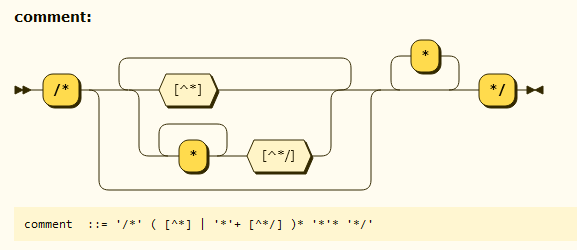
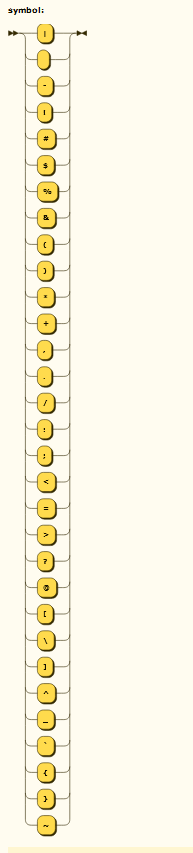
# **Синтаксис языка:** Go on Procedural Parametric Language (Go2P)

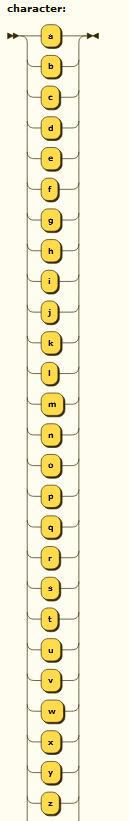
1. Коментарии (сomment)
2. Набор символов (symbol)
3. Набор латинских символов (character)
4. Набор чисел от 1-9 (digit\_excluding\_zero)
5. Число от 0-9 (digit)
6. Натуральные числа (natural\_number)
7. Целые числа (integer)
8. Название переменных функций или обобщений (name)
9. Название переменных (nameLabel)
10. Название функций (nameFunc)
11. Название Структур (nameStruct)
12. Название Обобщений ( nameRecord)
13. Ключ указывающий на объявления типа с пробелом (keyType)
14. Описание струтуры (structNewType)
15. Описание обобщения (recordNewType)
16. Сложный тип данных (hardType)
17. Различные вырианты типо челого чилса (typeInteger)
18. Различные варианты числа с плавающей точкой ( floatType)
19. Ссылочный тип данных (hrefType)
20. Указатель ( pointType)
21. Cтандартный тип данных (type)
22. Возвращаемый тип данных (returnType)
23. Объявление объекта или переменой (object)
24. Объявление объектов или переменных аргументов в вункции ( objectFunArg)
25. Перечисление объектов или переменных аргументов при вызове функции (objectFunCallArg)
26. Возможная инициализация объекта (objectVarInit)
27. Стандартное объявление переменных (objectInitStandartType)
28. Объявление переменных с последующей инициализацией (objectInit)
29. Объявление переменных с автоматическим определением типа (objectInitAuto)
30. Реализация данных в теле функции (bodyFigure)
31. Реализация данных в инициализации сложного типа (bodyFigureInitHardType)
32. Инициализация функции (func)
33. Вызов функции (funCall)
34. Различные переходы (tab)
35. Стандартные арифметические операторы (operators)
36. Выражение (expression)
37. Инициализация строки (stringStr)
38. Иницциализации сивола (charStr)
39. Значение или данные (value)
40. Краткая запись арифметической операции над объектом (shortOperators)
41. Коментарии (сomment)

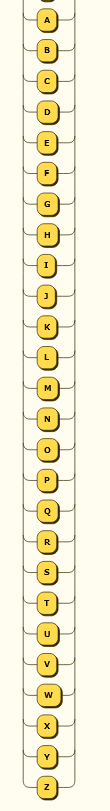


1. Набор символов (symbol)

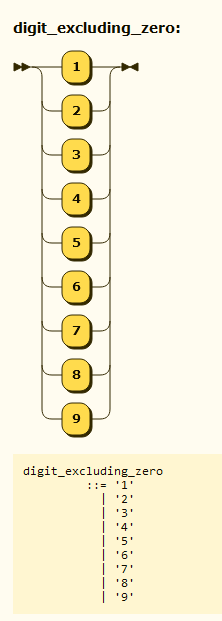


1. Набор латинских символов (character)

