Программирование на языке SAS

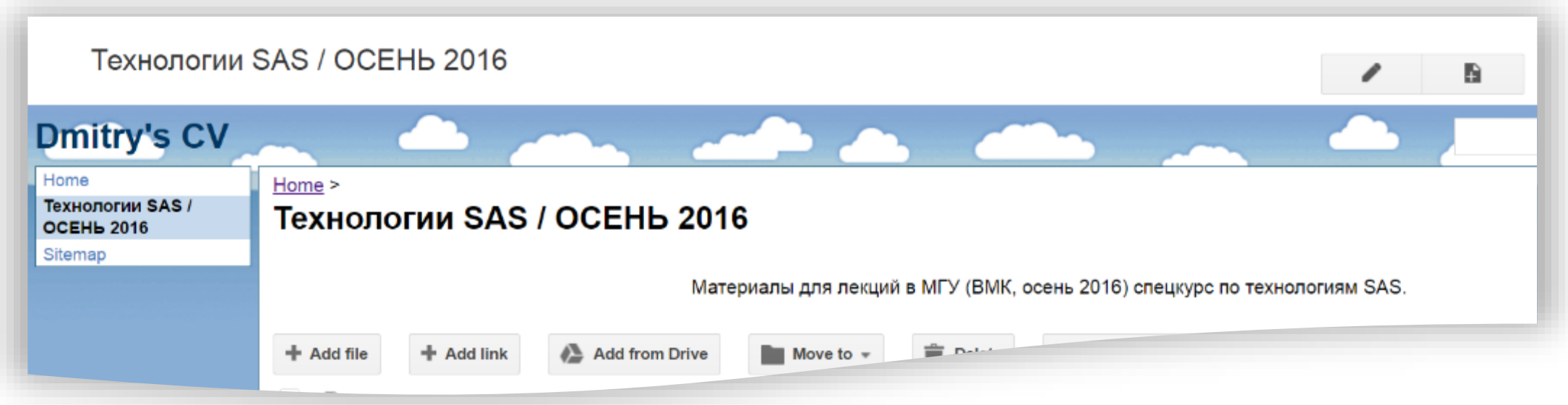
Осень 2016 – ВМК МГУ

Лекция 1. Основы.

Павел Гребенников,
Pavel.Grebennikov@sas.com

Где взять эти лекции и примеры программ?

http://bit.ly/VMK_FALL_2016



Слайды лекций нужно брать на этой странице.

Проверяйте, что вы берёте домашнее задание из АКТУАЛЬНЫХ слайдов 2016 года.

ДОМАШНЕЕ ЗАДАНИЕ

0) Установить SAS U

1) Настройте "общую директорию", чтобы вы могли сохранять программы и данные у себя на компьютере.

2) Программно создайте библиотеку MYDATA, которая будет связана с созданной "общей директорией".

3) С помощью шага DATA скопируйте в созданную библиотеку MYDATA набор данных AIR из библиотеки SASHELP. Убедитесь, что соответствующий файл появился на вашем компьютере.

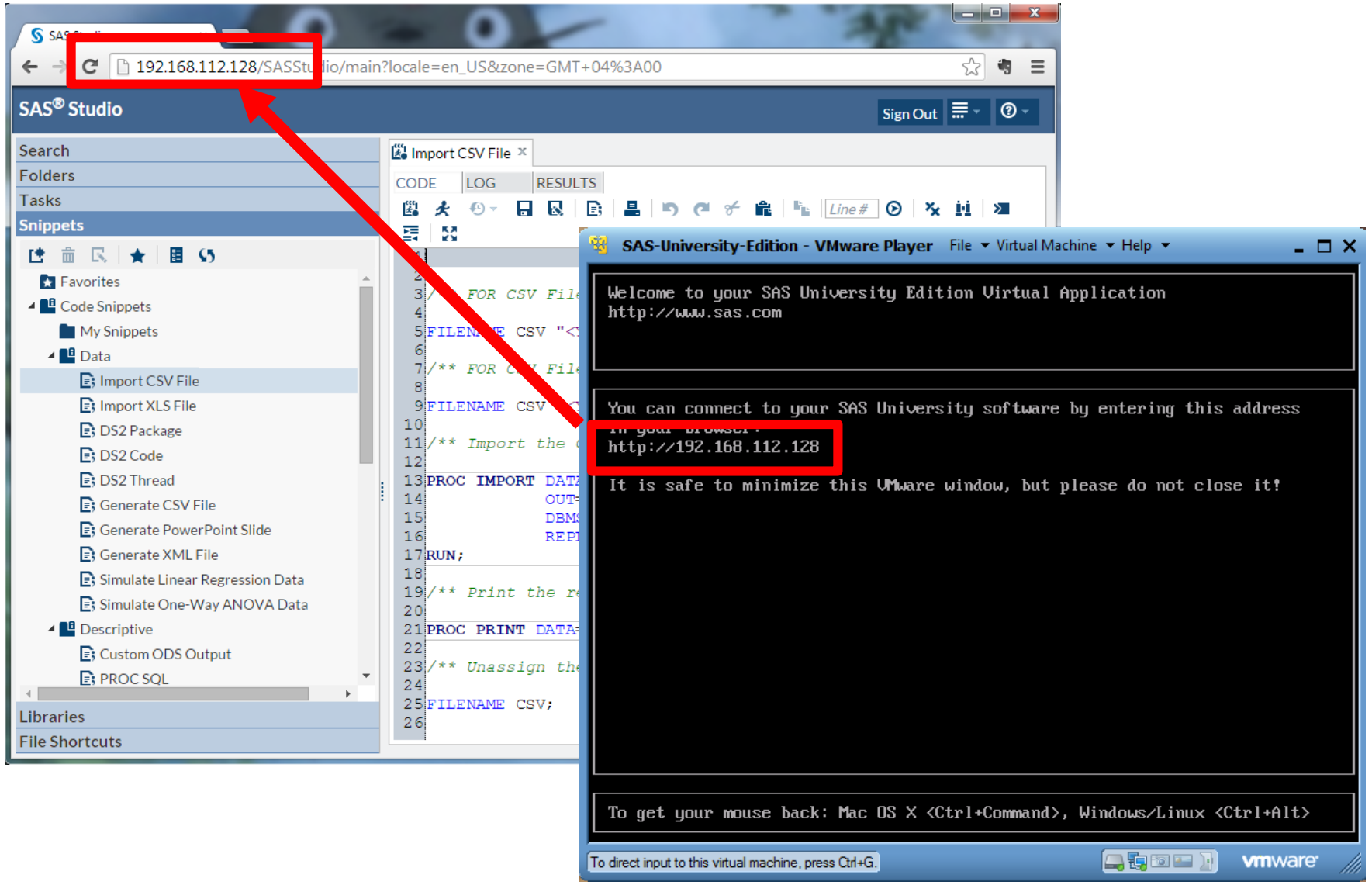
4) Найдите в документации SAS пример программы, которая печатает содержимое всех наборов данных SAS из заданной библиотеки. Выполните эту программу для библиотеки MYDATA.

