**Конвертер программы с подмножества языка Си в Паскаль с использованием LL(1) метода синтаксического анализа (выражения)"**

Курсовая работа

Выполнили студенты группы ИАС-00 Мардасова У. А. Шалудько В. А.

Кемеровский Государственный Университет, Факультет Информационных Технологий

Новокузнецк, 2002г.

**Введение**

При знакомстве с языком СИ, особенно после изучения Паскаля и Бейсика, погружение в детали его изобразительных средств может затушевать важную мысль: хотя на СИ можно написать практически любую прикладную программу, он изначально для этого не предназначен. СИ является результатом эволюционного развития языков создания системных программных средств. Если в прикладном программировании эволюция шла от Фортрана к Алголу, Коболу, Паскалю и т.д., то в системном - от Ассемблеров, привязанных к архитектуре ЭВМ, к СИ, для которого созданы трансляторы, делающие его хоть и независимым от архитектуры, но не меняющим основного предназначения.

С помощью СИ можно сделать то, что на Паскале сделать невозможно (или почти невозможно) - например, написать фрагмент операционной системы (или новую операционную систему), утилиты и т.п. Так, ряд трансляторов с Паскаля написаны на СИ; обратное невозможно представить. В то же время, не раз отмечалось, что прикладные программы, написанные на Паскале, отличаются большей надежностью, чем написанные на СИ; их легче читать, передавать от одного программиста другому для совершенствования и сопровождения. Это связано с тем, что Паскаль содержит существенно больше ограничений и является языком более высокого уровня с сильной типизацией данных. Для языка же, который предназначен для разработки системного программного обеспечения, чем меньше ограничений, тем лучше; так, в СИ возможны неявные преобразования всех базовых типов данных и указателей друг в друга, что крайне желательно при создании системных средств, но при невнимательности программиста приводит к ошибкам, не улавливаемым транслятором с СИ (Паскаль же подобные недопустимые операции пресекает немедленно).

Разумеется, сказанное выше не следует абсолютизировать. Программисты, привыкшие к СИ, успешно пишут на нем программы различных классов. Это касается не только СИ - вспомните об экспертных системах, написанных на Бейсике. В то же время, при массовом программировании придерживаться "разделение труда" между языками представляется более естественным.

Широкой популярности Паскаля среди программистов способствуют следующие причины:

Благодаря своей компактности, удачному первоначальному описанию Паскаль оказался достаточно лёгким для изучения.

Язык программирования Паскаль отражает фундаментальные и наиболее важные концепции (идеи) алгоритмов в очевидной и легко воспринимаемой форме, что предоставляет программисту средства, помогающие проектировать программы.

Язык Паскаль позволяет чётко реализовать идеи структурного программирования и структурной организации данных.

Язык Паскаль сыграл большую роль в развитии методов аналитического доказательства правильности программ и позволил реально перейти от методов отладки программ к системам автоматической проверки правильности программ.

Применение языка Паскаль значительно подняло "планку" надёжности разрабатываемых программ за счёт требований Паскаля к описанию используемых в программе переменных, проверки согласованности программы при компиляции без её выполнения.

**Использование в Паскале простых и гибких структур управления: ветвлений, циклов.**

С другой стороны язык программирования СИ - это универсальный язык с богатым набором операторов и компактным способом записи выражений. Благодаря гибкости, выразительности и компактности своей конструкции СИ завоевал наибольшую популярность в среде профессиональных программистов и широко используется при разработке системных и прикладных программ.

Язык СИ представляет собой удачный компромисс между желанием располагать теми возможностями, которые обычно предоставляют программисту столь понятные и удобные языки высокого уровня, и стремлением эффективно использовать особенности компьютера. Кроме набора средств, присущих современным языкам программирования высокого уровня (структурность, модульность, определяемые типы данных) в него включены средства для программирования "почти" на уровне ассемблера (использование указателей, побитовые операции, операции сдвига). Большой набор операторов позволяет писать компактные и эффективные программы. Однако, такие мощные средства требуют от программиста осторожности, аккуратности и хорошего знания языка со всеми его преимуществами и недостатками. В отличие от языков программирования типа Паскаль, требующих жесткой дисциплины программирования ограничивающих свободу программиста, содействующих устранению многих ошибок еще на стадии их трансляции, язык СИ предоставляет программисту наибольшую свободу. Однако, ответственность за корректность программ при этом полностью ложится на программиста.

В настоящее время имеется большое количество систем программирования на СИ для разных типов компьютеров. Разработано много библиотек модулей, инструментальных средств разработки и отладки, облегчающих создание новых программ. Программы на СИ обладают высокой мобильностью, без каких либо изменений они переносятся, транслируются и выполняются на машинах различного типа.

В рамках учебного проекта реализована программная система, называемая в дальнейшем конвертором, позволяющая автоматизировать процесс преобразования программ на Паскале в программы на языке СИ. На вход конвертора подается синтаксически правильная программа на Паскале, на выходе его формируется эквивалентная ей программа на языке СИ.

Задача разработки конвертора относится к классу задач автоматизации реинженеринга программ с устаревших языков на современные языки программирования и новые платформы. Разработка конвертора для языка Паскаль является достаточно трудоёмкой в силу особенностей синтаксиса и семантики языка Паскаль.

Язык Паскаль не допускает использования принципов умолчаний и сокращений, требует строгого соблюдения соответствия типов данных, в нём жёстко регламентированы структура и место описания программных объектов.

В силу выше сказанного, даже ручной перевод программы на Паскале на язык СИ требует от программиста приложения значительных интеллектуальных усилий, а реализация конвертора для языка Паскаль возможна лишь при наличии существенных временных и материальных ресурсов. Помимо синтаксических ограничений входная программа конвертора имеет ряд чисто семантических ограничений, связанных с реализацией собственно конвертора. Перечислим основные из них:

Ограниченное использование типов данных, в полном объёме поддерживаются только арифметические типы данных.

Вложенность блоков, в которых используются одноименные объекты, не должна превышать двух уровней.

Процедурные блоки не должны быть внутри BEGIN-блоков, вложенность процедурных блоков не ограничивается.

Допускается использовать только одномерные массивы с открытой правой границей (динамические массивы) в качестве параметров процедурных блоков; при этом адресуются элементы массива в Паскаль программе, начиная с нулевого элемента.

**Ограниченное использование смешения различных типов данных.**

Выражения в Паскале

Конструкция языка, задающая порядок выполнения действия над элементами данных, называется выражением. Выражение состоит из операндов (operand - элемент данных, участвующий в операции), - величин и выражений, над которыми производится операция (константы и переменные всех типов, обращения к функциям); круглых скобок и знаков операций. Операции определяют действия, которые надо выполнить над операндами. Например, в выражении (X+Y-10) X, Y и 10 - операнды; а "+" и "-" - знаки операций сложения и вычитания.

В простейшем случае выражение может состоять из одной переменной или константы. Круглые скобки ставятся так же, как и в обычных арифметических выражениях для управления ассоциативностью и порядком выполнения операций.

Операции в языке Паскаль делятся на арифметические, отношения, логические (булевские), операцию @, строковые и др. Выражения соответственно называются арифметическими, отношения, булевскими, строковыми и т.д. в зависимости от того, какого типа операнды и операции в них используются.

Тип значения, вычисляемого с помощью выражения, определяется типом его операндов и знаками выполняемых над ними операций.

Операции могут быть унарными и бинарными. В первом случае операция относится к одному операнду и всегда записывается перед ним, во втором - операция выражает отношение между двумя операндами и записывается между ними.

