Feladat: Rendezzük egy lista elemeit növekvő sorrendbe

```
module proba
import StdEnv

Insert :: a [a] -> [a] | Ord a
Insert e [ ] = [e]
Insert e [x:xs]
        |e<=x    = [e, x:xs]
                 = [x: Insert e xs]

mysort [x:xs] = foldr Insert [x] xs

Start = mysort [5,3,1]
```



```
module proba
import StdEnv

Insert :: a [a] -> [a] | Ord a
Insert  e  [ ]  = [e]
Insert  e  [x:xs]
        |e<=x           = [e,x :xs]
                        = [x: Insert e xs]

mysort [x:xs] = foldr Insert [x] xs

Start = mysort [5,3,1]
```

Helyes-e ez a jelölés?

Insert xs_3 (Insert xs_2 (Insert xs_1 [x]))

Feladat: Keressük meg egy egész számokból álló lista legnagyobb elemét!

$g(0) = x_1$

$$g(i+1) = \begin{cases} x_{i+1} & \text{ha } x_{i+1} > g(i) \\ g(i) & \text{e.k.} \end{cases}$$

```
g :: [a] a ->a | Ord a

g [ ] y  = y
g [x:xs] y
          |x>y      =  g xs x
                    =  g xs y
myMax [x:xs] = g xs x

Start = myMax [1,3,2,7,4]
```

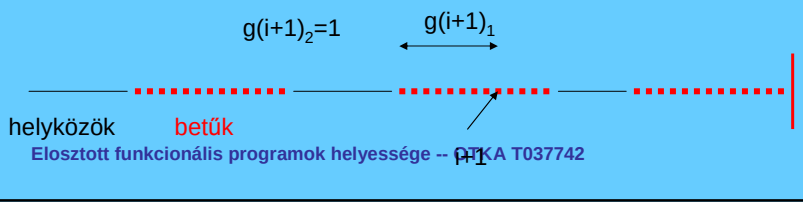

Feladat: Számoljuk meg, hány legalább k hosszúságú szó van a szövegben

Az aktuális szó hossza az adott pontig.

A k-nál hosszabb szavak darabszáma az adott pontig

$g(0) = (0 , 0)$

$$g(i+1) = \begin{cases} (g(i)_1+1 , g(i)_2) & \textbf{ha } \neg \textbf{ helyköz } (x_{i+1}) \\ (0 , g(i)_2+1) & \text{ha helyköz } (x_{i+1}) \text{ és } g(i)_1 \geq k \\ (0 , g(i)_2) & \text{ha helyköz } (x_{i+1}) \text{ és } g(i)_1 < k \end{cases}$$

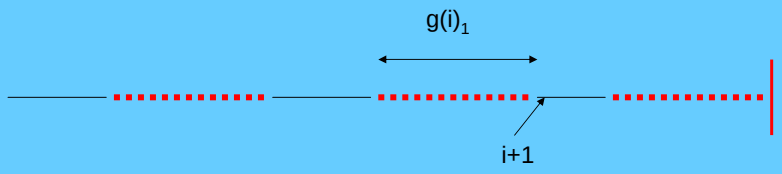


Aktuális szó hossza az adott pontig

a k-nál hosszabb szavak darabszáma az adott pontig

$$g(0) = (0 , 0)$$

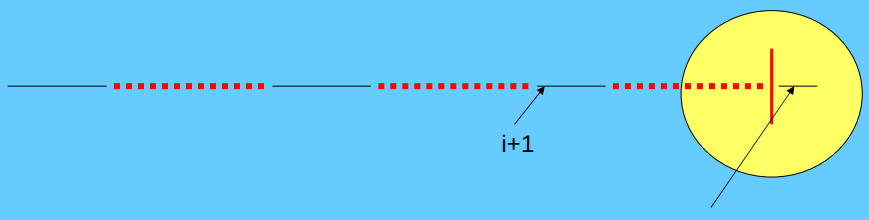
$$g(i+1) = \begin{cases} (g(i)_1+1 , g(i)_2) & \text{ha } \neg \text{ helyköz } (x_{i+1}) \\ (0 , g(i)_2+1) & \textbf{ha helyköz } (x_{i+1}) \textbf{ és } g(i)_1 \geq k \\ (0 , g(i)_2) & \textbf{ha helyköz } (x_{i+1}) \textbf{ és } g(i)_1 < k \end{cases}$$



Beleszámoltuk-e az utolsó szót?

$$g(0) = (0 , 0)$$

$$g(i+1) = \begin{cases} (g(i)_1+1 , g(i)_2) & \text{ha } \neg \text{ helyköz } (x_{i+1}) \\ (0 , g(i)_2+1) & \textbf{ha helyköz } (x_{i+1}) \textbf{ és } g(i)_1 \geq k \\ (0 , g(i)_2) & \text{ha helyköz } (x_{i+1}) \text{ és } g(i)_1 < k \end{cases}$$



A listához hozzáveszünk egy „extremális” elemet (egy helyközt), mert egyébként az utolsó k-nál hosszabb szót nem számolná bele.