Пункты 2), 3), 4) оформите в виде одной программы, в качестве ДЗ пришлите лог ее выполнения.

Где взять SAS?
SAS University Edition

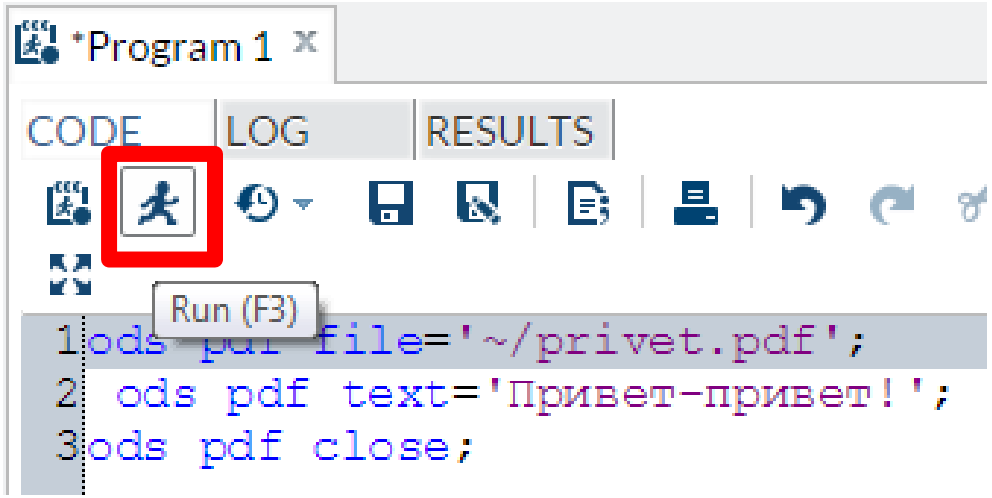
- http://www.sas.com/en_us/software/university-edition.html
  - Приложение в виде образа виртуальной машины для VirtualBox или VMWare
  - Не требуется постоянное подключение к интернету, просто настроить (инструкции см. на сайте)

SAS University Edition

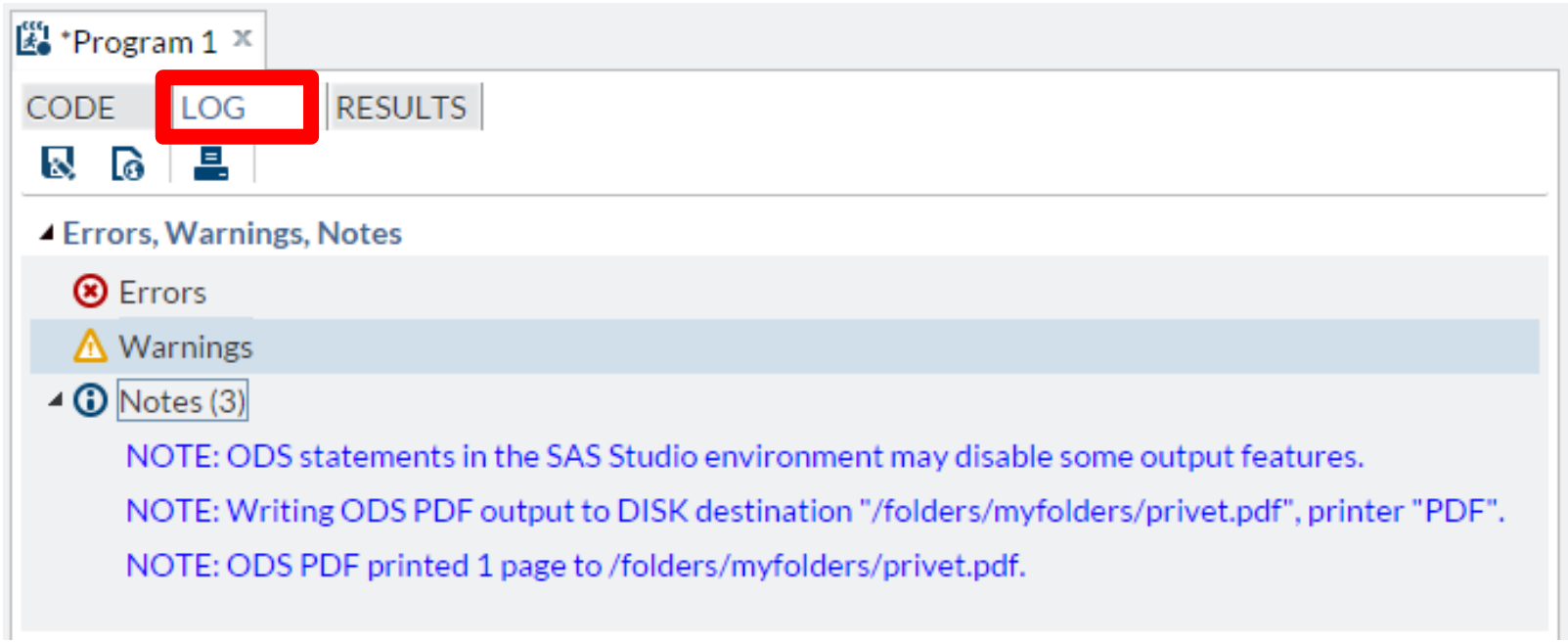


Как написать программу?

