

Universidad Autónoma de Baja California
Facultad de Ingeniería, Arquitectura y Diseño



Programación Estructurada
Prof. Pedro Núñez Yepiz
Aarón Alejandro Parra Velarde
Actividad 9 - 3/4 - Anexos

Grupo: 432
Matrícula: 372193
Ensenada, B.C., 23 de Octubre de 2023

```

1 ///////////////////////////////////////////////////
2 /* Porro Velasco Aaron Alejandro 172191 */
3 /* 04/Oct/2023 */
4 /* Programa que genera el CURP de una persona. */
5 /* Pasa por 344,432,41 */
6 ///////////////////////////////////////////////////
7
8 #include <stdio.h>
9 #include <stdlib.h>
10 #include <time.h>
11 #include "arbol.h"
12
13 char inicializa(char apat[], char amat[], char nombre[], char CURP[]);
14 char fecha(char yr[], char mes[], char dia[], char CURP[]);
15 char sexo(int sx, char CURP[]);
16 int shat();
17 char letrastr(int est, char CURP[]);
18 char cons(char apat[], char amat[], char nombre[], char yr[], char CURP[]);
19 void muestra(char CURP[]);
20 char excep(char apat[], char amat[], char nombre[], char CURP[], int sx);
21 char mariajos(char cadena[], char nombre[], char CURP[]);
22 char busamos(char CURP[]);
23
24 int main()
25 {
26     char apat[30], amat[30], nombre[30], CURP[20], yr[5], mes[3], dia[3];
27     int opc, est, i, n, sx;
28     srand((time(NULL)));
29     do
30     {
31         system("CLS");
32         printf("\nApellido Paterno: ");
33         fflush(stdin);
34         gets(apat);
35         mayus(apat);
36
37         printf("\nEn caso de no tener segundo apellido, presione 0. ");
38         printf("\nApellido Materno: ");
39         gets(amat);
40         mayus(amat);
41
42         system("CLS");
43         printf("\nNombre(s): ");
44         gets(nombre);
45         mayus(nombre);
46
47         system("CLS");
48         printf("\nAño de nacimiento: ");
49         gets(yr);
50         printf("\nMes de nacimiento: ");
51         gets(mes);
52         printf("\nDía de nacimiento: ");
53         gets(dia);
54         system("CLS");
55
56         sx = valid_num(0,1,"Sexo (1-hombre, 0-mujer): ");
57
58         system("CLS");
59         est = shat();
60
61         inicializa(apat,amat,nombre,CURP);
62         fecha(yr,mes,dia,CURP);
63         sexo(sx,CURP);
64         letrastr(est,CURP);
65         cons(apat,amat,nombre,yr,CURP);
66         excep(apat,amat,nombre,CURP,sx);
67         muestra(CURP);
68         system("CLS");
69
70         if (1 == 0)
71         {
72             puts(CURP);
73         }
74         else
75         {
76             CURP[17] = CURP[18];
77             CURP[18] = '0';
78             puts(CURP);
79         }
80         system("Pause");
81         opc = valid_num(0,9999,"Para salir, presione 0. Para continuar, cualquier numero");
82         if (opc != 0)
83             while(opc != 0);
84         return 0;
85     }
86 }
87
88 char inicializa(char apat[], char amat[], char nombre[], char CURP[])
89 {
90     int i = 0, j = 0;
91     // apellido paterno
92     if(apat[0] != 0)
93     {
94         CURP[i] = apat[i];
95     }
96     else
97     {
98         CURP[i] = 'X';
99     }
100
101     if(amat[0] != 'A' || apat[0] != 'E' || apat[0] != 'I' || apat[0] != 'O' || apat[0] != 'U')
102     {
103         if(apat[i] == 'A' || apat[i] == 'E' || apat[i] == 'I' || apat[i] == 'O' || apat[i] == 'U')
104         {
105             CURP[i] = apat[i];
106         }
107         else
108         {
109             do
110             {
111                 CURP[i] = apat[i+1];
112                 i++;
113             }while(apat[i] != 'A' || apat[i] != 'E' || apat[i] != 'I' || apat[i] != 'O' || apat[i] != 'U');
114         }
115     }
116     else
117     {
118         do
119         {
120             CURP[i] = apat[i+1];
121             i++;
122         }while(apat[i] != 'A' || apat[i] != 'E' || apat[i] != 'I' || apat[i] != 'O' || apat[i] != 'U');
123     }
124
125     // apellido materno
126     if (amat[0] != 0)
127     {
128         CURP[i] = amat[i];
129     }