Например, -А - унарная операция, Х+У - бинарная.

Арифметические выражения и операции.

Арифметическим называется выражение, составленное из операндов арифметического типа и использующее только знаки арифметических операций и круглые скобки. Порядок вычисления определяется скобками и старшинством операций.

Арифметическое выражение порождает целое или действительное (вещественное) значение. Наиболее простыми формами арифметических выражений являются:

Целая или действительная константа без знака;

Целая или действительная переменная;

Элемент массива целого или действительного типа;

Функция, принимающая целое или действительное значение.

Значение переменной или элемента массива должно быть определено до их появления в арифметическом выражении. Другие арифметические выражения составляются из вышеперечисленных простых форм путем применения круглых скобок и арифметических операций.

Арифметические операции выполняют арифметические действия в выражениях над значениями операндов целочисленных и вещественных типов. Арифметические операции языка Паскаль представлены в таблице1.

Арифметические операции

|  |  |  |  |
| --- | --- | --- | --- |
| Операция | Действие | Типы операндов | Тип результата |
| Бинарные | | | |
| +  -  \*  /  DIV  MOD  AND  SHL  SHR  OR  XOR | Сложение  Вычитание  Умножение  Деление  Целочисленное деление  Остаток от деления  Арифметическое И  Сдвиг влево  Сдвиг вправо  Арифметическое ИЛИ  Исключающая дизъюнкция | Целый  Вещественный  Целый  Вещественный  Целый  Вещественный  Целый  Вещественный  Целый  Целый  Целый  Целый  Целый  Целый  Целый | Целый  Вещественный  Целый  Вещественный  Целый  Вещественный  Вещественный  Вещественный  Целый  Целый  Целый  Целый  Целый  Целый  Целый |
| Унарные | | | |
| +  -  NOT | Сохранение знака  Отрицание знака  Арифметическое отрицание | Целый  Вещественный  Целый  Вещественный  Целый | Целый  Вещественный  Целый  Вещественный  Целый |

Выражения и операции отношения.

Выражением отношения называется словосочетание языка, в котором два выражения связаны знаком операции отношения. Выражение отношения определяет истинность или ложность результата. Операции отношения выполняют сравнение двух операндов и определяют, истинно значение выражения или ложно.

В языке Паскаль операции отношения и рассмотренные ниже булевские операции более важны при написании программ, чем в других языках, так как они интенсивно используются для реализации разветвляющихся и циклических алгоритмов. В таблице 2 приведены операции отношения, допустимые в версии языка Паскаль для ПЭВМ.

Сравниваемые величины могут принадлежать к любому скалярному или перечисляемому типу данных. Результат всегда имеет булевский тип и принимает одно из двух значений: True (истина) или False (ложь).

Операции отношения.

|  |  |  |  |
| --- | --- | --- | --- |
| Операция | Название | Выражение | Результат |
| =  <>  >  <  >=  <=  in | Равно  Не равно  Больше  Меньше  Больше или равно  Меньше или равно  Принадлежность | A=B  A<>B  A>B  A<B  A>=B  A<=B  A in M | True, если А равно В  True, если А не равно В  True, если А больше В  True, если А меньше В  True, если А больше или равно В  True, если А меньше или равно В  True, если А находится в списке М |

**Логические выражения и операции.**

Результатом выполнения логического (булевского) выражения является логическое значение True или False. Операндами служат данные только булевского типа.

Простейшими видами логических выражений являются следующие:

Логическая константа;

Логическая переменная;

Элемент массива логического типа;

Логическая функция;

Выражение отношения.

Другие логические выражения строятся из вышеперечисленных путем применения логических операций и круглых скобок. Список логических операций приведен в таблице 3.

Логические операции.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Операция | Действие | Выражение | А | В | Результат |
| Not  And  Or  xor | Логическое отрицание  Логическое И  Логическое ИЛИ  Исключающее ИЛИ | not A  A and B  A or B  A xor B | True  False  True  True  False  False  True  True  False  False  True  True  False  False | True  False  True  False  True  False  True  False  True  False  True  False | False  True  True  False  False  False  True  True  True  False  False  True  True  False |

**Операция @.**

С помощью операции @ можно создать указатель на переменную. В таблице 4 показаны операнд и типы результата.

Операция создания указателя.

|  |  |  |  |
| --- | --- | --- | --- |
| Операция | Действие | Тип операнда | Тип результата |
| @ | Получение указателя | Ссылка на переменную, процедуру или идентификатор функции | Указатель (совместимый с nil) |

Операция @ является унарной. В качестве операнда может использоваться ссылка на переменную, процедуру или идентификатор функции. После выполнения операнду возвращается соответствующий указатель, тип которого является таким же, как тип указателя nil, и, следовательно, его можно присвоить любому указателю переменной.

**Выражения в СИ.**

Конструкции, включающие константы (литералы), переменные, знаки операций, скобки для управления порядком выполнения операций, обращения к функциям, называют выражениями.

Если в выражениях встречаются операнды различных типов, то они преобразуются к общему типу в соответствии с определенными правилами:

Переменные типа char интерпретируются как целые без знака (unsigned);

Переменные типа short автоматически преобразуются в int; если один из операндов имеет тип unsigned, то другой (другие) также преобразуется к типу unsigned и результат имеет тип unsigned;

Если один из операндов имеет тип int, то другой (другие) также преобразуется к типу int и результат имеет тип int;

Если один из операндов имеет тип char, то другой (другие) также преобразуется к типу char и результат имеет тип char;

Во время операции присваивания значения правой части преобразуются к типу левой части, который и становится типом результата;

В процессе преобразования int в char лишние 8 бит просто отбрасываются.

Кроме того, существует возможность точно указывать требуемый тип данных, к которому необходимо привести некоторую величину (в скобках перед этой величиной). Скобки и имя типа вместе образуют операцию, называемую приведением типов.

Например: z=(int)x+(int)y;

Комбинация знаков операций и операндов, результатом которой является определенное значение, называется выражением. Знаки операций определяют действия, которые должны быть выполнены над операндами. Каждый операнд в выражении может быть выражением. Значение выражения зависит от расположения знаков операций и круглых скобок в выражении, а также от приоритета выполнения операций.

В языке СИ присваивание также является выражением, и значением такого выражения является величина, которая присваивается.

При вычислении выражений тип каждого операнда может быть преобразован к другому типу. Преобразования типов могут быть неявными, при выполнении операций и вызовов функций, или явными, при выполнении операций приведения типов.

Операнд - это константа, литерал, идентификатор, вызов функции, индексное выражение, выражение выбора элемента или более сложное выражение, сформированное комбинацией операндов, знаков операций и круглых скобок. Любой операнд, который имеет константное значение, называется константным выражением. Каждый операнд имеет тип.

Выражения со знаками операций могут участвовать в выражениях как операнды. Выражения со знаками операций могут быть унарными (с одним операндом), бинарными (с двумя операндами) и тернарными (с тремя операндами).

Унарное выражение состоит из операнда и предшествующего ему знаку унарной операции и имеет следующий формат:

знак-унарной-операции операнд

Бинарное выражения состоит из двух операндов, разделенных знаком бинарной операции:

операнд1 знак-бинарной-операции операнд2

Тернарное выражение состоит из трех операндов, разделенных знаками тернарной операции (?) и (:), и имеет формат:

операнд1 ? операнд2 : операнд3

По количеству операндов, участвующих в операции, операции также подразделяются на унарные, бинарные и тернарные.

