使用 RE 與 bash 實作試算表語言

作者：一卡
時間： 2013 年 05月 22日

字串分析

整個語言限制只使用 bash 的 if、case、echo、grep、sed、tr、變數及 function 實作，因遞迴函數較直覺，實作中的重複計算皆使用尾遞迴，當需要加快計算速度時，可優先將尾遞迴函數全部替換成迴圈，實作時將程式碼分成兩個部份，第一為字串分析的部份，第二為數值計算的部份，語言的固定格式為 [command] [parameter 1] [parameter 2]，根據用途與實作，parameter 1 可能是一個數值或 command內的一個函數，實作上使用 5個全域變數，用途如下所述。

FILENAME:	目前使用的試算表名稱
SIZE_ROW:	試算表 ROW的大小
SIZE_COLUMN:	試算表 COLUMN 的大小
ROW:	目前指向試算表的那一個 ROW
COLUMN:	目前指向試算表的那一個 COLUMN

寫入資料或讀出資料時，根據 ROW與 COLUMN 決定操作那一格，目前尚未加入用來存放資料的GRID變數，只能寫入資料到試算表中，實作程式碼請參考 re_spreatsheet ，相關資料及程式碼可在 https://github.com/drm343/RE_Spreadsheet 取得。

數值計算

加法包含純數值計算與資料檢查兩部份，純數值計算的部份以加法器為基礎原理，透過查表法處理位數相加與進位，檢查部份則判別輸入資料是否為整數，如果數值處理正確，則回傳值必為整數，因此不檢查回傳值的型別，確定兩者皆為整數，則分析正負號，根據正負號決定計算方式，請參考以下演算法。

```
type Sign = + | -

Check_Sign :: Sign -> Sign -> (Int -> Int -> Int)
Check_Sign sign_1 sign_2
        | sign_1 == sign_2 = Calculate
        | otherwise = (\int_1 int_2 -> Calculate int_1 (Complement int_2))
```

決定計算方式後，根據兩數的絕對值大小，決定何者為先計算的數值，較大的數值將決定計算後數值的正負數，由於使用加法來處理兩數相減的情況，如果先將數值轉為二進位計算後再轉回十進位將浪廢許多時間，因此需要使用補數法來轉換數值，並透過補數法與加法取代減法，這裡採用「n 進位補數法」，證明如下。

[證明]
於 n 進位時，設有一個數值 a，若存在一個數值 b 使以下條件成立，則我們稱 a 與 b 互補，若稱 a 為正數，則 b 即稱負數，反之亦同。

```
a + b = 0
```

假設存在兩個數值 a、 b，若 a 為 r 位數， b 為 k 位數，假設 b 的補數為 b' ，則可獲得以下結果。

```
|b| < |a| < |b'| or |b'| < |a| < |b|
```

因為 a、 b 正負號相反，相加後所得結果必小於兩者絕對值中較大者，即代表從 r 與 k 中取較大者 max(r, k)，若相加後得到的位數 j 大於 max(r, k)，則我們只取 max(r, k) 位數，稱此為「補數法最大位數取位原則」。

設 n 進位 r 位數的最大值為 b'，根據取位原則我們可以得到如下結果。

```
b' + 1
  = 0
  = a + b
=> b = b' - a + 1
```