130     else
131     {
132         CURP[i] = 'X';
133     }
134
135     // nombre
136     CURP[i] = nombre[0];
137     return CURP[20];
138 }

```

```

1 char fecha(char yr[], char mes[], char dia[], char CURP[])
2 {
3     CURP[4] = yr[2];
4     CURP[5] = yr[3];
5
6     if(tam_cadena(mes) != 1)
7     {
8         CURP[6] = mes[0];
9         CURP[7] = mes[1];
10    }
11    else
12    {
13        CURP[6] = '0';
14        CURP[7] = mes[0];
15    }
16
17    if(tam_cadena(dia) != 1)
18    {
19        CURP[8] = dia[0];
20        CURP[9] = dia[1];
21    }
22    else
23    {
24        CURP[8] = '0';
25        CURP[9] = dia[0];
26    }
27    return CURP[20];
28 }
29
30 char sexo(int sx, char CURP[])
31 {
32     if (sx == 1)
33     {
34
35         CURP[10] = 'H';
36     }
37     else
38     {
39         CURP[10] = 'M';
40     }
41     return CURP[20];
42 }
43
44 int shtat()
45 {
46     int est;
47     printf("\nSi nacio en el extranjero, presione '0': ");
48     printf("\n1.-Aguascalientes");
49     printf("\n2.-Baja California");
50     printf("\n3.-Baja California Sur");
51     printf("\n4.-Campeche");
52     printf("\n5.-Chiapas");
53     printf("\n6.-Chihuahua");
54     printf("\n7.-Ciudad de Mexico");
55     printf("\n8.-Coahuila");
56     printf("\n9.-Colima");
57     printf("\n10.-Durango");
58     printf("\n11.-Guanajuato");
59     printf("\n12.-Guerrero");
60     printf("\n13.-Hidalgo");
61     printf("\n14.-Jalisco");
62     printf("\n15.-Estado de Mexico");
63     printf("\n16.-Michoacan");
64     printf("\n17.-Morelos");
65     printf("\n18.-Nayarit");
66     printf("\n19.-Nuevo Leon");
67     printf("\n20.-Oaxaca");
68     printf("\n21.-Puebla");
69     printf("\n22.-Queretaro");
70     printf("\n23.-Quintana Roo");
71     printf("\n24.-San Luis Potosi");
72     printf("\n25.-Sinaloa");
73     printf("\n26.-Sonora");
74     printf("\n27.-Tabasco");
75     printf("\n28.-Tamaulipas");
76     printf("\n29.-Tlaxcala");
77     printf("\n30.-Veracruz");
78     printf("\n31.-Yucatan");
79     printf("\n32.-Zacatecas");
80     est = valid_num(0,32,"Entidad de Nacimiento: ");
81     return est;
82 }

```

```

1  char *strdup(char *str)
2  {
3      if (!str)
4          return NULL;
5      int len = strlen(str);
6      char *p = (char *)malloc(len + 1);
7      if (!p)
8          return NULL;
9      memcpy(p, str, len);
10     p[len] = '\0';
11     return p;
12 }
13
14 char *strncpy(char *dest, const char *src, rsize_t n)
15 {
16     if (!src)
17         return dest;
18     if (!dest)
19         return NULL;
20     char *p = dest;
21     while (n-- > 0)
22     {
23         *p = *src;
24         p++;
25         src++;
26     }
27     *p = '\0';
28     return dest;
29 }
30
31 char *strncat(char *dest, const char *src, rsize_t n)
32 {
33     if (!src)
34         return dest;
35     if (!dest)
36         return NULL;
37     char *p = dest;
38     while (*p)
39         p++;
40     while (n-- > 0)
41     {
42         *p = *src;
43         p++;
44         src++;
45     }
46     *p = '\0';
47     return dest;
48 }
49
50 char *strnchr(const char *s, int c, rsize_t n)
51 {
52     if (!s)
53         return NULL;
54     char *p = s;
55     while (n-- > 0)
56     {
57         if (*p == c)
58             return p;
59         p++;
60     }
61     return NULL;
62 }
63
64 char *strnstr(const char *s, const char *t, rsize_t n)
65 {
66     if (!s || !t)
67         return NULL;
68     char *p = s;
69     char *q = t;
70     while (n-- > 0)
71     {
72         if (*p == *q)
73             q++;
74         p++;
75     }
76     if (*q == '\0')
77         return p - t;
78     return NULL;
79 }
80
81 char *strndup(const char *s, rsize_t n)
82 {
83     if (!s)
84         return NULL;
85     char *p = (char *)malloc(n + 1);
86     if (!p)
87         return NULL;
88     memcpy(p, s, n);
89     p[n] = '\0';
90     return p;
91 }
92
93 char *strnlen(const char *s, rsize_t n)
94 {
95     if (!s)
96         return 0;