В языке Си имеются следующие унарные операции:

-арифметическое отрицание (отрицание и дополнение);

~ побитовое логическое отрицание (дополнение);

! логическое отрицание;

& вычисление адреса;

+ унарный плюс;

++ увеличение (инкремент);

--уменьшение (декремент);

sizeof размер .

Унарные операции выполняются справа налево.

Операции увеличения и уменьшения увеличивают или уменьшают значение операнда на единицу и могут быть записаны как справа так и слева от операнда. Если знак операции записан перед операндом (префиксная форма), то изменение операнда происходит до его использования в выражении. Если знак операции записан после операнда (постфиксная форма), то операнд вначале используется в выражении, а затем происходит его изменение.

В отличие от унарных, бинарные операции, список которых приведен в табл.7, выполняются слева направо.

Таблица 7

|  |  |  |
| --- | --- | --- |
| Знак операции | Операция | Группа операций |
| \* | Умножение | Мультипликативные |
| / | Деление |  |
| % | Остаток от деления |  |
| + | Сложение | Аддитивные |
| - | Вычитание |  |
| << | Сдвиг влево | Операции сдвига |
| >> | Сдвиг вправо |  |
| < | Меньше | Операции отношения |
| <= | Меньше или равно |  |
| >= | Больше или равно |  |
| = = | Равно |  |
| != | Не равно |  |
| & | Поразрядное И | Поразрядные операции |
| | | Поразрядное ИЛИ |  |
| ^ | Поразрядное исключающее ИЛИ |  |
| && | Логическое И | Логические операции |
| || | Логическое ИЛИ |  |
| , | Последовательное вычисление | Последовательного вычисления |
| = | Присваивание | Операции присваивания |
| \*= | Умножение с присваиванием |  |
| /= | Деление с присваиванием |  |
| %= | Остаток от деления с присваиванием |  |
| -= | Вычитание с присваиванием |  |
| += | Сложение с присваиванием |  |
| <<= | Сдвиг влево с присваиванием |  |
| >>= | Сдвиг вправо присваиванием |  |
| &= | Поразрядное И с присваиванием |  |
| |= | Поразрядное ИЛИ с присваиванием |  |
| ^= | Поразрядное исключающее ИЛИ с присваиванием |  |

**Мультипликативные операции**

К этому классу операций относятся операции умножения (\*), деления (/) и получение остатка от деления (%). Операндами операции (%) должны быть целые числа. Отметим, что типы операндов операций умножения и деления могут отличаться, и для них справедливы правила преобразования типов. Типом результата является тип операндов после преобразования.

Операция умножения (\*) выполняет умножение операндов. Операция деления (/) выполняет деление первого операнда на второй. Если две целые величины не делятся нацело, то результат округляется в сторону нуля. При попытке деления на ноль выдается сообщение во время выполнения. Операция остаток от деления (%) дает остаток от деления первого операнда на второй.

**Аддитивные операции**

К аддитивным операциям относятся сложение (+) и вычитание (-). Операнды могут быть целого или плавающего типов. В некоторых случаях над операндами аддитивных операций выполняются общие арифметические преобразования. Однако преобразования, выполняемые при аддитивных операциях, не обеспечивают обработку ситуаций переполнения и потери значимости. Информация теряется, если результат аддитивной операции не может быть представлен типом операндов после преобразования. При этом сообщение об ошибке не выдается.

Результатом выполнения операции сложения является сумма двух операндов. Операнды могут быть целого или плавающего типа или один операнд может быть указателем, а второй - целой величиной.

Операция вычитания (-) вычитает второй операнд из первого. Возможна следующая комбинация операндов:

1. Оба операнда целого или плавающего типа.

2. Оба операнда являются указателями на один и тот же тип.

3. Первый операнд является указателем, а второй - целым.

Отметим, что операции сложения и вычитания над адресами в единицах, отличных от длины типа, могут привести к непредсказуемым результатам.

**Логические операции**

К логическим операциям относятся операция логического И (&&) и операция логического ИЛИ (||). Операнды логических операций могут быть целого типа, плавающего типа или типа указателя, при этом в каждой операции могут участвовать операнды различных типов.

Операнды логических выражений вычисляются слева направо. Если значения первого операнда достаточно, чтобы определить результат операции, то второй операнд не вычисляется.

Логические операции не вызывают стандартных арифметических преобразований. Они оценивают каждый операнд с точки зрения его эквивалентности нулю. Результатом логической операции является 0 или 1, тип результата int.

Операция логического И (&&) вырабатывает значение 1, если оба операнда имеют нулевые значения. Если один из операндов равен 0, то результат также равен 0. Если значение первого операнда равно 0, то второй операнд не вычисляется.

Операция логического ИЛИ (||) выполняет над операндами операцию включающего ИЛИ. Она вырабатывает значение 0, если оба операнда имеют значение 0, если какой-либо из операндов имеет ненулевое значение, то результат операции равен 1. Если первый операнд имеет ненулевое значение, то второй операнд не вычисляется.

**Операция последовательного вычисления**

Операция последовательного вычисления обозначается запятой (,) и используется для вычисления двух и более выражений там, где по синтаксису допустимо только одно выражение. Эта операция вычисляет два операнда слева направо. При выполнении операции последовательного вычисления, преобразование типов не производится. Операнды могут быть любых типов. Результат операции имеет значения и тип второго операнда. Отметим, что запятая может использоваться также как символ разделитель, поэтому необходимо по контексту различать, запятую, используемую в качестве разделителя или знака операции.

**Условная операция**

В языке СИ имеется одна тернарная операция - условная операция, которая имеет следующий формат:

операнд-1 ? операнд-2 : операнд-3

Операнд-1 должен быть целого или плавающего типа или быть указателем. Он оценивается с точки зрения его эквивалентности 0. Если операнд-1 не равен 0, то вычисляется операнд-2 и его значение является результатом операции. Если операнд-1 равен 0, то вычисляется операнд-3 и его значение является результатом операции. Следует отметить, что вычисляется либо операнд-2, либо операнд-3, но не оба. Тип результата зависит от типов операнда-2 и операнда-3, следующим образом.

1. Если операнд-2 или операнд-3 имеет целый или плавающий тип (отметим, что их типы могут отличаться), то выполняются обычные арифметические преобразования. Типом результата является тип операнда после преобразования.

2. Если операнд-2 и операнд-3 имеют один и тот же тип структуры, объединения или указателя, то тип результата будет тем же самым типом структуры, объединения или указателя.

3. Если оба операнда имеют тип void, то результат имеет тип void.

4. Если один операнд является указателем на объект любого типа, а другой операнд является указателем на vold, то указатель на объект преобразуется к указателю на vold, который и будет типом результата.

Если один из операндов является указателем, а другой константным выражением со значением 0, то типом результата будет тип указателя.

**Операции увеличения и уменьшения**

Операции увеличения (++) и уменьшения (--) являются унарными операциями присваивания. Они соответственно увеличивают или уменьшают значения операнда на единицу. Операнд может быть целого или плавающего типа или типа указатель и должен быть модифицируемым. Операнд целого или плавающего типа увеличиваются (уменьшаются) на единицу. Тип результата соответствует типу операнда. Операнд адресного типа увеличивается или уменьшается на размер объекта, который он адресует. В языке допускается префиксная или постфиксная формы операций увеличения (уменьшения), поэтому значения выражения, использующего операции увеличения (уменьшения) зависит от того, какая из форм указанных операций используется.

