Варианты заданий
на лабораторные работы и РГР (год приема 2022)

В качестве задания на выполнение лабораторных работ и РГР является описание синтаксиса и семантических соглашений некоторого учебного языка программирования.

Синтаксис учебного языка разрабатывается студентом самостоятельно на основе некоторого языка-прототипа.

Общие требования к учебному языку:

- язык должен быть со строгой явной статической типизацией, т.е. типы всех объектов должны быть объявлены в специальном разделе описаний и не допускается неявное преобразование типов;
- должны быть ключевые слова, обозначающие начало и конец программы;
- программа должна состоять из раздела описаний и раздела (последовательности) операторов;
- должно быть, как минимум, три простых предопределенных базовых типа (целый, вещественный, логический);
- для арифметических выражений должны быть определены, как минимум, операции сложения, вычитания, умножения, деления (унарные плюс и минус – по желанию);
- для логических выражений должны быть определены операции дизъюнкции (ИЛИ), конъюнкции (И), отрицания (НЕ) и шесть операций отношения;
- обязательным оператором является оператор присваивания;
- текст программы должен допускать использование комментариев.

Приоритетность арифметических и логических операций определяется языком-прототипом.

Если в языке-прототипе отсутствуют какие-либо пункты перечисленных выше общих требований, следует разработать их самостоятельно.

К общим требованиям добавляются производные типы и другие операторы в зависимости от номера варианта.

Номер варианта состоит из двух цифр. Первая цифра означает выбор оператора, вторая цифра – выбор языка-прототипа.

Операторы:

0) цикл с предусловием типа **while**
1) цикл с постусловием типа **repeat**
2) цикл с параметром типа **for**
3) условный оператор типа **if**
4) оператор варианта типа **case**
5) процедуры (описание и вызов)
6) функции (описание и вызов)

Языки-прототипы:

0) Ada	5) Fortran
1) Kotlin	6) Ruby
2) C, C++, C#, Java	7) Python
3) Delphi, Pascal	8) GoLang
4) Swift	9) Rust

Производные типы данных: для четных номеров – массив, для нечетных – запись (структура).

Для заданий с процедурами и функциями нет производных типов (выделены желтым)!!!

Студент должен описать синтаксис языка в нормальной или расширенной форме Бэкуса-Наура (БНФ или РБНФ). Поскольку БНФ не позволяет задавать контекстные условия, раскрывающие особенности семантики языка, их можно записать в словесной форме в виде перечня неформальных семантических соглашений.

Существуют различные модификации синтаксиса РБНФ. Можно рекомендовать следующий вариант РБНФ.

Металингвистическая переменная (нетерминал) обозначается произвольной символьной строкой (без использования угловых скобок как в БНФ). Если нетерминал состоит из нескольких смысловых слов, то они записываются слитно или разделяются символом подчеркивания. При этом для удобства восприятия целесообразно каждое ее слово начинать с прописной буквы.

Терминальные символы изображаются словами, написанными буквами латинского алфавита (ключевые слова) или цепочками символов, заключенными в одиночные (′) или двойные (″) кавычки. Для удобства восприятия ключевые слова можно выделить вдобавок и жирным шрифтом.

Левая и правая части правила разделяются метасимволом "=" (вместо "::=" в БНФ), альтернативные варианты разделяются метасимволом "|". Каждое правило заканчивается точкой.

Квадратные скобки "[" и "]" означают, что заключенная в них синтаксическая конструкция может отсутствовать. Фигурные скобки "{" и "}" означают нуль или более повторений заключенной в них синтаксической конструкции.

Ниже приведен пример описания синтаксиса учебного языка с использованием РБНФ. Курсивом выделены понятия, подчеркивающие семантический смысл синтаксической конструкции.

1. Модуль = "**prog**" Имя_ *программы* "**;**" Блок "**.**".
2. Имя = Буква { Буква | Цифра }.
3. Блок = Объявления "**beg**" ПоследОператоров "**end**".
4. Объявления = ["**type**" ОпределТипов] "**var**" ОписПеременных.
5. ОпределТипов = ОпределТипа ";"{ ОпределТипа ";" }.
6. ОпределТипа = Имя_*типа* "**is**" Тип.
7. ОписПеременных = ТипПеременных "**;**" { ТипПеременных "**;**" }.
8. ТипПеременных = СписокИмен "**:**" Тип.
9. СписокИмен = Имя { "**,**" Имя }.
10. Тип = ПростойТип | ТипМассив.
11. ПростойТип = Имя_*типа* | Диапазон.
12. Диапазон = Число "**..**" Число.
13. ТипМассив = "**array**" "**[**" СписИндексов "**]**" "**of**" Тип.
14. СписИндексов = Индекс { "**,**" Индекс }.
15. Индекс = ПростойТип.
16. ПоследОператоров = Оператор { "**;**" Оператор }.
17. Оператор = Присваивание | Цикл.
18. Присваивание = Переменная "**:=**" Выражение.
19. Переменная = Имя | ИндексПеременная.
20. ИндексПеременная = Имя_*массива* "**[**" ПростоеВыраж { "**,**" ПростоеВыраж } "**]**".
21. Цикл = "**while**" Выражение "**do**" ПоследОператоров "**end**".
22. Выражение = ПростоеВыраж [Отношение ПростоеВыраж].
23. Отношение = "<" | "<=" | ">" | ">=" | "=" | "#".
24. ПростоеВыраж = ["+" | "–"] Терм { АддитОперация Терм }.
25. АддитОперация = "+" | "–" | "**or**".
26. Терм = Фактор { МультОперация Фактор }.
27. МультОперация = "*****" | "/" | "**and**" | "**div**".
28. Фактор = Константа | Переменная | "**(**" Выражение "**)**" | "**not**" Фактор.
29. Константа = Число | ЛогическаяКонст.
30. Число = Целое ["**.**" Целое].
31. Целое = Цифра { Цифра }.
32. ЛогическаяКонст = "**true**" | "**false**".

Определение нетерминалов «Буква» и «Цифра» здесь не приведено ввиду их очевидности. «Буква» определяется выбранным алфавитом (обычно строчные и прописные буквы латинского алфавита), а «Цифра» – арабские цифры от 0 до 9.

Краткая характеристика языка и семантические соглашения:

- Язык удовлетворяет семантическим соглашениям, характерным для многих языков программирования (единственность именования различных объектов программы, необходимость описания идентификатора до его использования и т.п.).
- В ключевых словах и идентификаторах прописные и строчные буквы не различаются.
- Ключевые слова языка зарезервированы, их нельзя использовать в качестве идентификаторов.
- Отсутствуют именованные константы.
- Предопределенные (базовые) типы: *integer, float, Boolean, char*.
- В конструкции «Диапазон» (правило 12) числа должны быть целого типа, кроме того, значение первого числа должно быть меньше значения второго числа.
- Если в конструкции «Индекс» (правило 15) в качестве «ПростойТип» используется имя типа («Имя_*типа*»), оно должно представлять тип «Диапазон».
- В операторе цикла конструкция «Выражение» после ключевого слова **while** должно быть типа Boolean (правило 21).
- Комментарий представляет собой любую последовательность символов, заключенную в фигурные скобки "{" и "}".