97     char *p = s;
98     while (n-- > 0)
99     {
100         if (*p == '\0')
101             return p - s;
102         p++;
103     }
104     return p - s;
105 }
106
107 char *strnset(char *s, int c, rsize_t n)
108 {
109     if (!s)
110         return NULL;
111     char *p = s;
112     while (n-- > 0)
113         *p++ = c;
114     return s;
115 }
116
117 char *strnchr(const char *s, int c, rsize_t n)
118 {
119     if (!s)
120         return NULL;
121     char *p = s;
122     while (n-- > 0)
123     {
124         if (*p == c)
125             return p;
126         p++;
127     }
128     return NULL;
129 }
130
131 char *strnstr(const char *s, const char *t, rsize_t n)
132 {
133     if (!s || !t)
134         return NULL;
135     char *p = s;
136     char *q = t;
137     while (n-- > 0)
138     {
139         if (*p == *q)
140             q++;
141         p++;
142     }
143     if (*q == '\0')
144         return p - t;
145     return NULL;
146 }
147
148 char *strndup(const char *s, rsize_t n)
149 {
150     if (!s)
151         return NULL;
152     char *p = (char *)malloc(n + 1);
153     if (!p)
154         return NULL;
155     memcpy(p, s, n);
156     p[n] = '\0';
157     return p;
158 }
159
160 char *strnlen(const char *s, rsize_t n)
161 {
162     if (!s)
163         return 0;
164     char *p = s;
165     while (n-- > 0)
166     {
167         if (*p == '\0')
168             return p - s;
169         p++;
170     }
171     return p - s;
172 }
173
174 char *strnset(char *s, int c, rsize_t n)
175 {
176     if (!s)
177         return NULL;
178     char *p = s;
179     while (n-- > 0)
180         *p++ = c;
181     return s;
182 }
183
184 char *strnchr(const char *s, int c, rsize_t n)
185 {
186     if (!s)
187         return NULL;
188     char *p = s;
189     while (n-- > 0)
190     {
191         if (*p == c)
192             return p;
193         p++;
194     }
195     return NULL;
196 }
197
198 char *strnstr(const char *s, const char *t, rsize_t n)
199 {
200     if (!s || !t)
201         return NULL;
202     char *p = s;
203     char *q = t;
204     while (n-- > 0)
205     {
206         if (*p == *q)
207             q++;
208         p++;
209     }
210     if (*q == '\0')
211         return p - t;
212     return NULL;
213 }
214
215 char *strndup(const char *s, rsize_t n)
216 {
217     if (!s)
218         return NULL;
219     char *p = (char *)malloc(n + 1);
220     if (!p)
221         return NULL;
222     memcpy(p, s, n);
223     p[n] = '\0';
224     return p;
225 }
226
227 char *strnlen(const char *s, rsize_t n)
228 {
229     if (!s)
230         return 0;
231     char *p = s;
232     while (n-- > 0)
233     {
234         if (*p == '\0')
235             return p - s;
236         p++;
237     }
238     return p - s;
239 }
240
241 char *strnset(char *s, int c, rsize_t n)
242 {
243     if (!s)
244         return NULL;
245     char *p = s;
246     while (n-- > 0)
247         *p++ = c;
248     return s;
249 }
250
251 char *strnchr(const char *s, int c, rsize_t n)
252 {
253     if (!s)
254         return NULL;
255     char *p = s;
256     while (n-- > 0)
257     {
258         if (*p == c)
259             return p;
260         p++;
261     }
262     return NULL;
263 }
264
265 char *strnstr(const char *s, const char *t, rsize_t n)
266 {
267     if (!s || !t)
268         return NULL;
269     char *p = s;
270     char *q = t;
271     while (n-- > 0)
272     {
273         if (*p == *q)
274             q++;
275         p++;
276     }
277     if (*q == '\0')
278         return p - t;
279     return NULL;
280 }
281
282 char *strndup(const char *s, rsize_t n)
283 {
284     if (!s)
285         return NULL;
286     char *p = (char *)malloc(n + 1);
287     if (!p)
288         return NULL;
289     memcpy(p, s, n);
290     p[n] = '\0';
291     return p;
292 }
293
294 char *strnlen(const char *s, rsize_t n)
295 {
296     if (!s)
297         return 0;
298     char *p = s;
299     while (n-- > 0)
300     {
301         if (*p == '\0')