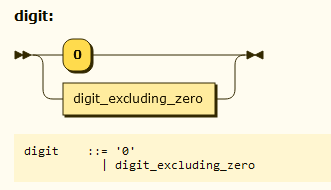




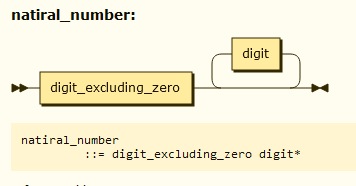
1. Набор чисел от 1-9 (digit\_excluding\_zero)



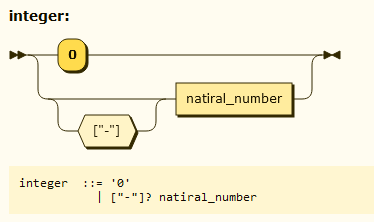
1. Число от 0-9 (digit)



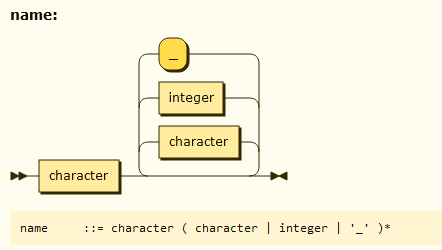
1. Натуральные числа (natural\_number)



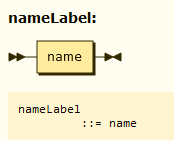
1. Целые числа (integer)



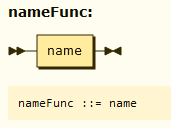
1. Название переменных функций или обобщений (name)



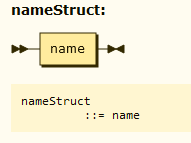
1. Название переменных (nameLabel)



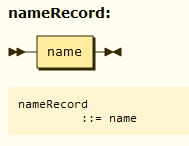
1. Название функций (nameFunc)



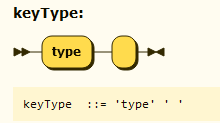
1. Название Структур (nameStruct)



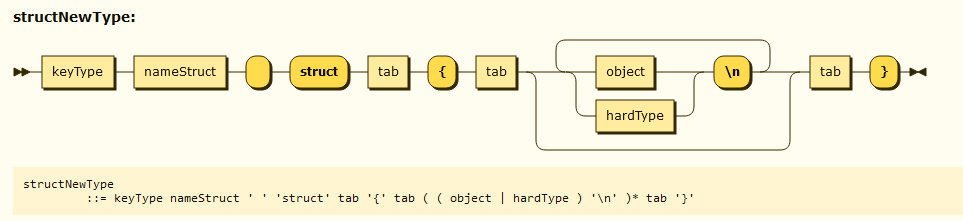
1. Название Обобщений ( nameRecord)



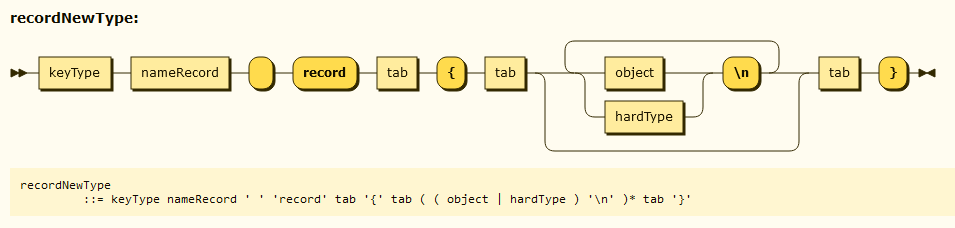
1. Ключ указывающий на объявления типа с пробелом (keyType)



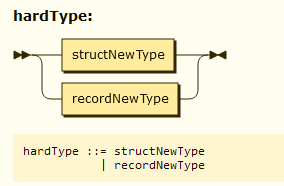
1. Описание струтуры (structNewType)