g(0) = (0 , 0) ⟵

g(i+1) =
- $(g(i)_1+1 , g(i)_2)$ ha ¬ helyköz (x_{i+1})
- $(0 , g(i)_2+1)$ ha helyköz (x_{i+1}) és $g(i)_1 \geq k$
- $(0 , g(i)_2)$ ha helyköz (x_{i+1}) és $g(i)_1 < k$

```
g [ ] g1 g2 k = g2

g [x:xs] g1 g2 k
        | x<>' '                = g  xs  (g1+1)   g2       k
        | x==' ' && g1 >=k      = g  xs  0        (g2+1)   k
        | x==' ' && g1 <k       = g  xs  0         g2      k

count [x:xs] k = g ([x:xs]++[' ']) 0 0 k   ⟵

Start = count ['a','R','a','R',' ','a','R',' ','a','R','a',' '] 3
```

g(0) = (0 , 0)

g(i+1) =
- **$(g(i)_1+1 , g(i)_2)$ ha ¬ helyköz (x_{i+1})** ⟵
- $(0 , g(i)_2+1)$ ha helyköz (x_{i+1}) és $g(i)_1 \geq k$
- $(0 , g(i)_2)$ ha helyköz (x_{i+1}) és $g(i)_1 < k$

```
g [ ] g1 g2 k = g2

g [x:xs] g1 g2 k

        | x<>' '                = g  xs  (g1+1)   g2       k

        | x==' ' && g1 >=k      = g  xs  0        (g2+1)   k
        | x==' ' && g1 <k       = g  xs  0         g2      k

count [x:xs] k = g ([x:xs]++[' ']) 0 0 k

Start = count ['a','R','a','R',' ','a','R',' ','a','R','a',' '] 3
```

8

$g(0) = (0 , 0)$

$$g(i+1) = \begin{cases} (g(i)_1+1 , g(i)_2) & \text{ha } \neg \text{ helyköz } (x_{i+1}) \\ \mathbf{(0 , g(i)_2+1)} & \mathbf{\text{ha helyköz } (x_{i+1}) \text{ és } g(i)_1 \geq k} \\ (0 , g(i)_2) & \text{ha helyköz } (x_{i+1}) \text{ és } g(i)_1 < k \end{cases}$$

```
g [ ] g1 g2 k = g2

g [x:xs] g1 g2 k
        | x<>' '                = g  xs  (g1+1)   g2       k
        | x==' ' && g1 >=k      = g  xs  0        (g2+1)   k
        | x==' ' && g1 <k       = g  xs  0        g2       k

count [x:xs] k = g ([x:xs]++[' ']) 0 0 k

Start = count ['a','R','a','R',' ','a','R',' ','a','R','a',' '] 3
```

9

$g(0) = (0 , 0)$

$$g(i+1) = \begin{cases} (g(i)_1+1 , g(i)_2) & \text{ha } \neg \text{ helyköz } (x_{i+1}) \\ (0 , g(i)_2+1) & \text{ha helyköz } (x_{i+1}) \text{ és } g(i)_1 \geq k \\ \mathbf{(0 , g(i)_2)} & \mathbf{\text{ha helyköz } (x_{i+1}) \text{ és } g(i)_1 < k} \end{cases}$$

```
g [] g1 g2 k = g2

g [x:xs] g1 g2 k
        | x<>' '                = g  xs  (g1+1)   g2       k
        | x==' ' && g1 >=k      = g  xs  0        (g2+1)   k
        | x==' ' && g1 <k       = g  xs  0        g2       k

count [x:xs] k = g ([x:xs]++[' ']) 0 0 k

Start = count ['a','R','a','R',' ','a','R',' ','a','R','a',' '] 3
```

10

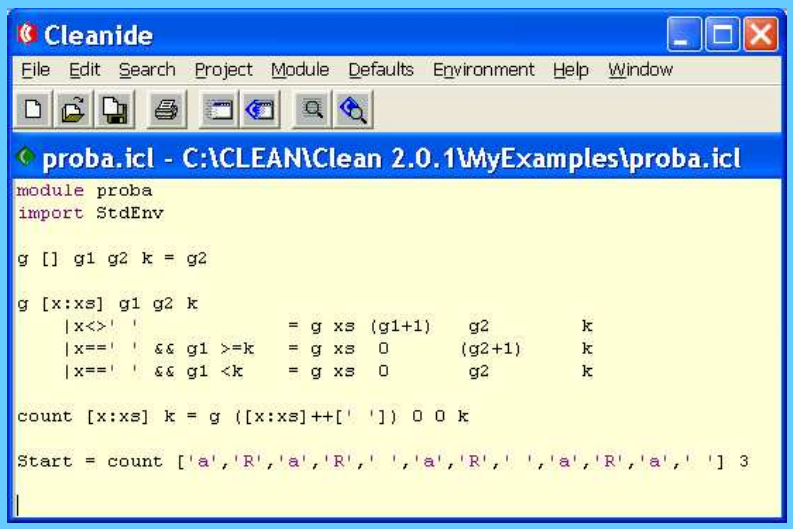
$$g(0) = (0 , 0)$$

$$g(i+1) = \begin{cases} (g(i)_1+1 , g(i)_2) & \text{ha } \neg \text{ helyköz } (x_{i+1}) \\ (0 , g(i)_2+1) & \text{ha helyköz } (x_{i+1}) \text{ és } g(i)_1 \geq k \\ (0 , g(i)_2) & \text{ha helyköz } (x_{i+1}) \text{ és } g(i)_1 < k \end{cases}$$

