

# Раздел 9

## Электронные таблицы – дополнительные возможности

# ЗАПИСЬ В БАНК ДАННЫХ С ИСПОЛЬЗОВАНИЕМ ФУНКЦИИ *PUT*

- n Команда *PUT* записывает блок ячеек электронной таблицы в открытый банк данных
- n Команда *PUT* противоположна команде *Select*
- n Команда *PUT* - часто заключительный шаг в шаблоне электронной таблицы, который уменьшает необработанные данные и записывает их в банк данных
- n Записываемые данные должны включать в себя по крайней мере один атрибут, входящий в отношение иерархии

# СОВЕТЫ ПО ИСПОЛЬЗОВАНИЮ ФУНКЦИИ *PUT*

- n Обратите внимание на информационное окно в процессе выполнения команды *PUT*
- n В конце выполнения появится окно, сообщающее об успехе выполнения операции *PUT*



- u **WARNING** - Операция успешно завершена; однако, данные могут быть неполными
- u **ERRORS** - Операция не была успешно завершена некоторой части данных
- u **INFORMATION** – Обзор

# СОЗДАНИЕ ИНСТРУМЕНТАРИЯ С ПОМОЩЬЮ ЭЛЕКТРОННЫХ ТАБЛИЦ

- n Электронные таблицы полезно использовать как интерфейс пользователя для обращения к банку данных
- n Возможно использование электронных таблиц для создания своих собственных инструментов
  - u Выбор определенных данных
  - u Помещение выбранных данных в банк данных
  - u Манипулирование данными в банках

# СОЗДАНИЕ ИНСТРУМЕНТАЛЬНЫХ СРЕДСТВ С ПОМОЩЬЮ ЭЛЕКТРОННЫХ ТАБЛИЦ

- n Функции электронных таблиц, используемые для создания инструментов, включают
  - и форматирование ячеек
  - и функции печати
  - и внешние функции

# ФОРМАТИРОВАНИЕ ЯЧЕЕК

- n Форматирование значения ячейки осуществляется при помощи команды меню **FORMAT**
- n Заданный по умолчанию формат ячейки - **%15.7g**

Ввод	%15.7g (Default)	%15.7f	%15.7e
123.456	123.456	123.456000	1.2345600e+02
123456789.0	1.2345678e+08	123456789.0000000	1.2345679e+08
123.4567890	123.4568	123.4567890	1.2345679e+02
.123456	0.123456	0.1234560	1.2345600e-01
.0012345678	0.001234568	0.0012346	1.2345678e-03

- n Можно включать текст в строку формата

Например, чтобы значение ячейки выводилось в виде " 1000 deg F ", надо определить формат след. образом " %15.7g deg F ".

# ФОРМАТИРОВАНИЕ С ПОМОЩЬЮ ФУНКЦИИ PRINT

Синтаксис:

```
=print(C_format, Value1, Value2...)
```

Примеры:

```
=print("The number in cell A3 is %8.5e", A3)
```

Где в ячейку A3 введено 123.45.

Вы увидите:

*The number in cell A3 is 1.23450e+02*

```
=print("%s/%s", A1, A2)
```

Где "/mydir" введено в ячейку A1 и "myfile.data" введено в ячейку A2.

Таким образом произойдет связка каталога и имени файла в виде "/mydir/myfile.data".

# ПОСТРОЕНИЕ КРИВЫХ В ЭЛЕКТРОННЫХ ТАБЛИЦАХ

- n Создавать и сохранять графики можно с использованием функции **POLYLINE**
- n Извлечь координаты точки из графика можно при помощи функции **Curve\_points** или **Scatter\_points**
- n Интерполяция значений X или Y по заданным значениям Y или X с помощью функций **interp\_y** или **interp\_x**, соответственно
  - u Пример: **interp\_y(C1, 1000)** возвращает значение X для Y=1000 для графика в ячейке C1
- n Аппроксимация заданного набора точек прямой методом наименьших квадратов с помощью функции **lin\_regres**
  - u Вы можете также использовать внешнюю функцию **FIT**, которая обеспечивает более точные методы сглаживания кривых.