Если знак операции стоит перед операндом (префиксная форма записи), то изменение операнда происходит до его использования в выражении и результатом операции является увеличенное или уменьшенное значение операнда.

В том случае если знак операции стоит после операнда (постфиксная форма записи), то операнд вначале используется для вычисления выражения, а затем происходит изменение операнда.

**Построение управляющих таблиц**

Определим выражение в виде БНФ для языка С\С++ и Turbo Pascal 7.0.

С\С++:

В::= ++i | --i | i++ | i-- | B\*B | B/B | B%B | B+B | B-B | B<B | B>B | B>=B | B<=B | B!=B | B==B | B&&B | B|| | (k)B | B?B:B | i=B | i\*=B | i-=B | i+=B | i/=B | i%=B | i | i(S) | (B)

S::=B | B, S

Где В – выражение;

S – список выражений;

i - индификатор.

Turbo Pascal 7.0.

В::=П | П=П | П<П | П> П | П<> П | П>= П | П<= П

П::=+C | -C | П+C | П-C | П or C

C::=M | C\*M | C/M | C div M | C mod M | C and M

M::=i | i(S) | (B)

S::=B | B, S

Где В – выражение;

S – список выражений;

П – простое выражение;

С – слагаемое;

М – множитель:

i - индификатор.

N

Теперь приведём данные БНФ к КС-грамматике: G=<N, T, P, S>

С\С++ Turbo Pascal 7.0

B (k)B BП

B++i BП=П

B--i BП<П

Bi++ BП>П

BB\*B BП<=П

BB/B BП>=П

BB+B BП<>П

BB-B ПП+C

BB<B ПП-C

BB>B ПП or C

BB>=B П+C

BB<=B П-C

BB!=B CM

BB==B CC\*M

BB&&B CC/M

BB||B CC div M

BB?B:B CC mod M

Bi=B CC and M

Bi\*=B Mi

Bi/=B Mi(S)

Bi%=B M(B)

Bi+=B SB

Bi-=B SB, S

Bi

Bi(S)

B(B)

S B

SB, S

N(C\C++)={B, П, S}

T(C\C++)={i, ++, --, +, -, ==, !=, <, >, <=, >=, \*=, -=, +=, /=, %=, =, \*, /, %, (, ), ?, :, ,, &&, ||}

N(TP)={B, П, C, M, S}

T(TP)={ i, +, -, =, <>, <, >, <=, >=, \*, /, div, mod, and, or, (, ), ,}

Устранив цепные правила, левую рекурсию, получим LL(1)-грамматику.

C\C++

Bi B1

B1= BB’

B1\*=BB’

B1+=BB’

B1-BB’

B1/BB’

B1--B’

B1++B’

B1(SS1

S1)B’

B(B2

B2i B1C

C)B’

B2(B2C

B2--C1 C

C1i B’

B2++C1 C

B2k C2

C2)BB’

B--C1

B++C1

B1%BB’

B1\*BB’

B1/BB’

B1+BB’

B1-BB’

B1>BB’

B1<BB’

B1<=BB’

B1>=BB’

B1==BB’

B1!=BB’

B1&&BB’

B1||BB’

B1?BB3

B3:BB’

B1$

B’%BB’

B’\*BB’

B’/BB’

B’+BB’

B’-BB’

B’>BB’

B’<BB’

B’>=BB’

B’<=BB’

B’==BB’

B’!=BB’

B’&&BB’

B’||BB’

B’?BB3

B1%=BB’

B’$

Si B1 S’

S (B2S’

S--C1S’

S++C1S’

S’, S

S’$

ДОПУСТИТЬ

0.Отвергнуть

Turbo Pascal 7.0

B+CП’B’

B- CП’B’

Bi M’C’П’B’

B(BM1C’П’B’

B’=П

B’<П

B’>П

B’<>П

B’>=П

B’<=П

B’$

П+CП’

П-CП’

П i M’C’П’

П(BM1C’П’

П’+CП’

П’-CП’

П’ or C’П’

П’$

MI M’

M(BM1

M’(SM1

M’$

M1)

Ci M’C’

C(BM’C’

C’\*MC’

C’/MC’

C’div MC’

C’mod MC’

C’and MC’

C’$

S+CП’B’S’

S-CП’B’S’

Si M’C’П’B’S’

S(B M1C’П’B’

S’, S

S’$

ДОПУСТИТЬ

Отвергнуть

Так как данные LL(1)-грамматики являются грамматиками Грейбаха, то по ним можно построить управляющие таблицы.

Управляющая таблица для выражения на языке С\С++

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |- |  |  | 37 |  |  | 53 |  |  |  | 59 |  | 60 |
| %= |  |  | 52 |  |  |  |  |  |  |  |  |  |
| /= |  |  | 6 |  |  |  |  |  |  |  |  |  |
| -= |  |  | 5 |  |  |  |  |  |  |  |  |  |
| += |  |  | 4 |  |  |  |  |  |  |  |  |  |
| \*= |  |  | 3 |  |  |  |  |  |  |  |  |  |
| = |  |  | 2 |  |  |  |  |  |  |  |  |  |
| , |  |  | 37 |  |  | 53 |  |  |  | 58 |  |  |
| : |  |  |  |  | 36 |  |  |  |  |  |  |  |
| ? |  |  | 35 |  |  | 51 |  |  |  |  |  |  |
| && |  |  | 33 |  |  | 49 |  |  |  |  |  |  |
| || |  |  | 34 |  |  | 50 |  |  |  |  |  |  |
| == |  |  | 31 |  |  | 47 |  |  |  |  |  |  |
| != |  |  | 32 |  |  | 48 |  |  |  |  |  |  |
| <= |  |  | 29 |  |  | 46 |  |  |  |  |  |  |
| >= |  |  | 30 |  |  | 45 |  |  |  |  |  |  |
| > |  |  | 27 |  |  | 43 |  |  |  |  |  |  |
| < |  |  | 28 |  |  | 44 |  |  |  |  |  |  |
| - |  |  | 26 |  |  | 42 |  |  |  |  |  |  |
| + |  |  | 25 |  |  | 41 |  |  |  |  |  |  |
| / |  |  | 24 |  |  | 40 |  |  |  |  |  |  |
| \* |  |  | 23 |  |  | 39 |  |  |  |  |  |  |
| % |  |  | 22 |  |  | 38 |  |  |  |  |  |  |
| -- | 56 | 20 | 7 | 15 |  |  |  |  |  |  |  |  |
| ++ | 57 | 21 | 8 | 17 |  |  |  |  |  |  |  |  |
| K |  |  |  | 18 |  |  |  |  |  |  |  |  |
| I | 54 | 1 |  | 12 |  |  |  | 16 |  |  |  |  |
| ) |  |  | 37 |  |  | 53 | 13 |  | 19 | 59 | 10 |  |
| ( | 55 | 11 | 9 | 14 |  |  |  |  |  |  |  |  |
|  | S | B | B1 | B2 | B3 | B’ | C | C1 | C2 | S’ | S1 | # |

