5.

Algorithm: serie1

Variable: T, S:real

Instructions:

T <- 1

S <- 0

for n from 1 to 20 do

T <- (-1) \* T \* (n + 1) / n

S <- S + T

endfor

write(S!)

Variable

T, S:real

Instructions

T <- 1

S <- 0

For n from 1 to 20 do

T <- (-1) \* (n + 1) / n \* T

S <- S + T

endFor

write("S is:", S!)

6.

Algorithm: serie2

Variable:

T, S: real

K: Integer

Instructions:

write("input K(integer >=0):")

read(KBD!K)

if K % 2 != 0 then

write("K is not definition!")

EXIT

endif

T <- 1

S <- 0

For n from 2 to K Step 2 do

T <- (-1) \* T \* (n + 2) / n

S <- S + T

endfor

write(S!)

Variable

T, S:real

K: Integer

Instructions

write("input K")

read(KBD!K)

T <- 1

S <- 0

For n from 2 to K Step 2 do

T <- (-1) \* (n + 2) / n \* T

S <- S + T

endFor

write("S is:", S!)

7.

(1)read the input text as String S

(2)apply a new empty String T

(3)foreach character C in S, check C if it is the upper letter; If it is true, T = T + c

(4)return T

1.读入字符串S，新建空字符串T

2.检查每一个S中的字符c，如果它是大写，就T = T + c

3.输出T

8.

Specifications

DATA:A String the user input it called S

RESULT:A String:all upper letter in the input text S

Specifications

DATA:一个用户输入的字符串S

RESULT:一个字符串，是S中的所有大写字母

9.

Algorithm: reverse String

Variable:

S, T:String

i,len:Integer

Instructions:

write("please input text:")

read(KBD!S)

T <- ""

len <- S.length()

for i from 0 to len - 1 step 1 do

T <- T + S[len-i-1]

endfor

write(T!)

Variable

T, S:String

p : integer

Instructions

write("input text:")

read(KBD!S)

T <- new String()

For p from 0 to S.length() - 1 step 1 do

T <- T + S[S.length() - i - 1]

endFor

write("the text is: ", S!)

10.

STRUCTURE ITEM

ID: Integer

data: DATE(day, month, year: Integer)

infectiousness: Integer

cases: Integer

death: Integer

level: Integer

ENDSTRUCTURE

STRUCTURE ITEM

id: Integer

Record DATE(day, month, year: Integer)

这里写四个自定义类型，有意义即可，比如level

ENDSTRUCTURE

11.

Algorithm: SAISIR

Variable:

item: ITEM

a, b, c: Integer

Instructions:

item <- new ITEM()

write("please input virus ID(A Integer):")

read(KBD!a)

item.ID <- a

write("please input virus Date(Three Integers):")

read(KBD!a,b,c)

item.data <- new DATE(a, b, c)

write("please input virus infectiousness(A Integer):")

read(KBD!a)

item.infectiousness <- a

write("please input virus cases(A Integer):")

read(KBD!a)

item.cases <- a

write("please input virus death(A Integer):")

read(KBD!a)

item.death <- a

write("please input virus level(A Integer):")

read(KBD!a)

item.level <- a

return ITEM

Variable:

item: ITEM

a, b, c: Integer

Instructions:

item <- new ITEM()

write("please input virus ID:")

read(KBD!a)

item.ID <- a

write("please input virus Date(Three Integers):")

read(KBD!a,KBD!b,KBD!c)

item.data <- DATE(a, b, c)

写四个你自己的属性

return ITEM

12.

Algorithm: SORT\_LIST\_ITEMS

Variable:

list: List

temp\_item: ITEM

min\_pos, i, j: Integer

Instructions

list <- get from System

for i from 1 to list.Number\_items - 1 step 1 do

min\_pos <- i

for j from i+1 to list.Number\_item step 1 do

if list.TAB[j].date is earlier than list.TAB[min\_pos].date then

min\_pos <- j

endif

endfor

temp\_item <- list.TAB[i]

list.TAB[i] <- list.TAB[min\_pos]

list.TAB[min\_pos] <- temp\_item

endfor

return list

Variable:

a: ITEM

j, I, position: Integer

Instructions

for i from 1 to list.Number\_items - 1do

position <- i

for j from i+1 to list.Number\_item do

if list.TAB[position].date later than list.TAB[i].date then

position <- j

endif

endfor

a <- list.TAB[i]

list.TAB[i] <- list.TAB[position]

list.TAB[position] <- a

endfor

return list

13.

Variable:

a: Integer

Function MODIFY(L:LIST, D:DATE, Begin:integer, End:integer)

if Begin > END then

return L

endif

if D is equal to L.TAB[Begin].date then

write("please input virus infectiousness(A Integer):")

read(KBD!a)

L.TAB[Begin].infectiousness <- a

write("please input virus cases(A Integer):")

read(KBD!a)

L.TAB[Begin].cases <- a

write("please input virus death(A Integer):")

read(KBD!a)

L.TAB[Begin].death <- a

write("please input virus level(A Integer):")

read(KBD!a)

L.TAB[Begin].level <- a

else

return MODIFY(L, D, Begin + 1, End)

endif

endFunction

Variable:

a: Integer

Function MODIFY(L:LIST, D:DATE, Begin:integer, End:integer)

if Begin > END then

return L

endif

if D is not equal to L.TAB[Begin].date then

return MODIFY(L, D, Begin + 1, End)

else

写四个键盘输入的属性

样例：

write("please input virus infectiousness(A Integer):")

read(KBD!a)

L.TAB[Begin].infectiousness <- a

endif

endFunction

14.

STRUCTURE ITEM

ID: Integer

data: DATE(day, month, year: Integer)

infectiousness: Integer

cases: Integer

death: Integer

level: Integer

next: ITEM

ENDSTRUCTURE

STRUCTURE ITEM

id: Integer

Record DATE(day, month, year: Integer)

这里写四个自定义类型，有意义即可，比如level

next: ITEM

ENDSTRUCTURE