# Графическое сравнение напряжений при разных температурах с допустимыми значениями из банка данных mil5

Шаг 1: Импортируйте данные  $\sigma$  vs T

Шаг 2: Сократите данные в случае необходимости (ie: Determine von Mises Stresses)

Шаг 3: Постройте график напряжений (отобразив только точки)

Шаг 4: Выберите из банка данных следующие данные:  
%YS11TvsT и YS11T

Шаг 5: Определите YS11TvsT из данных на шаге 4

Шаг 6: Постройте кривую YS11TvsT на существующем графике с точками разброса

# ЧТО ТАКОЕ ФУНКЦИИ и ЧЕМ ОНИ ПОЛЕЗНЫ

- n Функция - утилита электронной таблицы, которая может использоваться для управления данными.
- n Некоторые полезные встроенные функции включают сортировку, минимизирование/максимизирование, линейную регрессию, тригонометрические и арифметические функции.
- n Встроенные функции могут быть дополнены написанными пользователем внешними функциями
  - u Построение особых графиков
  - u Чтение или запись специальных файлов
  - u Пользовательская статистическая обработка (например по базисам A и B)

# Внешние функции электронных таблиц

# ПОДДЕРЖКА ВНЕШНИХ ФУНКЦИЙ

- n Вместе с MVISION поставляются различные внешние функции (в исходных текстах)
  - u Basis A & B
  - u Полиномиальное приближение
  - u Аппроксимация кривых (логарифмическая, линейная, степенная, экспоненциальная)
  - u Матричные операции (инверсия, транспонирование, масштабирование, детерминант...)

# ОСОБЕННОСТИ ВНЕШНИХ ФУНКЦИЙ

- n Внешние функции созданы с набором скриптов и библиотек, облегчающих доступ к электронным таблицам и передачу данных
- n Вызовы внешних функций описаны в руководстве пользователя
- n Примеры внешних функций (в исходных текстах) находятся в директории ExFun инсталляции MSC.Mvision

# ОСОБЕННОСТИ ВНЕШНИХ ФУНКЦИЙ

- n Могут обращаться к нескольким (до 8) блокам ячеек электронной таблицы и возвращать 1 блок результатов и/или выводить сообщения (до 64 символов).
  - Число ячеек в блоке не ограничено.
- n Использование языков программирования ФОРТРАН или С.
- n Доступ как к текстовым значениям, так и к значениям двойной точности.
- n Внешние функции работают как Remote Procedural Call (RPC) на Unix платформах.
- n RPC активизирует внешние функции и управляет передачей данных между банком данных и функцией.

# ПРИМЕР ФУНКЦИИ TRANSPOSE НА языке FORTRAN

```
INTEGER FUNCTION TRANSPOSE_FORTRAN( INPUT_DATA )
  INTEGER INPUT_DATA
C...
C
C Transpose an ARRAY of NUMBERS (text in the cells will produce
C an error) and send the transposed ARRAY back to the spreadsheet.
C The print statements will print diagnostic messages ( the before and
C after ARRAYS ) to the window where M/VISION was started.
C
C
C...
  INCLUDE '/mvision/include/mvef_type.incl'
C...
  INTEGER RETURN_DATA
C...
C...Declare TYPEs for local variables
  INTEGER IRET
  INTEGER NUM_COL, NUM_ROW, NUM_COL_T, NUM_ROW_T, ROW, COL
  INTEGER TEXT_FLAG
  CHARACTER*128 ERR_MSG
C *****MAX. ARRAY SIZE OF 100
  DOUBLE PRECISION ARRAY(100,100)
  INTEGER MAX
  MAX = 100
C *****
C
C
C... Initialize
  TEXT_FLAG = 0
C
```

Объявление  
входного  
дескриптора

Включение определения данных

Объявление выходного дескриптора

Объявление  
типов и  
инициализация

# FORTRAN TRANSPOSE

C234567

```
C... Get the size [NUM_COL,NUM_ROW] of argument #1
      IRET = GTARSZ( INPUT_DATA, 1, NUM_COL, NUM_ROW )
```

C

```
C... Check against maximum array size
      IF( (NUM_COL .GT. MAX) .OR. (NUM_ROW .GT. MAX) ) THEN
          IRET = ALRTDT( 1, 1, RETURN_DATA )
          IRET = STRTCD( RETURN_DATA, 999 )
          WRITE(ERR_MSG, 999) MAX
999   FORMAT('ARRAY IS TOO BIG (MAX IS 15)')
          IRET = STRTMS( RETURN_DATA, ERR_MSG)
          TRANSPOSE = RETURN_DATA
          RETURN
      ENDIF
```

C

```
C... Fill ARRAY with NUMBERS from Argument 1
      PRINT *, ''
      PRINT *, 'Array:'
      DO 10 ROW=1,NUM_ROW
          DO 20 COL=1,NUM_COL
              IRET = GTCLNM( INPUT_DATA, 1, COL, ROW, ARRAY(ROW,COL))
C          * check to see if cell is text *
              IF( IRET .EQ. EF_NOTNUMBER) THEN
                  TEXT_FLAG = 1
              ENDIF
              WRITE( 6, 101) ARRAY(ROW,COL)
101   FORMAT(F5.2,' ',',')
          20 CONTINUE
              PRINT *, ''
          10 CONTINUE
```