若 n 為二，即為常見的二補數法，設 n 為 10 ，則十進位補數法最大值如下。

$$9\times10^{r} + 9\times10^{(r-1)} + .. + 9$$

若 a 值以相同方式表示則如下。

$$a_{r}\times10^{r} + a_{(r-1)}\times10^{(r-1)} + .. a_{1}$$

則可得知對於 b' - a 的所有位數皆有以下公式。

```
f x = 9 - x
```

根據此公式求完 b' - a 的值後加一即為 a 的補數 b。

re_spreatsheet 程式碼

```
#!/bin/bash

#--------------------------------#
# Int Type                       #
#--------------------------------#
function Is_Digit {
  RESULT=`echo "$1" | grep "^[0123456789]\{1\}$"`

  case $RESULT in
    "")
      RESULT="false"
      ;;
    *)
      RESULT="true"
      ;;
  esac
}

function Is_Int {
  RESULT=`echo "$1" | grep "\-\?[0123456789]\{1,10\}$"`

  case $RESULT in
    "")
      RESULT="false"
      ;;
    *)
      RESULT="true"
      ;;
  esac
}

function Is_Negative {
  RESULT=`echo "$1" | grep "\-"`

  case $RESULT in
    "")
      RESULT="false"
      ;;
    *)
      RESULT="true"
      ;;
  esac
}

function Decimal_Complement {
  RESULT=`echo "$1" | tr "1234567890" "8765432109"`
  Succ "$RESULT"
}
```

```
function Succ {
  Uncheck_Add "$1" "1"
}

function Pred {
  Add "$1" "-1"
}

function Uncheck_Add {
  local carry_result="false"

  #------------------#
  # Check Carry      #
  #------------------#
  function No_Carry {
    RESULT=`echo "$1" | sed "s/00/0/" | sed "s/01/1/" | sed "s/02/2/" | sed
"s/11/2/" | sed "s/03/3/" | sed "s/12/3/" | sed "s/04/4/" | sed "s/13/4/" | sed
"s/22/4/" | sed "s/05/5/" | sed "s/14/5/" | sed "s/23/5/" | sed "s/06/6/" | sed
"s/15/6/" | sed "s/24/6/" | sed "s/33/6/" | sed "s/07/7/" | sed "s/16/7/" | sed
"s/25/7/" | sed "s/34/7/" | sed "s/08/8/" | sed "s/17/8/" | sed "s/26/8/" | sed
"s/35/8/" | sed "s/44/8/" | sed "s/09/9/" | sed "s/18/9/" | sed "s/27/9/" | sed
"s/36/9/" | sed "s/45/9/"`
    local local_result=$RESULT

    Is_Digit "$RESULT"

    case $RESULT in
      "false")
        RESULT="nothing"
        ;;
      *)
        RESULT=$local_result
        carry_result="false"
        ;;
    esac
  }

  function Carry {
    RESULT=`echo "$1" | sed "s/19/0/" | sed "s/28/0/" | sed "s/37/0/" | sed
"s/46/0/" | sed "s/55/0/" | sed "s/29/1/" | sed "s/38/1/" | sed "s/47/1/" | sed
"s/56/1/" | sed "s/39/2/" | sed "s/48/2/" | sed "s/57/2/" | sed "s/66/2/" | sed
"s/49/3/" | sed "s/58/3/" | sed "s/67/3/" | sed "s/59/4/" | sed "s/68/4/" | sed
"s/77/4/" | sed "s/69/5/" | sed "s/78/5/" | sed "s/79/6/" | sed "s/88/6/" | sed
"s/89/7/" | sed "s/99/8/"`
    local local_result=$RESULT

    Is_Digit "$RESULT"

    case $RESULT in
```

```
    "false")
      RESULT="nothing"
      ;;
    *)
      RESULT=$local_result
      carry_result="true"
      ;;
  esac
}

#------------------#
# Inline           #
#------------------#
function Inline_Add {
  No_Carry "$1$2"

  if [ $RESULT == "nothing" ]
  then
    Carry "$1$2"
  fi
}

function Check_Carry_When_Calculate_End {
  if [ "$carry_result" == "true" ]
  then
    Succ "$1"
  else
    RESULT=""
  fi
  carry_result="false"
}

function Inline_Add_Loop {
  local num_1_car=`echo "$1" | sed "s/.*\(.\)$/\1/"`
  local num_2_car=`echo "$2" | sed "s/.*\(.\)$/\1/"`
  local num_1_cdr=`echo "$1" | sed "s/\(.*\).$/\1/"`
  local num_2_cdr=`echo "$2" | sed "s/\(.*\).$/\1/"`
  local new_val=""

  if [ "$carry_result" == "true" ]
  then
    Succ "$num_1_car"
    num_1_car="$RESULT"
  fi

  Inline_Add "$num_1_car" "$num_2_car"

  if [ "$RESULT" == "nothing" ]
  then
    Inline_Add "$num_2_car" "$num_1_car"
```

```
    fi

