### Строковые литералы

Ограничения на длину строковых литералов, судя по всему, нет. ИРБИС воспринимает литералы длиной более миллиона знаков (проверено).

Строковые литералы могут располагаться на нескольких строках, это не является синтаксической ошибкой. Перед выводом на печать все символы перевода строки удаляются. Таким образом, конструкция

'Hello,  
world!'

выведет

Hello,world!

Пустые переводы строки исчезают. Поэтому конструкция

'Hello,  
  
world!'

полностью эквивалентна предыдущей.

Редактор GenPft64 сопротивляется переводу каретки после появления в тексте открывающего символа, однако, даёт вставить многострочный литерал из буфера обмена.

### Числа и числовые литералы

Внутреннее представление чисел – 8-байтовое плавающее с диапазоном от 2,3×10−308 до 1,7×10+308. Хотя функция f использует формат представления, характерный для 10-байтового плавающего расширенной точности. Конструкции вроде

f(2e308)

выводят

+Inf

f(1e-325)

выводит пустую строку, т. е. интерпретируется как 0.

### Команды вертикального размещения

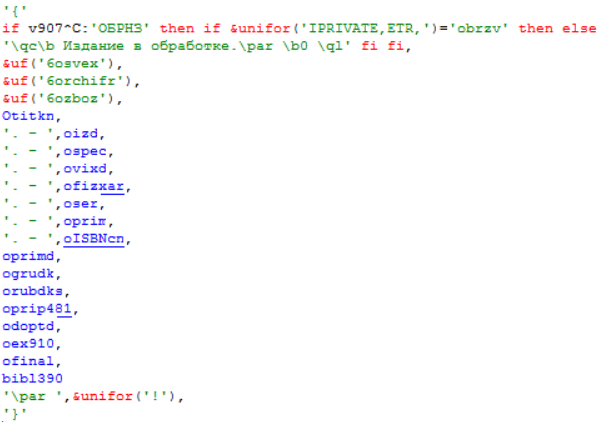
|  |  |
| --- | --- |
| Команда | Что делает |
| # | Безусловно выводит \r\n (кроме случая, когда стоит сразу за %) |
| / | Выводит \r\n только тогда, когда курсор не в первой колонке |
| % | Удаляет оконечные \r\n\r\n, убирая \r\n с конца.  Если текст не заканчивается \r\n, выводит \r\n.  Подавляет один следующий # |

### Вложенные форматы

Официальным способом вызова вложенного формата является

&uf('6fileName')

Однако, существует и неофициальный, когда имя файла формата (без расширения) заключается между байтами 0x1C (28) и 0x1D (29). По неизвестным причинам ГПНТБ использует оба способа сразу. Пример смешения двух подходов мы можем наблюдать, например, в файле KN.PFT в папке Deposit (см. снимок экрана GenPft64.exe):



Вложенные форматы на скриншоте подсвечены синим. Таким образом, GenPft64 знает о подобном способе вложения форматов, и даже открывает формат по щелчку на нём мышью. Более того, предусмотрена сочетание клавиш Ctrl+I, которое открывает диалог вставки вложенного формата.

### Вывод полей

Между элементами оператора допускаются пробелы:

v200 ^a

Более того, допускаются переводы строки:

v200   
^a  
\*˽10  
.˽3  
(˽5,˽1)

Между крышкой и последующим символом пробела быть не должно, т. к. он будет интерпретироваться как код подполя (таким образом, с точки зрения ИРБИС, пробел – вполне допустимый код подполя).

Забавно, но при этом пробел перед запятой или закрывающей скобкой в спецификации абзаца не допускается:

v200(5˽,1˽) /\* ошибка!

Ограничений на длину тега поля нет, конструкции вроде такой

v20000000000000000000000000000000000000000000000000000000000000000000000000^a

воспринимаются без проблем

Коды подполей с кириллическими буквами не воспринимаются:

v200^ю

выдаст ошибку

\*\* Format Error 99 - Неизвестная команда (например, ошибка в правильности написания имени функции или команды), возможен также пропуск закрывающего ограничителя литерала. \*\*

Смещения интерпретируются своеобразно:

v200^a\*100

вплоть до

v200^a\*10000000000000000000000000000000

воспринимается нормально. Но уже

v200^a\*100000000000000000000000000000000

вновь выдаёт значение подполя, хотя не должно этого делать. Дальнейшее увеличение длины не меняет поведения

v200^a\*100000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000

Повторения подполей игнорируются, выводится только первое повторения указанного подполя. Пусть у нас есть запись

#1/1: ^aA1^aA2^aA3^bB1

Тогда формат

v1^a

выведет

A1

Однако, формат v1 выведет

^aA1^aA2^A3^bB1

### Логические выражения

if p(v200) and p(v201) or p(v200)  
 then 'True'   
 else 'False'   
fi

выдаёт тот же результат (True), что и

if p(v200) or p(v200) and p(v201)  
 then 'True'   
 else 'False'   
fi

Т. е. «невидимых скобок» нет, и логические выражения интерпретируются слева направо без приоритетов операторов.

### Глюки

Формат

&uf('0') (&uf('0'))

выдаёт ошибку «\*\* Format Error 19 - Непарная открывающаяся скобка (. \*\*» (вместо «0» в &uf может быть любое выражение, даже пустое), хотя полностью эквивалентный ему формат

&uf('0'), (&uf('0'))

отрабатывает нормально (вместо запятой может быть любая конструкция PFT, например, / или #.

Аналогично дело обстоит с форматом

v1(&uf('0'))

но здесь хотя бы понятна причина – ИРБИС интерпретирует группу как спецификацию абзацного отступа.