JAPLA 研究会資料 2009/6/27

２進法足し算の Windows グラフィックス

西川　利男

今月、６月１３日（土）に、日本技術史教育学会の年会、研究発表講演会で筆者は下記の講演発表を行なった。

「コンピュータの計算はなぜ２進法で行なうのか？」

例えば、筆者がサイエンスボランティアとして毎週出かけている日本科学未来館で、一番人気の「インターネット物理モデル」では、コンピュータの情報は白と黒のボールで表され、「コンピュータの中では、０と１つまり２進法ですべてが行なわれる」ということはよく知られている。しかし、なぜ２進法でなければならないのかについては、それほどよく理解されていない。

その理由のひとつが、実際に２進法で、例えば足し算を行なったことがないことにあると思う。Ｊではこのような実験が容易に可能であり、パソコンのキー入力により、さらには色つきの Windows グラフィックスとして、上の発表を行なうことが出来た。

ここでは、そのＪプログラムの詳細をお目にかける。

１．２進法の足し算とは

１０進数と２進数との関係は以下のようになっている。

0	1	2	3	4	5	6	7	8	9	10
0	1	10	11	100	101	110	111	1000	1001	1010

２進数の足し算では、演算操作は次のようにたった４種類だけである。

```
    0    0    1    1
    0    1    0    1
-------------------------
    0    1    1   10
```

ここでのポイントは1と1とのときに上の桁に桁上がり(carry)を行い、そして、次の桁ではこれも加えることである。この操作を最低桁から順次、各桁に対して行なえば、２進数の足し算が行なえることになる。

あらためて１０進法の足し算を見てみると、１０種類の数字に対し１００通りの演算操作があり、なお桁上がりを考慮する必要がある。足し算は小学算数の基本であるが、真に理解するにはいかに大変かがわかる。

一方、２進法の足し算は、これに比べてずっと簡単であり、コンピュータの電子回路で行なうコンピュータで採用になったのである。

上の演算を以下のように考えれば

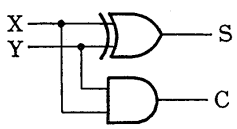
```
    X    0    0    1    1
    Y    0    1    0    1
----------------------------
    S    0    1    1    0
    C    0    0    0    1
```

S(結果)は X と Y との Exclusive OR 操作であり、
C(桁上がり)は X と Y との AND 操作である。

実際の内部ハードウェアとしては以下の次の半加算器(Half Adder)、全加算器(Full Adder)なる論理ＩＣでなされる。

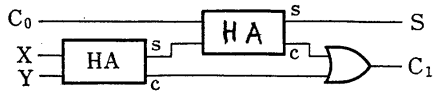
(a) 加算回路

半加算器（Half Adder）



X	0	1	0	1
Y	0	0	1	1
S	0	1	1	0
C	0	0	0	1

全加算器（Full Adder）



X	0	1	0	0	1	1	0	1
Y	0	0	1	0	1	0	1	1
C_0	0	0	0	1	0	1	1	1
S	0	1	1	1	0	0	0	1
C_1	0	0	0	0	1	1	1	1

２．Ｊによる２進法の足し算

まず、１０進数から２進数、逆に２進数から１０進数への変換は、Ｊでは例えば次のように簡単である。

```
   #: 7
1 1 1
   #: 13
1 1 0 1

   #. 1 1 1
7
   #. 1 1 0 1
13
```

２進法足し算の基本部分は、各桁に対する次の繰り返し操作である。

```
while. i > 0
  do.
    S =. (i{X) + (i{Y) + C
    select. S
      case. 0 do. T =. 0  [ C =. 0
      case. 1 do. T =. 1  [ C =. 0
      case. 2 do. T =. 0  [ C =. 1
      case. 3 do. T =. 1  [ C =. 1
    end.
    Z =. T, Z
    i =. i - 1
  end.
```