302             return p - s;
303         p++;
304     }
305     return p - s;
306 }
307
308 char *strnset(char *s, int c, rsize_t n)
309 {
310     if (!s)
311         return NULL;
312     char *p = s;
313     while (n-- > 0)
314         *p++ = c;
315     return s;
316 }
317
318 char *strnchr(const char *s, int c, rsize_t n)
319 {
320     if (!s)
321         return NULL;
322     char *p = s;
323     while (n-- > 0)
324     {
325         if (*p == c)
326             return p;
327         p++;
328     }
329     return NULL;
330 }
331
332 char *strnstr(const char *s, const char *t, rsize_t n)
333 {
334     if (!s || !t)
335         return NULL;
336     char *p = s;
337     char *q = t;
338     while (n-- > 0)
339     {
340         if (*p == *q)
341             q++;
342         p++;
343     }
344     if (*q == '\0')
345         return p - t;
346     return NULL;
347 }
348
349 char *strndup(const char *s, rsize_t n)
350 {
351     if (!s)
352         return NULL;
353     char *p = (char *)malloc(n + 1);
354     if (!p)
355         return NULL;
356     memcpy(p, s, n);
357     p[n] = '\0';
358     return p;
359 }
360
361 char *strnlen(const char *s, rsize_t n)
362 {
363     if (!s)
364         return 0;
365     char *p = s;
366     while (n-- > 0)
367     {
368         if (*p == '\0')
369             return p - s;
370         p++;
371     }
372     return p - s;
373 }
374
375 char *strnset(char *s, int c, rsize_t n)
376 {
377     if (!s)
378         return NULL;
379     char *p = s;
380     while (n-- > 0)
381         *p++ = c;
382     return s;
383 }
384
385 char *strnchr(const char *s, int c, rsize_t n)
386 {
387     if (!s)
388         return NULL;
389     char *p = s;
390     while (n-- > 0)
391     {
392         if (*p == c)
393             return p;
394         p++;
395     }
396     return NULL;
397 }
398
399 char *strnstr(const char *s, const char *t, rsize_t n)
400 {
401     if (!s || !t)
402         return NULL;
403     char *p = s;
404     char *q = t;
405     while (n-- > 0)
406     {
407         if (*p == *q)
408             q++;
409         p++;
410     }
411     if (*q == '\0')
412         return p - t;
413     return NULL;
414 }
415
416 char *strndup(const char *s, rsize_t n)
417 {
418     if (!s)
419         return NULL;
420     char *p = (char *)malloc(n + 1);
421     if (!p)
422         return NULL;
423     memcpy(p, s, n);
424     p[n] = '\0';
425     return p;
426 }
427
428 char *strnlen(const char *s, rsize_t n)
429 {
430     if (!s)
431         return 0;
432     char *p = s;
433     while (n-- > 0)
434     {
435         if (*p == '\0')
436             return p - s;
437         p++;
438     }
439     return p - s;
440 }
441
442 char *strnset(char *s, int c, rsize_t n)
443 {
444     if (!s)
445         return NULL;
446     char *p = s;
447     while (n-- > 0)
448         *p++ = c;
449     return s;
450 }
451
452 char *strnchr(const char *s, int c, rsize_t n)
453 {
454     if (!s)
455         return NULL;
456     char *p = s;
457     while (n-- > 0)
458     {
459         if (*p == c)
460             return p;
461         p++;
462     }
463     return NULL;
464 }
465
466 char *strnstr(const char *s, const char *t, rsize_t n)
467 {
468     if (!s || !t)
469         return NULL;
470     char *p = s;
471     char *q = t;
472     while (n-- > 0)
473     {
474         if (*p == *q)
475             q++;
476         p++;
477     }
478     if (*q == '\0')
479         return p - t;
480     return NULL;
481 }
482
483 char *strndup(const char *s, rsize_t n)
484 {
485     if (!s)
486         return NULL;
487     char *p = (char *)malloc(n + 1);
488     if (!p)
489         return NULL;
490     memcpy(p, s, n);
491     p[n] = '\0';
492     return p;
493 }
494
495 char *strnlen(const char *s, rsize_t n)
496 {
497     if (!s)
498         return 0;
499     char *p = s;
500     while (n-- > 0)
501     {
502         if (*p == '\0')
503             return p - s;
504         p++;