    new_val="$RESULT$3"

    if [ "$num_1_cdr" == "" ] && [ "$num_2_cdr" == "" ]
    then
      Check_Carry_When_Calculate_End "0"
      RESULT=`echo "$RESULT$new_val" | sed "s/\([1234567890]\{10\}\)$/\1/"`
    elif [ "$num_1_cdr" == "" ]
    then
      Check_Carry_When_Calculate_End "$num_2_cdr"
      RESULT=`echo "$num_2_cdr$new_val" | sed "s/\([1234567890]\{10\}\)$/\1/"`
    elif [ "$num_2_cdr" == "" ]
    then
      Check_Carry_When_Calculate_End "$num_1_cdr"
      RESULT=`echo "$num_1_cdr$new_val" | sed "s/\([1234567890]\{10\}\)$/\1/"`
    else
      Inline_Add_Loop "$num_1_cdr" "$num_2_cdr" "$new_val"
    fi
  }

  #------------------#
  # main uncheck add #
  #------------------#
  Inline_Add_Loop "$1" "$2" ""
}

function Add {
  #------------------#
  # Order            #
  #------------------#
  function Num_1_Big_Than_Num_2 {
    local num_1_car=""
    local num_2_car=""
    local num_1_cdr=""
    local num_2_cdr=""

    function Loop {
      RESULT=`echo "$1" | sed "s/0./false/" | sed "s/.*0/true/"`

      case $RESULT in
        "true" | "false")
          ;;
        *)
          RESULT=`echo "$RESULT" | tr "123456789" "012345678"`
          Loop "$RESULT"
          ;;
      esac
    }
```

```
  Car $1
  num_1_car=$RESULT
  Car $2
  num_2_car=$RESULT

  if [ "$num_1_car" != "$num_2_car" ]
  then
    Loop "$num_1_car$num_2_car"
  fi

  case $RESULT in
    "true" | "false")
      ;;
    *)
      Cdr $1
      num_1_cdr=$RESULT
      Cdr $2
      num_2_cdr=$RESULT

      if [ "$num_1_cdr" != "" ]
      then
        Num_1_Big_Than_Num_2 "$num_1_cdr" "$num_2_cdr"
      fi
      ;;
  esac
}

#------------------#
# List Process     #
#------------------#
function Car {
  RESULT=`echo "$1" | sed "s/\(.\).*/\1/"`
}

function Cdr {
  RESULT=`echo "$1" | sed "s/.\(.*\)/\1/"`
}

function Count_String {
  local new_val=""

  Cdr $1
  new_val=$RESULT
  Succ $2
  count=$RESULT

  if [ "$new_val" != "" ]
  then
    Count_String "$new_val" "$RESULT"
  fi
```

```
}

#------------------#
# Check Sign       #
#------------------#
function Get_Sign_Then_Add {
  local sign=`echo "$1" | tr "1234567890" "          " | sed "s/ *$//"`
  local row_re="s/.*\(.\{$3\}\)$/\1/"

  Is_Negative $1
  num_1_is_negative=$RESULT

  Is_Negative $2
  num_2_is_negative=$RESULT

  RESULT=`echo "$1" | sed "s/\-\(.\{1,10\}\)$/\1/"`
  num_1=$RESULT
  RESULT=`echo "$2" | sed "s/\-\(.\{1,10\}\)$/\1/"`
  num_2=$RESULT

  if [ "$num_1_is_negative" != "$num_2_is_negative" ]
  then
    RESULT=`echo "000000000$num_2" | sed "$row_re"`
    Decimal_Complement $RESULT
    num_2=$RESULT
  fi

  Uncheck_Add $num_1 $num_2
  RESULT=`echo $RESULT | sed "$row_re" | sed "s/^0*0?$\(.*\)/\1/"`
  RESULT="$sign$RESULT"
}

#------------------#
# Main Add         #
#------------------#
local num_1=""
local num_2=""
local num_1_is_digit=""
local num_2_is_digit=""
local count_1=""
local count_2=""

Is_Int "$1"
num_1_type=$RESULT
Is_Int "$2"
num_2_type=$RESULT

num_1=`echo "$1" | sed "s/\-\(.\{1,10\}\)$/\1/"`
Count_String "$num_1" 0
count_1=$RESULT
```

```
  num_1=`echo "000000000$num_1" | sed "s/.*\(.\{10\}\)$/\1/"`

  num_2=`echo "$2" | sed "s/\-\(.\{1,10\}\)$/\1/"`
  Count_String "$num_2" 0
  count_2=$RESULT
  num_2=`echo "000000000$num_2" | sed "s/.*\(.\{10\}\)$/\1/"`

  if [ $num_1_type == $num_2_type ] && [ $num_1_type == "true" ]
  then

    if [ "$num_1" == "$num_2" ]
    then
      RESULT="true"
    else