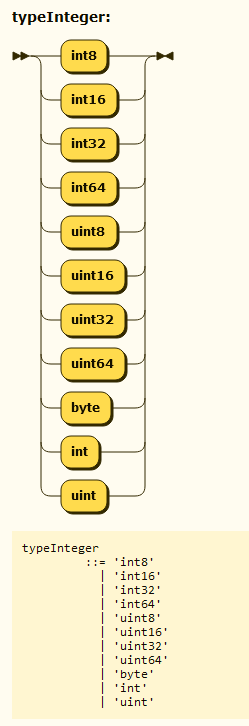
1. Описание обобщения (recordNewType)



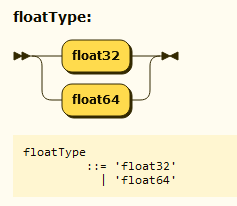
1. Сложный тип данных (hardType)



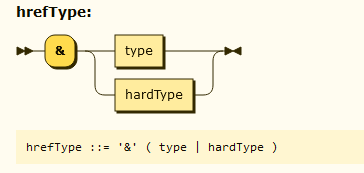
1. Различные вырианты типо челого чилса (typeInteger)



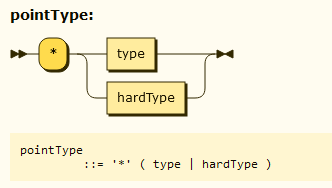
1. Различные варианты числа с плавающей точкой ( floatType)



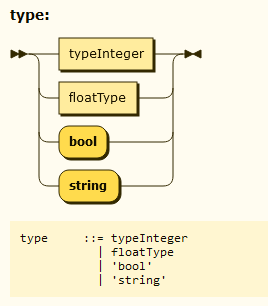
1. Ссылочный тип данных (hrefType)



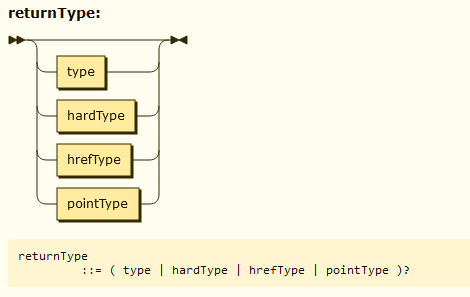
1. Указатель ( pointType)



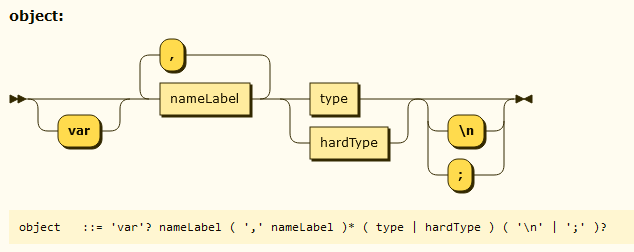
1. Cтандартный тип данных (type)



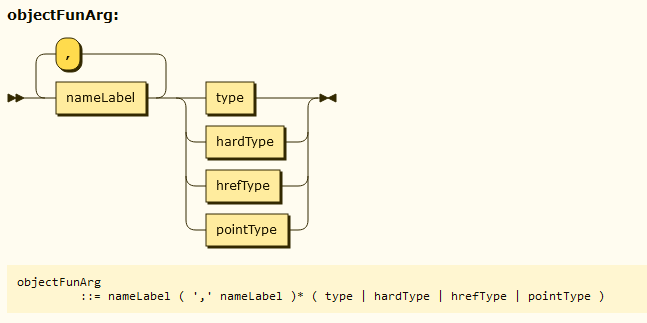
1. Возвращаемый тип данных (returnType)