505     }
506     return p - s;
507 }
508
509 char *strnset(char *s, int c, rsize_t n)
510 {
511     if (!s)
512         return NULL;
513     char *p = s;
514     while (n-- > 0)
515         *p++ = c;
516     return s;
517 }
518
519 char *strnchr(const char *s, int c, rsize_t n)
520 {
521     if (!s)
522         return NULL;
523     char *p = s;
524     while (n-- > 0)
525     {
526         if (*p == c)
527             return p;
528         p++;
529     }
530     return NULL;
531 }
532
533 char *strnstr(const char *s, const char *t, rsize_t n)
534 {
535     if (!s || !t)
536         return NULL;
537     char *p = s;
538     char *q = t;
539     while (n-- > 0)
540     {
541         if (*p == *q)
542             q++;
543         p++;
544     }
545     if (*q == '\0')
546         return p - t;
547     return NULL;
548 }
549
550 char *strndup(const char *s, rsize_t n)
551 {
552     if (!s)
553         return NULL;
554     char *p = (char *)malloc(n + 1);
555     if (!p)
556         return NULL;
557     memcpy(p, s, n);
558     p[n] = '\0';
559     return p;
560 }
561
562 char *strnlen(const char *s, rsize_t n)
563 {
564     if (!s)
565         return 0;
566     char *p = s;
567     while (n-- > 0)
568     {
569         if (*p == '\0')
570             return p - s;
571         p++;
572     }
573     return p - s;
574 }
575
576 char *strnset(char *s, int c, rsize_t n)
577 {
578     if (!s)
579         return NULL;
580     char *p = s;
581     while (n-- > 0)
582         *p++ = c;
583     return s;
584 }
585
586 char *strnchr(const char *s, int c, rsize_t n)
587 {
588     if (!s)
589         return NULL;
590     char *p = s;
591     while (n-- > 0)
592     {
593         if (*p == c)
594             return p;
595         p++;
596     }
597     return NULL;
598 }
599
600 char *strnstr(const char *s, const char *t, rsize_t n)
601 {
602     if (!s || !t)
603         return NULL;
604     char *p = s;
605     char *q = t;
606     while (n-- > 0)
607     {
608         if (*p == *q)
609             q++;
610         p++;
611     }
612     if (*q == '\0')
613         return p - t;
614     return NULL;
615 }
616
617 char *strndup(const char *s, rsize_t n)
618 {
619     if (!s)
620         return NULL;
621     char *p = (char *)malloc(n + 1);
622     if (!p)
623         return NULL;
624     memcpy(p, s, n);
625     p[n] = '\0';
626     return p;
627 }
628
629 char *strnlen(const char *s, rsize_t n)
630 {
631     if (!s)
632         return 0;
633     char *p = s;
634     while (n-- > 0)
635     {
636         if (*p == '\0')
637             return p - s;
638         p++;
639     }
640     return p - s;
641 }
642
643 char *strnset(char *s, int c, rsize_t n)
644 {
645     if (!s)
646         return NULL;
647     char *p = s;
648     while (n-- > 0)
649         *p++ = c;
650     return s;
651 }
652
653 char *strnchr(const char *s, int c, rsize_t n)
654 {
655     if (!s)
656         return NULL;
657     char *p = s;
658     while (n-- > 0)
659     {
660         if (*p == c)
661             return p;
662         p++;
663     }
664     return NULL;
665 }
666
667 char *strnstr(const char *s, const char *t, rsize_t n)
668 {
669     if (!s || !t)
670         return NULL;
671     char *p = s;
672     char *q = t;
673     while (n-- > 0)
674     {
675         if (*p == *q)
676             q++;
677         p++;
678     }
679     if (*q == '\0')
680         return p - t;
681     return NULL;
682 }
683
684 char *strndup(const char *s, rsize_t n)
685 {
686     if (!s)
687         return NULL;
688     char *p = (char *)malloc(n + 1);
689     if (!p)
690         return NULL;
691     memcpy(p, s, n);
692     p[n] = '\0';
693     return p;
694 }
695
696 char *strnlen(const char *s, rsize_t n)
697 {
698     if (!s)
699         return 0;
700     char *p = s;
701     while (n-- > 0)
702     {
703         if (*p == '\0')
704             return p - s;
705         p++;
706     }
707     return p - s;
708 }
709