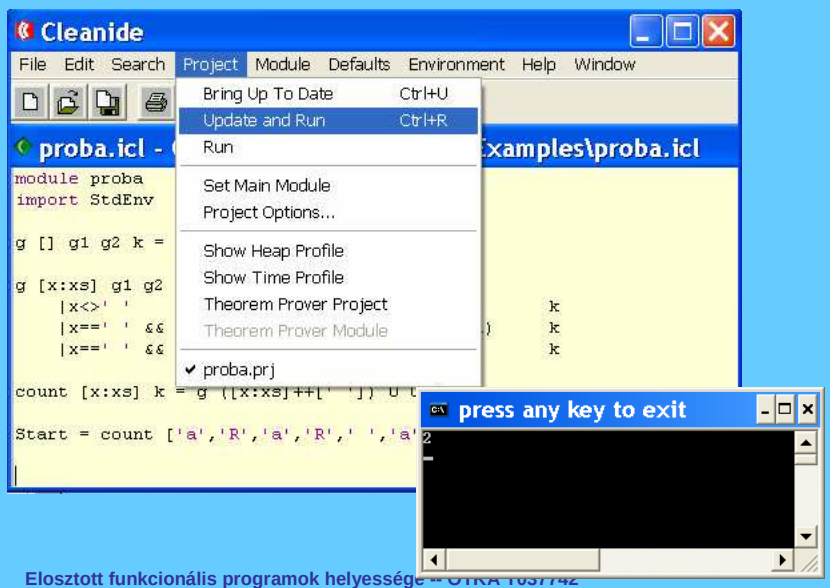
A végeredmény g2-ben van.

```
g [ ] g1 g2 k = g2

g [x:xs] g1 g2 k
        | x<>' '                = g  xs  (g1+1)   g2       k
        | x==' ' && g1 >=k      = g  xs  0        (g2+1)   k
        | x==' ' && g1 <k       = g  xs  0         g2      k

count [x:xs] k = g ([x:xs]++[' ']) 0 0 k

Start = count ['a','R','a','R',' ','a','R',' ','a','R','a',' '] 3
```


A program



Fordítás/Futtatás



Feladat: Számoljuk meg, hány legalább k hosszúságú szó van a szövegben

2. változat

Az aktuális szó hossza az adott pontig.

$$g(0) = 0$$

$$g(i+1) = \begin{cases} 0, & \text{ha helyköz } (x_{i+1}) \\ g(i)_1+1 & \text{ha } \neg\text{helyköz } (x_{i+1}) \end{cases}$$

g(i): 000000001234567890000000000012345600000000000123456 78

helyközök betűk

i+1

$$g(0) = 0$$

$$g(i+1) = \begin{cases} 0, & \text{ha helyköz } (x_{i+1}) \\ g(i)_1+1 & \text{ha } \neg\text{helyköz } (x_{i+1}) \end{cases}$$

```
g x y
  | x==' '  = 0
           = (y+1)

count g [ ] y k = 0
count g [x:xs] y k
          | g x y == k    = 1 + count g xs (g x y) k
                          = count g xs (g x y) k

Start = count g ( ['a','R','a','R',' ','a','R',' ','a','R','a']  ++[' ']) 0 5
```

15

Feladat:: Számoljuk meg, hány olyan szó van a szövegben, amely legalább k db 'R' betűt tartalmaz!

R betűk száma az aktuális szóban

Σ

$$g(0) = (0 \;,\; 0)$$

$$g(i+1) = \begin{cases} (g(i)_1+1 \;,\; g(i)_2) & \text{ha } x_{i+1}=\text{'R'} \\ (g(i)_1 \;,\; g(i)_2) & \text{ha } x_{i+1}\neq \text{'R' és } x_{i+1}\neq \text{' '} \\ (0 \;,\; g(i)_2+1) & \text{ha } x_{i+1}=\text{' ' és } g(i)_1 \geq k \\ (0 \;,\; g(i)_2) & \text{ha } x_{i+1}=\text{' ' és } g(i)_1 < k \end{cases}$$

$$g(0) = (0 , 0)$$

$$g(i+1) = \begin{cases} (g(i)_1+1 , g(i)_2) & \text{ha } x_{i+1}=\text{'R'} \\ (g(i)_1 , g(i)_2) & \text{ha } x_{i+1}\neq \text{'R' és } x_{i+1}\neq \text{' '} \\ (0 , g(i)_2+1) & \text{ha } x_{i+1}=\text{' ' és } g(i)_1 \geq k \\ (0 , g(i)_2) & \text{ha } x_{i+1}=\text{' ' és } g(i)_1 < k \end{cases}$$

```
g [] g1 g2 k = g2

g [x:xs] g1 g2 k
        |x=='R'          = g xs   (g1+1)   g2      k
        |x<>' ' && x<>'R' = g xs   g1       g2      k
        |x==' ' && g1>=k  = g xs   0        (g2+1)  k
        |x==' ' && g1<k   = g xs   0        g2      k

RS [x:xs] k = g ([x:xs] ++ [' ']) 0 0 k

Start =  RS ['a','R','a','R',' ','a','R','R','R',' ','a','R','R','R'] 3
```

17

$$g(0) = (0 , 0)$$

$$g(i+1) = \begin{cases} (g(i)_1+1 , g(i)_2) & \text{ha } x_{i+1}=\text{'R'} \\ (g(i)_1 , g(i)_2) & \text{ha } x_{i+1}\neq \text{'R' és } x_{i+1}\neq \text{' '} \\ (0 , g(i)_2+1) & \text{ha } x_{i+1}=\text{' ' és } g(i)_1 \geq k \\ (0 , g(i)_2) & \text{ha } x_{i+1}=\text{' ' és } g(i)_1 < k \end{cases}$$