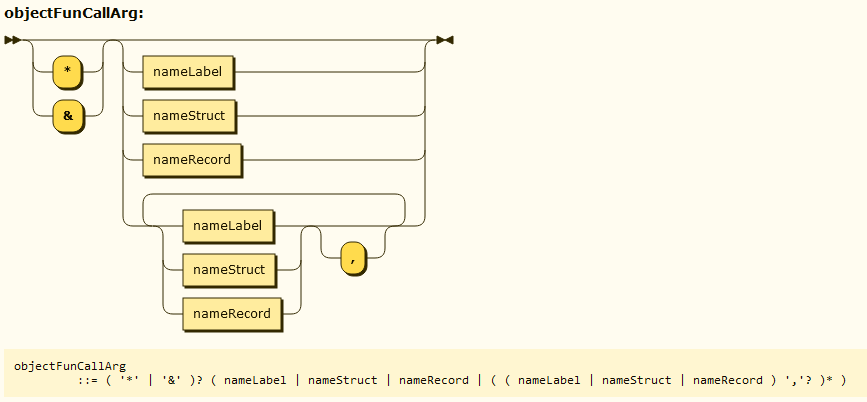
1. Объявление объекта или переменой (object)



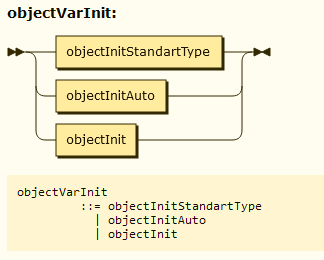
1. Объявление объектов или переменных аргументов в вункции ( objectFunArg)



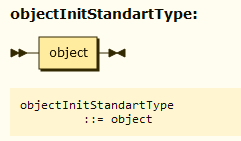
1. Перечисление объектов или переменных аргументов при вызове функции (objectFunCallArg)



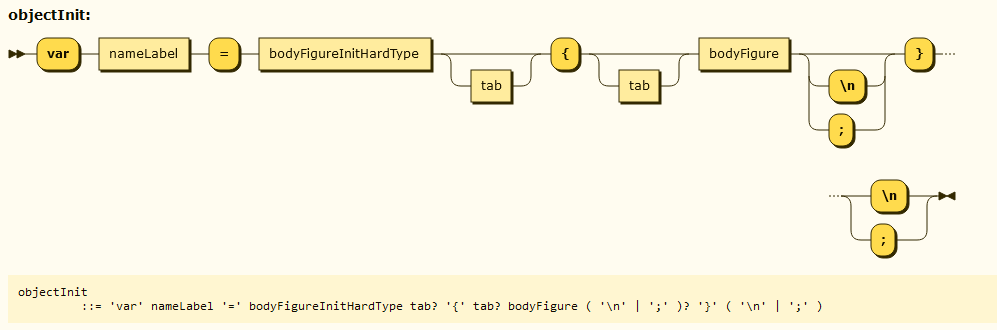
1. Возможная инициализация объекта (objectVarInit)



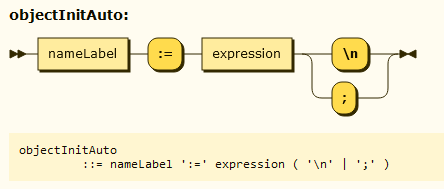
1. Стандартное объявление переменных (objectInitStandartType)



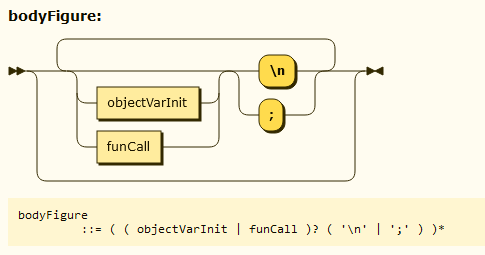
1. Объявление переменных с последующей инициализацией (objectInit)



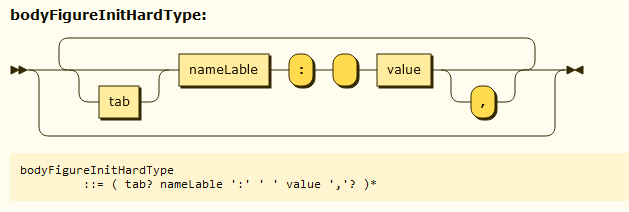
1. Объявление переменных с автоматическим определением типа (objectInitAuto)