実際のプログラムaddは末尾のリストを参照のこと。なお、２進数掛け算のプログラムmultも合わせてあげた。

次のように実行される。

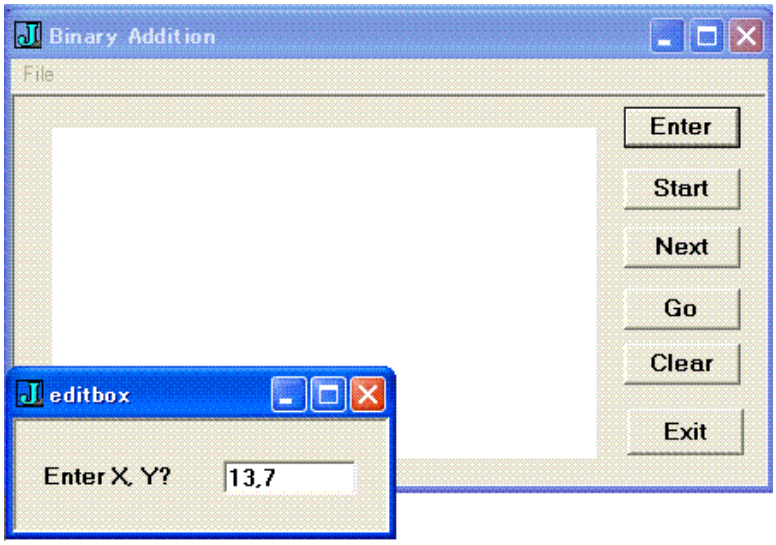
```
 13 add 7
X:0 0 1 1 0 1 (= 13 decimal)
Y:0 0 0 1 1 1 (= 7 decimal)
===============
step: 1
X:_ _ _ _ _ 1
Y:_ _ _ _ _ 1
C:_ _ _ _ _ 0
---------------
Z:_ _ _ _ 1 0 (Carry & Result)
===============
step: 2
X:_ _ _ _ 0 _
Y:_ _ _ _ 1 _
C:_ _ _ _ 1 _
---------------
Z:_ _ _ 1 0 _ (Carry & Result)
===============
step: 3
X:_ _ _ 1 _ _
Y:_ _ _ 1 _ _
C:_ _ _ 1 _ _
---------------
Z:_ _ 1 1 _ _ (Carry & Result)
===============
step: 4
X:_ _ 1 _ _ _
Y:_ _ 0 _ _ _
C:_ _ 1 _ _ _
---------------
Z:_ 1 0 _ _ _ (Carry & Result)
===============
step: 5
X:_ 0 _ _ _ _
Y:_ 0 _ _ _ _
C:_ 1 _ _ _ _
---------------
Z:0 1 _ _ _ _ (Carry & Result)

Final Result:
Z = X + Y
Z:0 1 0 1 0 0 (= 20 decimal)
```

```
===============
decimal calc.: 13 + 7 = 20
```

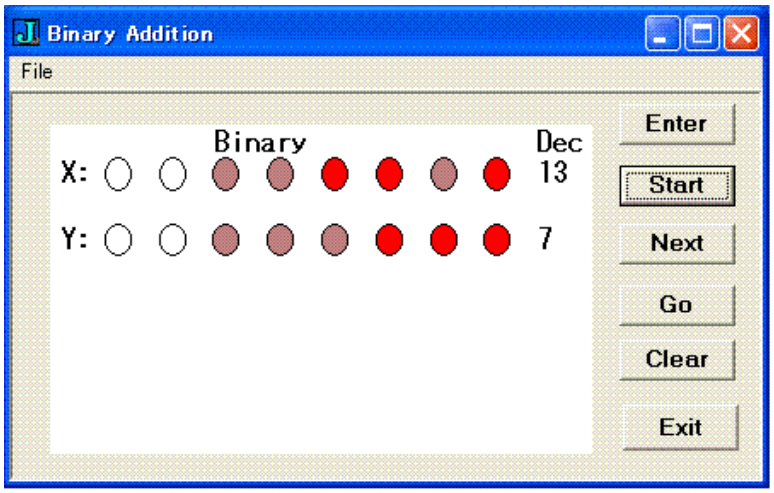
３．Ｊグラフィックスによる２進法の足し算

グラフィックスのためのＪのWindowsプログラミングでは、コード部分はそのままで良いが、入出力のフォームの作成が必要となる。　Form Editorでisigraphを指定して、かつ[Enter], [Start], [Next], [Go],..などのボタンを配置する。

