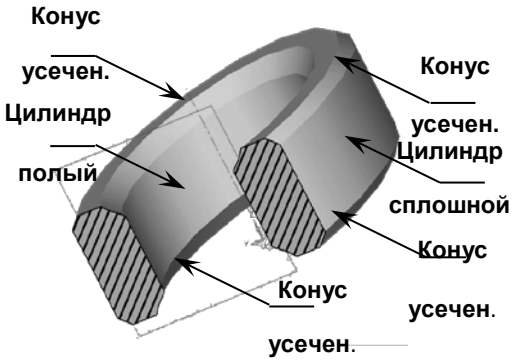
УНИВЕРСИТЕТ ИТМО

Филиппов А. Н., Путинцева А.А.

ПРИМЕНЕНИЕ МЕТОДОВ ВИРТУАЛЬНОГО СТРОКОВОГО ПРОСТРАНСТВА ТЕХНОЛОГИЧЕСКИХ ДАННЫХ И ЗНАНИЙ В САПР ТП



Геометрическая модель

Вал (1 ед.):

$CIM_CYL.RAD=30;

$CIM_CYL.LEN=20;

$CIM_CYL.TYPE=1;

$CIM_CYL.LINK=0;

Отверстие (1 ед.):

$CIM_CYL1.RAD=20;

$CIM_CYL1.LEN=20;

$CIM_CYL1.TYPE=0;

$CIM_CYL1.LINK=0;

Фаска (4 ед.):

$CIM_CHM.LEN=3;

$CIM_CHM.LINK=1;

$CIM_CHM.ANG=45;

Параметрическая модель

Санкт-Петербург

2015

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
УНИВЕРСИТЕТ ИТМО

А.Н. Филиппов, А.А. Путинцева

ПРИМЕНЕНИЕ МЕТОДОВ ВИРТУАЛЬНОГО СТРОКОВОГО ПРОСТРАНСТВА ТЕХНОЛОГИЧЕСКИХ ДАННЫХ И ЗНАНИЙ В САПР ТП

Описание применения

Методическое пособие

УНИВЕРСИТЕТ ИТМО

г. Санкт-Петербург
2015

УДК 658.512.011.56

А.Н. Филиппов, А.А. Путинцева. Применение методов виртуального строкового пространства Технологических данных и знаний в САПР ТП/ Методическое пособие // Описание применения. СПб: Университет ИТМО, 2015. – с.40

Настоящее пособие предназначено для студентов специализации «Технологии приборостроения». В пособии изложены наиболее важные темы, связанные с описанием методов организации данных и программирования алгоритмов с применением символьного представления объектов, как основы проектирования экспертных систем технологического назначения.

Адресовано студентам высших учебных заведений, обучающихся по направлению подготовки 09.03.01 – Информатика и вычислительная техника.

Рекомендовано к печати Ученым советом факультета Точной механики и технологии 10-го марта 2015 г, протокол № 3.

УНИВЕРСИТЕТ ИТМО

В 2009 году Университет стал победителем многоэтапного конкурса, в результате которого определены 12 ведущих университетов России, которым присвоена категория «Национальный исследовательский университет». Министерством образования и науки Российской Федерации была утверждена программа его развития на 2009–2018 годы. В 2011 году Университет получил наименование «Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики»

ОГЛАВЛЕНИЕ

Сокращения, принятые в пособии

БД	база данных
БЗ	база знаний
ВСПТД	виртуальное строковое пространство технологических данных
ГО	генератор отчетов
ДМВ	дедуктивная машина вывода
ИИ	искусственный интеллект
ИМ	интерфейсный модуль
ИПС	информационно-поисковая система
ЛПР	лицо, принимающее решение
МАС	многоагентная система
САПР ТП	система автоматизации проектирования технологических процессов
СУЗ	система управления записями
ТП	технологический процесс
ТС	триплексная строка
ФПЗ	формализма представления знаний
ЭС	экспертная система
ЯП	язык программирования

Введение

Новые методы автоматизации проектирования технологических процессов (ТП) во многих случаях опираются на методы искусственного интеллекта, одним из разделов которого являются экспертные системы (ЭС). В таком направлении на кафедре ТПС разработана теория виртуального строкового пространства технологических данных (ВСПТД), приемы работы с которым изложены ниже.

Для глубокого понимания изложенного материала рекомендуется ознакомиться с учебным пособием «Виртуальное строковое пространство технологических данных и знаний. Методы представления данных» [1], в котором приведено подробное изложение понятий, используемых в настоящей работе.

Применение инструментальных средств рассчитано на интегрированную среду разработки IDLE. Предполагается, что читатель знаком с данной средой и языком Python. Программный код, представленный в данном пособии, написан на языке программирования Python 3.4. Использование этого языка существенно упрощает работу с ВСПТД по сравнению с существующими наработками на языке C++[1] так как, во-первых, гарантирует отсутствие привязки к конкретной операционной системе и, во-вторых, позволяет объединить схожие по своему назначению функции и тем самым сократить их количество. Возможность объединения функций обусловлена динамической типизацией в языке Python. В данном пособии представлены базовые функции работы с ВСПТД, исходные тексты которых представлены в приложении А[2].

Учитывая многообразие задач, которые возникают перед разработчиками систем автоматизированного проектирования технологической подготовки производства САПР ТПП, встает вопрос выбора моделей представления баз данных (БД) и баз знаний (БЗ). Использование готовых инструментальных средств существенно сокращает время разработки системы. Однако программное обеспечение подобного типа далеко не во всех случаях предоставляет возможности для обеспечения гибкости системы: отсутствуют условия для развития интерфейса, существенно ограничивается выбор формализма представления знаний (ФПЗ). Кроме того, эти факторы накладывают значительные ограничения на принцип эволюционного развития САПР ТП.

В рамках данного пособия невозможно охватить весь спектр задач, которые встают перед разработчиками САПР технологического назначения. В работе описаны инструментальные программные средства и методы построения структур данных при создании САПР ТП. Предлагается базовый пакет программ и алгоритмов, который можно развивать путем подключения новых функций. Для применения данного пакета студенты должны овладеть

[1]С расширенной версией программного обеспечения ВСПТД, реализованного на языке программирования C++, можно ознакмиться здесь [2].

[2]Полный доступ к инструментариям ВСПТД обеспечивается на занятиях.

основами программирования на языках C++ и PYTHON.

Основные определения

Виртуальные строковое пространство технологических данных – язык и протокол для взаимодействия между программными агентами и системами, основанными на знаниях.

Фундаментальным понятием ВСПТД является триплет.

В широком смысле триплетом (лат. *triplus* – тройной) называют совокупность каких-либо трех объектов.

В ВСПТД под **триплетом** понимается специальная форма описания в символьном представлении, имеющая следующий вид:

объект – имя характеристики – отношение – значение – комментарий

Объект представляет собой математическую абстракцию, поставленную в соответствие некоторой реальной сущности, обладающей набором качественных и/или количественных характеристик.

Данные, известные системе в текущий момент, называют **фактами.**

Каждый факт представляется в виде триплета, имеющего в программном коде вид

Ф = <Prefix, Name, Value>,

где

- Prefix – префикс,
- Name – имя параметра,
- Value – значение параметра.

Здесь префикс обеспечивает контекстное понятие параметра, т.е. указывает на конкретный описываемый объект. Например, если L – имя параметра, указывающее в общепринятых обозначениях на длину описываемого объекта, то выражение Z.L будет обозначать длину заготовки при условии, что Z – это префикс, указывающий на объект «заготовка». Т.е., Z – это множество характеристик производственного объекта «заготовка», а Z.L – один из элементов множества Z. Третья составляющая факта – «значение» – всегда представляется в символьной форме.

Таким образом, поля Prefix, Name и Value триплета обозначают соответственно объект, имя характеристики и значение.

В классической теории ВСПТД синтаксис триплета представляет собой следующую структуру:

$<Префикс>.<Имя> = <Значение>["Комментарий"];

Служебный символ «$» указывает на начало триплета, «=» определяет отношение, а символ «;» служит завершением триплета.

Примеры триплетов факта:

- `$E.NM='Фреза 20';`
- `$W.NM='Скоба 3,1 h12';` и т.д.

Еще не известные системе данные называют **целями** системы.

Каждая цель представляется в виде соответствующего триплета цели, у которого префикс и имя параметра организованы по тому же принципу, что у

триплета факта, а значение может быть представлено следующим способом:

Префикс.Имя = Заявка;

Зарезервированным символом для обозначения заявки служит двоеточие («:»).

Примеры триплетов цели:

- `$R.I=:;`
- `$P.KWO=:;`

Совокупность триплетов называют **триплексной строкой**.