- Программа сохраняется на вашем компьютере (в настроенной директории My Folders = …/myfolders)
- Данные хранятся на вашем компьютере (можно загружать и создавать свои).



Журнал выполнения (Log)



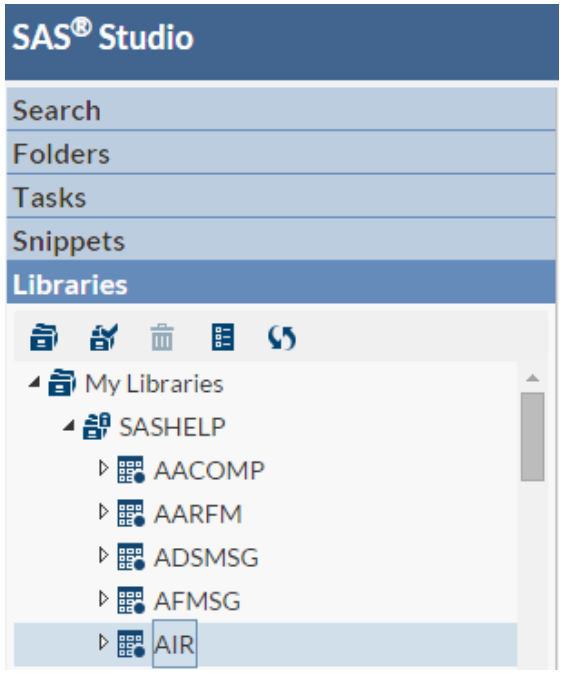
- Log – средство для отладки и диагностики выполнения программы. Работает для всех процедур SAS.
- Выводит текст запущенной программы и ошибки (если есть). Ошибки подробно описаны, иногда даются предложения по исправлению.
- Даже если ошибок сразу не видно, рекомендуется вдумчиво изучить Log.

Введение. Обзор языка SAS BASE

Структура программы на языке SAS BASE

- Линейная структура программы. Нет циклов, операторов условного перехода.
- Разные части программы обмениваются друг с другом данными в виде наборов данных SAS.
- Основная структура данных – набор данных (SAS data set). Все данные лежат на жестком диске в виде файлов. Поэтому данных может быть очень много!
- Только две (!) основные синтаксические конструкции:
  - Шаг DATA (работа с данными – создание, чтение, добавление, изменение)
  - Шаг PROC – всё остальное

Наборы данных SAS (datasets)



Наборы данных – обычные «плоские» таблицы

- Переменные (столбцы) – одного из двух типов, числовые или текстовые
- Для хранения чисел отводится 8 байт. Для хранения текста – от 1 до 32767 байт.
- Названия переменных – до 32 символов: _ a-z 0-9
- Регистр в названии переменной не важен
- Нельзя начинать название переменной с цифры

Именно наборы данных SAS (либо данные, которые представляются в этом виде) обязаны подаваться на вход процедур (шаг proc)

Каждый столбец имеет набор атрибутов (-> Columns), среди них: length (макс. кол-во байт, которые можно хранить в этой переменной для каждого наблюдения), name (название), label (ярлык, текстовое поле, где можно хранить более подробное описание переменной), type (тип), format (формат – правило для отображения данных в отчет или на экран)

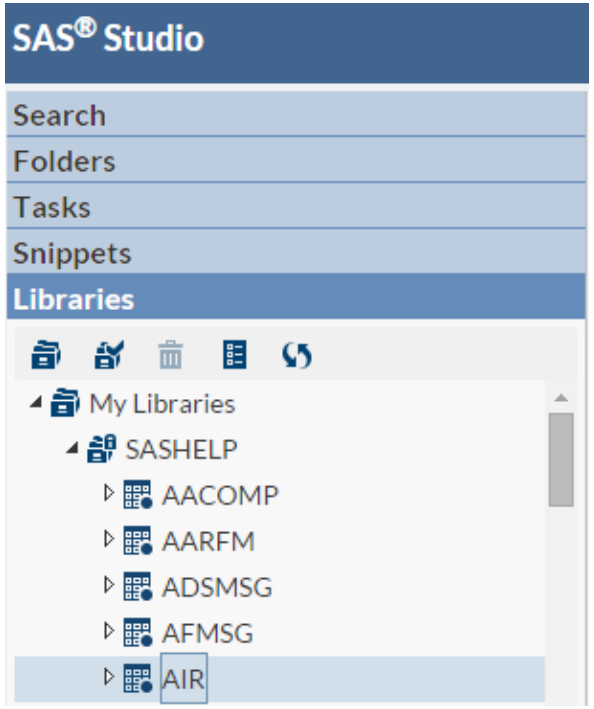
Для работы с данными в формате SAS dataset можно применять большой набор собственных алгоритмов, а также стандартный язык запросов (SQL)

Наборы данных SAS (продолжение)

- Наборы данных – обычные файлы (*.sas7bdat)
- Можно прозрачным образом подключать данные практически в любом виде (от excel/access до промышленных баз данных) через набор механизмов (engines).
- Импорт данных из большинства популярных форматов (xls и csv в SAS Studio; Code Snippets->Data->Import …)
- Нельзя создавать и редактировать прямо в SAS Studio ☹
- SAS – это НЕ СУБД (не гарантирует транзакционную целостность) и НЕ менеджер для организации хранения данных (разделение доступа, аудит данных и проч.), хотя возможности для этих целей у него имеются.