Csak akkor számol, ha ez a feltétel teljesül, ezért be kell vezetni egy „extremális” elemet.

```
g [] g1 g2 k = g2

g [x:xs] g1 g2 k
        |x=='R'          = g xs   (g1+1)   g2      k
        |x<>' ' && x<>'R' = g xs   g1       g2      k
        |x==' ' && g1>=k  = g xs   0        (g2+1)  k
        |x==' ' && g1<k   = g xs   0        g2      k

RS [x:xs] k = g ([x:xs] ++ [' '] ) 0 0 k

Start =  RS ['a','R','a','R',' ','a','R','R','R',' ','a','R','R','R'] 3
```

18

Feladat:: Számoljuk meg, hány olyan szó van a szövegben, amely legalább k db 'R' betűt tartalmaz!

2. változat

```
g [] g1 g2 k
        |(g1>=k)    =(g2+1)
                    =g2

g [x:xs] g1 g2 k
        |x=='R'                 = g xs  (g1+1)  g2      k
        |x<>' ' && x<>'R'       = g xs  g1      g2      k
        |x==' ' && g1>=k        = g xs  0       (g2+1)  k
        |x==' ' && g1<k         = g xs  0       g2      k

RS [x:xs] k = g [x:xs] 0 0 k
Start =    RS ['a','R','a','R',' ','a','R','R','R',' ','a','R','R','R'] 3
```

Az utolsó elem megszámolása.

Feladat: Adott egy egész számokat tartalmazó lista. Ha a lista tartalmaz pozitív elemeket, akkor keressük meg a legnagyobbat, különben a legkisebbet

Igaz (↑), ha maximumot kell számolni, hamis(↓), ha minimumot

Ha g1 igaz, akkor ez a maximum, ha g1 hamis, akkor ez a minimum

$$
\begin{aligned}
g(0) &= (x_1>0 \;,\; x_1) \\
g(i+1) &= \begin{cases}
(g(i)_1 \;,\; \max(g(i)_2, x_{i+1})) & \text{ha } g(i)_1=\uparrow \\
(\uparrow \;,\; x_{i+1}) & \text{ha } \neg g(i)_1 \text{ és } x>0 \\
(g(i)_1 \;,\; \min(g(i)_2, x_{i+1})) & \text{ha } \neg g(i)_1 \text{ és } x\leq 0
\end{cases}
\end{aligned}
$$

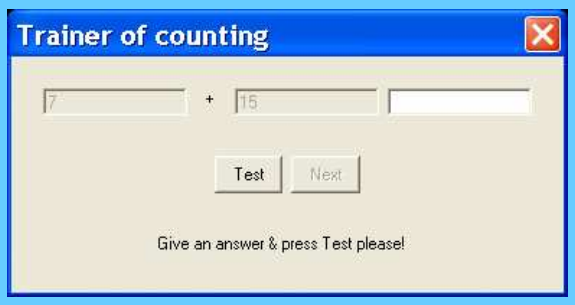
$g(0) = (x_1>0 \;,\; x_1)$

$$g(i+1) = \begin{cases} (g(i)_1 \;,\; \max(g(i)_2, x_{i+1})) & \text{ha } g(i)_1 = \uparrow \\ (\uparrow \;,\; x_{i+1}) & \text{ha } \neg g(i)_1 \text{ és } x>0 \\ (g(i)_1 \;,\; \min(g(i)_2, x_{i+1})) & \text{ha } \neg g(i)_1 \text{ és } x \leq 0 \end{cases}$$

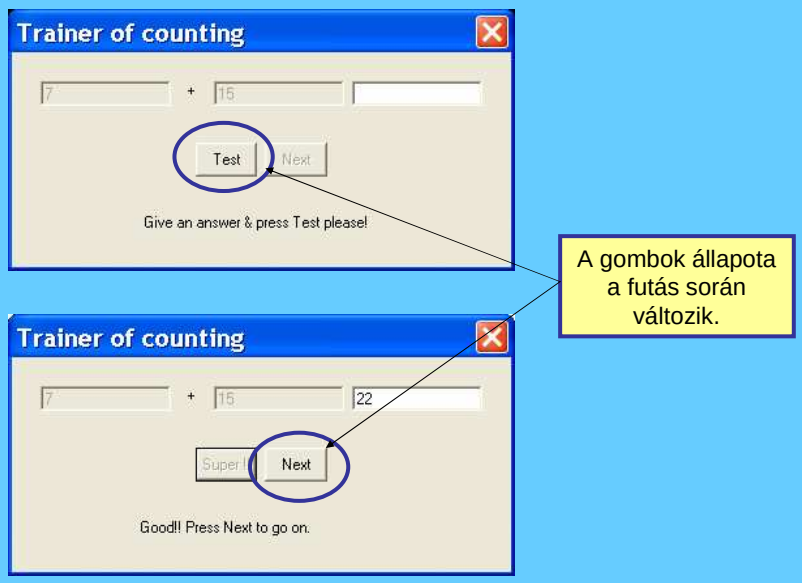
```
g [ ] g1 g2 = g2
g [x:xs] g1 g2
        |g1                     = g xs  g1      (Max g2 x)
        |(not g1) && (x>0)      = g xs  True    x
        |(not g1) && (x<=0)     = g xs  g1      (Min g2 x)
        where
                Max a b = if (a<b) b a
                Min  a b = if (a<b) a b
MinMax [x:xs] = g xs (x>0) x
Start = MinMax [-1,8,-12,-3,-5]
```