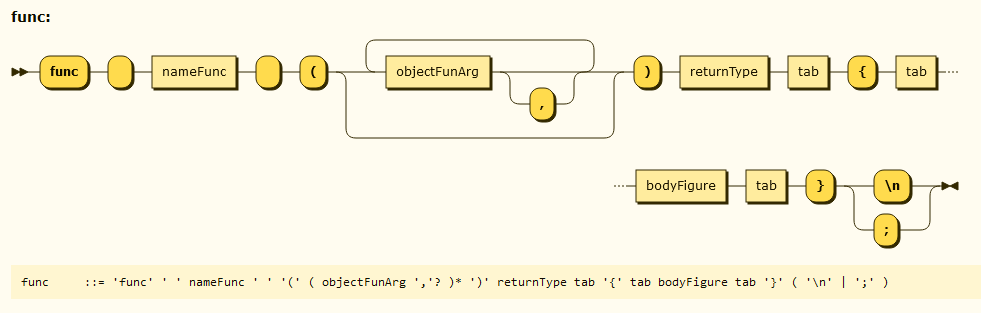
1. Реализация данных в теле функции (bodyFigure)



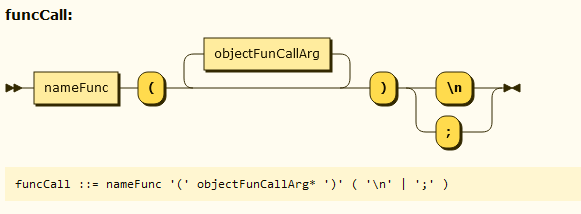
1. Реализация данных в инициализации сложного типа (bodyFigureInitHardType)



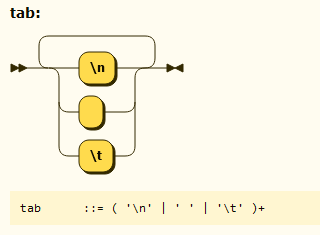
1. Инициализация функции (func)



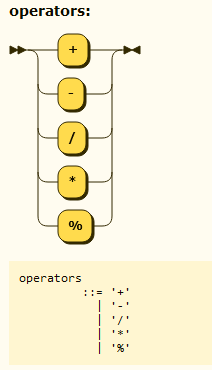
1. Вызов функции (funCall)



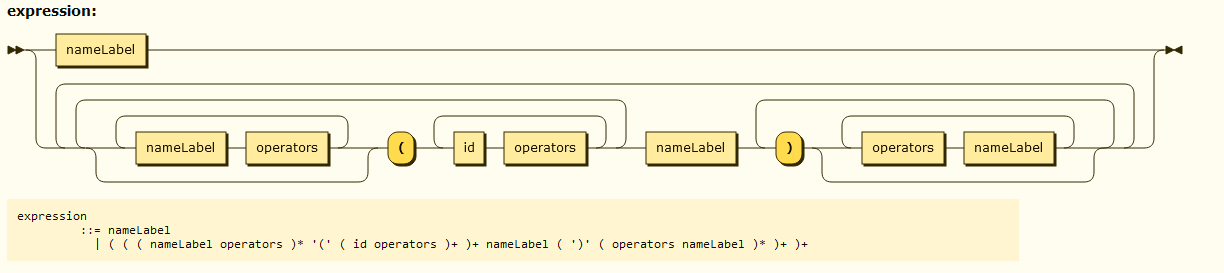
1. Различные переходы (tab)



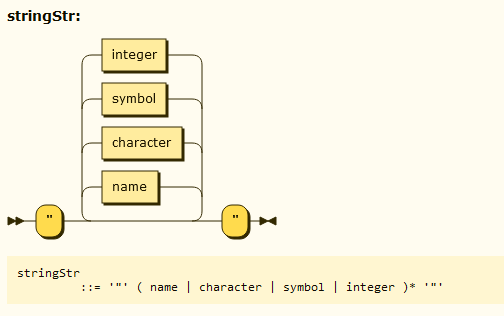
1. Стандартные арифметические операторы (operators)