710 char *strnset(char *s, int c, rsize_t n)
711 {
712     if (!s)
713         return NULL;
714     char *p = s;
715     while (n-- > 0)
716         *p++ = c;
717     return s;
718 }
719
720 char *strnchr(const char *s, int c, rsize_t n)
721 {
722     if (!s)
723         return NULL;
724     char *p = s;
725     while (n-- > 0)
726     {
727         if (*p == c)
728             return p;
729         p++;
730     }
731     return NULL;
732 }
733
734 char *strnstr(const char *s, const char *t, rsize_t n)
735 {
736     if (!s || !t)
737         return NULL;
738     char *p = s;
739     char *q = t;
740     while (n-- > 0)
741     {
742         if (*p == *q)
743             q++;
744         p++;
745     }
746     if (*q == '\0')
747         return p - t;
748     return NULL;
749 }
750
751 char *strndup(const char *s, rsize_t n)
752 {
753     if (!s)
754         return NULL;
755     char *p = (char *)malloc(n + 1);
756     if (!p)
757         return NULL;
758     memcpy(p, s, n);
759     p[n] = '\0';
760     return p;
761 }
762
763 char *strnlen(const char *s, rsize_t n)
764 {
765     if (!s)
766         return 0;
767     char *p = s;
768     while (n-- > 0)
769     {
770         if (*p == '\0')
771             return p - s;
772         p++;
773     }
774     return p - s;
775 }
776
777 char *strnset(char *s, int c, rsize_t n)
778 {
779     if (!s)
780         return NULL;
781     char *p = s;
782     while (n-- > 0)
783         *p++ = c;
784     return s;
785 }
786
787 char *strnchr(const char *s, int c, rsize_t n)
788 {
789     if (!s)
790         return NULL;
791     char *p = s;
792     while (n-- > 0)
793     {
794         if (*p == c)
795             return p;
796         p++;
797     }
798     return NULL;
799 }
800
801 char *strnstr(const char *s, const char *t, rsize_t n)
802 {
803     if (!s || !t)
804         return NULL;
805     char *p = s;
806     char *q = t;
807     while (n-- > 0)
808     {
809         if (*p == *q)
810             q++;
811         p++;
812     }
813     if (*q == '\0')
814         return p - t;
815     return NULL;
816 }
817
818 char *strndup(const char *s, rsize_t n)
819 {
820     if (!s)
821         return NULL;
822     char *p = (char *)malloc(n + 1);
823     if (!p)
824         return NULL;
825     memcpy(p, s, n);
826     p[n] = '\0';
827     return p;
828 }
829
830 char *strnlen(const char *s, rsize_t n)
831 {
832     if (!s)
833         return 0;
834     char *p = s;
835     while (n-- > 0)
836     {
837         if (*p == '\0')
838             return p - s;
839         p++;
840     }
841     return p - s;
842 }
843
844 char *strnset(char *s, int c, rsize_t n)
845 {
846     if (!s)
847         return NULL;
848     char *p = s;
849     while (n-- > 0)
850         *p++ = c;
851     return s;
852 }
853
854 char *strnchr(const char *s, int c, rsize_t n)
855 {
856     if (!s)
857         return NULL;
858     char *p = s;
859     while (n-- > 0)
860     {
861         if (*p == c)
862             return p;
863         p++;
864     }
865     return NULL;
866 }
867
868 char *strnstr(const char *s, const char *t, rsize_t n)
869 {
870     if (!s || !t)
871         return NULL;
872     char *p = s;
873     char *q = t;
874     while (n-- > 0)
875     {
876         if (*p == *q)
877             q++;
878         p++;
879     }
880     if (*q == '\0')
881         return p - t;
882     return NULL;
883 }
884
885 char *strndup(const char *s, rsize_t n)
886 {
887     if (!s)
888         return NULL;
889     char *p = (char *)malloc(n + 1);
890     if (!p)
891         return NULL;
892     memcpy(p, s, n);
893     p[n] = '\0';
894     return p;
895 }
896
897 char *strnlen(const char *s, rsize_t n)
898 {
899     if (!s)
900         return 0;
901     char *p = s;
902     while (n-- > 0)
903     {
904         if (*p == '\0')
905             return p - s;
906         p++;
907     }
908     return p - s;
909 }
910
911 char *strnset(char *s, int c, rsize_t n)
912 {
913     if (!s)