1. Библиотека функций для работы с триплексными строками в ВСПТД

В данном разделе приведен готовый набор функций работы с триплексными строками, вынесенный в отдельный модуль, который подключается к проекту при возникновении необходимости работы с триплексными строками.

Название модуля: trplib.py

Зарезервированные символы

В приведенном ниже модуле определены следующие служебные символы:

- $ – начало триплета;
- ; – конец триплета;
- ' – начало и конец текстового значения;
- " – начало и конец комментария;
- . – разделитель префикса и имени триплета;
- = – знак присваивания;
- : – запрос.

Использование служебных символов в префиксе и имени триплетов не допускается; значение и комментарий могут содержать любые символы. При этом параметризация служебных символов (хранение их в виде отдельных переменных, а не в виде строго определенных констант) позволяет при необходимости произвести их замену на другие символы.

Функции работы с триплексными строками

Для удобства ориентирования по данному пособию все функции приведены в алфавитном порядке.

Примечание: для корректной работы всех функций, кроме функций проверки синтаксиса (что очевидно), требуется соблюдение синтаксиса ВСПТД.

Функция trpAddStr

Функция производит добавление триплета в триплексную строку по лексикографическому возрастанию префиксов и имен триплетов. Если

триплет с таким же префиксом и именем уже есть в строке, производится замена значения триплета на новое.

Объявление:

```
trpAddStr(trpStr, prefix, name, value, typeValue = 1, rem = "")
```

Параметры:

- `trpStr` – триплексная строка, в которую производится добавление;
- `prefix` – префикс добавляемого триплета;
- `name` – имя добавляемого триплета;
- `value` – значение добавляемого триплета;
- `typeValue` – тип значения триплета (необязательный параметр, по умолчанию принимает значение 1):
  1) `0` – числовое значение;
  2) `1` – текстовое значение (заключается в апострофы);
- `rem` – комментарий к добавляемому триплету (необязательный параметр, по умолчанию принимает значение "", т.е. отсутствие комментария).

Пример применения:

```
s = "$L.NM='Элемент обработки';$L.DSZ='45.6';";
s = trpAddStr(s, "L", "LENGTH", "120", None, 0);
```

Результат:

```
"$L.LENGTH='120';        $L.NM='Элемент        обработки';
$L.DSZ='45.6';"
```

Функция trpCheck

Функция осуществляет проверку синтаксиса триплексной строки.

Объявление:

```
trpCheck(trpStr)
```

Параметры:

- `trpStr` – проверяемая триплексная строка.

Возвращаемые значения:

Коды возвращаемых значений представлены в таблице 1.

Таблица 1 – Коды возвращаемых значений функции trpCheck

Код возвращаемого значения	Вид ошибки
0	Ошибок нет, синтаксис триплексной строки верен.
1	Между триплетами есть посторонние символы.
2	Попытка начать новый триплет при незакрытом предыдущем или отсутствие конца последнего триплета.
3	Отсутствие префикса триплета.
4	Некорректный префикс.

5	Не задано имя триплета.

Продолжение таблицы 1

6	Наличие недопустимых символов в имени триплета (или отсутствие знака присваивания).
7	Отсутствие закрывающего апострофа в значении триплета.
8	Строковое значение триплета не обрамлено в апострофы.
9	Наличие посторонних символов после значения триплета, не оформленных в комментарий.
10	Отсутствие закрывающей кавычки в комментарии.

Пример применения:

```
trpCheck("$W.IST='ГОСТ 166-73';$W.NM=Скоба 'h12';");
```

Результат:

```
8
```

Функция tripletCheck

Функция реализует проверку синтаксиса отдельного триплета. Данная функция является вспомогательной к функции trpCheck(trpStr) и реализует лишь часть алгоритма, поэтому при работе с ВСПТД рекомендуется использовать функцию trpCheck(trpStr).

Объявление:

```
tripletCheck(trpStr)
```

Параметры:

- `trpStr` – проверяемый триплет.

Возвращаемые значения:

Функция возвращает результат проверки триплета. Интерпретация возвращаемого значения аналогична интерпретации выходных данных функции trpCheck.

Функция trpDel

Функция удаляет триплеты с заданными префиксом и именем из триплексной строки.

Объявление:

```
trpDel(trpStr, prefix, name)
```

Параметры:

- `trpStr` – триплексная строка;
- `prefix` – префикс, по которому производится удаление;
- `name` – имя, по которому при соответствующем префиксе производится удаление.

Пример применения:

```
trpDel("$V.TXT='013000    тара    ';$W.IST='ГОСТ    166-73';$W.NM='Скоба 3,1 h12';", "W", "IST");
```

Результат:
```
$V.TXT='013000 тара ';$W.NM='Скоба 3,1 h12';
```

Функция trpDelPref

Функция удаляет из триплексной строки все триплеты с заданным префиксом.

Объявление:
```
rpDelPref(trpStr, prefix)
```
Параметры:

- `trpStr` – триплексная строка;
- `prefix` – префикс, по которому производится удаление.

Пример применения:
```
trpDelPref("$V.TXT='013000 тара ';  $W.IST='ГОСТ 166-
73'; $W.NM='Скоба 3,1 h12';", "W");
```
Результат:
```
$V.TXT='013000 тара ';
```

Функция trpGet

Функция осуществляет поиск триплета с заданными префиксом и именем в триплексной строке и возвращает значение этого триплета.

Объявление:
```
trpGet(trpStr, prefix, name)
```
Параметры:

- `trpStr` – строка, в которой производится поиск;
- `prefix` – префикс искомого триплета;
- `name` – имя искомого триплета.

Пример применения:
```
trpGet("$P.WO='Фрезеровать';$R.I=1.0;$R.NOB=1120.0;",
"R", "NOB")
```
Результат:
```
1120.0
```

Функция trpGetNametrpStr

Функция возвращает имя триплета.

Объявление:
```
trpGetName(trpStr)
```
Параметры:

- `trpStr` – триплексная строка.

Пример применения:
```
trpGetName("$N.TW=0.265;")
```
Результат:
```
TW
```

Функция trpGetObject

Функция возвращает список триплетов заданного объекта.

Объявление:

```
trpGet(trpStr, prefix)
```

Параметры:

- `trpStr` – триплексная строка;
- `prefix` – префикс искомого объеста.

Пример применения:

```
trpGetObject("$D.D=34;$D.L=45;$N.TO=0.073;$N.TW=0.265;
","D")
```

Результат:

```
$D.D=34;$D.L=45;
```

Функция trpGetPrefix

Функция возвращает префикс триплета.

Объявление:

```
trpGetPrefix(trpStr)
```

Параметры:

- `trpStr` – триплексная строка.

Пример применения:

```
trpGetPrefix("$N.TW=0.265;")
```

Результат:

```
N
```

Функция trpGetValue

Функция возвращает значение триплета.

Объявление:

```
trpGetValue(trpStr)
```

Параметры:

- `trpStr` – триплексная строка.

Пример применения:

```
trpGetValue("$N.TW=0.265;")
```

Результат:

```
0.265
```

Функция trpGoNext

Функция осуществляет переход к следующему триплету: возвращаемым значением является индекс начала следующего триплета. Поиск производится с заданного индекса. Если триплет, с которого начинается поиск, – последний в триплексной строке, функция возвращает длину строки.

Объявление:

```
trpGoNext(trpStr, beginIndex)
```

Параметры:

– `trpStr` – триплексная строка;
– `currentIndex` – индекс, с которого нужно начинать поиск.

Пример применения:
```
trpGoNext("$V.OB='013000';$V.TXT='013000 тара ';", 2)
```
Результат:
```
15
```

Функция trpGetVoc

Функция выбирает из словаря метаданных, представленному в XML-формате (см. приложение Б), по префиксу и имени реквизита его описание в триплексной форме.

Объявление:
```
trpGetVoc (prefix, name, baseVoc)
```
Параметры:
– `prefix` – префикс реквизита;
– `name` – имя реквизита;
– `baseVoc` – имя базы, содержащей словари метаданных;

Пример применения:
```
trpGetVoc ("D","L");
```
Результат:
```
$Q.NAME='D.L';$Q.NM='Длина';$Q.FRMT='9999V99';
```

Функция trpMergeStr

Функция реализует слияние двух триплексных строк и возвращает результат этого слияния.

Объявление:
```
trpMergeStr(trpStr, trpStrAdd)
```
Параметры:
– `trpStr` – основная триплексная строка;
– `trpStrAdd` – добавляемая триплексная строка.

Пример применения:
```
trpMergeStr("$L.K=99;", "$E.NM='Фреза 20';");
```
Результат:
```
$L.K=99;$E.NM='Фреза 20';
```

Функция trpSort

Функция сортирует триплексную строку в лексикографическом порядке имен и возвращает отсортированную триплексную строку. Эту функцию рекомендуется применять в том случае, когда триплексная строка сформирована нестандартными методами, например, с применением текстового редактора.