Библиотеки

Возникли как подход к централизованному хранению и прозрачному использованию данных в программах SAS

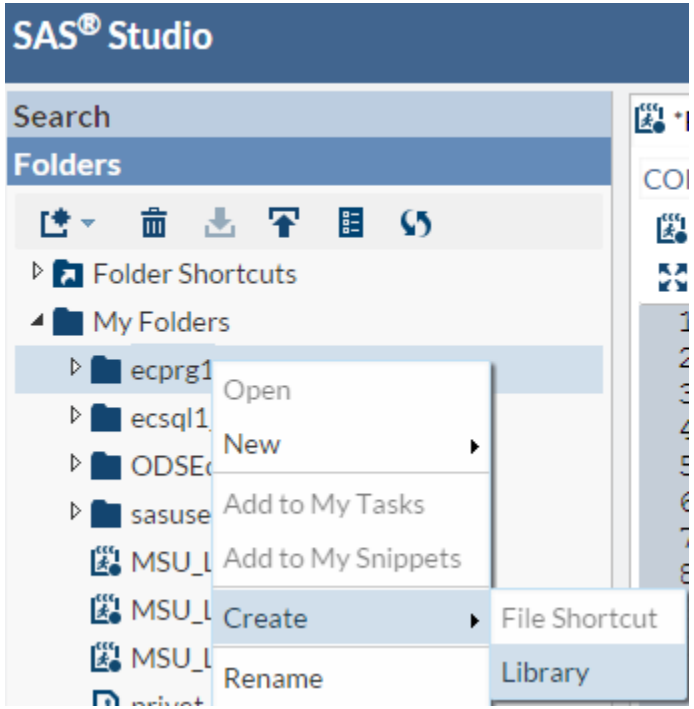


Самый простой случай библиотеки – это наборы данных, находящиеся в одной директории.

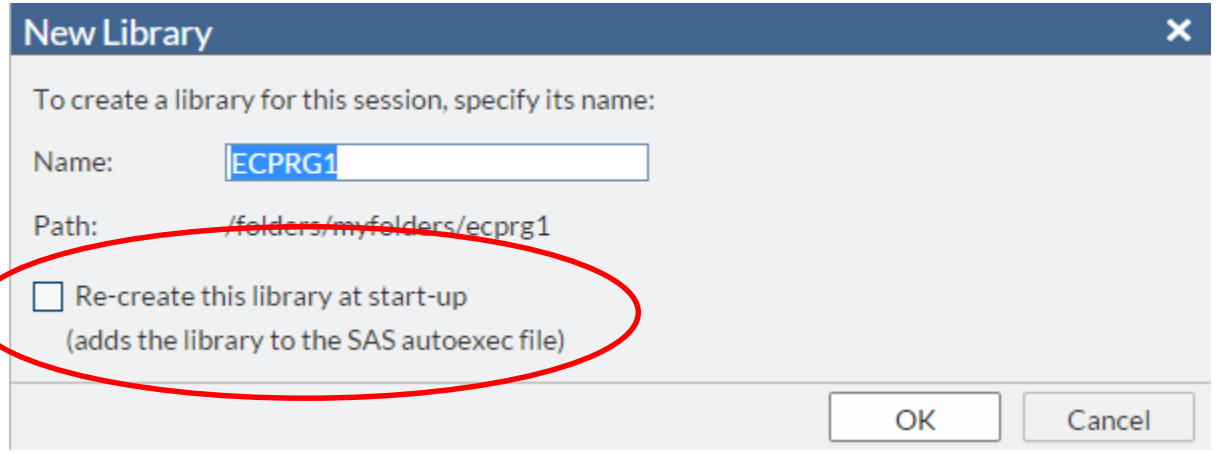
При запуске SAS сразу имеется несколько служебных библиотек, временная библиотека Work (она будет очищена при отключении от сервера или завершении процесса sas.exe) и персональная библиотека Sasuser (данные хранятся постоянно, но она отключена в SAS U).

В качестве библиотек можно подключать данные, которые физически находятся не в формате наборов данных SAS (промышленные БД, excel, access). Для большей части вашей программы будет всё равно, как (и где) данные хранятся.

Подключение библиотеки в SAS U



1. Копируете директорию с наборами данных в директорию …./SASUniversityEdition/myfolders
Открываете меню “My Folders”.



2. В SAS Studio создаёте библиотеку через меню.

Создание библиотек

- Подключение библиотеки может производиться:
  - кнопка «New Library»
  - либо программным путём (оператор libname):

    ```
    libname mylib “c:\dir_with_data”;
    ```

    mylib - название библиотеки: 0-9, a-z, _, не может начинаться с цифры, не более 8 символов длиной
- Подключенная таким образом библиотека «живет» до завершения процесса sas (для SAS U - до “Sign out”, если не включить волшебную галочку при создании библиотеки)
- Теперь в программе можно обращаться к набору данных не указывая полный путь (c:\dir_with_data\testdataset.sas7bdat), а с помощью 2-х уровневого имени (которое записывается через точку):

  mylib.testdataset

  Если первая часть не указана – предполагается библиотека Work

  Содержимое библиотеки можно посмотреть в окне «Server list», или с помощью процедуры:

  ```
  proc contents data=sasuser._all_;
  run;
  ```