Управляющая таблица для выражения на языке Turbo Pascal 7.0

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |- |  |  | 11 |  | 32 |  | 23 |  |  | 19 | 38 | 39 |
| Or |  |  |  |  | 32 |  | 23 |  |  | 18 |  |  |
| <= |  |  | 10 |  | 32 |  | 23 |  |  | 19 |  |  |
| >= |  |  | 9 |  | 32 |  | 23 |  |  | 19 |  |  |
| > |  |  | 7 |  | 32 |  | 23 |  |  | 19 |  |  |
| < |  |  | 6 |  | 32 |  | 23 |  |  | 19 |  |  |
| <> |  |  | 8 |  | 32 |  | 23 |  |  | 19 |  |  |
| = |  |  | 5 |  | 32 |  | 23 |  |  | 19 |  |  |
| And |  |  |  |  | 31 |  | 23 |  |  |  |  |  |
| Mod |  |  |  |  | 30 |  | 23 |  |  |  |  |  |
| Div |  |  |  |  | 29 |  | 23 |  |  |  |  |  |
| / |  |  |  |  | 28 |  | 23 |  |  |  |  |  |
| \* |  |  |  |  | 27 |  | 23 |  |  |  |  |  |
| - | 34 | 2 |  |  | 32 |  | 23 |  | 13 | 17 |  |  |
| + | 33 | 1 |  |  | 32 |  | 23 |  | 12 | 16 |  |  |
| , |  |  | 11 |  | 32 |  | 23 |  |  | 19 | 37 |  |
| I | 35 | 3 |  | 25 |  | 20 |  |  | 14 |  |  |  |
| ) |  |  | 11 |  | 32 |  | 23 | 24 |  | 19 | 38 |  |
| ( | 36 | 4 |  | 26 |  | 21 | 22 |  | 15 |  |  |  |
|  | S | B | B’ | C | C’ | M | M’ | M1 | П | П’ | S’ | # |

Пример работы программы

Ввели выражение на языке СИ:

КУРСОВАЯ РАБОТА ПО ЯПМТ

Выберете язык: 1 - СИ\СИ++ 2 - Turbo Pascal 7.0

1

Напишите выражение на языке С:

(e+e==7)

Дескрипторный текст:

0 2 9 2 16 2 1

Заменить(S',В2) Сдвиг

Заменить(C,В1) Сдвиг

Заменить(В',В) Сдвиг

Заменить(В1) Сдвиг

Заменить(В',В) Сдвиг

Заменить(В1) Сдвиг

Вытолкнуть Держать

Вытолкнуть Держать

Вытолкнуть Держать

Заменить(В') Сдвиг

Вытолкнуть Держать

Вытолкнуть Держать

ДОПУСТИТЬ

Выходная лента:

55 12 25 1 31 1 37 53 53 13 53 59 60

Ввели не выражение:

КУРСОВАЯ РАБОТА ПО ЯПМТ

Выберете язык: 1 - СИ\СИ++ 2 - Turbo Pascal 7.0

1

Напишите выражение на языке С:

t+

Дескрипторный текст:

2 9

Заменить(S',В1) Сдвиг

Заменить(В',В) Сдвиг

Отвергнуть

Выходная лента:

54 25 0

Ввели выражение на Паскале:

КУРСОВАЯ РАБОТА ПО ЯПМТ

Выберете язык: 1 - СИ\СИ++ 2 - Turbo Pascal 7.0

2

Напишите выражение на языке TP:

ww=s+1

Дескрипторный текст:

2 11 2 4 2

Заменить(В'П'C'M') Сдвиг

Вытолкнуть Держать

Вытолкнуть Держать

Вытолкнуть Держать

Заменить(П) Сдвиг

Заменить(П'C'M') Сдвиг

Вытолкнуть Держать

Вытолкнуть Держать

Заменить(П'C) Сдвиг

Заменить(C'M') Сдвиг

Вытолкнуть Держать

Вытолкнуть Держать

Вытолкнуть Держать

ДОПУСТИТЬ

Выходная лента:

3 23 32 19 5 14 23 32 16 25 23 32 19 39

Ввели не выражение

КУРСОВАЯ РАБОТА ПО ЯПМТ

Выберете язык: 1 - СИ\СИ++ 2 - Turbo Pascal 7.0

2

Напишите выражение на языке TP:

f=s+

Дескрипторный текст:

2 11 2 4

Заменить(В'П'C'M') Сдвиг

Вытолкнуть Держать

Вытолкнуть Держать

Вытолкнуть Держать

Заменить(П) Сдвиг

Заменить(П'C'M') Сдвиг

Вытолкнуть Держать

Вытолкнуть Держать

Заменить(П'C) Сдвиг

Отвергнуть

Выходная лента:

3 23 32 19 5 14 23 32 16 0

Листинг программы:

#include<stdio.h>

#include<conio.h>

#define n 100

#define t 17

#define m 53

#define g 11

char s1[n],s[n],a[n];

int j=0,h,vl[n],y=0;

/\*---------------------------ВВОД ВЫРАЖЕНИЯ------------------------------\*/

int vvod()

{

int i;

printf("\t\t\tКУРСОВАЯ РАБОТА ПО ЯПМТ\n");

printf("Выберете язык: 1 - СИ\\СИ++ 2 - Turbo Pascal 7.0\n");

scanf("%d",&y);

if(y==1)

printf("Напишите выражение на языке С:\n");

{for(i=0;i<=n-1;i++)

{

scanf("%c",&s1[i]);

if (s1[i]=='\n'){j=i;break;}

}

}

if(y==2)

printf("Напишите выражение на языке TP:\n");

{for(i=0;i<=n-1;i++)

{

scanf("%c",&s1[i]);

if (s1[i]=='\n'){j=i;break;}

}

}

return y;

};

/\*-----------------------------------------------------------------------\*/

/\*--------------------------ИНДИФИКАТОР----------------------------------\*/

int perem(int be,int l)

{

int i,j,k,d(0),i1,d1=0,z(0),chiclo(0),di(0);

char b[m]={'a','b','c','d','e','f','g','h','i','j','k','l','m','n','o',

'p','q','r','s','t','u','v','w','x','y','z','A','B','C','D',

'F','G','H','I','J','K','L','M','N','O','P','Q','R','S','T',

'U','V','W','X','W','Z','E','\_'};

char c[g]={'1','2','3','4','5','6','7','8','9','0','.'};

for(j=0;j<=m-1;j++)

{

if(s1[be]==b[j]) {d=1;break;}

}

if(d==1)

{

for(i=be+1;i<=l;i++)

{

d=0;

for(j=0;j<=m-1;j++)

{

if(s1[i]==b[j]) {d=1;break;}

}

if (d==0)

{

d1=0;

for(k=0;k<=g-1;k++)

{

if(s1[i]==c[k]) {d1=1;break;}

}

if(d1==0)break;

}

}

}

chiclo=0;

for(i=be;i<=l;i++)

for(j=0;j<=g-1;j++)

{

if(s1[i]==c[j]) {chiclo=chiclo++;break;}

}

if(d1==1||d==1||chiclo==l-be+1&&z==0) {z=1;/\*printf("DA");\*/} else {z=0;/\*printf("NET");\*/}

return z;

};

/\*--------------------КОНЕЦ ИНДИФИКАТОРА---------------------------------\*/

/\*--------------------------LL(1) - АНАЛИЗАТОР-----------------------------\*/

int analiz()

{

int z[6],v,z1,i(0),j,k;

int tab[12][29]={{55,0,54,0,57,56,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

{11,0,1,0,21,20,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

{9,37,0,0,8,7,22,23,24,25,26,28,27,30,29,32,31,34,33,35,0,0,2,3,4,5,6,52,37},

{14,0,12,18,17,15,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,36,0,0,0,0,0,0,0,0},