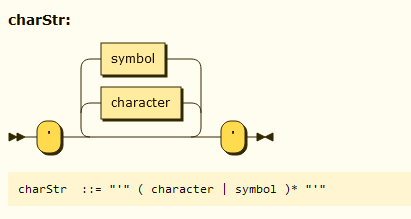
1. Выражение (expression)



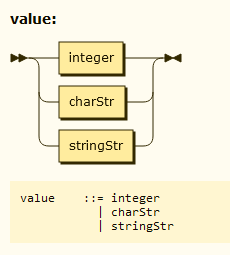
1. Инициализация строки (stringStr)



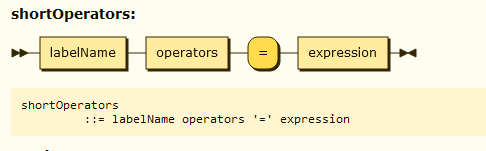
1. Иницциализации сивола (charStr)



1. Значение или данные (value)



1. Краткая запись арифметической операции над объектом (shortOperators)



::= - присвоение

| - ИЛИ

‘\*’ – Повторение 0 или более раз

‘+’ – Повторение 1 или более раз

? – может как быть так и не быть ( без повтарений)

() – связывание

# БНФ:

сomment ::= '/\*' ( [^\*] | '\*'+ [^\*/] )\* '\*'\* '\*/'

symbol ::= "|"|""|"-"|"!"|"#"|"$"|"%"|"&"|"("|")"|"\*"|"+"|","|"-"|"."|"/"|":"|";"|"<"|"="|">"|"?"|"@"|"["|"\"|"]"|"^"|"\_"|"`"|"{"|"|"|"}"|"~"

character ::= "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"|"E"|"F"|"G"|"H"|"I"|"J"|"K"|"L"|"M"|"N"|"O"|"P"|"Q"|"R"|"S"|"T"|"U"|"V"|"W"|"X"|"Y"|"Z"

digit\_excluding\_zero ::= "1"|"2"|"3"|"4"|"5"|"6"|"7"|"8"|"9"

digit ::= "0"|digit\_excluding\_zero

natiral\_number ::= digit\_excluding\_zero digit\*

integer ::= "0"|(["-"])? natiral\_number

name ::= character (character|integer|"\_")\*

nameLabel ::= name

nameFunc ::= name

nameStruct ::= name

nameRecord ::= name

keyType ::= "type" " "

structNewType ::= keyType nameStruct " " "struct" tab "{" tab ((object|hardType) "\n")\* tab "}"

recordNewType ::= keyType nameRecord " " "record" tab "{" tab ((object|hardType) "\n")\* tab "}"

hardType ::= structNewType|recordNewType

typeInteger ::= "int8"|"int16"|"int32"|"int64"|"uint8"|"uint16"|"uint32"|"uint64"|"byte"|"int"|"uint"

floatType ::= "float32"|"float64"

boolType ::= "bool"

stringType ::= "string"

hrefType ::= "&" (type|hardType)

pointType ::= "\*" (type|hardType)

type ::= typeInteger|floatType|boolType|stringType

returnType ::= (type|hardType|hrefType|pointType)?

object ::= ("var")? nameLabel("," nameLabel)\* (type|hardType) ("\n"|";")?

objectFunArg ::= nameLabel("," nameLabel)\* (type|hardType|hrefType|pointType)

objectFunCallArg ::= (("\*"|"&")? (nameLabel|nameStruct|nameRecord) | ("\*"|"&")? ((nameLabel|nameStruct|nameRecord)(",")?)\*)

objectVarInit ::= objectInitStandartType|objectInitAuto|objectInit

objectInitStandartType ::= object

objectInit ::= "var" nameLabel "=" bodyFigureInitHardType (tab)? "{" (tab)? bodyFigure ("\n"|";")? "}" ("\n"|";")

objectInitAuto ::= nameLabel ":=" expression ("\n"|";")

bodyFigure ::= (((objectVarInit|funCall|) ("\n"|";")))\*

bodyFigureInitHardType ::= ((tab)? nameLable ":" " " value (",")?)\*

func ::= "func" " " nameFunc " " "(" (objectFunArg|(objectFunArg) ",")\* ")" returnType tab "{" tab bodyFigure tab "}" ("\n"|";")