C

Определение размера аргумента #1

Вывод сообщения об ошибке, если размер массива превышает допустимый размер

Получение информации о данных из INPUT\_DATA и помещение их в массив



# FORTRAN TRANSPOSE

```
C... ----- Transpose Array -----
NUM_ROW_T = NUM_COL
NUM_COL_T = NUM_ROW

C... Allocate return_data structure
   IRET = ALRTDT( NUM_COL_T, NUM_ROW_T, RETURN_DATA )

C... Put data into return structure
PRINT *, ''
PRINT *, 'Transposed Array:'
DO 50 ROW=1,NUM_ROW_T
  DO 60 COL=1,NUM_COL_T
    IRET = PTCLNM( RETURN_DATA, 1, COL, ROW, ARRAY(COL,ROW) )
    WRITE( 6, 102) ARRAY(COL,ROW)
102  FORMAT(F5.2,' ', '$)
  60  CONTINUE
    PRINT *, ''
  50  CONTINUE
C... Clear the buffer to standard output
PRINT *, ''

C... Setup error message if the TEXT_FLAG was set
IF( TEXT_FLAG.EQ.1 ) THEN
  IRET = STRICD( RETURN_DATA, 999)
  ERR_MSG = 'There was TEXT in the cells of the input matrix'
  IRET = STRTMS( RETURN_DATA, ERR_MSG )
ENDIF

C
C... This function !MUST! return the allocated return data
TRANSPOSE_FORTRAN = RETURN_DATA
RETURN
END
```

Создание области  
памяти return\_data

Транспонирование  
данных массива в  
область памяти  
return\_data

Создание сообщения  
об ошибке, если  
массив содержит  
текст

Возвращение  
транспонированных  
данных обратно в  
электронную таблицу

# ПРИМЕР ФУНКЦИИ TRANSPOSE НА ЯЗЫКЕ C

```
#include <stdio.h>
#include <stdlib.h>
#include I/mvision/include/mvef_type.h

void *
transpose( input_data )
/*
 *
 *   Transpose an array of numbers (text will produce 0's in the tranposed
 *   array) and send the transposed array back to the spreadsheet.
 *   The print statements will print diagnostic messages ( the before and
 *   after arrays ) to the window where M/VISION was started.
 *
 */

void *input_data;
{

    int    num_input_args;
    int    i, j, type, iret;
    int    num_col, num_row, num_col_t, num_row_t, col, row;
    int    text_flag = 0;
    double number;
    double *array;

    void *return_data;

    /* Get the size [num_col,num_row] of argument #1 */
    iret = EfGetArgSize( input_data, 1, &num_col, &num_row );
```

# C TRANSPOSE

```
/* Create array of appropriate size */
array = (double *) malloc( (num_col+1) * (num_row+1) * sizeof( double ));

/* Fill array with numbers from argument 1 */
printf(1\narray1);
for( row=1; row<=num_row; row++){
    printf(1\n1);
    for( col=1; col<=num_col; col++){
        if( EfGetCellNumber( input_data, 1, col, row, &array[row * num_col + col])
=
        EF_NOTNUMBER)
            text_flag = 1; /* cell is text instead of number */
        printf(1%5.2f 1,array[row*num_col+col]);
    }
}

/* ----- Transpose Array ----- */
num_row_t = num_col;
num_col_t = num_row;

/* Allocate return_data structure */
iret = EfAllocReturnData( num_col_t, num_row_t, &return_data );

/* Put data into return structure */
printf(1\ntransposed1);
for( row=1; row<=num_row_t; row++){
    printf(1\n1);
    for( col=1; col<=num_col_t; col++){
        EfPutCellNumber( return_data, 1, col, row, array[col*num_col+row] );
        printf(1%5.2f 1,array[col*num_col+row]);
    }
}
```

# C TRANSPOSE

```
printf( "!\n!");

/* Setup error message if the text_flag was set */
if( text_flag){
    EfSetReturnCode( return_data, 999);
    EfSetReturnMessage( return_data, "There were text cells in the input matrix
!");
}

return( return_data );
}
```

# СПОСОБЫ УЛУЧШИТЬ ЭТУ ПРОГРАММУ

- n Использовать код возврата для того, чтобы проверять каждый запрос
- n Динамическое выделение памяти под массив
- n Нет потребности использовать массив. Все транзакции могут быть осуществлены с помощью указателей (ссылок)
- n Обратитесь к Документации для получения дополнительной информации

# УПРАЖНЕНИЯ

- n Выполните Упражнение 8
- n **Спрашивайте, если Вы что-нибудь не понимаете**