Объявление:
```
trpSort(trpStr)
```

Параметры:

– `trpStr` – триплексная строка.

Пример применения:

```
trpSort("$N.TO=0.073;$W.IST='ГОСТ 166-73';$L.OB='(1)';")
```

Результат:

```
$L.OB='(1)';$N.TO=0.073;$W.IST='ГОСТ 166-73';
```

Функция trpSplit

Функция разбивает триплексную строку на список триплетов.

Объявление:

```
trpSplit (trpStr)
```

Параметры:

– `trpStr` – триплексная строка.

Пример применения:

```
trpSplit("$N.TO=0.073;$W.IST='ГОСТ 166-73';
$L.OB='(1)';")
```

Результат:

```
['$N.TO=0.073;', "$W.IST='ГОСТ 166-73';",
"$L.OB='(1)';"]
```

Класс для работы с триплексными строками

Класс CVSATripletString создан для работы с триплексной строкой как с объектом (VSA – virtual string area – виртуальное строковое пространство). Такой подход значительно упрощает работу с данными, так как освобождает программиста от необходимости применения стандартных функций работы со строками языка Python и позволяет добиться большей степени абстракции. Данный класс реализует основные методы работы с триплексными строками.

Единственный атрибут класса – vsaString – представляет собой триплексную строку, над которой производятся все операции.

Рассмотрим методы описываемого класса:

– `GetPrefName(ParName, Pref, Name)` – получение префикса и имени триплета. Если триплексная строка содержит несколько триплетов, возвращаются префикс и имя первого;

– `DelParam(self, Prefix, Name)` – удаление триплета с заданными префиксом и именем;

– `SetParam(self, Prefix, Name, Value, TypeValue = 1, Rem = "")` – добавление триплета в триплексную строку;

– `GetParam(self, Prefix, Name)` – получение значения триплета с заданными префиксом и именем.

Рассмотрим пример практического применения приведенного выше класса в прикладной программе:

```
ts = CVSATripletString();
```

```
ts.SetParam("Q", "REZ", "Сверло"); # Добавим первый триплет.
ts.SetParam("Q", "MT", "Сталь"); # Добавим второй триплет.
trpStr = ts.vsaString;
```
Результат добавлений: $Q.REZ='Сверло';$Q.MT='Сталь';

```
print (trpStr);

Value = ts.GetParam("Q", "REZ"); # Запросим значение первого триплета.
print(Value); # Результат запроса: Сверло.

ts.DelParam("Q", "MT"); # Удалим второй триплет.
trpStr = ts.vsaString; # Результат удаления: $Q.REZ='Сверло';
print (trpStr);
```

2. Функции формирования анкет (фреймов)

В данном разделе рассматривается модуль формирования анкет (фреймов).

Назначение: процедура предназначена для динамического формирования анкеты при отображении и ввода параметров проектирования на основе их мнемонических имен. Данная процедура может быть вызвана из различных приложений при условии выполнения нижеописанных соглашений.

Методы организации

Процедура формирования анкеты texDialog имеет вид:

```
texDialog (texMode, forma, trpstr, config, BaseVoc, TableName)
```

Рассмотрим список параметров:

- `texMode` – метод формирования анкеты;
- `forma` – параметры для организации окна анкеты;
- `trpstr` – параметры для отображения в анкете;
- `config` – имя конфигурационного файла;
- `BaseVoc` – имя базы, содержащей словари метаданных;
- `TableName` – имя таблицы локального словаря метаданных. Применяется, когда имя определено не по правилам, иначе NULL.

Анкета типа «текст перехода»

Такая анкета формируется путем передачи в качестве параметра `texMode` (метод формирования анкеты) нулевого значения:

```
texMode = 0;
```

Этот вариант предназначен для ввода переменной информации в

логико-лингвистические фразы типа «текст перехода». В данной ситуации фразы могут иметь, например, такой вид:

```
"Зенкеровать " $L.KOL " отв., выдерживая размеры " $L.D " " $L.QV "(" $L.WD "|" $L.ND ")" <ПС $L.SH $L.OSSH> // $Q.ZAGA = "Текст перехода"; $Q.ZAGP = "Переменная информация"; $Q.W=600; $Q.X=100; $Q.Y=200; $Q.H=400;
```

Первая часть окна анкеты (до символа «<») формируется примерно так же, как она выглядит в параметрах, исключая кавычки. Слоты (триплеты) формируются в соответствии с форматом, взятым из словаря.

Заголовок окна формируется из значения параметра $Q.ZAGA. Заголовок переменной части формируется из параметра $Q.ZAGP. Параметры расположения и размеры окна формируются в соответствии с W (ширина окна), X (координата по оси абсцисс), Y (координата по оси ординат), H (высота окна).

Для совместимости с накопленными базами анкет введены параметры Q.WIDTH, Q.LEFT, Q.TOP, Q.NSTR. Они позволяют использовать DOS-размерность независимо от разрешения (перевод условных единиц в пиксели производятся анкетным модулем).

При нормальном завершении работы возвращается триплексная строка в виде:

```
$P1.NAME1=VALUE1; …; $Pn.NAMEn=VALUEn;
```

где $Pi.NAMEi=VALUEi; – триплет i-го поля, включая переменную информацию.

При некорректном завершении возвращается NULL.

Переходы в базе данных могут быть сформированы таким образом, что пользователю достаточно будет ввести минимум переменной информации.

Например, для хранящегося в таблице текста перехода:

```
"Шлифовать отв-е в разм. " $L.D " " $L.QV "(" $L.WD "|" $L.ND ")" //
```

Согласно тексту перехода, в режиме проектирования пользователю будет достаточно ввести диаметр элемента обработки ($L.D), квалитет на диаметр ($L.QV), верхнее отклонение на диаметр ($L.WD) и нижнее отклонение на диаметр ($L.ND).

Для того, чтобы данные $L.D и $L.QV были записаны через пробел, между ними необходимо ввести открывающие кавычки, пробел и закрывающие кавычки (" "); для ввода в текст перехода после переменной информации, например, скобки, следует ввести "(".

Текст перехода в графе P.TXT можно вводить в несколько строк; он считается законченным, если в конце его написания стоят две наклонные черты.

Анкета общего типа

Анкета общего типа задается следующим образом:

```
texMode = 1;
```

Этот вариант предназначен для ввода переменной информации с автоматической генерацией наименований полей.

```
$$T.FR $T.FP|Падун Б.С. $:T.FU $T.FN $T.LI // $Q.ZAGA = "Фамилии (нижний штамп)"; $Q.W=600; $Q.X=100; $Q.Y=200; $Q.H=400;
```

Значение используемых здесь символьных конструкций раскрывается в следующем пункте.

Анкета общего типа для индексированных полей

Этот вариант предназначен для ввода переменной информации для индексированных полей. Пример индексированных триплетов:

```
$E2.NM='Резец'; $E2.NM='Сверло'; $E4.NM='Фреза';
```

В этом случае алгоритм поиска характеристик из словаря сводится к следующим шагам:

Поиск по полному имени (включая цифры);

Если на предыдущем шаге ничего не найдено, то изъять хвостовые цифры из имени и повторить поиск.

Дополнительные условия:

- `$$T.FR` – символ «$» два раза подряд означает недоступность соответствующего поля для редактирования;
- `$T.FP | Сидоров В.С.` – значение по умолчанию, если его нет в строке параметров для отображения;
- `$:T.FU` – символ «:» после «$» означает необходимость обязательного заполнения соответствующего поля.

Любое поле может быть снабжено возможностью вызова функции, взятой из параметров конфигурационного файла.

Вычисление формул

Общие сведения

Один из видов представления технологических знаний – это некоторые выражения, которые включают в себя арифметические операции (сложение, вычитание, умножение, деление, возведение в степень), а также тригонометрические (синус, косинус и др.), алгебраические, трансцендентные и другие математические функции. Многочисленные литературные источники по технологии машиностроения и приборостроения указывают на то, что такого рода выражения содержатся почти в каждом виде работ, начиная от расчетов стойкости инструмента, режимов резания и кончая расчетом времени на личные надобности. Эти формулы носят как теоретический, так и эвристический характер. В процессе эксплуатации некоторые из них подвергаются корректировке, но будучи «зашитыми» в программы, не могут быть исправлены без участия программиста. В этих условиях формулы удобно хранить вне программ, обеспечивая возможность их корректировки силами эксперта в любой удобный для него момент. В

контексте применения ВСПТД разработано соответствующее представление формул, методы их хранения и способы реализации.

Фрейм-формула – арифметическое выражение, представленное в виде символьной строки, в которой значения переменных заданы в виде триплетов целей и постоянных величин (заданных коэффициентов).