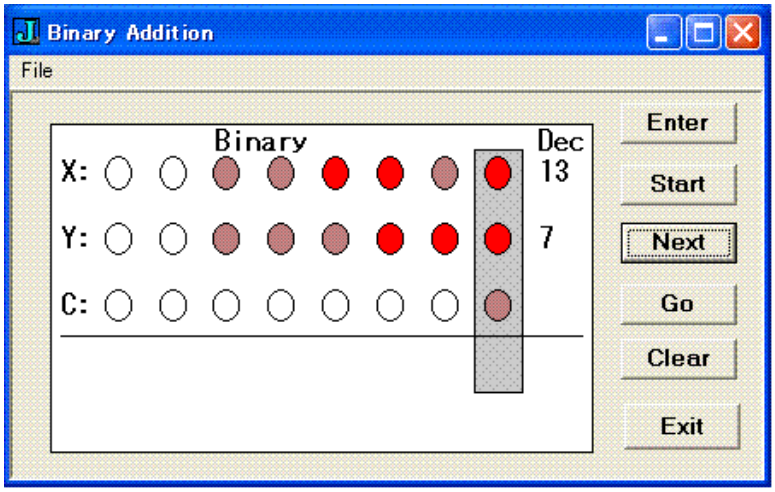


まずEnterボタンを押すと、数値入力の画面が現れデータが入力できる。

この操作は、基本のFormからさらに動的にFormを生成し、値を得るものだが、プログラムの詳細は末尾のリストを参照のこと。

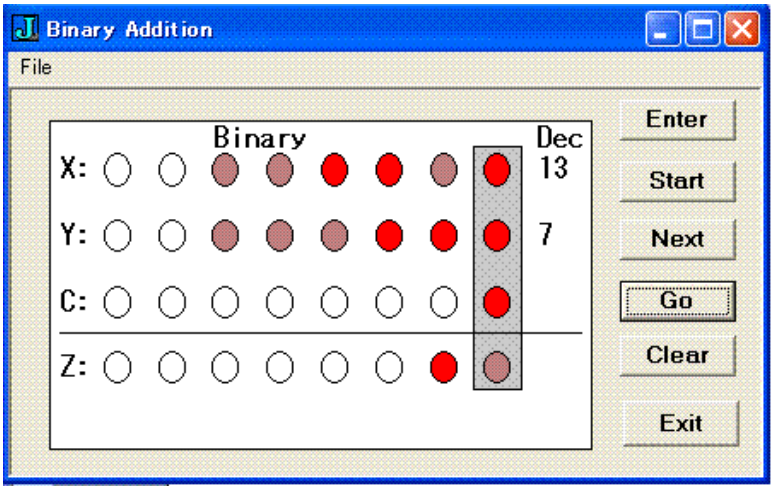


Startボタンを押すと、２進数に変換したスタート画面が表示される。

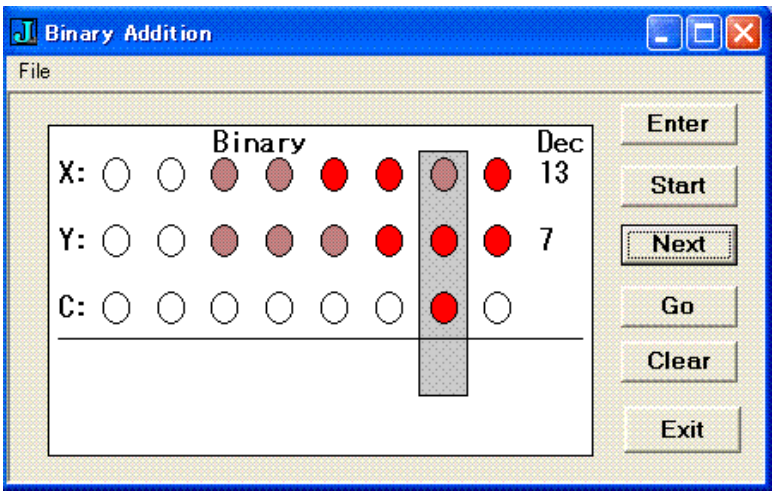


続いてNextボタンを押すと、演算を行なう最も右の桁を指示し、２進演算の準備を行なう。

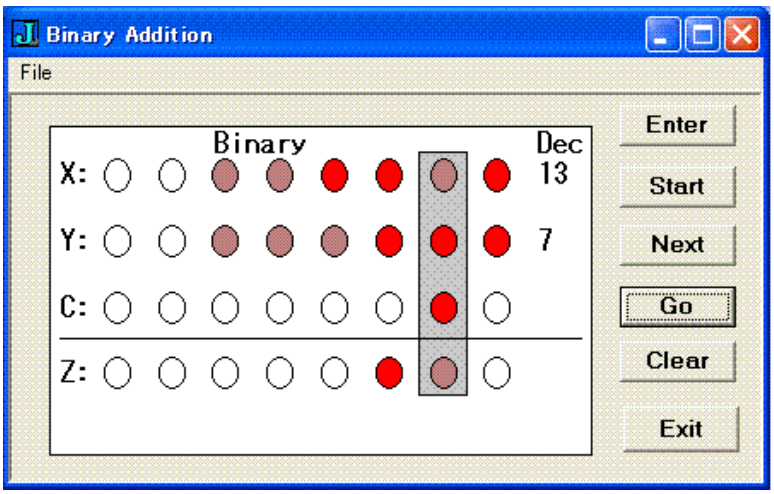
そしてGoボタンを押すと、その桁について２進数の足し算が行なわれる。



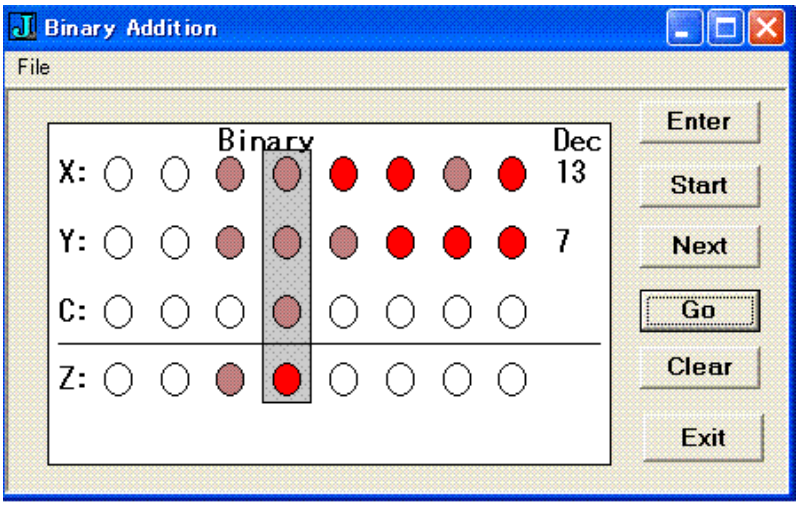
再び、Nextボタンを押すと、演算のための次の桁を準備指示する。



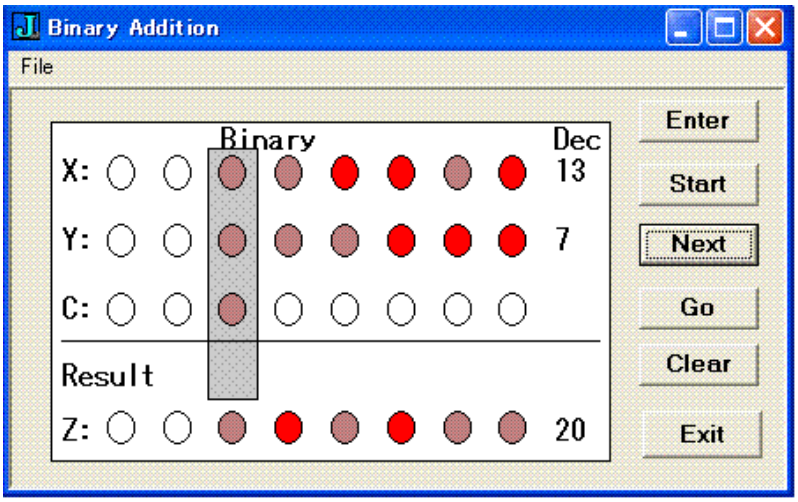
ここで、Goボタンを押すと、指示した桁で２進数の足し算を行なう。



このように、Go ボタンと Next ボタンとを交互に押して、桁を次々に上げて行く。



やがて、すべての桁の演算を終えると、結果は２進数とともに、１０進数でも表示される。



グラフィック Windows プログラムの詳細は、かなり長くなっているが、後に挙げる。Ｊ本来のコードというより、Basic にも似た手続き的なプログラムであるのでそれほど難しいことはない。

```
NB. Binary Addition 2008/12/13
add =: 3 : 0
:
N =. 2 + <. 2 ^. x. + y.
xb =. #: x.
yb =. #: y.
wr 'X:', (": X =. (-N) {. xb), ' (= ', (":x.), ' decimal)'
wr 'Y:', (": Y =. (-N) {. yb), ' (= ', (":y.), ' decimal)'
i =. N - 1
C =. 0
Z =. ''
while. i > 0
  do.
    wr '==============='
    wr 'step: ', ": N - i
    wr 'X:', ": ((i)#_), (i{X), ((N-1)-i)#_
    wr 'Y:', ": ((i)#_), (i{Y), ((N-1)-i)#_
    wr 'C:', ": ((i)#_),  C, ((N-1)-i)#_
    S =. (i{X) + (i{Y) + C
    select. S
      case. 0 do. T =. 0  [ C =. 0
      case. 1 do. T =. 1  [ C =. 0
      case. 2 do. T =. 0  [ C =. 1
      case. 3 do. T =. 1  [ C =. 1
    end.
    wr '---------------'
    wr 'Z:', (": ((i-1)#_), C, T, ((N-1)-i)#_ ), ' (Carry & Result)'
    yn =. rd 1
    if. 1 = #yn do. return. end.