{0,53,0,0,0,0,38,39,40,41,42,44,43,45,46,48,47,50,49,51,0,53,0,0,0,0,0,0,53},

{0,13,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

{0,0,16,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

{0,19,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

{0,59,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,58,0,0,0,0,0,0,59},

{0,10,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,60}};

printf("\n");

z1=0;

z[z1]=11;z1++;z[z1]=0;

v=0;j=0;

do

{

switch(tab[z[z1]][a[v]])

{

case 1: z[z1]=2;v++;

printf("\tЗаменить(В1)\tСдвиг\n");vl[j]=1;break;

case 2: z[z1]=5;z1++;z[z1]=1;v++;

printf("\tЗаменить(В',В)\tСдвиг\n");vl[j]=2;break;

case 3: z[z1]=5;z1++;z[z1]=1;v++;

printf("\tЗаменить(В',В)\tСдвиг\n");vl[j]=3;break;

case 4: z[z1]=5;z1++;z[z1]=1;v++;

printf("\tЗаменить(В',В)\tСдвиг\n");vl[j]=4;break;

case 5: z[z1]=5;z1++;z[z1]=1;v++;

printf("\tЗаменить(В',В)\tСдвиг\n");vl[j]=5;break;

case 6: z[z1]=5;z1++;z[z1]=1;v++;

printf("\tЗаменить(В',В)\tСдвиг\n");vl[j]=8;break;

case 7: z[z1]=5;v++;

printf("\tЗаменить(В')\tСдвиг\n");vl[j]=7;break;

case 8: z[z1]=5;v++;

printf("\tЗаменить(В')\tСдвиг\n");vl[j]=8;break;

case 9: z[z1]=10;z1++;z[z1]=0;v++;

printf("\tЗаменить(S1,S)\tСдвиг\n");vl[j]=9;break;

case 10: z[z1]=5;v++;

printf("\tЗаменить(В')\tСдвиг\n");vl[j]=10;break;

case 11: z[z1]=3;v++;

printf("\tЗаменить(В2)\tСдвиг\n");vl[j]=11;break;

case 12: z[z1]=6;z1++;z[z1]=2;v++;

printf("\tЗаменить(C,В1)\tСдвиг\n");vl[j]=12;break;

case 13: z[z1]=5;v++;

printf("\tЗаменить(В')\tСдвиг\n");vl[j]=13;break;

case 14: z[z1]=6;z1++;z[z1]=3;v++;

printf("\tЗаменить(C,В2)\tСдвиг\n");vl[j]=14;break;

case 15: z[z1]=6;z1++;z[z1]=7;v++;

printf("\tЗаменить(C,C1)\tСдвиг\n");vl[j]=15;break;

case 16: z[z1]=5;v++;

printf("\tЗаменить(В')\tСдвиг\n");vl[j]=16;break;

case 17: z[z1]=6;z1++;z[z1]=7;v++;

printf("\tЗаменить(C,C1)\tСдвиг\n");vl[j]=17;break;

case 18: z[z1]=8;v++;

printf("\tЗаменить(C2)\tСдвиг\n");vl[j]=18;break;

case 19: z[z1]=5;z1++;z[z1]=1;v++;

printf("\tЗаменить(В',В)\tСдвиг\n");vl[j]=19;break;

case 20: z[z1]=7;v++;

printf("\tЗаменить(C1)\tСдвиг\n");vl[j]=20;break;

case 21: z[z1]=7;v++;

printf("\tЗаменить(C1)\tСдвиг\n");vl[j]=21;break;

case 22: z[z1]=5;z1++;z[z1]=1;v++;

printf("\tЗаменить(В',В)\tСдвиг\n");vl[j]=22;break;

case 23: z[z1]=5;z1++;z[z1]=1;v++;

printf("\tЗаменить(В',В)\tСдвиг\n");vl[j]=23;break;

case 24: z[z1]=5;z1++;z[z1]=1;v++;

printf("\tЗаменить(В',В)\tСдвиг\n");vl[j]=24;break;

case 25: z[z1]=5;z1++;z[z1]=1;v++;

printf("\tЗаменить(В',В)\tСдвиг\n");vl[j]=25;break;

case 26: z[z1]=5;z1++;z[z1]=1;v++;

printf("\tЗаменить(В',В)\tСдвиг\n");vl[j]=26;break;

case 27: z[z1]=5;z1++;z[z1]=1;v++;

printf("\tЗаменить(В',В)\tСдвиг\n");vl[j]=27;break;

case 28: z[z1]=5;z1++;z[z1]=1;v++;

printf("\tЗаменить(В',В)\tСдвиг\n");vl[j]=28;break;

case 29: z[z1]=5;z1++;z[z1]=1;v++;

printf("\tЗаменить(В',В)\tСдвиг\n");vl[j]=29;break;

case 30: z[z1]=5;z1++;z[z1]=1;v++;

printf("\tЗаменить(В',В)\tСдвиг\n");vl[j]=30;break;

case 31: z[z1]=5;z1++;z[z1]=1;v++;

printf("\tЗаменить(В',В)\tСдвиг\n");vl[j]=31;break;

case 32: z[z1]=5;z1++;z[z1]=1;v++;

printf("\tЗаменить(В',В)\tСдвиг\n");vl[j]=32;break;

case 33: z[z1]=5;z1++;z[z1]=1;v++;

printf("\tЗаменить(В',В)\tСдвиг\n");vl[j]=33;break;

case 34: z[z1]=5;z1++;z[z1]=1;v++;

printf("\tЗаменить(В',В)\tСдвиг\n");vl[j]=34;break;

case 35: z[z1]=4;z1++;z[z1]=1;v++;

printf("\tЗаменить(В3,В)\tСдвиг\n");vl[j]=35;break;

case 36: z[z1]=5;z1++;z[z1]=1;v++;

printf("\tЗаменить(В',В)\tСдвиг\n");vl[j]=36;break;

case 37: z1--;

printf("\tВытолкнуть\tДержать\n");vl[j]=37;break;

case 38: z[z1]=5;z1++;z[z1]=1;v++;

printf("\tЗаменить(В',В)\tСдвиг\n");vl[j]=38;break;

case 39: z[z1]=5;z1++;z[z1]=1;v++;

printf("\tЗаменить(В',В)\tСдвиг\n");vl[j]=39;break;

case 40: z[z1]=5;z1++;z[z1]=1;v++;

printf("\tЗаменить(В',В)\tСдвиг\n");vl[j]=40;break;

case 41: z[z1]=5;z1++;z[z1]=1;v++;

printf("\tЗаменить(В',В)\tСдвиг\n");vl[j]=41;break;

case 42: z[z1]=5;z1++;z[z1]=1;v++;

printf("\tЗаменить(В',В)\tСдвиг\n");vl[j]=42;break;

case 43: z[z1]=5;z1++;z[z1]=1;v++;

printf("\tЗаменить(В',В)\tСдвиг\n");vl[j]=43;break;

case 44: z[z1]=5;z1++;z[z1]=1;v++;

printf("\tЗаменить(В',В)\tСдвиг\n");vl[j]=44;break;

case 45: z[z1]=5;z1++;z[z1]=1;v++;

printf("\tЗаменить(В',В)\tСдвиг\n");vl[j]=45;break;

case 46: z[z1]=5;z1++;z[z1]=1;v++;

printf("\tЗаменить(В',В)\tСдвиг\n");vl[j]=46;break;

case 47: z[z1]=5;z1++;z[z1]=1;v++;