Формула для расчета нормы расхода для плоских материалов с известным весом 1 кв. м:

$$N = \frac{B \cdot L \cdot K_1 \cdot K_2}{n \cdot 10^6},$$

где

- N – норма расхода;
- B – ширина листа;
- L – длина листа;
- K_1 – весовая характеристика материала;
- K_2 – коэффициент отходов в заготовительном производстве;
- n – количество деталей из заготовки.

Здесь в представлении ВСПТД

- параметру N соответствует имя реквизита `Z.NRM`;
- параметру B соответствует имя реквизита `Z.B`;
- параметру L соответствует имя реквизита `Z.L`;
- параметру K_1 соответствует имя реквизита `M.VH`;
- параметру K_2 соответствует имя реквизита `M.KTOT`;
- параметру n соответствует имя реквизита `Z.KD`.

Тогда формула в базе знаний ВСПТД будет иметь вид:

```
$Z.NRM=$Z.B*$Z.L*$M.VH*$M.KTOT/($Z.KD*10**6);
```

Алгоритм вычисления значения по формуле

При вычислении какого-либо значения с использованием фрейма-формулы применяется следующий алгоритм:

1) Выбор очередного триплета из формулы.
2) Выбор значения триплета из строки ВСПТД по его префиксу и имени:
   a. если в ВСПТД нет такого триплета, то получение его значения осуществляется по правилам МАС с привлечением агента-диспетчера или же с помощью ИПС или ЛПР;
   b. подстановка значения триплета в формулу.
3) Если формула еще не кончилась, переход к п.1.
4) Вычисление полученного выражения с применением методов обратной польской записи.

3. Библиотека селекции/модификации данных ВСПТД

Модуль предназначен для получения данных из таблиц ВСПТД, минуя процесс обращения к другим модулям в прикладной программе.

Организация интерфейса функций

Функция диалога выбора значений таблиц данных или их модификации

```
LoadList(BaseBtr, TblCode, TypeBox, ShowCol, CodeCol, Razd, MarkerString, Res);
```

Параметры функции:

- `BaseBtr` – имя файла данных;
- `TblCode` – имя таблицы для селекции/редактирования;
- `TypeBox` – тип селекции:
  1) Однострочный выбор (`TypeBox = 0`);
  2) Многострочный выбор (`TypeBox = 1`);
- `ShowCol` – номер поля таблицы для отображения (0 – отобразить все поля таблицы);
- `CodeCol` – номер поля для взятия значения (0 – сформировать триплексную строку по выбранной записи таблицы);
- `Razd` – разделитель значений для многострочного выбора, в случае однострочного выбора Razd = ‘N’;
- `MarkerString` – строка маркировки схожей строки в таблице диалога;
- `Res` – результат отработки функции.

Функция диалога выбора значений таблиц данных или их модификации с применением модуля работы с таблицами

```
LoadDBList(Table, X, Y, W, H, Edit, key);
```

Параметры функции:

- `Table` – таблица для отображения в диалоге;
- `X`, `Y`, `W`, `H` – размерности и положение диалога;
- `Edit` – флаг редактирования (0 – только чтение);
- `key` – возвращаемый код нажатой клавиши.

4. Представление технологической информации в САПР ТП механической обработки заготовок

Система может получать данные (и их сочетания) с помощью следующих методов:

- анкетный ввод данных;
- поиск данных;
- поиск данных с пересчетом;
- расчет данных;
- выбор данных из единичной таблицы.

Данные о технологическом процессе размещены в файле с именем VSP<регистрационный номер>.tpp, где «регистрационный номер» – это номер технологического процесса в общем каталоге системы.

Пример: vsp0001.tpp – информация о техпроцессе № 1 по каталогу (см. рисунок 1).

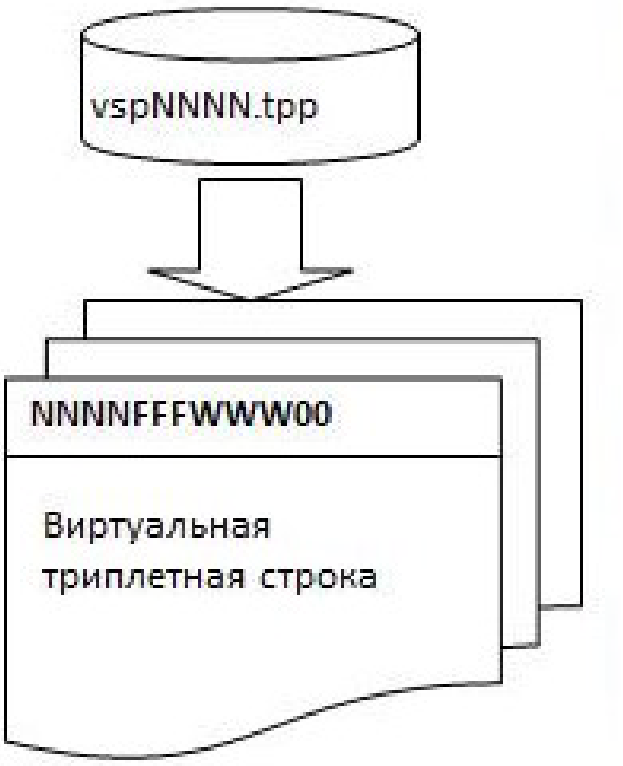


Рисунок 1 – Представление данных в ВСПТД

Файл с информацией по технологическому процессу имеет древовидную структуру. Каждая вершина этого дерева описывается с помощью уникальных ключей. В вершинах находятся данные в виде триплетов.

Структура ключей имеет следующий вид:

- NNNNFFFWWW00 – ключ хранения информации о переходе, где:
  1) NNNN – номер операции в техпроцессе;
  2) FFF – номер перехода в данной операции;
  3) WWW – код вида обработки в соответствии с базой данных переходов;
- NNNNFFFWWW02 – ключ хранения текста перехода после введения переменной информации;
- 000000000000 – ключ хранения общей информации о технологическом процессе;
- NNNN00000000 – ключ хранения сведений об операции, где:
  1) NNNN – номер операции в техпроцессе, например, 001000000000 – по десятой операции.

Система управления записями ВСПТД

При первом практическом применении методов ВСПТД СУЗ

Btrieve[1][1]. Однако, данная СУЗ не нашла распространения на рынке программных средств в России. В настоящее время имеется опыт программирования ВСПТД с применением современных реляционных СУБД. В таких случаях виртуальная строка представляется в реляционной таблице как поле переменной длины. Остальные поля таблицы отвечают за первичный ключ. Вопросы доступа к виртуальным строкам решаются средствами применяемой СУБД (см. рисунок 2).

Здесь графа `$K.OP` – ключ записи, а `$K.TXTX` – содержимое строки ВСПТД. Следует иметь в виду, что максимальная длина записи ограничивается характеристиками применяемой СУБД.