Программа на языке SAS Base

```
options linesize=95 pagesize=52;

data work.NewSalesEmps;
   length First_Name $ 12 Last_Name $ 18
          Job_Title $ 25;
   infile 'newemps.csv' dlm=',';
   input First_Name $ Last_Name $
         Job_Title $ Salary;
run;

proc print data=work.NewSalesEmps;
run;

proc means data=work.NewSalesEmps;
   class Job_Title;
   var Salary;
run;
```

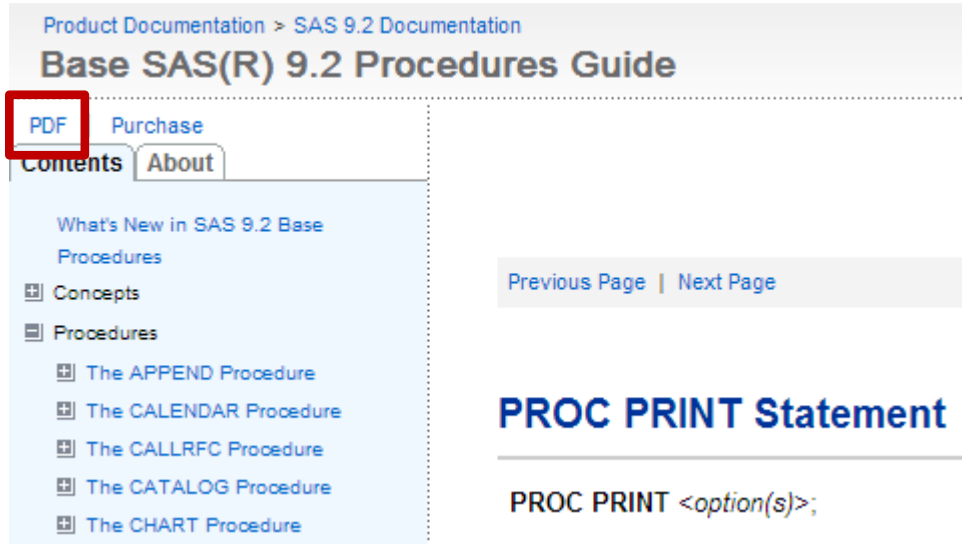
Программа на языке SAS Base

- Программа не требует явного подключения «библиотек» с процедурами, которые есть в установленном дистрибутиве SAS («#include»)
- Программа выполняется «сверху вниз» по шагам
- Минимальный «кусочек» программы, который SAS может выполнить, это один шаг (начинается с операторов «data», «proc»)
- Можно выполнить один или несколько шагов программы (выделить код мышкой и нажать кнопку RUN)
- Каждый оператор должен заканчиваться «**;**»
- Каждый шаг должен заканчиваться операторами «run;», «quit;», или началом следующего шага
- Форматирование свободное (программу можно записать в одну строку, оператор можно разбивать на несколько строк)

Справка

Справка в свободном доступе (веб-страница либо pdf) http://support.sas.com/documentation

1) Поиск справки по элементам языка SAS BASE нужно начинать с описания интересующей процедуры (и уже в ней искать справку о конкретном операторе)
2) Пункт overview – общие слова о том, что может делать процедура (шаг PROC)
3) Пункт examples – для самых нетерпеливых



4) Дополнительные материалы

*) Материалы конференции SUGI (обмен опытом между пользователями)

*) Справки по конкретным продуктам SAS

Рекомендуется иметь под рукой:

SAS/STAT(R) 9.4 User's Guide
SAS(R) 9.4 Functions and CALL Routines: Reference
Base SAS(R) 9.4 Procedures Guide
SAS(R) 9.4 SQL Procedure User's Guide
…

Шаг DATA

- Обработку данных нужно было делать ещё до того, как был принят стандарт SQL (wiki: в 1986 году первый стандарт языка SQL был принят ANSI, SAS развивается с 1966г, с 1976 как коммерческий продукт).
- Шаг DATA дополняет SQL и наоборот (SQL – работа с множеством наблюдений, шаг DATA – работа с единичными наблюдениями). Чем пользоваться для конкретных целей – нужно думать.
- Шаг DATA очень часто требует существенно меньше ресурсов по сравнению с SQL (важно если данных много), но нужно изучить логику его работы, которая бывает нетривиальной.
- Позволяет создавать и манипулировать наборами данных SAS (включая запись, чтение наборов данных, чтение необработанных данных, изменение структуры, создание агрегатов, объединение таблиц, фильтрацию наблюдений, можно использовать условные переходы, циклы, вызов пользовательских и встроенных функций, …).
- Мы посмотрим малую часть того, что может шаг DATA. Справка по DATA STEP:
  - **SAS(R) 9.4 Language Reference: Concepts, Second Edition -> DATA Step Concepts**
  - **SAS(R) 9.4 Language Reference: Concepts, Second Edition -> Dictionary of Language Elements -> SAS Data Set Options**

