

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

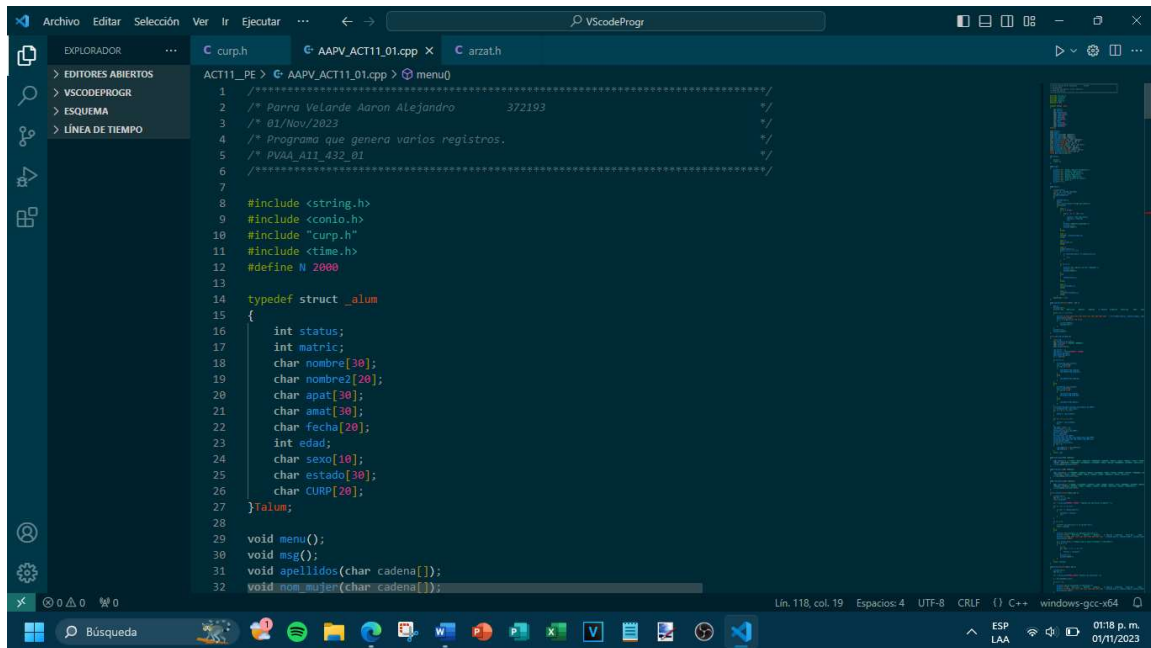


Programación Estructurada
Prof. Pedro Núñez Yepiz
Aarón Alejandro Parra Velarde
Actividad 11 - Anexos