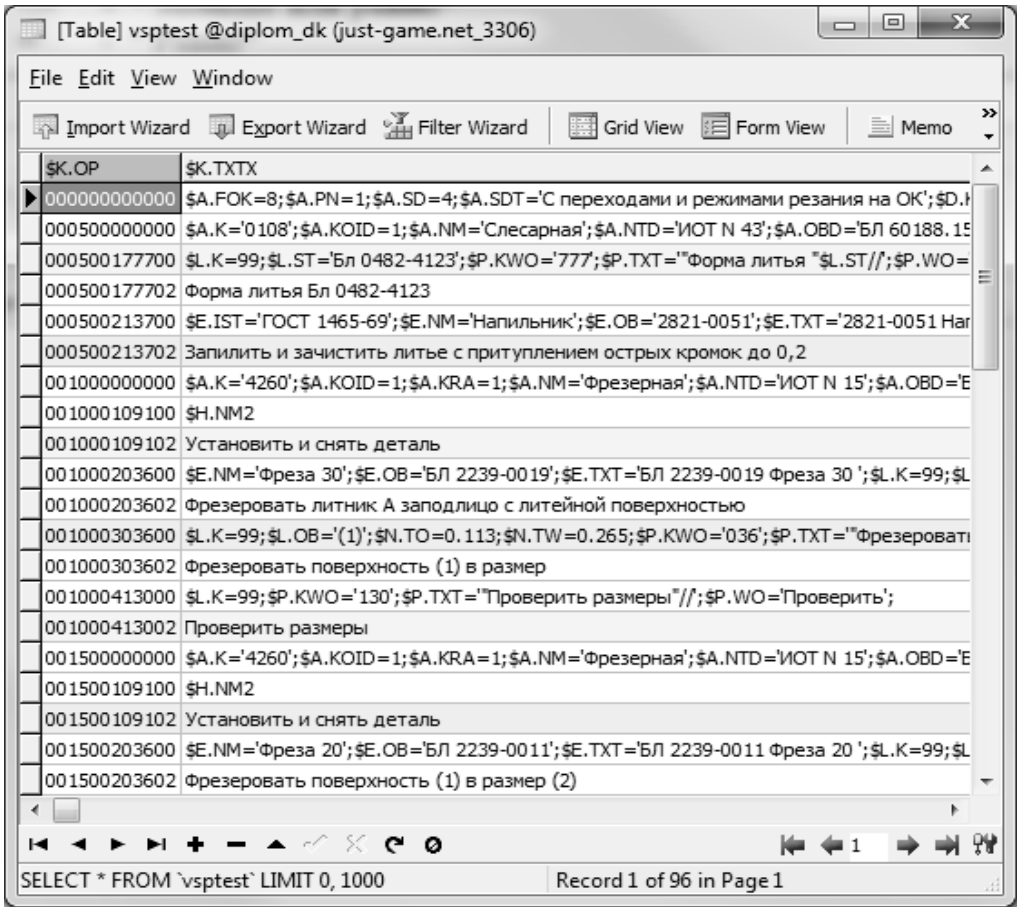


Рисунок 2 – Пример представления виртуальных строк

5. Методика формирования отчетов по данным, хранимым в ВСПТД

Общие сведения

Эффективность, простота и удобство работы с текстовыми и табличными данными присущи приложениям семейства MS Office. Приложения Office – универсальные продукты, которые могут использоваться в различных предметных областях.

Программное обеспечение, предназначенное для работы в области

[1] СУЗ Btrieve поставлялась фирмой Novell в составе сетевой операционной системы NetWare.

технологии приборостроения, напротив достаточно специфично и позволяет получать на выходе документы, как правило, пригодные для применения только внутри самой системы и ей подобных из данной области.

Мало кто будет спорить с тем фактом, что отчет, представленный в приложении MS Word будет более предпочтительным, нежели его аналог, построенный средствами самого приложения специфичной предметной области, в частности технологии приборостроения.

В связи с этим, очень привлекательно выглядит возможность комбинирования получения данных путем использования всей специфики приложения, нацеленного на некоторую предметную область, с простым и удобным оформлением этих данных в приложениях Office.

Далее речь идет о представлении данных ВСПТД в текстовом документе Word. Данные могут быть табличными, числовыми, текстовыми, т.е. любыми из возможных типов данных, представленных в ВСПТД. Представление графических элементов не рассматривается в данной пособии, но сама возможность их внедрения в документ реальна и может быть реализована.

Рассматриваемый программный модуль позволяет формировать отчеты на базе табличных данных и триплетов, представленных в ВСПТД. В качестве приложения по формированию отчетов выбрано приложение Microsoft Word из пакета Microsoft Office.

В библиотеке представлен базовый набор функций, позволяющий автоматизировать процесс создания документов MS Word на заранее подготовленных шаблонах.

Совместимость: MS Word 2000 и выше.

Что описывается далее в разделе:

1) объектная модель документа;
2) принцип формирования шаблона по принятым в ВСПТД соглашениям;
3) программный интерфейс модуля работы с Word и сопряжение с программой на языке Python.

Принцип работы с отчетами

Таблицы, хранимые в ВСПТД, могут быть быстро и эффективно подготовлены к печати при использовании технологии автоматизации MS Word OLE Automation. Причем детали применения технологии автоматизации скрыты в самом программном модуле, а пользователю (программисту) предлагается лишь небольшой набор функций.

Пользователь сам определяет желаемый набор полей таблицы, указывая его в документе Word. Документ сохраняется и в дальнейшем может быть использован как шаблон при формировании отчета.

В качестве слота, в который в процессе получения отчета подставляются реальные данные из ВСПТД, используется объект документа Word «поле».

Хотя в основе лежит объектная модель приложения Office (Word), пользователю не обязательно вдаваться в подробности ее реализации. Достаточно иметь лишь общие представления о работе с полями.

Поле в понятии Word – вычисляемый элемент, который вставлен в документ.

Поле содержит определенный код и может иметь несколько параметров документа.

Обзор типов полей можно получить в диалоговом окне, выбрав в меню Word пункт «Вставка/Поле».

Объект поля в контексте ВСПТД используется для хранения имени триплета. Указанное в поле наименование служит ключом для извлечения данных из ВСПТД и их установки на место слота в документе. После подстановки значения само поле удаляется.

Преимущество такого подхода заключается, во-первых, в увеличении скорости прохода по тексту, т.к. перебирается массив объектов модели Word, а не весь текст. Во-вторых, если слот остался нетронутым (например, нет данных по некоторому триплету в ВСПТД), он никак не отображается в документе и не выводится на печать.

Абстрактная объектная модель документа Word

Данная схема (рисунок 3) позволяет увидеть объект поля в составе объектной модели Word.

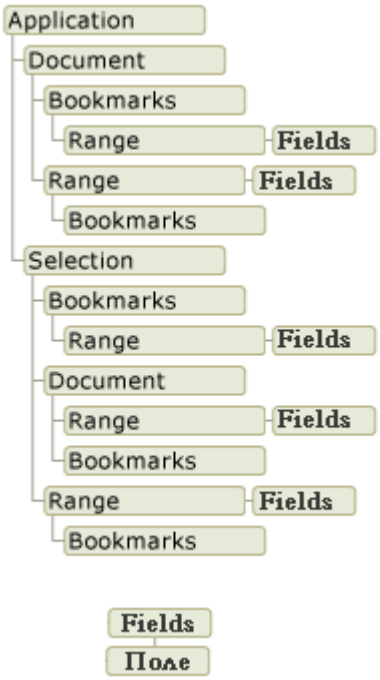


Рисунок 3 – Объектная модель документа Word

На приведенной схеме (Рисунок 3):

- Application – объект самого приложения Word;
- Document – документ, с которым мы работаем;
- Bookmarks – объект закладок;
- Selection – объект, определяющий весь набор подобъектов,

попавших в область выделения;
- Range – собственно текст;
- Fields – объект, определяющий массив полей в составе текущего документа Document;
- В Fields, в виде его элементов, находятся те самые поля, о которых мы говорим.

Объектная модель документа, как и положено любой объектной модели, предлагает ряд свойств и методов для каждого объекта. Как было упомянуто выше, программист, работающий с ВСПТД, не имеет необходимости знакомиться с методами и свойствами этих объектов.

Принцип формирования документа-шаблона для отчетов в ВСПТД

Разработаны определенные правила представления слотов в документе, выполняющем роль шаблона при формировании отчета.

Шаблон, прежде всего, служит для разграничения самих технологических данных и их оформления в конечном документе.

Напомним, что поле в документе Word с настройками по умолчанию скрыто.

Для удобной работы с полями желательно включить отображение полей. Это можно сделать двумя способами:

1) В меню «Сервис/Параметры», на вкладке «Вид» в категории «показывать» устанавливаем галочку «Коды полей». При этом принятые настройки глобальны, т.е. при открытии любого документа коды полей будут отображаться.
2) Выделяем тот фрагмент текста, с которым планируем работать, и в контекстном меню выбираем пункт «коды/значения полей». В результате этого действия будут отображены/скрыты коды полей. Результат применяется только к текущему документу в данном сеансе работе.

Как добавить новое поле в документ. В меню вставка/поле выбираем категория «автоматизация», поле DocVariable, в окне ввода «новое имя» вводим имя триплета, например, «$Q.NAME», в результате получим:

{ DOCVARIABLE $Q.VOC * MERGEFORMAT }

Удалим ненужные наименования и получим:

{$Q.VOC}

Так представляется триплет-слот в документе-шаблоне в составе поля. Любое поле в документе заключено между символами {}.

Приняты следующие соглашения:

1) Имена, начинающиеся со знака «$» соответсвуют одноименному триплету в ВСПТД.

   Примеры наименований: {$Q.VAL} – будет подставлено значение триплета Q.VAL из ВСПТД.

2) Имена, начинающиеся со знака «#$» соответсвуют имени поля в таблице ВСПТД, для которого делается отчет. В отличие от первого случая, данное поле обязательно располагается в ячейке таблицы. Это связано, прежде всего, с тем, что подстановка соотвествует значению столбца таблицы в целом, а не конкретному значению. Т.е. в данном случае идет подстановка значений всех строк таблицы, по указанному столбцу.
Примеры наименований: {#$Q.LENGHT} – будет подставлен столбец с именем Q.LENGHT текущей таблицы.

3) Имена, начинающиеся со знака «$$» соответствуют наименованию графы (т.е., названию столбца). На место данного слота подставляется буквенное обозначение (заголовок) указанной графы.
Примеры наименований: {$$Q.D} – будет подставлено название столбца с именем Q.D текущей таблицы, например, результатом может быть слово «Диаметр».