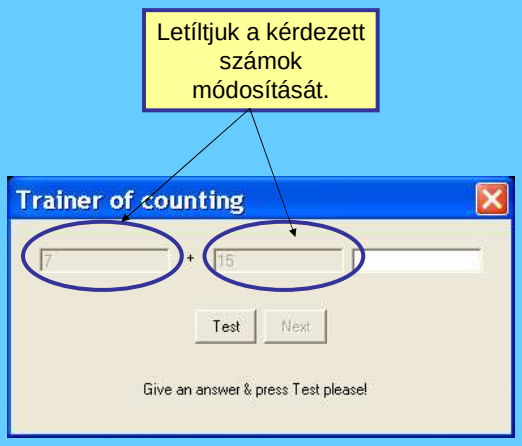

Egy GUI program

Feladat: Készítsünk el egy alkalmazást „összeadás gyakoroltatására.

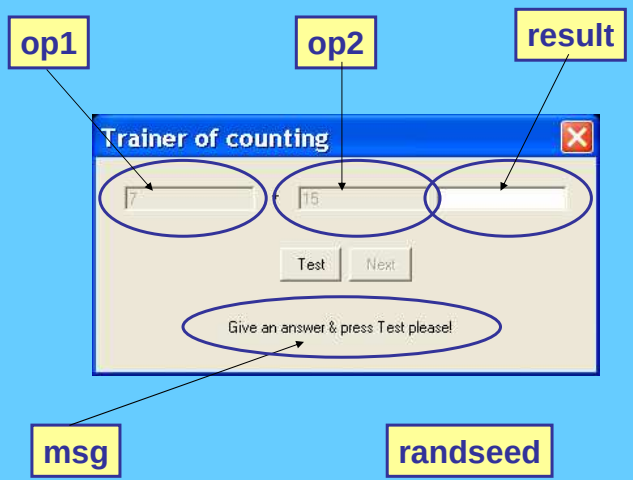


Cél: Gombok, adatbeviteli mezők megadása
Adatbeviteli mező adatának lekérdezése
Adatbeviteli mező módosításának tiltása/engedélyezése
Üzenetek kezelése
Gomb állapotának dinamikus módosítása

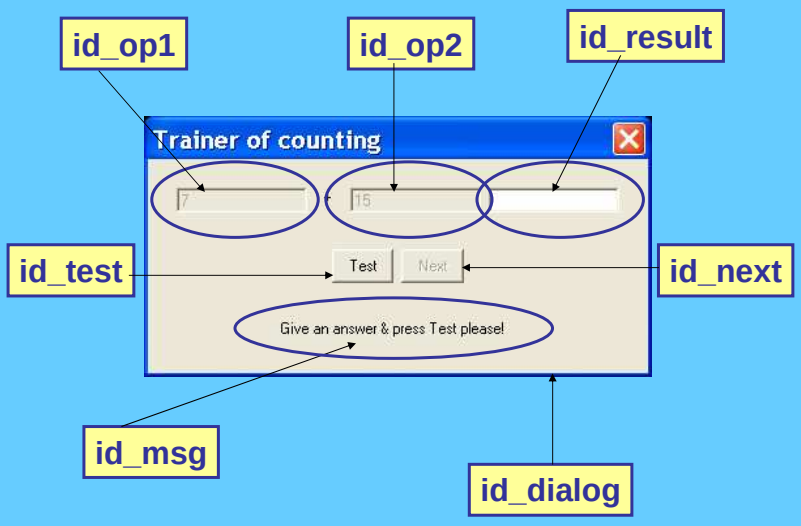



Letíltjuk a kérdezett számok módosítását.
Trainer of counting
7
+
15
Test
Next
Give an answer & press Test please!

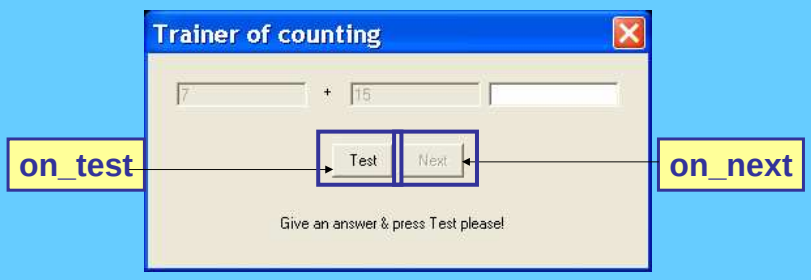
Local state


op1
op2
result
Trainer of counting
7
+
15
Test
Next
Give an answer & press Test please!
msg
randseed