printf("\tЗаменить(В',В)\tСдвиг\n");vl[j]=47;break;

case 48: z[z1]=5;z1++;z[z1]=1;v++;

printf("\tЗаменить(В',В)\tСдвиг\n");vl[j]=48;break;

case 49: z[z1]=5;z1++;z[z1]=1;v++;

printf("\tЗаменить(В',В)\tСдвиг\n");vl[j]=49;break;

case 50: z[z1]=5;z1++;z[z1]=1;v++;

printf("\tЗаменить(В',В)\tСдвиг\n");vl[j]=50;break;

case 51: z[z1]=4;z1++;z[z1]=1;v++;

printf("\tЗаменить(В3,В)\tСдвиг\n");vl[j]=51;break;

case 52: z[z1]=5;z1++;z[z1]=1;v++;

printf("\tЗаменить(В',В)\tСдвиг\n");vl[j]=52;break;

case 53: z1--;

printf("\tВытолкнуть\tДержать\n");vl[j]=53;break;

case 54: z[z1]=9;z1++;z[z1]=2;v++;

printf("\tЗаменить(S',В1)\tСдвиг\n");vl[j]=54;break;

case 55: z[z1]=9;z1++;z[z1]=3;v++;

printf("\tЗаменить(S',В2)\tСдвиг\n");vl[j]=55;break;

case 56: z[z1]=9;z1++;z[z1]=7;v++;

printf("\tЗаменить(S',C1)\tСдвиг\n");vl[j]=56;break;

case 57: z[z1]=9;z1++;z[z1]=7;v++;

printf("\tЗаменить(S',C1)\tСдвиг\n");vl[j]=57;break;

case 58: z[z1]=0;v++;

printf("\tЗаменить(S)\tСдвиг\n");vl[j]=58;break;

case 59: z1--;

printf("\tВытолкнуть\tДержать\n");vl[j]=59;break;

case 60: printf("ДОПУСТИТЬ\n");i=1;vl[j]=60;break;

case 0: printf("Отвергнуть\n");i=2;vl[j]=0;break;

}

if(i==1|i==2)break;else j++;

}

while(tab[z[z1]][a[v]]!=0||tab[z[z1]][a[v]]!=60);

printf("Выходная лента:\n");

for(k=0;k<=j;k++)

{

printf("%d ",vl[k]);

}

return i;

};

/\*-------------------------------------------------------------------------\*/

/\*--------------------------ТЕРМИНАЛЬНЫЕ СИМВОЛЫ---------------------------\*/

int termin(char s)

{

char term[t]={'+','-','&','|','>','<','=','\n','!','/','\*',':','?','%','(',')',','};

int k,di=0;

for(k=0;k<=t-1;k++)

{

if(s==term[k]||s=='d'||s=='i'||s=='v'||s=='m'||s=='o'||s=='r'||s=='a'||s=='n')

{di=1;break;}

}

return di;

};

/\*---------------------КОНЕЦ ТЕРМИНАЛЬНЫЕ СИМВОЛЫ--------------------------\*/

/\*-----------------------ДЕСКРИПТОРНЫЙ ТЕКСТ---------------------------------\*/

int lexica()

{ int di(0),q(0),w(0),i1,i;

i=0;

printf("Дескрипторный текст:\n");

do

{

di=termin(s1[i]);

if(di==1)

{

switch(s1[i])

{

case '(': a[w]=0;break;

case ')': a[w]=1;break;

case '%': if(s1[i+1]=='='){a[w]=27;i++;}else a[w]=6;break;

case '\*': if(s1[i+1]=='='){a[w]=23;i++;}else a[w]=7;break;

case '/': if(s1[i+1]=='='){a[w]=26;i++;}else a[w]=8;break;

case '=': if(s1[i+1]=='='){a[w]=16;i++;}else a[w]=22;break;

case '!': if(s1[i+1]=='=')a[w]=15;i++;break;

case '>': if(s1[i+1]=='='){a[w]=13;i++;}else a[w]=12;break;

case '<': if(s1[i+1]=='='){a[w]=14;i++;}else a[w]=11;break;

case '+': if(s1[i+1]=='+'){a[w]=4;i++;}else if(s1[i+1]=='='){a[w]=24;i++;}else a[w]=9;break;

case '-': if(s1[i+1]=='-'){a[w]=5;i++;}else if(s1[i+1]=='='){a[w]=25;i++;}else a[w]=10;break;

case '&': if(s1[i+1]=='&')a[w]=18;i++;break;

case '|': if(s1[i+1]=='|')a[w]=17;i++;break;

case ',': a[w]=21;break;

case '?': a[w]=19;break;

case ':': a[w]=20;break;

}

i++;

}

else

{

i1=i;

while(di!=1)

{

i++;

di=termin(s1[i]);

}

q=perem(i1,i-1);

if(q==1)a[w]=2;else {printf("ERROR.\nЛЕКСИЧЕСКАЯ ОШИБКА");break;}

}

printf("%d ",a[w]);

w++;

}

while(s1[i]!='\n');

a[w]=28;

};

/\*---------------------КОНЕЦ ДЕСКРИПТОРНОГО ТЕКСТА--------------------------\*/

int lexica1()

{ int di(0),q(0),w(0),i1,i;

i=0;

printf("Дескрипторный текст:\n");

do

{

di=termin(s1[i]);

if(di==1)

{

switch(s1[i])

{

case '(': a[w]=0;break;

case ')': a[w]=1;break;

case '\*': a[w]=6;break;

case '/': a[w]=7;break;

case '=': a[w]=11;break;

case '>': if(s1[i+1]=='='){a[w]=15;i++;}else a[w]=14;break;

case '<': if(s1[i+1]=='='){a[w]=16;i++;}else a[w]=13;break;

case '+': a[w]=4;break;

case '-': a[w]=5;break;

case ',': a[w]=3;break;

case 'd': if(s1[i+1]=='i'&&s1[i+2]=='v')a[w]=8;i++;i++;break;

case 'm': if(s1[i+1]=='o'&&s1[i+2]=='d')a[w]=9;i++;i++;break;

case 'a': if(s1[i+1]=='n'&&s1[i+2]=='d')a[w]=10;i++;i++;break;

case 'o': if(s1[i+1]=='r')a[w]=17;i++;break;

}

i++;

}

else

{

i1=i;

while(di!=1)

{

i++;

di=termin(s1[i]);

}

q=perem(i1,i-1);

if(q==1)a[w]=2;else

{printf("ERROR.\nЛЕКСИЧЕСКАЯ ОШИБКА");break;}

}

printf("%d ",a[w]);

w++;

}

while(s1[i]!='\n');

a[w]=18;

};

int analiz1()

{

int z[10],v,z1,i(0),j,k;

int tab[12][19]={{36,0,35,0,33,34,0,0,0,0,0,0,0,0,0,0,0,0,0},

{4,0,3,0,1,2,0,0,0,0,0,0,0,0,0,0,0,0,0},

{0,11,0,11,0,0,0,0,0,0,0,5,8,6,7,9,10,0,11},

{26,0,25,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

{0,32,0,32,32,32,27,28,29,30,31,32,32,32,32,32,32,32,32},

{21,0,20,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

{22,23,0,23,23,23,23,23,23,23,23,23,23,23,23,23,23,23,23},

{0,24,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

{15,0,14,0,12,13,0,0,0,0,0,0,0,0,0,0,0,0,0},

{0,19,0,19,16,17,0,0,0,0,0,19,19,19,19,19,19,18,19},

{0,38,0,37,0,0,0,0,0,0,0,0,0,0,0,0,0,0,38},

{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,39}};

printf("\n");

z1=0;

z[z1]=11;z1++;z[z1]=1;

v=0;j=0;

do

{

switch(tab[z[z1]][a[v]])