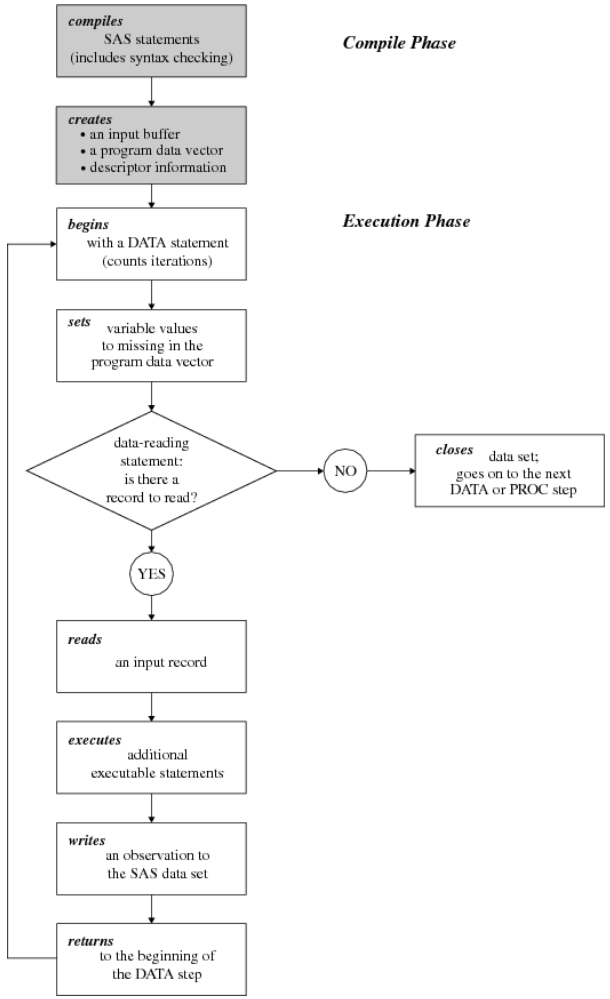
Шаг DATA

Самая простая операция – создание копии набора данных:

```
data employees;
  set employee_list;
run;
```

новый набор данных

исходный набор данных



Что происходит при запуске такого шага?

(см. Overview of DATA Step Processing: Flow of Action)

1) Фаза «компиляции»:

- проверка синтаксиса
- создание вектора данных (Program Data Vector, PDV) – области в памяти, где хранятся значения переменных **для одного наблюдения** (строки) из исходного набора данных, включая новые и служебные переменные.
- создание дескриптора (служебной части) для нового набора данных (хранит информацию о типах, названиях переменных, свойства набора данных и проч.)

Шаг DATA

Что происходит после фазы «компиляции»?

```
data employees;
  set employee_list;
run;
```

Ошибок в синтаксисе нет

Названия, типы и свойства переменных взяты из исходной таблицы на фазе компиляции

PDV:
(в памяти)

VAR1	VAR2	VAR3	…	…	…	…	…	…

employees:
(на диске)

Data Set Name	…	Observations	…
Member Type	…	Variables	..
Engine	..	Indexes	…
Created	…	Observation Length	…
Last Modified	…	Deleted Observations	…

VAR1 (Num)	VAR2 (Char)	VAR3 (Num)

2) Фаза «выполнения» – выполнение “цикла” внутри шага data. Одна итерация цикла – это обработка одного наблюдения (строчки)

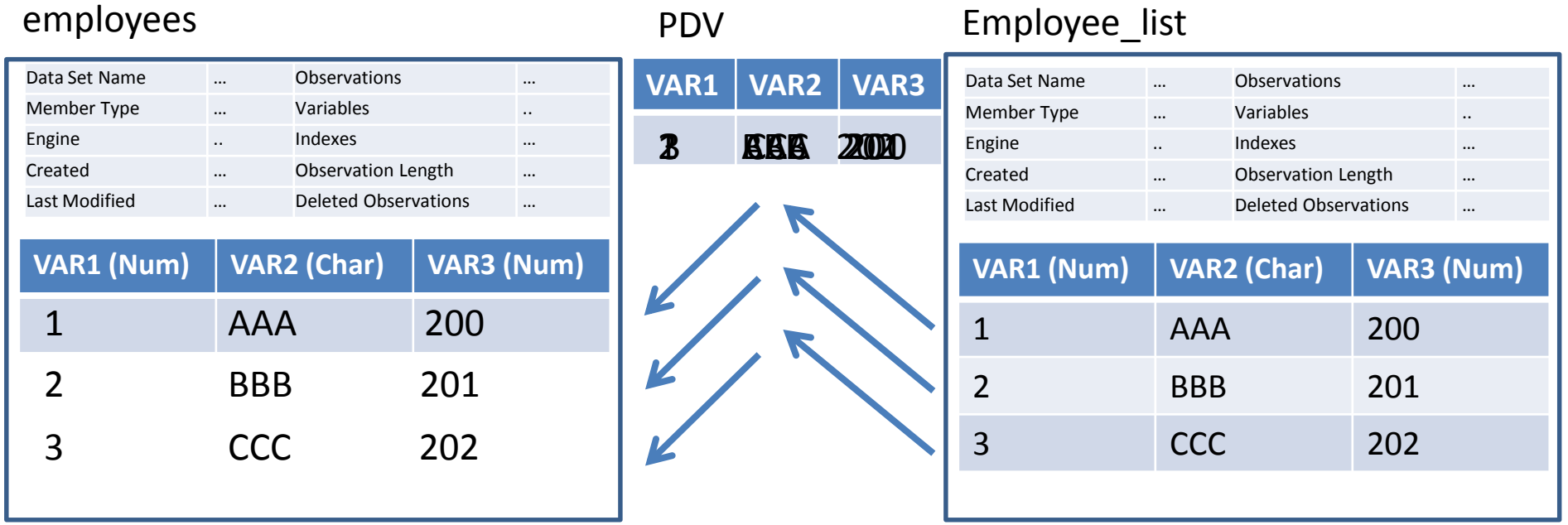
У нас он состоит только из одного оператора SET и неявных операторов, которых мы в программе не видим.

Вопрос: Когда этот цикл закончится?

