоритету верхнего элемента стека, то этот элемент удаляется из стека и помещается в формируемую строку, после чего сравнивается приоритеты очередного символа и нового верхнего символа.

Каждый раз при изменении обратной польской записи модифицируется ранг результирующего выражения.