    Z =. T, Z
    i =. i - 1
  end.
wr 'Final Result:'
wr 'Z = X + Y'
ZA =. C, Z
wr 'Z:', (": ZA =. C, Z), ' (= ', (":#. ZA), ' decimal)'
wr '===============  '
'decimal calc.: ', (":x.), ' + ', (":y.), ' = ', ": #. ZA
)


left_shift =: 4 : '(# y.) {. x. |. y. , x. # 0'

mult =: 3 : 0
:
```

```
N =. 8
L =. 4
NB. N =: 1 + <. 2 ^. x. + y.
xb =. #: x.
yb =. #: y.
wr 'M   :', (": M =. (-N) {. xb), ' (= ', (": x.), ' decimal)'
wr 'Y   :', (": Y =. (-N) {. yb), ' (= ', (": y.), ' decimal)'
i =. N - 1
j =. 0
C =. 0
Z =. (#Y)#0
while. j < L
  do.
    wr '====================='
    wr 'step: ', ": N - i
    if. 1 = (i{M)
      do.
        wr 'M=', (": (i{M))
        wr 'Y shifted by ', (": ((N-1) - i)), ' digits'
        wr '  MY:', ": T =. ((N-1) - i) left_shift Y
        wr '+)Z0:', ": Z
        wr '--------------------'
        wr '  Z :', ": Z =. Z b_add T
      else.
        wr 'M=', (": (i{M))
        wr 'skipped'
        wr 'Z   :', ": Z
      end.
    yn =. rd 1
    if. 1 = #yn do. return. end.
    i =. i - 1
    j =. j + 1
  end.
wr '=================='
wr 'Result:  ', ": Z =. C, Z
'decimal: ', ": #. Z
)

NB. Graphic Binary Add ================================
NB. 2009/4/4
NB. revised 4/8
require 'gl2'
require 'isigraph'

BINADD=: 0 : 0
```

```
pc binadd;pn "Binary Addition";
menupop "File";
menu new "&New" "" "" "";
menu open "&Open" "" "" "";
menusep ;
menu exit "&Exit" "" "" "";
menupopz;
xywh 179 81 34 12;cc cancel button;cn "Exit";
xywh 11 8 159 86;cc bindisplay isigraph;