4) Возможны специальные управляющие имена, начинающиеся на «$Setup.».
Пример: {$Setup.OrientationLandscape} – при наличии такого поля в документе будет применен альбомный тип листа.

Пример шаблона

Следуя рассмотренным правилам, можно составить шаблон для технологического документа.

Полученный шаблон сохраним под именем example.doc

Сведения о режущем инструменте **({$INSTR.NAME})** в цехе № **{$CEH.NUMBER}**

№ п/п	**{$$Q.NAME}**	**{$$Q.D}**	**{$$Q.LENGTH}**
{#$Q.N}	**{#$Q.NAME}**	**{#$Q.D}**	**{#$Q.LENGTH}**

Ответственный за цех **{$WORKER.NAME}**

Возможный результат обработки этого шаблона:

Сведения о режущем инструменте (сверло) в цехе № 38

№ п/п	Наименование	Диаметр	Длина
1	Правое, шлифованое	0.2-0.35	19

2	Правое, шлифованое	0.4-0.45	20
3	Правое, шлифованое	0.50	22
4	Правое	0.55-0.60	24
5	Правое, шлифованое	0.65	26
6	Правое, шлифованое	0.70-0.75	28
7	Правое, шлифованое	2.75-3.0	61
8	Левое, шлифованое	5.0-5.1	86
9	Правое, шлифованое	9.0	125
10	Правое, шлифованое	10.0	133
11	Правое	17.5	191
12	Правое, шлифованое	20.0	205

Ответственный за цех Комаров А.Н.

6. Дедуктивная машина вывода (ДМВ) в ВСПТД

Описание синтаксиса языка ДМВ

Для описания синтаксиса языка ДМВ использована расширенная форма Бэкуса-Наура.

Типы данных:

- ВещЧисло = [“+”|“-”] НатЧисло [“.”[НатЧисло]]
- [(“e”|”E”) [“+”|“-”]НатЧисло].
- НатЧисло = Цифра {Цифра}.
- Строка = “‘” {ЛатСимвол|КирСимвол|Цифра|Знак} “’”.
- Цифра = “0”|“1”|“2”|“3”|“4”|“5”|“6”|“7”| “8”|“9”.
- ЛатСимвол = “a”|“b”|“c”|“d”|“e”|“f”|“g”|“h”|“i”|“j”|“k”|“l”|“m”|“n”| “o”|“p”|“q”|“r”|“c”|“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”|“C”|“T”|“U”|“V”|“W”|“X”|“Y”|“Z”.
- КирСимвол = “а”|“б”|“в”|“г”|“д”|“е”|“ё”|“ж”|“з”|“и”|“й”|“к”|“л”|“м”| “н”|“о”|“п”|“р”|“с”|“т”|“у”|“ф”|“х”|“ц”|“ч”|“ш”|“щ”|“Ъ”| “ы”|“ь”|“э”|“ю”|“я”| “А”|“Б”|“В”|“Г”|“Д”|“Е”|“Ё”|“Ж”|“З”|“И”|“Й”|“К”|“Л”|“М”| “Н”|“О”|“П”|“Р”|“С”|“Т”|“У”|“Ф”|“Х”|“Ц”|“Ч”|“Ш”|“Щ”|“Ъ”| “Ы”|“Ь”|“Э”|“Ю”|“Я”.

- Кортеж = “{” Элемент {“,” Элемент} “}”.
- Элемент = ВещЧисло|Строка|Переменная|Константа|
- Промежуток|ВероятПараметр.
- Промежуток = (“[”|“(”)ВещЧисло “;” ВещЧисло(“[”|“(”).
- ВероятПараметр = (ВещЧисло|Строка)“@”[“+”|“-”] НатЧисло.

Пример применения:

Кортежи:

$X.A = {[5; 11), (14; 23.6), 30} – аналогично:
($X.A >= 5 И $X.A < 11) ИЛИ ($X.A > 14 И $X.A < 23.6)
$X.A = {‘Токарный’, ‘Фрезерный’, ‘Сверлильный’} – аналогично:
$X.A =‘Токарный’ ИЛИ $X.A=‘Фрезерный’ ИЛИ $X.A=‘Сверлильный’
$X.A = {8@100, 7@70, 2.4@-100}

Переменные:

Переменная = “$” Префикс “.” ИмяПеременной.
Префикс = ЛатСимвол {ЛатСимвол}.
ИмяПеременной = ЛатСимвол {ЛатСимвол|Цифра|“_”}.

Константы:

Константа = “$” ИмяКонстанты.
ИмяКонстанты = ЛатСимвол {ЛатСимвол|Цифра|“_”}.
Все константы хранятся в файле `config.xml`

Операции:

МатОпер = “+”|“-”|“*”|“/”|“^”|“%”.
ЛогОпер = “И”|“ИЛИ”.
Отношения = “>”|“<”|“=”|“>=”|“<=”|“<>”.
ОперПрисвоения = “:=”.
ОперПеречисления = “,”.

Функции:

Функция= ИмяФункции“(”Аргумент{ОперПеречисления Аргумент}“)”.
ИмяФункции = ЛатСимвол {ЛатСимвол}
Аргумент = ВещЧисло|Строка|Кортеж|Переменная|Константа|Функция.
Функции, реализованные в ДМВ, представлены в таблице 2.

Таблица 2 – Функции, реализованные в ДМВ

Функция	Краткое описание
Тригонометрические функции	
SIN(x)	Синус x
COS(x)	Косинус x
TAN(x)	Тангенс x
ACOS(x)	Арккосинус x
ATAN(x)	Арктангенс x
SINH(x)	Гиперболический синус x

COSH(x)	Гиперболический косинус x
TANH(x)	Гиперболический тангенс x
Алгебраические и арифметические функции	
EXP(x)	Экспонента x
ABS(x)	Модуль x
LN(x)	Натуральный логарифм x
LOG(x)	Десятичный логарифм x
SQRT(x)	Квадратный корень x
MAX(a, b)	Минимальное из a и b
MIN(a, b)	Максимальное из a и b
Строковые функции	
STRCAT(a, b)	Конкатенация (сложение) строк a и b
Логические функции	
НЕ(x)	Логическое отрицание x
Функции работы с базами данных	
SQL(dbn, sql)	Обработка запроса sql в базе данных dbn
Функции редуцирования баз знаний	
REDUCE(x)	Выбрать из базы знаний правила, заданные условием x
Предикаты	
НЕТ(x)	Отсутствие параметра x в контексте задачи
ЕСТЬ(x)	Наличие параметра x в контексте задачи

Структура логического правила:

Правило = “ЕСЛИ” Условие “ТО” Действие {ОперПеречисления Действие}.
Условие = ЛогВыражение{ЛогОпер(ЛогВыражение|“(” Условие “)”)}.
Действие = Переменная ОперПрисвоения Формула.
ЛогВыражение = Формула Отношение Формула.
Формула = Компонент {МатОпер (Элемент|“(” Формула “)”)}.
Компонент = ВещЧисло | Строка | Кортеж | Переменная | Константа | Функция.

Пример применения:

Правило
ЕСЛИ ($L.WOB=25 ИЛИ $L.WOB=27) И НЕТ($A.TT) ТО $E.KTS:='21'
Вербальная интерпретация

ЕСЛИ вид_обработки рассверлить ИЛИ сверлить И НЕ заданы технические требования ТО сверло общего назначения.