914         return NULL;
915     char *p = s;
916     while (n-- > 0)
917         *p++ = c;
918     return s;
919 }
920
921 char *strnchr(const char *s, int c, rsize_t n)
922 {
923     if (!s)
924         return NULL;
925     char *p = s;
926     while (n-- > 0)
927     {
928         if (*p == c)
929             return p;
930         p++;
931     }
932     return NULL;
933 }
934
935 char *strnstr(const char *s, const char *t, rsize_t n)
936 {
937     if (!s || !t)
938         return NULL;
939     char *p = s;
940     char *q = t;
941     while (n-- > 0)
942     {
943         if (*p == *q)
944             q++;
945         p++;
946     }
947     if (*q == '\0')
948         return p - t;
949     return NULL;
950 }
951
952 char *strndup(const char *s, rsize_t n)
953 {
954     if (!s)
955         return NULL;
956     char *p = (char *)malloc(n + 1);
957     if (!p)
958         return NULL;
959     memcpy(p, s, n);
960     p[n] = '\0';
961     return p;
962 }
963
964 char *strnlen(const char *s, rsize_t n)
965 {
966     if (!s)
967         return 0;
968     char *p = s;
969     while (n-- > 0)
970     {
971         if (*p == '\0')
972             return p - s;
973         p++;
974     }
975     return p - s;
976 }
977
978 char *strnset(char *s, int c, rsize_t n)
979 {
980     if (!s)
981         return NULL;
982     char *p = s;
983     while (n-- > 0)
984         *p++ = c;
985     return s;
986 }
987
988 char *strnchr(const char *s, int c, rsize_t n)
989 {
990     if (!s)
991         return NULL;
992     char *p = s;
993     while (n-- > 0)
994     {
995         if (*p == c)
996             return p;
997         p++;
998     }
999     return NULL;
1000 }
1001
1002 char *strnstr(const char *s, const char *t, rsize_t n)
1003 {
1004     if (!s || !t)
1005         return NULL;
1006     char *p = s;
1007     char *q = t;
1008     while (n-- > 0)
1009     {
1010         if (*p == *q)
1011             q++;
1012         p++;
1013     }
1014     if (*q == '\0')
1015         return p - t;
1016     return NULL;
1017 }
1018
1019 char *strndup(const char *s, rsize_t n)
1020 {
1021     if (!s)
1022         return NULL;
1023     char *p = (char *)malloc(n + 1);
1024     if (!p)
1025         return NULL;
1026     memcpy(p, s, n);
1027     p[n] = '\0';
1028     return p;
1029 }
1030
1031 char *strnlen(const char *s, rsize_t n)
1032 {
1033     if (!s)
1034         return 0;
1035     char *p = s;
1036     while (n-- > 0)
1037     {
1038         if (*p == '\0')
1039             return p - s;
1040         p++;
1041     }
1042     return p - s;
1043 }
1044
1045 char *strnset(char *s, int c, rsize_t n)
1046 {
1047     if (!s)
1048         return NULL;
1049     char *p = s;
1050     while (n-- > 0)
1051         *p++ = c;
1052     return s;
1053 }
1054
1055 char *strnchr(const char *s, int c, rsize_t n)
1056 {
1057     if (!s)
1058         return NULL;
1059     char *p = s;
1060     while (n-- > 0)
1061     {
1062         if (*p == c)
1063             return p;
1064         p++;
1065     }
1066     return NULL;
1067 }
1068
1069 char *strnstr(const char *s, const char *t, rsize_t n)
1070 {
1071     if (!s || !t)
1072         return NULL;
1073     char *p = s;
1074     char *q = t;
1075     while (n-- > 0)
1076     {
1077         if (*p == *q)
1078             q++;
1079         p++;
1080     }
1081     if (*q == '\0')
1082         return p - t;
1083     return NULL;
1084 }
1085
1086 char *strndup(const char *s, rsize_t n)
1087 {
1088     if (!s)
1089         return NULL;
1090     char *p = (char *)malloc(n + 1);
1091     if (!p)
1092         return NULL;
1093     memcpy(p, s, n);
1094     p[n] = '\0';
1095     return p;
1096 }
1097
1098 char *strnlen(const char *s, rsize_t n)
1099 {
1100     if (!s)
1101         return 0;
1102     char *p = s;
1103     while (n-- > 0)
1104     {
1105         if (*p == '\0')
1106             return p - s;
1107         p++;
1108     }
1109     return p - s;
1110 }
1111