Оператор SET

```
data employees;
   set employee_list;
run;
```

0. Проверка на достижение конца исх. файла
1. Явный оператор Set = Чтение одного наблюдения из Employee_list в PDV
2. Неявный оператор Output = сброс содержимого PDV в employees
3. Неявный оператор Return = возврат к началу цикла внутри шага DATA



Выбор наблюдений по условию

```
data employees;
  set employee_list;
  where age <= 65 and  upcase(gender) = "M" ;
run;
```

Symbol	Mnemonic Equivalent	Symbol	Mnemonic Equivalent
=	EQ	&	AND
^=	NE	|	OR
¬=	NE	!	OR
~=	NE	¦	OR
>	GT		
<	LT	¬	NOT
>=	GE	^	NOT
<=	LE	~	NOT
	IN		

Вхождение в список: **state in ('NY','NJ','PA')**

Замечания:

1. Выражение в условии обязано возвратить число:
   0 или . = False
   всё остальное = True

(. – это пропущенное значение, missing)

2. Операторы min (><) и max (<>):
Например, если A<B, то A><B вернёт значение A

3. Оператор конкатенации для символьных значений:
'grade '||'A'
Этот оператор не обрабатывает пробелы, которые могут содержаться в переменных. Особенность хранения символьных переменных в SAS: значения добиваются пробелами справа до максимального размера переменной (см атриб. length)

Выбор наблюдений по условию

```
data employees;
   set employee_list;
   where age <= 65 and  upcase(gender) = "M" ;
run;
```

Замечания:

4. Where умеет работать только с теми переменными, которые есть во входном наборе данных. Никакие новые переменные формально не должны появляться в условии.

5. При использовании оператора «where» в PDV попадают не все наблюдения, а только те, которые удовлетворяют условию.

6. Если данные не индексированы, то с диска читается весь входной набор данных (даже если часть выборки не удовлетворяет условию).

Операторы условного перехода

1. Выполнение нескольких операторов в одной ветви условного перехода на шаге DATA:

```
IF expression THEN do;
        …;
        …;
        …;
end;
ELSE do;
        …;
end;
```

2. Ещё один вариант условного перехода (аналог switch-case):

```
SELECT<(select-expression)>;
        WHEN (when-expression-1) statement;
        <WHEN (when-expression-n) statement;>
        <OTHERWISE statement;>
END;
```

Действия с переменными на шаге DATA

Изменение значения переменной (или создание новой переменной):

```
data employees;
  set employee_list;
  if upcase(Country)='US' then Bonus = 500;
  else Bonus=0;
run;
```

При компиляции в оперативной памяти формируется Program Data Vector, который содержит место для всех переменных, имеющихся внутри шага DATA. Если переменная «Bonus» была в наборе данных «employee_list» – его свойства берутся оттуда, если нет – в PDV появляется новая переменная.

Откуда берутся свойства новой переменной?

1. Можно явно задавать самим (напр., тип, формат, длину)
2. Если они не заданы явно, то берутся свойства по умолчанию или из контекста программы (длина, тип), см. далее

*** См. SAS(R) 9.4 Language Reference: Dictionary**

Функции*

Функции – это подпрограммы, которые возвращают одно значение

function-name (*argument-1*<, …*argument-n*>)

function-name (OF *шаблон-названия*)

Могут применяться на шаге DATA и PROC (например, в условиях-фильтрах)

Примеры функций:

SUM(*argument*,*argument*,...) – сумма аргументов (пропущенные значения игнор.)

Sum и оператор «+» могут работать по разному. Чему равны a1 и a2 в temp?

```
data temp;
  set in_tab;
  a1=b+c;
  a2=sum(b,c);
run;
```

in_tab

B (Num)	C (Num)
.	1

UPCASE(*argument*) – перевод символьной строки в верхний регистр

PUT(*source*, *format*.) – перевод числового аргумента в символьный с исползованием формата (format)

INPUT(*source*,*informat*.) – перевод символьного аргумента в числовой с использованием формата для ввода (informat)

Функции можно применять друг к другу без создания промежуточных переменных

* Справка: SAS(R) 9.4 Functions and CALL Routines: Reference

Функции для даты и времени

Внутренний формат даты в SAS – число дней с 01 января 1960 года

Внутренний формат времени – число секунд с полуночи

Внутренний формат дата/время (datetime) – число секунд с полуночи 01 января 1960 года

Для работы с такими данными (все хранятся в переменных числового типа) есть, в частности, функции:

DATE	Текущая дата в формате SAS
DATEPART	Извлекает дату в формате SAS из даты/времени
DATETIME	Текущая дата/время (timestamp)
DAY	Число из даты во внутреннем формате: day ('01jan2013'd) -> 1 ; day (19359) -> 1
DHMS	Вычисляет дату/время (datetime) из даты, часов, минут, секунд.
HMS	Вычисляет время в формате SAS из часов, минут и секунд
HOUR	Возвращает часы из внутреннего формата времени или даты/времени
INTCK	Возвращает количество временных промежутков, укладывающихся на данный интервал (годы, месяцы, недели …)
INTNX	Увеличивает дату, время или дату/время на данную величину, и возвращает полученное значение в формате SAS
MDY	Сформировать дату во внутреннем формате из месяца, числа, года: Mdy(1,1,2013) -> 19359
MINUTE	Извлечь минуты из внутреннего формата времени или даты/времени
MONTH	Извлечь месяц из внутреннего формата даты
QTR	Вычисляет квартал (года) из значения даты SAS.
TIME	Возвращает текущее время в формате SAS
TIMEPART	Извлекает часть, содержащую время, из временной метки
TODAY	Возвращает текущую дата во внутреннем формате даты
WEEK	Номер недели из внутреннего формата даты
WEEKDAY	День недели из внутреннего формата даты
YEAR	Год из внутреннего формата даты
YRDIF	Возвращает разницу между двумя датами в годах. Можно выбрать формулу для перевода.

Как создаются новые переменные в наборах данных?

1) Явно, с помощью оператора length:

```
length a $ 10; /*текстовая переменная,length=10 байт*/
length b 6; /*числовая переменная, length=6 байт*/
```

- В этом случае оператор length должен появиться на шаге data до операций с переменными (в т.ч. оператора присваивания, если он есть).