Приложения

Приложение А

Библиотека trplib.py

```
# -*- coding: utf-8 -*-

# Зарезервированные символы ВСПТД:
trpBeg = '$'; # Начало триплета.
trpEnd = ';'; # Конец триплета.
trpAps = '\''; # Начало и конец текстового значения.
trpKv = '\"'; # Начало и конец комментария.
trpPoint = '.'; # Точка, разделяющая префикс и имя триплета.
trpEqual = '='; # Знак присваивания.
trpRequest = ':'; # Запрос.

systemSymbols = [trpBeg, trpEnd, trpAps, trpKv, trpPoint,
trpEqual, trpRequest];

def trpAddStr(trpStr, prefix, name, value, typeValue = 1, rem =
""):
 '''
 Добавляет триплет в триплетную строку по лексикографическому
возрастанию префиксов триплетов.

 Параметры:
 trpStr - триплексная строка, в которую производится добавление;
 prefix - префикс добавляемого триплета;
 name - имя добавляемого триплета;
 value - значение добавляемого триплета;
 typeValue - тип значения триплета:
 0 - числовое значение;
 1 - текстовое значение (заключается в апострофы);
 rem - комментарий к добавляемому триплету.
 '''
 if (typeValue == 0):
 value = int(value);
 else:
 value = str(value);

 prefix = str(prefix);
 name = str(name);
 rem = str(rem);

 currentIndex = 0; # Текущий индекс при проходе по триплексной
строке.
 lenStr = len(trpStr); # Длина триплексной строки.

 # Формирование добавляемого триплета.
 triplet = trpBeg + prefix + trpPoint + name + trpEqual;
 if (typeValue == 0):
 triplet += str(value);
```

```
 else:
 triplet += trpAps + value + trpAps;
 if (rem != ""): triplet += trpKv + rem + trpKv;
 triplet += trpEnd;

 while(currentIndex < lenStr):

 tmp = trpStr[currentIndex:]; # Часть строки, начинающаяся с текущего индекса.

 # Получение префикса и имени текущего триплета.
 currentPrefix = trpGetPrefix(tmp);
 currentName = trpGetName(tmp);

 # Вставка триплета.
 if (currentPrefix >= prefix):
 # 1. Вставка с заменой (если триплет с такими же префиксом и именем уже есть в исходной строке).
 if (currentPrefix == prefix and currentName == name):
 lenOld = trpStr[currentIndex:].index(trpEnd) + 1; # Длина заменяемого триплета.
 res = trpStr[:currentIndex] + triplet + trpStr[currentIndex + lenOld:];
 return res;

 # 2. Вставка с добавлением (если такого же трилета в исходной строке нет).
 res = trpStr[:currentIndex] + triplet + trpStr[currentIndex:];
 return res;

 currentIndex = trpGoNext(trpStr, currentIndex + 1); # Переход к следующему триплету.

 res = trpStr + triplet; # Добавление триплета в конец строки.
 return res;

def trpCheck(trpStr):
 '''
 Проверка корректности триплексной строки.

 Коды ошибок:
 1 - между триплетами есть посторонние символы;
 2 - попытка начать новый триплет при незакрытом предыдущем или отсутствие конца последнего триплета;
 3 - отсутствие префикса триплета;
 4 - некорректный префикс;
 5 - не задано имя триплета;
 6 - наличие недопустимых символов в имени триплета (или отсутствие знака присваивания);
 7 - отсутствие закрывающего апострофа в значении триплета;
 8 - строковое значение триплета не обрамлено в апострофы;
 9 - наличие посторонних символов после значения триплета, не оформленных в комментарий;
```

```
 10 - отсутствие закрывающей кавычки в комментарии (или есть посторонние символы после комментария).

 Параметры:
 trpStr - проверяемая строка.
 '''
 # ПЕРВЫЙ ЭТАП ПРОВЕРКИ: можно ли строку корректно разбить на триплеты?

 inTrp = 0; # Увеличивается на 1 при входе в триплет и уменьшается на 1 при выходе из него.

 trpStr = str(trpStr);
 lenStr = len(trpStr); # Количество символов в строке.

 i = 0;
 while (i < lenStr):

 if (trpStr[i] == trpBeg): inTrp += 1;
 if (trpStr[i] == trpEnd): inTrp -= 1;

 if (inTrp == 0):
 if (trpStr[i] not in [trpEnd, " ", "\n", "\0"]):
 return 1; # Ошибка: посторонние символы между триплетами.
 else:
 if (trpStr[i] != trpEnd):
 trpStr = trpStr[:i] + trpStr[i+1:];
 lenStr -= 1;
 i -= 1;

 if (abs(inTrp) > 1 or (inTrp != 0 and i == lenStr - 1)):
 return 2; # Ошибка: несоответствие начала и конца триплета (начало нового триплета при незакрытом предыдущем).

 i += 1;

 triplets = trpSplit(trpStr); # Разбиение строки на отдельные триплеты.
 kolTrp = len(triplets);

 for i in range(kolTrp):
 # ВТОРОЙ ЭТАП ПРОВЕРКИ: проверка синтаксиса каждого триплета.
 isTrpRegular = tripletCheck(triplets[i]);
 if (isTrpRegular != 0):
 return isTrpRegular;

 return 0;

def tripletCheck(trpStr):
 '''
 Проверка триплета.

 Параметры:
```

```
 trpStr - проверяемый триплет.
 '''
 trpStr = str(trpStr);

 # Проверка префикса.
 i = 1;
 while ((trpStr[i] not in systemSymbols) and (trpStr[i] != "
")): i += 1;
 if (i == 1):
 if (trpStr[i] == trpPoint):
 return 3; # Ошибка: отсутствие префикса триплета.
 if (trpStr[i] == trpEqual):
 return 5; # Ошибка: не задано имя триплета.
 if (trpStr[i] != trpPoint):
 return 4; # Ошибка: некорректный префикс.

 # Проверка имени.
 i += 1;
 while ((trpStr[i] not in systemSymbols) and (trpStr[i] != "
")): i += 1;
 if (trpStr[i-1] == trpPoint):
 return 5; # Ошибка: не задано имя триплета.
 if (trpStr[i] == " "):
 while (trpStr[i] == " "): i += 1;
 if (trpStr[i] != trpEqual):
 return 6; # Ошибка: наличие недопустимых символов в имени
триплета (или отсутствие знака присваивания).
 while (trpStr[i] not in systemSymbols): i += 1;
 if (trpStr[i] != trpEqual):
 return 6; # Ошибка: наличие недопустимых символов в имени
триплета (или отсутствие знака присваивания).

 # Проверка значения.
 i += 1;
 while (trpStr[i] == " "): i += 1;
 value = "";
 if (trpStr[i] == trpAps): # Если триплет имеет строковое
значение.
 i += 1;
 while ((trpStr[i] != trpAps) and (i != len(trpStr) - 1)):
 i += 1;
 if (i == len(trpStr) - 1):
 return 7; # Ошибка: отсутствие закрывающего апострофа в
значении триплета.
 i += 1;
 else:
 if (trpStr[i] == trpRequest): # Если значение триплета
представляет собой запрос.
 i += 1;
 else: # Если триплет имеет численное значение.
 while ((trpStr[i] != trpKv) and (i != len(trpStr) - 1)):
 value += trpStr[i];
 i += 1;
```

```
 try:
 value = float(value);
 except ValueError:
 return 8; # Ошибка: строковое значение триплета не обрамлено в апострофы.

 # Проверка комментария.
 if (i >= len(trpStr) - 2): return 0;

 while (trpStr[i] == " "): i += 1;
 if (trpStr[i] != trpKv and trpStr[i] != trpEnd): return 9; # Ошибка: наличие посторонних символов после значения триплета, не оформленных в комментарий.
 if (trpStr[-2] != trpKv): return 10; # Ошибка: отсутствие закрывающей кавычки в комментарии (или есть посторонние символы после комментария).
 if (trpStr[i+1:-2].count(trpKv) != 0): return 9; # Ошибка: наличие посторонних символов после значения триплета, не оформленных в комментарий.

 return 0;

def trpDel(trpStr, prefix, name):
 '''
 Удаляет все триплеты с заданными префиксом и именем из триплексной строки.

 Параметры:
 trpStr - триплексная строка;
 prefix - префикс, по которому производится удаление;
 name - имя, по которому при соответствующем префиксе производится удаление.
 '''
 triplets = trpSplit(trpStr);
 kolTrp = len(triplets);
 result = "";

 for i in range(kolTrp):
 if ((trpGetPrefix(triplets[i]) != prefix) or (trpGetName(triplets[i]) != name)): result += triplets[i];

 return result;

def trpDelPref(trpStr, prefix):
 '''
 Удаляет все триплеты с заданным префиксом из триплексной строки.

 Параметры:
 trpStr - триплексная строка;
 prefix - префикс, по которому производится удаление.
 '''
 triplets = trpSplit(trpStr);
```

```
 kolTrp = len(triplets);
 result = "";

 for i in range(kolTrp):
 if   (trpGetPrefix(triplets[i])   !=   prefix):   result   +=
triplets[i];

 return result;

def trpGet(trpStr, prefix, name):
 '''
 Возвращает значение триплета с заданными префиксом и именем.

 Параметры:
 trpStr - строка, в которой производится поиск;
 prefix - префикс искомого триплета;
 name - имя искомого триплета.
 '''
 triplets = trpSplit(trpStr);
 kolTrp = len(triplets);

 for i in range(kolTrp):
 if     ((trpGetPrefix(triplets[i])     ==     prefix)     and
(trpGetName(triplets[i]) == name)):
 return trpGetValue(triplets[i]);

def trpGetName(trpStr):
 '''
 Возвращает имя триплета в соответствии с синтаксисом:
 $<префикс>.<имя>=<значение>["Комментарий"];

 Параметры:
 trpStr - триплексная строка.
 '''
 result     =     trpStr[trpStr.index(trpPoint)     +     1     :
trpStr.index(trpEqual)];
 result = result.strip();
 return result;

def trpGetPrefix(trpStr):
 '''
 Возвращает префикс триплета в соответствии с синтаксисом:
 $<префикс>.<имя>=<значение>["Комментарий"];

 Параметры:
 trpStr - триплексная строка.
 '''
 result     =     trpStr[trpStr.find(trpBeg)     +     1     :
trpStr.index(trpPoint)];
 return result;

def trpGetValue(trpStr):
 '''
```

```
 Возвращает значение триплета в соответствии с синтаксисом:
 $<префикс>.<имя>=<значение>["Комментарий"];

 Параметры:
 trpStr - триплексная строка.
 '''
 result = trpStr[trpStr.index(trpEqual) + 1 :
trpStr.find(trpKv)];
 result = result.strip();
 if (result[0] == trpAps): result = result[1 : -1];
 return result;


def trpGoNext(trpStr, beginIndex):
 '''
 Возвращает индекс начала следующего триплета, осуществляя поиск
с заданного индекса.

 Параметры:
 trpStr - триплексная строка;
 currentIndex - индекс, с которого нужно начинать поиск.
 '''
 index = beginIndex;
 lenStr = len(trpStr);

 while (index < lenStr and trpStr[index] != trpBeg): index += 1;
 print(lenStr);

 return index;

def trpMergeStr(trpStr, trpStrAdd):
 '''
 Сливает две триплексных строки. Возвращает результат слияния.

 Параметры:
 trpStr - основная триплексная строка;
 trpStrAdd - добавляемая триплексная строка.
 '''
 return trpStr.strip() + trpStrAdd.strip();

def trpSort(trpStr):
 '''
 Сортирует триплексную строку в лексикографическом порядке имен.
Возвращает отсортированную триплексную строку.

 Параметры:
 trpStr - триплексная строка.
 '''
 triplets = trpSplit(trpStr);
 kolTrp = len(triplets);
 for i in range(kolTrp):
 triplets[i] = triplets[i][1:]; # Удаление символа $ в
триплетах.
```

```
 triplets.sort(); # Сортировка строки.

 result = "";

 for i in range(kolTrp): # Генерация результирующей строки.
 result += trpBeg + str(triplets[i]);

 return result;

def trpSplit(trpStr):
 '''
 Разбивает триплексную строку на триплеты. Возвращает список триплетов.

 Параметры:
 trpStr - триплексная строка.
 '''
 arr = trpStr.split(trpEnd);
 kolTrp = len(arr); # Количество триплетов в исходной строке.

 for i in range(kolTrp):
 try:
 arr[i] = arr[i].strip();
 arr[i] = arr[i][1:];
 except IndexError:
 break;
 if (len(arr[i]) == 0):
 del arr[i];
 else:
 arr[i] = trpBeg + arr[i] + trpEnd;

 return arr;
```