1112 char *strnset(char *s, int c, rsize_t n)
1113 {
1114     if (!s)
1115         return NULL;
1116     char *p = s;
1117     while (n-- > 0)
1118         *p++ = c;
1119     return s;
1120 }
1121
1122 char *strnchr(const char *s, int c, rsize_t n)
1123 {
1124     if (!s)
1125         return NULL;
1126     char *p = s;
1127     while (n-- > 0)
1128     {
1129         if (*p == c)
1130             return p;
1131         p++;
1132     }
1133     return NULL;
1134 }
1135
1136 char *strnstr(const char *s, const char *t, rsize_t n)
1137 {
1138     if (!s || !t)
1139         return NULL;
1140     char *p = s;
1141     char *q = t;
1142     while (n-- > 0)
1143     {
1144         if (*p == *q)
1145             q++;
1146         p++;
1147     }
1148     if (*q == '\0')
1149         return p - t;
1150     return NULL;
1151 }
1152
1153 char *strndup(const char *s, rsize_t n)
1154 {
1155     if (!s)
1156         return NULL;
1157     char *p = (char *)malloc(n + 1);
1158     if (!p)
1159         return NULL;
1160     memcpy(p, s, n);
1161     p[n] = '\0';
1162     return p;
1163 }
1164
1165 char *strnlen(const char *s, rsize_t n)
1166 {
1167     if (!s)
1168         return 0;
1169     char *p = s;
1170     while (n-- > 0)
1171     {
1172         if (*p == '\0')
1173             return p - s;
1174         p++;
1175     }
1176     return p - s;
1177 }
1178
1179 char *strnset(char *s, int c, rsize_t n)
1180 {
1181     if (!s)
1182         return NULL;
1183     char *p = s;
1184     while (n-- > 0)
1185         *p++ = c;
1186     return s;
1187 }
1188
1189 char *strnchr(const char *s, int c, rsize_t n)
1190 {
1191     if (!s)
1192         return NULL;
1193     char *p = s;
1194     while (n-- > 0)
1195     {
1196         if (*p == c)
1197             return p;
1198         p++;
1199     }
1200     return NULL;
1201 }
1202
1203 char *strnstr(const char *s, const char *t, rsize_t n)
1204 {
1205     if (!s || !t)
1206         return NULL;
1207     char *p = s;
1208     char *q = t;
1209     while (n-- > 0)
1210     {
1211         if (*p == *q)
1212             q++;
1213         p++;
1214     }
1215     if (*q == '\0')
1216         return p - t;
1217     return NULL;
1218 }
1219
1220 char *strndup(const char *s, rsize_t n)
1221 {
1222     if (!s)
1223         return NULL;
1224     char *p = (char *)malloc(n + 1);
1225     if (!p)
1226         return NULL;
1227     memcpy(p, s, n);
1228     p[n] = '\0';
1229     return p;
1230 }
1231
1232 char *strnlen(const char *s, rsize_t n)
1233 {
1234     if (!s)
1235         return 0;
1236     char *p = s;
1237     while (n-- > 0)
1238     {
1239         if (*p == '\0')
1240             return p - s;
1241         p++;
1242     }
1243     return p - s;
1244 }
1245
1246 char *strnset(char *s, int c, rsize_t n)
1247 {
1248     if (!s)
1249         return NULL;
1250     char *p = s;
1251     while (n-- > 0)
1252         *p++ = c;
1253     return s;
1254 }
1255
1256 char *strnchr(const char *s, int c, rsize_t n)
1257 {
1258     if (!s)
1259         return NULL;
1260     char *p = s;
1261     while (n-- > 0)
1262     {
1263         if (*p == c)
1264             return p;
1265         p++;
1266     }
1267     return NULL;
1268 }
1269
1270 char *strnstr(const char *s, const char *t, rsize_t n)
1271 {
1272     if (!s || !t)
1273         return NULL;
1274     char *p = s;
1275     char *q = t;
1276     while (n-- > 0)
1277     {
1278         if (*p == *q)
1279             q++;
1280         p++;
1281     }
1282     if (*q == '\0')
1283         return p - t;
1284     return NULL;
1285 }
1286
1287 char *strndup(const char *s, rsize_t n)
1288 {
1289     if (!s)
1290         return NULL;
1291     char *p = (char *)malloc(n + 1);
1292     if (!p)
1293         return NULL;
1294     memcpy(p, s, n);
1295     p[n
```


FXL0970131HGRRPM02

Presione una tecla para continuar . . .

CAXC020328HNEHWA7

Presione una tecla para continuar . . .

LOXA071228HYNMXNA4

Presione una tecla para continuar . . .