      Num_1_Big_Than_Num_2 "$num_1" "$num_2"
    fi

    case $RESULT in
      "true")
        Get_Sign_Then_Add "$1" "$2" "$count_1"
        ;;
      "false")
        Get_Sign_Then_Add "$2" "$1" "$count_2"
        ;;
      *)
        RESULT="nothing"
        ;;
    esac
  fi
}

function Mul {
  local constant_number="$1"
  local new_val="$1"

  function Mul_Loop {
    case $2 in
      "1")
        RESULT="$1"
        ;;
      *)
        Add "$1" "$constant_number"
        new_val="$RESULT"
        Pred "$2"
        Mul_Loop "$new_val" "$RESULT"
        ;;
    esac
  }

  Mul_Loop "$1" "$2"
```

```
}

#-------------------------------#
# file set                      #
#-------------------------------#
function File {
  function New {
    FILENAME="$1"
    echo "" > $FILENAME
  }

  case $1 in
    "New")
      $1 $2
      ;;
    *)
      echo "no file $1 or command error"
      exit
      ;;
  esac
}

function Size {
  local local_file="tmp_$FILENAME"
  local defalut_grid="          0"
  local column_grid="          0"

  function Set_Column {
    function Set_Column_Loop {
      case "$1" in
        "1")
          return 0
          ;;
        *)
          column_grid="$column_grid$defalut_grid"
          Pred "$1"
          Set_Column_Loop "$RESULT"
          ;;
      esac
    }
    Set_Column_Loop $1
  }

  function Set_Row {
    function Set_Row_Loop {
      case "$1" in
        "0")
          ;;
        *)
          echo "$column_grid" >> $local_file
```

```
          Pred "$1"
          Set_Row_Loop "$RESULT"
          ;;
      esac
    }
    Set_Row_Loop "$1"
  }

  SIZE_ROW="$1"
  SIZE_COLUMN="$2"

  Set_Column "$2"
  Set_Row "$1"
  mv $local_file $FILENAME
}

#-------------------------------#
# move grid register            #
#-------------------------------#
function Move {
  function Row {
    ROW=$1
  }

  function Column {
    COLUMN=$1
  }

  case $1 in
    "Row" | "Column")
      $1 $2
      ;;
    *)
      echo "move $1 error"
      ;;
  esac
}

#-------------------------------#
# set grid value                #
#-------------------------------#
function Grid {
  function Set {
    local set_value="           $1"
    set_value=`echo "$set_value" | sed "s/^.*\(.\{11\}\)$/\1/"`

    Pred "$COLUMN" "1"

    case $RESULT in
      "0")
```

```
        RESULT=0
        ;;
      *)
        Mul "11" "$RESULT"
        ;;
    esac

    search="\(.\{$RESULT\}\).\{11\}\(.*\)"
    row_re=$ROW"s/$search/\1$set_value\2/"
    sed -i "$row_re" "$FILENAME"
  }

  case $1 in
    "Set")
      $1 $2
      ;;
    *)
      echo "Grid set error"
      ;;
  esac
}

#--------------------------------#
# main                           #
#--------------------------------#
function Get_Command {
  local command=`echo "$1" | sed "s/\(.*\) .* .*/\1/"`
  local parameter=`echo "$1" | sed "s/.* \(.*\) .*/\1/"`
  local target=`echo "$1" | sed "s/.* .* \(.*\)/\1/"`
  $command $parameter $target
}

function Main_Loop {
  local file_number=$2"q;d"
  local new_line=`sed "$file_number" $1`
  Get_Command "$new_line"
  echo "$new_line"

  if [ ! "$2" == "$3" ]
  then
    Add $2 "1"
    Main_Loop "$1" "$RESULT" "$3"
  fi
}

# FILENAME
# SIZE_ROW
# SIZE_COLUMN
# ROW
# COLUMN
```

```
case $1 in
  "")
    echo "no data"
    ;;
  *)
    LAST_LINE=`grep -n ".*" $1 | tail -n 1 | sed "s/\([1234567890]*\):.*/\1/"`
    SIZE_ROW=1
    SIZE_COLUMN=1
    ROW=1
    COLUMN=1
    Main_Loop $1 1 $LAST_LINE
    ;;
esac
```