Расширенная БНФ и примеры в C

Расширенная форма Бэкуса-Наура (БНФ) состоит из ряда следующих обозначений и правил:

* символы в кавычках переносятся в конструкцию языка так, как они записаны. Кавычки при этом отбрасываются. Например, "while" означает, что в конструкции языка присутствует while;
* имена, записанные слитно русскими и латинскими буквами, обозначают различные конструкции языка. Например, оператор\_цикла;
* квадратные скобки охватывают элементы языка, которые могут повторяться 0 или 1 раз. Например, "AB"["C"] означает, что в конструкции языка может присутствовать или AB или ABC;
* фигурные скобки охватывают элементы языка, которые могут повторяться 0 или много раз. Например, "AB" { "C" } означает, что в конструкции языка может присутствовать или AB, или ABC, или ABCC и т.д.;
* символ | обозначает или, то есть используется для задания альтернативных значений, из списка элементов, разделенных знаком |. Например, "AB"|"C"|"ff" означает, что в конструкции языка может присутствовать или AB или C или ff;
* круглые скобки используются для группировки. Например, "A"("B"|"C")"D" означает, что в конструкции языка может присутствовать или ABD или ACD;
* многоточие используется для обозначения очевидных пропущенных значений в перечислении;
* символ = обозначает - слово есть. Например, буква = "A"|"B"|"C".

# Идентификаторы языка Си

имя = ( буква | "\_" ) { буква | цифра | "\_" }

буква = |"A"|"B"|...|"Y"|"Z"|"a"|"b"|...|"y"|"z"

цифра = "0"|"1"|...|"9"

## Основные типы данных языка Си

описание\_типа = ["const "] имя\_типа " " имя ["=" константа]

{ "," имя ["=" константа] } ";"

имя\_типа = "int" | "short" | "long" | "char" | "float" | "double"

## Правила записи констант различных типов

целая\_константа =( десятичная | восьмеричная | шестнадцатеричная ) ["l"|"L"]

десятичная = цифра {цифра}

восьмеричная = "0" цифра8 {цифра8}

шестнадцатеричная = "0" ( "x" | "X" ) цифра16 {цифра16}

цифра8 = "0"|"1"|...|"7"

цифра16 = "0"|"1"|...|"9"|"A"|"B"|"C"|"D"|"E"|"F"

вещ\_константа = цифра {цифра} "." {цифра} [ "e" [ "+" | "-" ] цифра {цифра} ]

## Беззнаковый тип для целых данных

имя\_беззнакового\_типа = "unsigned "("int"|"long"|"short"|"char" )

## Символьные строки

описание\_массива\_символов = "char " имя "["размер"]" { "," имя "["размер"]" } ";"

# Понятие функции

вызов\_функции = имя\_ функции "(" [ аргумент { "," аргумент } ] ")"

## Стандартная функция printf

"printf" "(" формат { "," аргумент } ")"

модификатор = ["-"] {цифра1} [ "." {цифра2} ][l]

## Стандартная функция scanf

"scanf" "(" формат { "," аргумент } ")"

# Операции и выражения

## Оператор-выражение

оператор = выражение ";"

## Операции преобразования типов

преобразование\_типа = "("имя\_типа ")" выражение

## Простейшие функции, определяемые программистом

заголовок\_функции = тип имя\_функции"("[тип параметр {"," тип параметр}]")"

"return " [выражение];

## Операция определения размера данных

"sizeof(" тип ")"

"sizeof " имя\_данного

# Операторы языка Си и приемы программирования

составной\_оператор = "{" оператор { оператор } "}"

## Оператор цикла while

"while" "(" выражение ")" оператор

## Условный оператор и условная операция

условный\_оператор = "if" "("выражение")" оператор\_1 ["else" оператор\_2]

условное\_операция = выр\_0 "?" выр\_1 ":" выр\_2

## Множественный выбор. Оператор переключения

оператор\_переключения =

"switch" "(" выражение ")"

"{"

"case" константа ":" { оператор }

"case" константа ":" { оператор }

[ "default" ":" { оператор } ]

"}"

## Оператор цикла do-while

цикл\_do-while = "do" оператор "while" "("выражение")" ";"

## Перечисления

перечисление =

"enum" [ имя\_перечисления ]

"{"

имя\_конст [ "=" конст\_выр ]

{ "," имя\_конст [ "=" конст\_выр ] }

"};"