xywh 178 50 34 11;cc Go button;
xywh 178 19 34 11;cc Start button;
xywh 178 34 34 11;cc Next button;
xywh 178 3 34 11;cc Enter button;
xywh 178 64 34 11;cc Clear button;
pas 6 6;pcenter;
rem form end;
)

run =: binadd_run
binadd_run=: 3 : 0
wd BINADD
NB. initialize form here
wd 'pshow;'
)

binadd_close=: 3 : 0
wd'pclose'
)

binadd_cancel_button=: 3 : 0
binadd_close''
)


bin_dis =: 3 : 0
:
'I J' =. x.
B =: y.
XX =: (I + 100*i.8)
YY =: 8#J
(XX,.YY) bidis B
)

bidis =: 3 : 0"(1 0)
:
```

```
'I J' =. x.
C =. y.
if. C = 0 do. glrgb 192 128 128 end.  NB. PINK
if. C = 1 do. glrgb 255 0 0 end.      NB. Red
if. C = _ do. glrgb 255 255 255 end.  NB. White
glbrush ''
glellipse I, J, 50, 100
)

binadd_ok_button=: 3 : 0
glfont '"FixedSys" 9 bold default'
NB. X:
gltextxy 20 900
gltext 'X:'
100 800 bin_dis _ _ _ 0 1 0 0 1
NB. Y:
gltextxy 20 700
gltext 'Y:'
100 600 bin_dis _ _ _ 0 1 1 1 0
NB. C:
gltextxy 20 500
gltext 'C:'
100 400 bin_dis _ _ _ 0 0 0 0 1
gllines 20 300 900 300
NB. Z:
gltextxy 20 200
gltext 'Z:'
100 100 bin_dis _ _ _ 0 0 0 0 1
glshow ''
)


binadd_Start_button=: 3 : 0
glfont '"FixedSys" 4 bold default'
gltextxy 20 997
gltext '          Binary               Dec'
glfont '"FixedSys" 9 bold default'
'XA YA' =: XYDA
NB. XA =. 6
NB. YA =. 7
NB. N =: 8
N =: 2 + <. 2 ^. XA + YA
xb =. #: XA
yb =. #: YA
NB. X:
```

```
gltextxy 20 900
gltext 'X:'
gltextxy 900 900
gltext ": XA
X =: (-N) {. xb
XXX =: ((8-N)#_), X
100 800 bin_dis XXX
NB. 100 800 bin_dis _ _ _ 0 1 0 0 1