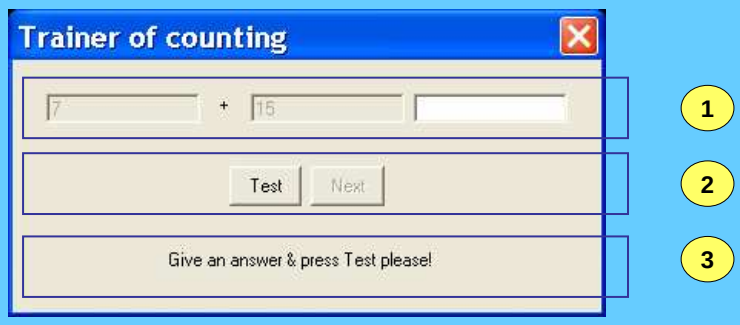
Control id



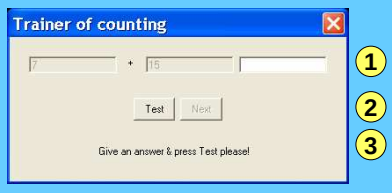
Control functions



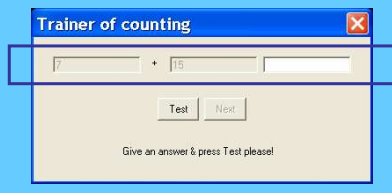
Layout controls



Layout controls



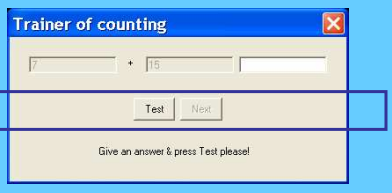

```
LayoutControl
(   EditControl …
:+: TextControl …
:+: EditControl …                    1
:+: EditControl …
) [ControlPos (Center,zero)]

:+: LayoutControl
(   ButtonControl                    2
:+: ButtonControl …
)[ControlPos (Center,zero)]

:+: LayoutControl
(   TextControl                                3
)           [ControlPos (Center,zero)]
```



1

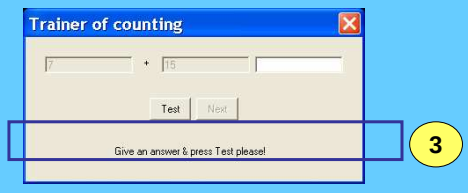
```
LayoutControl
(   EditControl x_str          displaywidth displayheight
            [           ControlId                   id_op1
            ,           ControlSelectState          Unable
            ]
:+: TextControl "  +  " []
:+: EditControl y_str          displaywidth displayheight
            [           ControlId                   id_op2
            ,           ControlSelectState          Unable
            ]
:+: EditControl ""   displaywidth displayheight
            [           ControlId                   id_result
            ,           ControlSelectState          Able
            ]
) [ControlPos (Center,zero)]
```

31



2

```
:+: LayoutControl
(   ButtonControl "Test"
            [           ControlFunction             onTest
            ,           ControlTip                  "Checks the result."
            ,           ControlId                   id_test
            ]
:+: ButtonControl "Next"
            [           ControlFunction             onNext
            ,           ControlSelectState          Unable
            ,           ControlTip                  "Gives a new task."
            ,           ControlId                   id_next
            ]
)           [ControlPos (Center,zero)]
```




```
:+: LayoutControl
(   TextControl init_msg
          [ ControlId                    id_msg ]
) [ControlPos (Center,zero)]
```


onNext

```
onNext :: (LocalSt,PSt .l) -> (LocalSt,PSt .l)
onNext ({op1,op2,result,msg},pst)
   # good              = toInt(op1)+toInt(op2)==toInt(result)
   # (randseed,pst) = getNewRandomSeed(pst)
   # (x,randseed)    = random randseed
   # (y,randseed)    = random randseed
   # x_str           = toString(x/500)
   # y_str           = toString(y/500)
   # op1             = if good x_str op1
   # op2             = if good y_str op2
   # result          = if good ""  result
   # msg             = if (toInt(result)==0) "Give an answer && press Test please! " ""
   # io              = disableControl id_next  pst.io
   # io              = enableControl id_test  io
   # io              = (setControlText id_test "Test" io)
                     = ({op1=op1,op2=op2,result=result,msg=msg,randseed}
                       ,appPIO (setControlTexts [(id_op1,op1)
                                                 ,(id_op2,op2)
                                                 ,(id_result,result)
                                                 ,(id_msg,msg)
                                                 ]) {pst & io= io} )
```

onNext

```
onNext :: (LocalSt,PSt .l) -> (LocalSt,PSt .l)
onNext ({op1,op2,result,msg},pst)
   # good              = toInt(op1)+toInt(op2)==toInt(result)
   # (randseed,pst) = getNewRandomSeed(pst)
   # (x,randseed)    = random randseed
   # (y,randseed)    = random randseed
   # x_str           = toString(x/500)
   # y_str           = toString(y/500)
   # op1             = if good x_str op1
   # op2             = if good y_str op2
   # result          = if good ""  result
   # msg             = if (toInt(result)==0) "Give an answer && press Test please! " ""
   # io              = disableControl id_next  pst.io
   # io              = enableControl id_test  io
   # io              = (setControlText id_test "Test" io)
                     = ({op1=op1,op2=op2,result=result,msg=msg,randseed}
                       ,appPIO (setControlTexts [(id_op1,op1)
                                                 ,(id_op2,op2)
                                                 ,(id_result,result)
                                                 ,(id_msg,msg)
                                                 ]) {pst & io= io} )
```

onNext

```
onNext :: (LocalSt,PSt .l) -> (LocalSt,PSt .l)
onNext ({op1,op2,result,msg},pst)
   # good              = toInt(op1)+toInt(op2)==toInt(result)
   # (randseed,pst) = getNewRandomSeed(pst)
   # (x,randseed)    = random randseed
   # (y,randseed)    = random randseed
   # x_str           = toString(x/500)
   # y_str           = toString(y/500)
   # op1             = if good x_str op1
   # op2             = if good y_str op2
   # result          = if good ""  result
   # msg             = if (toInt(result)==0) "Give an answer && press Test please! " ""
   # io              = disableControl id_next  pst.io
   # io              = enableControl id_test  io
   # io              = (setControlText id_test "Test" io)
                     = ({op1=op1,op2=op2,result=result,msg=msg,randseed}
                       ,appPIO (setControlTexts [(id_op1,op1)
                                                 ,(id_op2,op2)
                                                 ,(id_result,result)
                                                 ,(id_msg,msg)
                                                 ]) {pst & io= io} )
```