{

case 1: z[z1]=2;z1++;z[z1]=9;z1++;z[z1]=3;v++;

printf("\tЗаменить(В'П'C)\tСдвиг\n");vl[j]=1;break;

case 2: z[z1]=2;z1++;z[z1]=9;z1++;z[z1]=3;v++;

printf("\tЗаменить(В'П'C)\tСдвиг\n");vl[j]=2;break;

case 3: z[z1]=2;z1++;z[z1]=9;z1++;z[z1]=4;z1++;z[z1]=6;v++;

printf("\tЗаменить(В'П'C'M')\tСдвиг\n");vl[j]=3;break;

case 4: z[z1]=2;z1++;z[z1]=9;z1++;z[z1]=4;z1++;z[z1]=7;z1++;z[z1]=1;v++;

printf("\tЗаменить(В'П'C'M1,B)\tСдвиг\n");vl[j]=4;break;

case 5: z[z1]=8;v++;

printf("\tЗаменить(П)\tСдвиг\n");vl[j]=5;break;

case 6: z[z1]=8;v++;

printf("\tЗаменить(П)\tСдвиг\n");vl[j]=6;break;

case 7: z[z1]=8;v++;

printf("\tЗаменить(П)\tСдвиг\n");vl[j]=7;break;

case 8: z[z1]=8;v++;

printf("\tЗаменить(П)\tСдвиг\n");vl[j]=8;break;

case 9: z[z1]=8;v++;

printf("\tЗаменить(П)\tСдвиг\n");vl[j]=9;break;

case 10:z[z1]=8;v++;

printf("\tЗаменить(П)\tСдвиг\n");vl[j]=10;break;

case 11:z1--;

printf("\tВытолкнуть\tДержать\n");vl[j]=11;break;

case 12:z[z1]=9;z1++;z[z1]=3;v++;

printf("\tЗаменить(П'C)\tСдвиг\n");vl[j]=12;break;

case 13:z[z1]=9;z1++;z[z1]=3;v++;

printf("\tЗаменить(П'C)\tСдвиг\n");vl[j]=13;break;

case 14:z[z1]=9;z1++;z[z1]=4;z1++;z[z1]=6;v++;

printf("\tЗаменить(П'C'M')\tСдвиг\n");vl[j]=14;break;

case 15:z[z1]=9;z1++;z[z1]=4;z1++;z[z1]=7;v++;

printf("\tЗаменить(П'C'M1,B)\tСдвиг\n");vl[j]=15;break;

case 16:z[z1]=9;z1++;z[z1]=3;v++;

printf("\tЗаменить(П'C)\tСдвиг\n");vl[j]=16;break;

case 17:z[z1]=9;z1++;z[z1]=3;v++;

printf("\tЗаменить(П'C)\tСдвиг\n");vl[j]=17;break;

case 18:z[z1]=9;z1++;z[z1]=3;v++;

printf("\tЗаменить(П'C)\tСдвиг\n");vl[j]=18;break;

case 19:z1--;

printf("\tВытолкнуть\tДержать\n");vl[j]=19;break;

case 20:z[z1]=6;v++;

printf("\tЗаменить(M')\tСдвиг\n");vl[j]=20;break;

case 21:z[z1]=7;z1++;z[z1]=1;v++;

printf("\tЗаменить(M1,B)\tСдвиг\n");vl[j]=21;break;

case 22:z[z1]=7;z1++;z[z1]=0;v++;

printf("\tЗаменить(M1,S)\tСдвиг\n");vl[j]=22;break;

case 23:z1--;

printf("\tВытолкнуть\tДержать\n");vl[j]=23;break;

case 24:z1--;v++;

printf("\tВытолкнуть\tСдвиг\n");vl[j]=24;break;

case 25:z[z1]=4;z1++;z[z1]=6;v++;

printf("\tЗаменить(C'M')\tСдвиг\n");vl[j]=25;break;

case 26:z[z1]=4;z1++;z[z1]=6;z1++;z[z1]=1;v++;

printf("\tЗаменить(C'M'B)\tСдвиг\n");vl[j]=26;break;

case 27:z[z1]=4;z1++;z[z1]=6;v++;

printf("\tЗаменить(C'M')\tСдвиг\n");vl[j]=27;break;

case 28:z[z1]=4;z1++;z[z1]=6;v++;

printf("\tЗаменить(C'M')\tСдвиг\n");vl[j]=28;break;

case 29:z[z1]=4;z1++;z[z1]=5;v++;

printf("\tЗаменить(C'M)\tСдвиг\n");vl[j]=29;break;

case 30:z[z1]=4;z1++;z[z1]=5;v++;

printf("\tЗаменить(C'M)\tСдвиг\n");vl[j]=30;break;

case 31:z[z1]=4;z1++;z[z1]=5;v++;

printf("\tЗаменить(C'M)\tСдвиг\n");vl[j]=31;break;

case 32:z1--;

printf("\tВытолкнуть\tДержать\n");vl[j]=32;break;

case 33:z[z1]=10;z1++;z[z1]=2;z1++;z[z1]=9;z1++;z[z1]=3;v++;

printf("\tЗаменить(S'B'П'С)\tСдвиг\n");vl[j]=33;break;

case 34:z[z1]=10;z1++;z[z1]=2;z1++;z[z1]=9;z1++;z[z1]=3;v++;

printf("\tЗаменить(S'B'П'С)\tСдвиг\n");vl[j]=34;break;

case 35:z[z1]=10;z1++;z[z1]=2;z1++;z[z1]=9;z1++;z[z1]=4;z1++;z[z1]=6;v++;

printf("\tЗаменить(S'B'П'С'M')\tСдвиг\n");vl[j]=35;break;

case 36:z[z1]=2;z1++;z[z1]=9;z1++;z[z1]=4;z1++;z[z1]=7;z1++;z[z1]=1;v++;

printf("\tЗаменить(B'П'С'M1,B)\tСдвиг\n");vl[j]=36;break;

case 37:z[z1]=0;v++;

printf("\tЗаменить(S)\tСдвиг\n");vl[j]=37;break;

case 38:z1--;

printf("\tВытолкнуть\tДержать\n");vl[j]=38;break;

case 39: printf("ДОПУСТИТЬ\n");i=1;vl[j]=39;break;

case 0: printf("Отвергнуть\n");i=2;vl[j]=0;break;

}

if(i==1|i==2)break;else j++;

}

while(tab[z[z1]][a[v]]!=0||tab[z[z1]][a[v]]!=39);

printf("Выходная лента:\n");

for(k=0;k<=j;k++)

{

printf("%d ",vl[k]);

}

return i;

};

/\*--------------------------УДАЛЕНИЕ ПРОБЕЛОВ---------------------------------------\*/

/\*int probel()

{

int i(0),k(0);

for(i=0;i<=j-1;i++)

{

if(s[i]!=' '){s1[k]=s[i];k++;}

}

j=k;

};\*/

/\*-------------------------------------------------------------------------\*/

int main()

{

int w,i;

clrscr();

//probel();

y=vvod();

w=0;

a[w]=0;

if(y==1)

{lexica();

i=analiz();}

if(y==2)

{

lexica1();

i=analiz1();

}

getch();

return 0;

}