NB. Y:
gltextxy 20 700
gltext 'Y:'
gltextxy 900 700
gltext ": YA
Y =: (-N) {. yb
YYY =: ((8-N)#_), Y
100 600 bin_dis YYY
glshow ''
NB. 100 600 bin_dis _ _ _ 0 1 1 1 0
NB. wr 'X:', (": X =. (-N) {. xb), ' (= ', (":x.), ' decimal)'
NB. wr 'Y:', (": Y =. (-N) {. yb), ' (= ', (":y.), ' decimal)'
i =: N - 1
C =: 0
NB. C =: (7#_), C
Z =: ''
ZZ =: 8#_
CC0 =: 8#_
)

binadd_Next_button=: 3 : 0
XX =. i{X
YY =. i{Y
glbrush glrgb 255 255 255
IC =. 780 - 100 * (N-(i+2))
NB. glrect IC, 180, 90, 740
glrect 0 0 1000 1000
glbrush glrgb 200 200 200   NB. Light Gray
IC =. 780 - 100 * (N-(i+1))
glrect IC, 180, 90, 740
glbkmode 1
glfont '"FixedSys" 4 bold default'
gltextxy 20 997
gltext '          Binary               Dec'
NB. gltext 'i=', (": i), ', X:', (": XX), ', Y:', (": YY), ' , C:', (": C)
gltextxy 900 900
gltext ": XA
```

```
gltextxy 900 700
gltext ″: YA
NB. X:
glfont '″FixedSys″ 9 bold default'
gltextxy 20 900
gltext 'X:'
100 800 bin_dis XXX
NB. Y:
gltextxy 20 700
gltext 'Y:'
100 600 bin_dis YYY
NB. C:
gltextxy 20 500
gltext 'C:'
CCC =: (i#_), C, ((N-1)-i)#_
C0 =: ((8-#CCC)#_), CCC
100 400 bin_dis C0
NB. Draw Line
gllines 20 360 980 360
NB. Final Result
if. i = 0
  do. gltextxy 20 300
      gltext 'Result'
      gltextxy 20 140
      gltext 'Z:'
      ZA =: C, Z
      ZR =. ((8-#ZA)#_), ZA
      100 50 bin_dis ZR
      gltextxy 900 140
      gltext (″: #. ZA)
  end.
glshow ''
)

binadd_Go_button=: 3 : 0
if. i = 0 do. return. end.
XX =. i{X
YY =. i{Y
glfont '″FixedSys″ 4 bold default'
