Документация

По теме: Постфиксная, префиксная запись.

Выполнил: Маландий Иван 11в

Постановка задачи: Программа нацелена на использование учащимся или группой учащихся для изучения постфиксной и префиксной записи. Генерируются алгебраические выражения в какой-либо форме записи(инфиксная, постфиксная, префиксная), задача ученика записать это выражение в другую форму записи(указано в условии задачи).

Функции, реализуемые модулем:

* 1. Выбор режима: тренировочный, контрольный.
  2. Генерация случайного алгебраического выражения.
  3. Вычисление стековым методом формы записи.
  4. Сравнение ответов(показ результата)
  5. Отправка результатов в базу данных(контрольный режим)

Примеры записей:

|  |  |  |
| --- | --- | --- |
| Инфиксная | Префиксная | Постфиксная |
| A+B | +AB | AB+ |
| A+B\*C | +A\*BC | ABC\*+ |

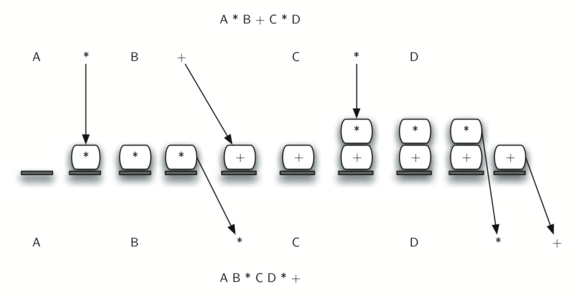
Подробнее про стековый метод:

Предположим, что инфиксное выражение есть строка токенов, разделённых пробелами. Токенами операторов являются \*, /, + и - вместе с правой и левой скобками, ( и ). Токены операндов - это однобуквенные идентификаторы A, B, C и так далее. Следующая последовательность шагов даст строку токенов в постфиксном порядке.

1. Создать пустой стек с названием opstack для хранения операторов. Создать пустой список для вывода.
2. Преобразовать инфиксную строку в список, используя строковый метод split.
3. Сканировать список токенов слева направо.
   * Если токен является операндом, то добавить его в конец выходного списка.
   * Если токен является левой скобкой, положить его в opstack.
   * Если токен является правой скобкой, то выталкивать элементы из opstack пока не будет найдена соответствующая левая скобка. Каждый оператор добавлять в конец выходного списка.
   * Если токен является оператором \*, /, + или -, поместить его в opstack. Однако, перед этим вытолкнуть любой из операторов, уже находящихся в opstack, если он имеет больший или равный приоритет, и добавить его в результирующий список.

#. Когда входное выражение будет полностью обработано, проверить opstack. Любые операторы, всё ещё находящиеся в нём, следует вытолкнуть и добавить в конец итогового списка.

Рисунок демонстрирует алгоритм преобразования, работающий над выражением A \* B + C \* D. Заметьте, что первый оператор \* удаляется до того, как мы встречаем оператор +. Также + остаётся в стеке, когда появляется второй \*, поскольку умножение имеет приоритет перед сложением. В конце инфиксного выражения из стека дважды происходит выталкивание, удаляя оба оператора и помещая + как последний элемент в результирующее постфиксное выражение.



Руководство пользователя:

1. Выбор режима: тренировочный, контрольный(для контрольного требуется ввести имя, фамилию и класс).
2. Перед вами случайно сгенерированный пример, ваша задача решить его и следующие, пока не закончатся примеры или время. Вводим ответ в поле для ответа и нажимаем далее. В тренировочном режиме есть возможность подсмотреть ответ.
3. В конце можно посмотреть ваш результат

Если наблюдаются трудности с запуском:

1. Открываем архив ivan\_500.7z, извлекаем все в удобную папку.
2. скачиваем Python и модули eel, requests.
3. Запускаем main.py

Создание exe через Pyinstaller:

1. Открываем командную строку.
2. качиваем Pyinstaller через команды: pip install wheel, pip install pyinstaller
3. Вписываем cd <путь к файлу>
4. Вписываем pyinstaller -w -F -i