onNext

```
onNext :: (LocalSt,PSt .l) -> (LocalSt,PSt .l)
onNext ({op1,op2,result,msg},pst)
   # good               = toInt(op1)+toInt(op2)==toInt(result)
   # (randseed,pst) = getNewRandomSeed(pst)
   # (x,randseed)    = random randseed
   # (y,randseed)    = random randseed
   # x_str              = toString(x/500)
   # y_str              = toString(y/500)
   # op1                = if good x_str op1
   # op2                = if good y_str op2
   # result             = if good ""  result
   # msg                = if (toInt(result)==0) "Give an answer && press Test please! " ""
   # io                 = disableControl id_next  pst.io
   # io                 = enableControl id_test  io
   # io                 = (setControlText id_test "Test" io)
                        = ({op1=op1,op2=op2,result=result,msg=msg,randseed}
                          ,appPIO (setControlTexts [(id_op1,op1)
                                                   ,(id_op2,op2)
                                                   ,(id_result,result)
                                                   ,(id_msg,msg)
                                                   ]) {pst & io= io} )
```

onNext

```
onNext :: (LocalSt,PSt .l) -> (LocalSt,PSt .l)
onNext ({op1,op2,result,msg},pst)
   # good               = toInt(op1)+toInt(op2)==toInt(result)
   # (randseed,pst) = getNewRandomSeed(pst)
   # (x,randseed)    = random randseed
   # (y,randseed)    = random randseed
   # x_str              = toString(x/500)
   # y_str              = toString(y/500)
   # op1                = if good x_str op1
   # op2                = if good y_str op2
   # result             = if good ""  result
   # msg                = if (toInt(result)==0) "Give an answer && press Test please! " ""
   # io                 = disableControl id_next  pst.io
   # io                 = enableControl id_test  io
   # io                 = (setControlText id_test "Test" io)
                        = ({op1=op1,op2=op2,result=result,msg=msg,randseed}
                          ,appPIO (setControlTexts [(id_op1,op1)
                                                   ,(id_op2,op2)
                                                   ,(id_result,result)
                                                   ,(id_msg,msg)
                                                   ]) {pst & io= io} )
```

onNext

```
onNext :: (LocalSt,PSt .l) -> (LocalSt,PSt .l)
onNext ({op1,op2,result,msg},pst)
   # good               = toInt(op1)+toInt(op2)==toInt(result)
   # (randseed,pst) = getNewRandomSeed(pst)
   # (x,randseed)    = random randseed
   # (y,randseed)    = random randseed
   # x_str              = toString(x/500)
   # y_str              = toString(y/500)
   # op1                = if good x_str op1
   # op2                = if good y_str op2
   # result             = if good ""  result
   # msg              = if (toInt(result)==0) "Give an answer && press Test please! " ""
   # io                 = disableControl id_next     pst.io
   # io                 = enableControl id_test  io
   # io                 = (setControlText id_test "Test" io)
                        = ({op1=op1,op2=op2,result=result,msg=msg,randseed}
                          ,appPIO (setControlTexts [(id_op1,op1)
                                                   ,(id_op2,op2)
                                                   ,(id_result,result)
                                                   ,(id_msg,msg)
                                                   ]) {pst & io= io} )
```

“Visszatérési érték”

onTest

```
onTest :: (LocalSt,PSt .l) -> (LocalSt,PSt .l)
onTest ({op1,op2,result,msg,randseed},pst)
   # (wst,pst)  = accPIO (getParentWindow id_dialog) pst
   # result     = fromJust (snd (getControlText id_result (fromJust wst)))
   # good       = toInt(op1)+toInt(op2)==toInt(result)
   # io         = if good (enableControl id_next pst.io) (disableControl id_next pst.io)
   # io         = if good (disableControl id_test io) (enableControl id_test io)
   # io         = if good (setControlText id_test "Super !!" io)
                        (setControlText id_test "Test"  io)
   # msg        = if good  "Good!! Press Next to go on."
                      if (toInt(result)==0) "Give the answer first!" "Bad!! Try it again!"
                = ({op1=op1,op2=op2,result=result,msg=msg,randseed=randseed}
                    ,appPIO  (setControlTexts[(id_op1,op1)
                                             ,(id_op2,op2)
                                             ,(id_result,result)
                                             ,(id_msg,msg)
                                             ]) {pst & io= io})
```

onTest

```
onTest :: (LocalSt,PSt .l) -> (LocalSt,PSt .l)
onTest ({op1,op2,result,msg,randseed},pst)
   # (wst,pst)  = accPIO (getParentWindow id_dialog) pst
   # result     = fromJust (snd (getControlText id_result (fromJust wst)))
   # good       = toInt(op1)+toInt(op2)==toInt(result)
   # msg        = if good  "Good!! Press Next to go on."
    # io         = if good (enableControl id_next pst.io) (disableControl id_next pst.io)
   # io         = if good (disableControl id_test io) (enableControl id_test io)
   # io         = if good (setControlText id_test "Super !!" io)
                        (setControlText id_test "Test"  io)
                    if (toInt(result)==0) "Give the answer first!" "Bad!! Try it again!"
               = ({op1=op1,op2=op2,result=result,msg=msg,randseed=randseed}
                   ,appPIO  (setControlTexts[(id_op1,op1)
                                             ,(id_op2,op2)
                                             ,(id_result,result)
                                             ,(id_msg,msg)
                                             ]) {pst & io= io})
```


