```
;;;; #!/usr/bin/clisp
;;;;
;;;
(defpackage "C-STRING"
(:use "COMMON-LISP" )
(:nicknames "C")
(:export "STRCPY"
         "STRCAT"
         "SUBSTR?"
         "STRSTR"
         "STRSPL"
         "STRTOK"
         "STRRPL"
         "VERSION"
         "PRINT-OBJ"
         "SET-DEBUGED"
         ))
(in-package c-string)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;; "В чистом С много полезного:"
;;;; #ifndef
;(setf *debuged* t)
(defun set-debuged ( &optional true )
  "Включение трассировки отладочного режима"
  (if true
    (setf *debuged* t)
    (setf *debuged* nil)))
(defun print-obj ( o &optional s )
  "Трассировка: вывод объекта и строки пояснения"
  (when *debuged*
     (terpri)
     (prin1 (type-of s))
     (prin1 #\:)
     (prin1 s)
     (terpri)
     (prin1 (type-of o))
     (prin1 #\:)
     (prin1 o)
     (terpri))
      o)
;;;; #endif
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(defun version ()
  "0.4.1")
;;
(defun strcpy (str &key (start 0) (end (- (length str) 1)))
  "копирует str (от :start до :end) возвращает str[start:end]"
  (setf strlen (- end start))
  (when (>= strlen 0)
    (do ((j start (+ 1 j)) (i 0 (+ 1 i))
       (new (make-string (+ 1 strlen) :initial-element #\+ )))
      ((= i (+ 1 strlen)) new)
      (setf (char new i) (char str j)) )))

(defun strcat (str cat &key (start1 0) (end1 (- (length str) 1))
                            (start2 0) (end2 (- (length cat) 1)))
  "приписывает cat[s:e] к str[s:e]"
  (setf s1 (strcpy str :start start1 :end end1))
  (setf s2 (strcpy cat :start start2 :end end2))
  (setf l1 (length s1))
  (setf l2 (length s2))
  (setf l (+ (length s1) (length s2)))
        ;(print-obj l1)
        ;(print-obj l2)
        ;(print-obj s2)
        ;(print-obj l)
  (setf s0 (make-string l :initial-element #\+))
  (do ((j start1 (+ 1 j)) (i 0 (+ 1 i)))
```

```
       ((= i l1) s0)
       (setf (char s0 i) (char str j)))
  (do ((j start2 (+ 1 j)) (i l1 (+ 1 i)))
       ((= i l) s0)
       (setf (char s0 i) (char cat j))) )

(defun substr? ( sub str &key (start 0) )
  "Поиск подстроки в строке
      возврат nil или индекс (начало с 0 индекса)"
  (setf lst (length str)
        lsu (length sub)
        lraz (- lst lsu))
  (when (> lraz 0)
      (do ((j lraz (- j 1))
           (i start (+ 1 i)))
          ((or (string-equal str sub :start1 i  :end1 (+ lsu i))
               (= j start))
           (if (= i lraz) nil i)))))

(defun strstr ( str sub  )
  "Аналог substr?"
  (substr? sub str ))

(defun strrpl (str ct nw &optional (integer 1) )
  "Замена в str  ct на nw возвращается str"
        (print-obj "start strrpl")
        (print-obj nw)
  (setf *str!* str)
  (do ((j 0 (+ 1 j)) (s ""))
       ((or (= j integer) (eql nil (strstr *str!* ct))) s)
       (setf s (strtok *str!* ct))
       (setf s (strcat s nw))
       (setf s (strcat s *str!*))
       (setf *str!* s)))

(defun strspl (str &optional (ct " "))
  "Аналог split : str разбивается по ct; возвращается list"
  (setf v '())
  (setf *str!* str)
  (push (strtok *str!* ct) v)
  (push (strtok *str!* ct) v)
  (do ((j 1 (+ 1 j)))
      ((or (string-equal (first v ) (second v )) (= j 10))  v)
      (push (strtok *str!* ct) v)
      )
  (pop v)
  (push *str!* v)
  (reverse v))

(defun strtok (str! &optional (ct " "))
  "От str! отрезается 1я часть по ct и возвращается строкой; str изменяется"
  (setf n (strstr str! ct))
  (when n
    (setf s (strcpy str! :end (- n 1)))
    (setf *str!* (strcpy str! :start (+ n (length ct)))))
    s)
```