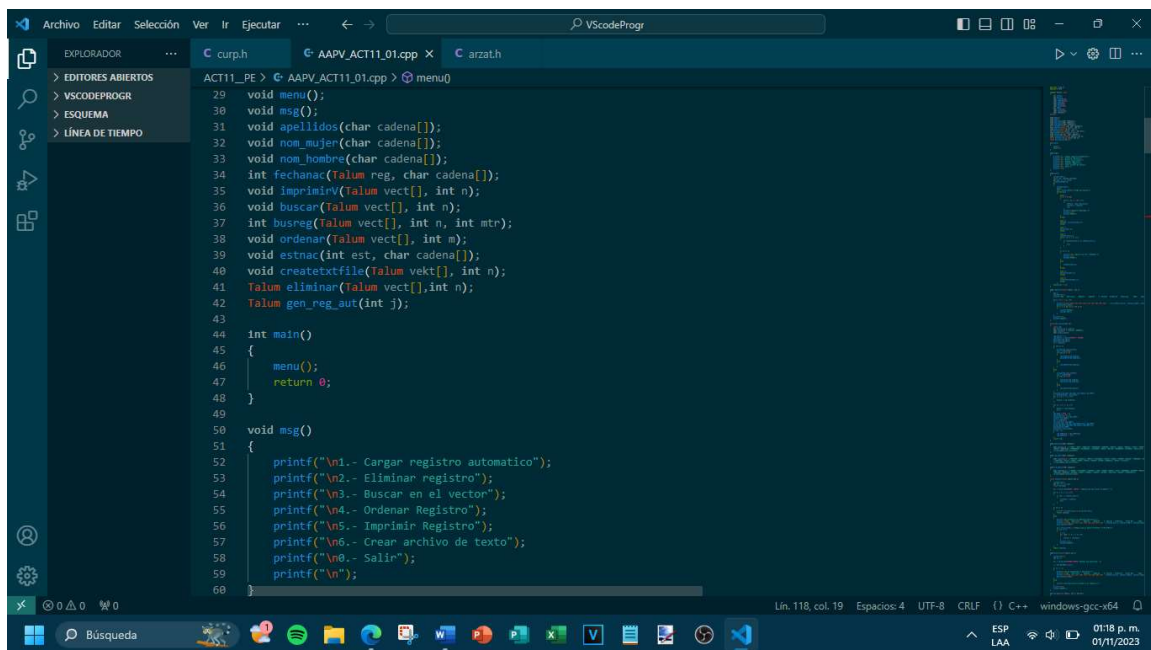
Grupo: 432

Matrícula: 372193

Ensenada, B.C., 01 de Noviembre de 2023



```
1  /* *****  
2  /* Parra Velarde Aaron Alejandro      372193  
3  /* 01/Nov/2023  
4  /* Programa que genera varios registros.  
5  /* PVAA A11_432_01  
6  /* *****  
7  
8  #include <string.h>  
9  #include <conio.h>  
10 #include "curp.h"  
11 #include <time.h>  
12 #define N 2000  
13  
14 typedef struct _alum  
15 {  
16     int status;  
17     int matric;  
18     char nombre[30];  
19     char nombre2[20];  
20     char apat[30];  
21     char amat[30];  
22     char fecha[20];  
23     int edad;  
24     char sexo[10];  
25     char estado[30];  
26     char CURP[20];  
27 }Talun;  
28  
29 void menu();  
30 void msg();  
31 void apellidos(char cadena[]);  
32 void nom_mujer(char cadena[]);
```



```
29 void menu();  
30 void msg();  
31 void apellidos(char cadena[]);  
32 void nom_mujer(char cadena[]);  
33 void nom_hombre(char cadena[]);  
34 int fechanac(Talun reg, char cadena[]);  
35 void imprimirV(Talun vect[], int n);  
36 void buscar(Talun vect[], int n);  
37 int busreg(Talun vect[], int n, int mtr);  
38 void ordenar(Talun vect[], int m);  
39 void estnac(int est, char cadena[]);  
40 void createtxtfile(Talun vekt[], int n);  
41 Talun eliminar(Talun vect[], int n);  
42 Talun gen_reg_aut(int j);  
43  
44 int main()  
45 {  
46     menu();  
47     return 0;  
48 }  
49  
50 void msg()  
51 {  
52     printf("\n1.- Cargar registro automatico");  
53     printf("\n2.- Eliminar registro");  
54     printf("\n3.- Buscar en el vector");  
55     printf("\n4.- Ordenar Registro");  
56     printf("\n5.- Imprimir Registro");  
57     printf("\n6.- Crear archivo de texto");  
58     printf("\n0.- Salir");  
59     printf("\n");  
60 }
```

```
void menu()
{
    system("CLS");
    int reg, vekt[N], vekt2[N];
    int opc, i = 0, j, k = 0;
    srand(time(NULL));
    do
    {
        system("CLS");
        msg();
        opc = valid_num(0, 6, "Elige una opción");
        switch(opc)
        {
            case 1:
                if (i < N-100)
                {
                    for(j = 0; j < 100; j++)
                    {
                        vekt[i] = gen_reg_aut(j);
                        vekt2[i] = vekt[i];
                        i++;
                    }
                    printf("\nRegistro generado.");
                    printf("\n");
                    system("PAUSE");
                }
                break;
            case 2:
                vekt[i] = eliminar(vekt, i);
```

```
            case 4:
                ordenar(vekt2, i);
                for(j = 0; j < i; j++)
                {
                    if (vekt2[j].matric == vekt[j].matric)
                    {
                        k++;
                    }
                }
                if (k == i)
                {
                    printf("\nEl registro ya está ordenado.");
                    printf("\n");
                    system("PAUSE");
                }
                else
                {
                    ordenar(vekt, i);
                }
                break;
            case 5:
                imprimirV(vekt, i);
                break;
            case 6:
                createtxtfile(vekt, i);
                break;
        }
    } while(opc != 0);
}
```

```
130
131 void imprimirV(Talum vekt[], int n)
132 {
133     int i;
134     system("CLS");
135     printf("\tho. Matricula Nombre1 Nombre2 A. Paterno A.Materno Fecha Nac. Edad Sexo Lugar N\n");
136
137     for(i = 0; i < n; i++)
138     {
139         printf("\n\t%2d %10d %13s %12s %10s %13s %14s %8d %10s %10s ", i+1, vekt[i].matric, vekt[i].nombre, vekt[i].nombre2,
140             puts(vekt[i].CURP);
141             if (i != 0 && (i+1) % 40 == 0)
142             {
143                 system("PAUSE");
144                 system("CLS");
145             }
146     }
147     printf("\n");
148     system("PAUSE");
149 }
150
151 Talum gen_reg_aut(int jj)
152 {
153     Talum reg;
154     int nmb, est, sx, a, i, k = 0, j;
155     char sx2[2][8] = {"MUJER", "HOMBRE"};
156     char anio[5];
157     char dia[3], mes[3];
158
159     reg.status = 1;
160     reg.matric = rand() % 3999999 + 3000000;
161     apellidos(reg.apat);
162     apellidos(reg.amat);
163     sx = rand() % 2;
164
165     if (sx == 1)
166     {
167         strcpy(reg.sexo, sx2[1]);
168         nmb = rand() % 100;
169         if (nmb % 2 == 0)
170         {
171             nom_hombre(reg.nombre);
172             nom_hombre(reg.nombre2);
173         }
174         else
175         {
176             nom_hombre(reg.nombre);
177         }
178     }
179     else
180     {
181         strcpy(reg.sexo, sx2[0]);
182         nmb = rand() % 100;
183         if (nmb % 2 == 0)
184         {
185             nom_mujer(reg.nombre);
186             nom_mujer(reg.nombre2);
187         }
188     }
189 }
```