Приложение Б

Фрагмент словаря метаданных. Список реквизитов детали

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<DATAPACKET Version="2.0">
 <METADATA>
 <FIELDS>
 <FIELD    attrname="Q.NAME"    fieldtype="string"    WIDTH="-1"
DisplayLabel="Имя_реквизита" />
 <FIELD    attrname="Q.FRMT"    fieldtype="string"    WIDTH="-1"
DisplayLabel="Формат" />
 <FIELD     attrname="Q.NM"     fieldtype="string"     WIDTH="-1"
DisplayLabel="Наименование" />
 <FIELD     attrname="Q.K"     fieldtype="string"     WIDTH="-1"
DisplayLabel="Код_реквизита" />
 </FIELDS>
 <PARAMS      CAPTION_TABLE="Список      реквизитов      детали"
AdditionChar="$A.BLD='1';//" />
 </METADATA>
 <ROWDATA>
 <ROW  RowState="4"  Q.NAME="D.D"  Q.FRMT="9999V99"  Q.NM="Диаметр"
Q.K="" />
 <ROW  RowState="4"  Q.NAME="D.L"  Q.FRMT="9999V99"  Q.NM="Длина"
Q.K="" />
 <ROW  RowState="4"  Q.NAME="D.LKD"  Q.FRMT="9"  Q.NM="Отношение  L
детали к ее D" Q.K="" />
 <ROW RowState="4" Q.NAME="D.MAS" Q.FRMT="99999V999" Q.NM="Масса
детали" Q.K="" />
 <ROW       RowState="4"       Q.NAME="dsa"       Q.FRMT="X(20)"
Q.NM="Наименование детали" Q.K="" />
 <ROW  RowState="4"  Q.NAME="D.NM1"  Q.FRMT="X(20)"  Q.NM=""  Q.K=""
/>
 <ROW  RowState="4"  Q.NAME="D.NM2"  Q.FRMT="X(20)"  Q.NM=""  Q.K=""
/>
 <ROW       RowState="4"       Q.NAME="D.OB"       Q.FRMT="X(24)"
Q.NM="Обозначение детали" Q.K="" />
 <ROW  RowState="4"  Q.NAME="D.OSSH"  Q.FRMT="X(2)"  Q.NM="Обозн.
шкалы шерох." Q.K="" />
 <ROW  RowState="4"  Q.NAME="D.OST"  Q.FRMT="X(5)"  Q.NM="Обозн.
шкалы тверд." Q.K="" />
 <ROW RowState="4" Q.NAME="D.SH" Q.FRMT="999V999" Q.NM="Значение
шерох." Q.K="" />
 <ROW  RowState="4"  Q.NAME="D.TW1"  Q.FRMT="9999"  Q.NM="Твердость
миним." Q.K="" />
 <ROW  RowState="4"  Q.NAME="D.TW2"  Q.FRMT="9999"  Q.NM="Твердость
максим." Q.K="" />
 <ROW  RowState="4"  Q.NAME="D.VI"  Q.FRMT="9999"  Q.NM="Вылет"
Q.K="" />
 </ROWDATA>
</DATAPACKET>
```

Приложение В

Схема взаимосвязи модулей ВСПТД

Рисунок В.1 – Схема взаимосвязи модулей ВСПТД

Список использованных источников

1. Филиппов А.Н. [Текст]: Виртуальное строковое пространство технологических данных и знаний, Учеб. пособие // СПб. : Университет ИТМО, 2015. – 81 с.;

2. Филиппов А.Н., Путинцева А.А. [Электронный ресурс] Применение виртуального строкового пространства в САПР технологической подготовки производства – Методические указания к лабораторным работам. – Режим доступа: http://tps.faculty.ifmo.ru/download/2013_Book_VSPTD_methodical_instructions.pdf. – 2014, 89 с.

Миссия университета – генерация передовых знаний, внедрение инновационных разработок и подготовка элитных кадров, способных действовать в условиях быстро меняющегося мира и обеспечивать опережающее развитие науки, технологий и других областей для содействия решению актуальных задач.

КАФЕДРА ТЕХНОЛОГИИ ПРИБОРОСТРОЕНИЯ

Кафедра технологии приборостроения относится к числу ведущих кафедр института со дня его основания в 1931 году. Тогда она называлась кафедрой механической технологии и возглавлялась известным ученым в области разработки инструмента профессором Александром Павловичем Знаменским. Позже она была переименована в кафедру технологии приборостроения.

За время своего существования кафедра выпустила из стен института более тысячи квалифицированных инженеров, более сотни кандидатов и докторов наук. В разные годы ее возглавляли известные ученые и педагоги профессора Николай Павлович Соболев и Сергей Петрович Митрофанов.

Кафедра имеет выдающиеся научные достижения. Заслуженным деятелем науки и техники РСФСР, профессором С.П. Митрофановым были разработаны научные основы группового производства, за что он был удостоен Ленинской премии СССР. Методы группового производства с успехом применяются в промышленности и постоянно развиваются его учениками. Заслуженным деятелем науки и техники РСФСР, Заслуженным изобретателем СССР Юрием Григорьевичем Шнейдером разработаны метод и инструментарий нанесения регулярного микрорельефа на функциональной поверхности.

В настоящее время кафедра осуществляет выпуск бакалавров и магистров по направлениям 12.03.01, 12.04.01 "Приборостроение" и 09.03.01, 09.04.01 "Информатика и вычислительная техника".

А.Н. Филиппов, А.А. Путинцева

ПРИМЕНЕНИЕ МЕТОДОВ ВИРТУАЛЬНОГО СТРОКОВОГО ПРОСТРАНСТВА ТЕХНОЛОГИЧЕСКИХ ДАННЫХ И ЗНАНИЙ В САПР ТП

Методическое пособие

В авторской редакции
Редакционно-издательский отдел Университета ИТМО
Зав. РИО Н.Ф. Гусарова
Тираж 50 экз.
Отпечатано на ризографе

Редакционно-издательский отдел НИУ
Университета ИТМО
197101, Санкт-Петербург**,** Кронверкский пр.**, 49**