gltextxy 20 997
gltext '          Binary               Dec'
NB. gltext 'i=', (″: i), ', X:', (″: XX), ', Y:', (″: YY), ' , C:', (″: C)
gltextxy 900 900
gltext ″: XA
gltextxy 900 700
```

```
gltext ″: YA
NB. X:
glfont '″FixedSys″ 9 bold default'
gltextxy 20 900
gltext 'X:'
NB. wr X
100 800 bin_dis XXX
NB. Y:
gltextxy 20 700
gltext 'Y:'
NB. wr Y
100 600 bin_dis YYY
NB. Draw Line
gllines 20 360 980 360
NB. Calculation
S =. (i{X) + (i{Y) + C
select. S
      case. 0 do. T =. 0  [ C =: 0
      case. 1 do. T =. 1  [ C =: 0
      case. 2 do. T =. 0  [ C =: 1
      case. 3 do. T =. 1  [ C =: 1
end.
NB. C:
gltextxy 20 500
gltext 'C:'
NB. wr C
CCC =: (i#_), C, ((N-1)-i)#_
C0 =: ((8-#CCC)#_), CCC
100 400 bin_dis C0
NB. Z:
ZZZ =. ((i-1)#_), C, T, ((N-1)-i)#_
Z0 =. ((8-#CCC)#_), ZZZ
NB. gltextxy 20 120
NB. gltext 'Z: ', (″: ((i-1)#_), C, T, ((N-1)-i)#_)
NB. Z:
gltextxy 20 300
gltext 'Z:'
100 200 bin_dis Z0
Z =: T, Z
NB. Final Result
goto_Next.
if. i = 2
  do. gltextxy 20 140
      gltext 'R:'
      ZA =: C, Z
```

```
      ZR =. ((8-#ZA)#_), ZA
      100 50 bin_dis ZR
      gltextxy 900 140
      gltext (": #. ZA)
  end.
label_Next.
i =: i - 1
glshow ''
)

binadd_Enter_button=: 3 : 0
wd 'pc editbox;'
wd 'xywh 8 11 50 8;cc s0 static;cn "Enter X, Y?";'
wd 'xywh 60 10 40 10;cc e0 edit ws_border;'
wd 'setfocus e0;'
wd 'pas 8 8;pcenter;pshow;wait;'
wd 'pclose;'
DA =. wd 'q;'
XYDA =: ". , >((<'e0') = 0{"(1) DA) # (i.#DA) { 1{"(1) DA
)

binadd_Clear_button=: 3 : 0
glclear ''
glshow ''
)
```