funcCall ::= nameFunc "(" (objectFunCallArg)\* ")" ("\n"|";")

tab ::= ("\n"|" "|"\t")+

operators ::=("+"|"-"|"/"|"\*"|"%")

expression ::= nameLabel|((((nameLabel operators)\* "(" (id operators)+)+ nameLabel (")" (operators nameLabel)\*)+))+

stringStr ::= ["] (name|character|symbol|integer)\* ["]

charStr ::= ['] (character|symbol)\* [']

value ::= integer|charStr|stringStr

shortOperators ::= labelName operators "=" expression

Объявление переменных:

<имя переменной> <тип данных>

Example:

g float64

x , y , z int32

rect Rectangle

Объявление указателей:

rect \*Rectangle

Объявление указателей на функцию:

<имя переменной> <имя функции>(<тип аргумента>)<тип возвращаемого значения>

point func (int) int

Получить указатель на выделенную память объекта :

new(<Тип данных>)

**Объявление сложного типа данных (специализация):**

(То же самое что и в языке Go)

type <имя тип> struct {

<имя поля> <тип поля>

<имя типа наследника>

}

**Объявление обобщения:**

Обобщение – тип данных являющемся общим для определенных специализаций

Type <имя обобщения> record {

<тип данных уточнения>

}

**Инициализация обобщенного объекта (всегда указатель):**

<имя переменной> \*<тип обобщения> (<тип специализации уточнения>);

<имя переменной> := new(<тип обобщения> (<тип специализации уточнения>));

**Добавления типа данных уточнения в объявленное обобщение:**

<тип данных обобщения> += <тип данных уточнения>

**Объявление процедур(функций):**

func <имя функции> (<имя аргумента> <тип аргумента>)<возвращаемый тип> {

return <возвращаемый тип>

}

**Инкапсуляция достигается объявлением открытых функций (pulic func) с помощью заглавной буквы имени функии:**

**Public:**

func <Имя функции> (<имя аргумента> <тип аргумента>) <возвращаемый тип> {

return <возвращаемый тип>

}

**Private:**

func <имя функции> (<имя аргумента> <тип аргумента>) <возвращаемый тип> {

return <возвращаемый тип>

}

**Пример программы**

//------------------------------------------------------------------

// Специализация Прямоугольника

type Rectangle struct {

a,b float64

}

//------------------------------------------------------------------

// Специализация Треугольника

type Triangle struct {

a,b,c float64

}

//------------------------------------------------------------------

// Специализация Круга

type Circl struct {

r float64

}

//------------------------------------------------------------------

// Обобщение фигуры

type Shape record

{

Rectangle

Triangle

}

//------------------------------------------------------------------

// Обобщенная функция (различие в перегруженных функций по типу принимаемых значений)

func Square(item \*Shape) {

name := square(item)

printf("\nPerimeter %s = %f\n",name,ans)

}

func main() int {

//При инициализации, обобщение может быть только указателем

rec := new(Shape(Rectangle)) //Указатель на обобщенный Прямоугольник

tri := new(Shape(Triangle)) //Указатель на обобщенный Триугольник

cir := new(Shape(Circl)) //Указатель на обобщенный Круг

Square(rec) //Вызов обобщенной функции вычисления Площади фигуры Прямоугольник

Square(tri) //Вызов обобщенной функции вычисления Площади фигуры Триугольник

Shape += Circl //Добавить тип данных для обобщения Круга в фигуру

Square(cir) //Вызов обобщенной функции вычисления Площади фигуры Круга

return 0

}

//------------------------------------------------------------------

// Сигнатуры требуемых функций

func Square(item \*Rectangle) float64 {

s := item.a \* item.b

return s

}

func Square(item \*Triangle) float64 {

s := 1/2\*(item.a + item.b + item.c)

return s

}

func Square(item \*Circl) float64 {

s := 3.14\*item.r

return s

}

//------------------------------------------------------------------