2) С помощью оператора присваивания:

```
aa = 10; /*справа стоит число, значит aa – числ. перем.*/
         /*по умолчанию length = 8 байт*/
bb = ` PRIVET! `; /*справа стоит текстовая константа*/
       /* bb – текст. перем. с length=9 (чтобы вместить */
       /*все символы, которые есть в текстовой константе)*/
```

Как создаются новые переменные в наборах данных?

2.1) А если справа от оператора присваивания стоит функция? Тогда тип переменной и её длина зависят от функции (см. help). Например:

> **Details**
>
> In a DATA step, if the UPCASE function returns a value to a variable that has not previously been assigned a length, then that variable is given the length of the argument.

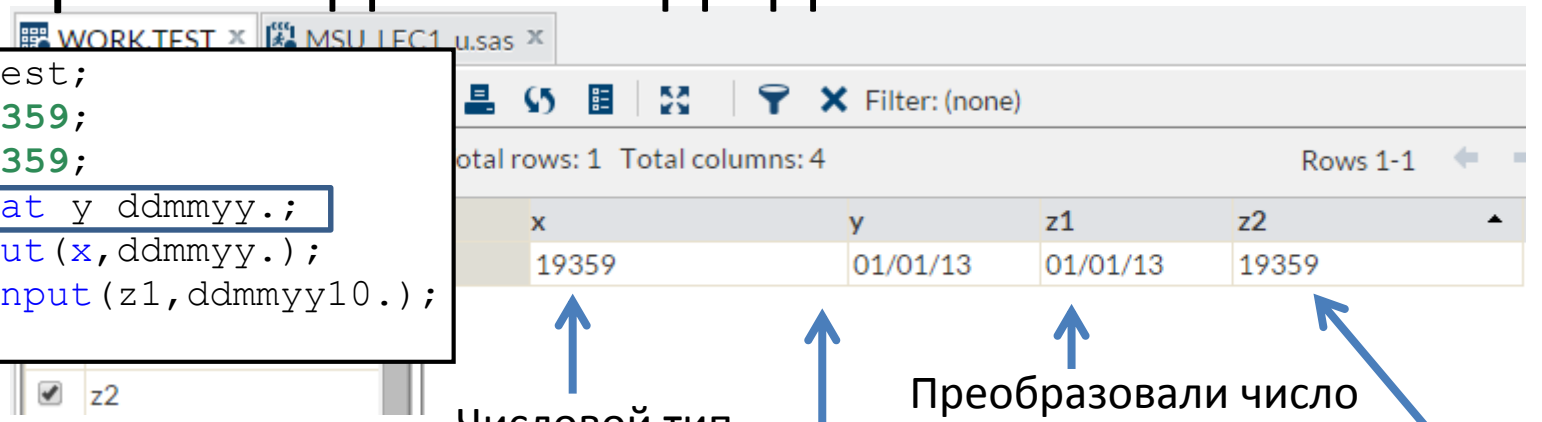
Вопрос: чему равна длина переменной в такой программе? (Если сеанс настроен для работы с utf-8 – многобайтовая кодировка):

```
data nnn;
var="Дима!";
run;
```

Формат*

- Это правило для вывода данных

```
data test;
  x=19359;
  y=19359;
  format y ddmmyy.;
  z1=put(x,ddmmyy.);
  z2=input(z1,ddmmyy10.);
run;
```



Информация о формате (для переменной Y) сохраняется в дескрипторе набора данных.

Числовой тип

Тоже числовой тип, но выводится как дата

Преобразовали число в строку символов

… и наоборот

1) При применении формата сами данные не меняются (переменная y - число)
2) Формат в SAS – это не только форматирование данных, но и метод табличного поиска (создание и поддержка централизованных справочников). Часто применяется, чтобы избежать хранение избыточных данных.
3) Шаблоны для вывода на экран можно создавать самому (proc format, Tasks -> Data -> Create format …).

* см. SAS(R) 9.3 Formats and Informats: Reference

Вывод наблюдений в наборы данных

```
   data test1 test2;
А    x=19359;
Б    output test1;
В    x=19360;
Г    output test1 test2; * output ;
     format x ddmmyy.;
   run;
```

```
NOTE: The data set WORK.TEST1 has 2 observations and 1 variables.
NOTE: The data set WORK.TEST2 has 1 observations and 1 variables.
```

Компиляция:

1) Формирование PDV (только 1 переменная: x),
2) Формирование дескрипторов наборов данных test1, test2
3) Информация о формате записана в дескрипторы test1 и test2

Выполнение:

А) Заносим x=19359 в PDV,
Б) Сбрасываем содержимое PDV в test1 (явный оператор output)
В) Заносим x=19360 в PDV,
Г) Сбрасываем PDV в test1, test2 (явный оператор output)
Д) Завершение шага DATA

Опции набора данных (keep, drop, where, end)

```
data test1 (keep=x y);
  x=today();
  y=datetime();
  z=time();
  keep x y;
run;
```

Опция набора данных – в test1 будут выведены только x,y

или

Оператор keep - во все наборы данных будут выведены только x,y

Keep: оставить в наборе данных только указанные переменные
Drop: оставить все переменные за исключением указанных

```
data test;
   set ecprg1.accounts
      (where=(Employee_ID between 11000 and 11999))
run;
```

Опция во входящем наборе данных, фильтрация при чтении данных (если в новом наборе данных – при записи)

```
data test;
   set ecprg1.accounts
      (rename=(account=code));
run;
```

Опция работает в исходящем и входящем наборе данных, переменная account будет переименована в code (в данном случае- в PDV и исходящем наборе данных)

Опции набора данных (keep, drop, where, end)

Замечания по опциям:

1) Внимательно читайте описание опции в справке, есть опции только для входных или только для выходных наборов данных
2) Опции действуют не только на шаге DATA, но и в процедурах (шаги PROC), в том числе PROC SQL.
3) Документация по опциям: SAS® 9.4 Data Set Options Reference


Спасибо!
pikabu.ru