onTest

```
onTest :: (LocalSt,PSt .l) -> (LocalSt,PSt .l)
onTest ({op1,op2,result,msg,randseed},pst)
   # (wst,pst)  = accPIO (getParentWindow id_dialog) pst
   # result     = fromJust (snd (getControlText id_result (fromJust wst)))
   # good       = toInt(op1)+toInt(op2)==toInt(result)
   # msg          = if good  "Good!! Press Next to go on."
                    if (toInt(result)==0) "Give the answer first!" "Bad!! Try it again!"
   # io         = if good (enableControl id_next pst.io) (disableControl id_next pst.io)
   # io         = if good (disableControl id_test io) (enableControl id_test io)
   # io         = if good (setControlText id_test "Super !!" io)
                        (setControlText id_test "Test"  io)
               = ({op1=op1,op2=op2,result=result,msg=msg,randseed=randseed}
                   ,appPIO  (setControlTexts[(id_op1,op1)
                                             ,(id_op2,op2)
                                             ,(id_result,result)
                                             ,(id_msg,msg)
                                             ]) {pst & io= io})
```

Ez egy „else” ág!

onTest

```
onTest :: (LocalSt,PSt .l) -> (LocalSt,PSt .l)
onTest ({op1,op2,result,msg,randseed},pst)
   # (wst,pst)  = accPIO (getParentWindow id_dialog) pst
   # result     = fromJust (snd (getControlText id_result (fromJust wst)))
   # good       = toInt(op1)+toInt(op2)==toInt(result)
   # msg         = if good  "Good!! Press Next to go on."
                     if (toInt(result)==0) "Give the answer first!" "Bad!! Try it again!"
   # io          = if good (enableControl id_next pst.io) (disableControl id_next pst.io)
   # io          = if good (disableControl id_test io) (enableControl id_test io)
   # io          = if good (setControlText id_test "Super !!" io)
                           (setControlText id_test "Test"  io)
               = ({op1=op1,op2=op2,result=result,msg=msg,randseed=randseed}
                   ,appPIO  (setControlTexts[(id_op1,op1)
                                             ,(id_op2,op2)
                                             ,(id_result,result)
                                             ,(id_msg,msg)
                                             ]) {pst & io= io})
```

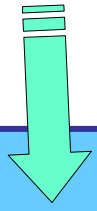

onTest

```
onTest :: (LocalSt,PSt .l) -> (LocalSt,PSt .l)
onTest ({op1,op2,result,msg,randseed},pst)
   # (wst,pst)  = accPIO (getParentWindow id_dialog) pst
   # result     = fromJust (snd (getControlText id_result (fromJust wst)))
   # good       = toInt(op1)+toInt(op2)==toInt(result)
   # msg         = if good  "Good!! Press Next to go on."
                     if (toInt(result)==0) "Give the answer first!" "Bad!! Try it again!"
   # io          = if good (enableControl id_next pst.io) (disableControl id_next pst.io)
   # io          = if good (disableControl id_test io) (enableControl id_test io)
   # io          = if good (setControlText id_test "Super !!" io)
                           (setControlText id_test "Test"  io)
               = ({op1=op1,op2=op2,result=result,msg=msg,randseed=randseed}
                   ,appPIO  (setControlTexts[(id_op1,op1)
                                             ,(id_op2,op2)
                                             ,(id_result,result)
                                             ,(id_msg,msg)
                                             ]) {pst & io= io})
```

“Visszatérési érték”

Start + LocalSt

```
module count
import StdEnv,StdIO,Random

Start :: *World -> *World
Start world
        = startIO NDI Void initIO [] world

::LocalSt
        ={op1:: String
          ,op2:: String
          , result::String
          , msg:: String
          ,randseed:: RandomSeed
          }
```



initIO

```
initIO pst
   # randseed = nullRandomSeed
   # (randseed,pst) = getNewRandomSeed(pst)
   # (id_dialog, pst)          = accPIO openId pst
   # (id_op1,pst)              = accPIO openId pst
   # (id_op2,pst)              = accPIO openId pst
   # (id_result,pst)           = accPIO openId pst
   # (id_msg,pst)              = accPIO openId pst
   # (id_next,pst)             = accPIO openId pst
   # (id_test,pst)             = accPIO openId pst
   # (_,pst)                   = openDialog Void (dialog id_dialog
                                                    id_op1
                                                    id_op2
                                                    id_result
                                                    id_msg
                                                    id_next
                                                    id_test
                                                    randseed)  pst
= pst
```

dialog

```
where
   dialog id_dialog id_op1 id_op2 id_result id_msg id_next id_test randseed
      #(x,randseed) = random randseed
      #(y,randseed) = random randseed
      # x_str       = toString(x/500)
      # y_str       = toString(y/500)
                    = Dialog "Trainer of counting"
                             { newLS  = {op1=x_str,op2=y_str,result="",
                                         msg=init_msg,randseed=randseed }
                             , newDef = LayoutControl
                               . . .
                             }
                             [   WindowClose   (noLS closeProcess)
                              ,  WindowId      id_dialog
                             ]
```