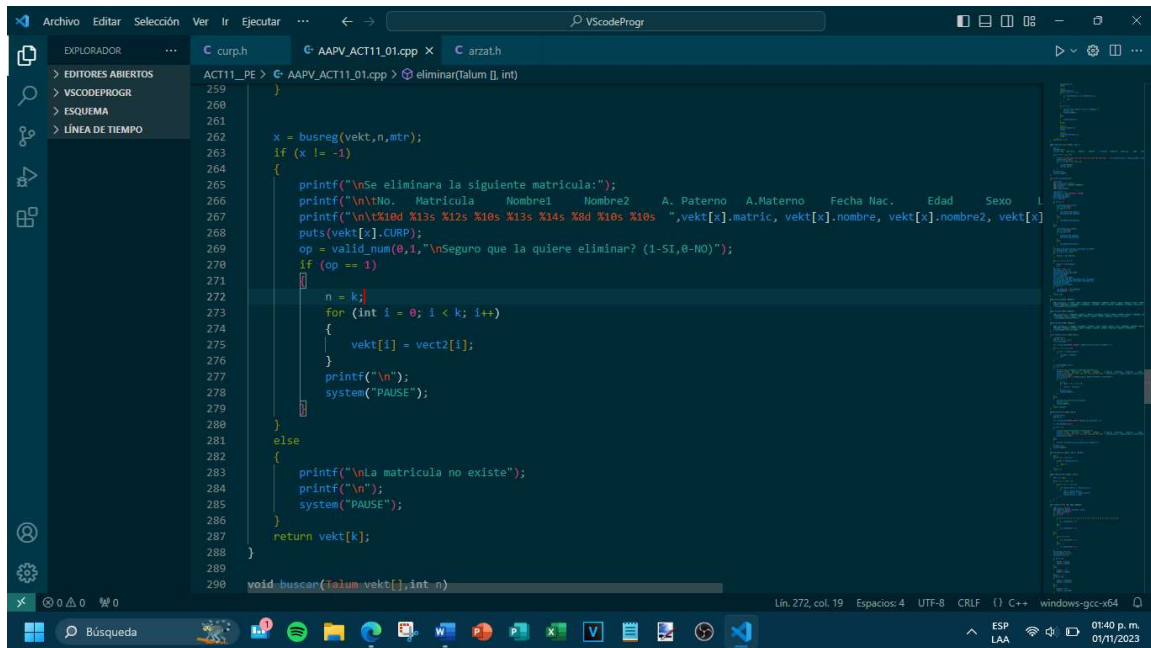
```
157 char dia[3], mes[3];
158
159 reg.status = 1;
160 reg.matric = rand() % 3999999 + 3000000;
161 apellidos(reg.apat);
162 apellidos(reg.amat);
163 sx = rand() % 2;
164
165 if (sx == 1)
166 {
167     strcpy(reg.sexo, sx2[1]);
168     nmb = rand() % 100;
169     if (nmb % 2 == 0)
170     {
171         nom_hombre(reg.nombre);
172         nom_hombre(reg.nombre2);
173     }
174     else
175     {
176         nom_hombre(reg.nombre);
177     }
178 }
179 else
180 {
181     strcpy(reg.sexo, sx2[0]);
182     nmb = rand() % 100;
183     if (nmb % 2 == 0)
184     {
185         nom_mujer(reg.nombre);
186         nom_mujer(reg.nombre2);
187     }
188 }
189
190 int main()
191 {
192     menu0;
193 }
```

This screenshot shows the implementation of a 'reg' struct and its initialization in 'curp.h'. The code includes functions for setting initials, date, month, and name, as well as a function to generate a CURP number. The 'reg' struct is defined with fields for 'apat', 'amat', 'nombre', 'CURP', 'edad', 'anio', 'mes', 'dia', 'sx', 'est', 'letrasEst', 'cons', 'excep', 'buscanom', 'estnac', and 'estado'.

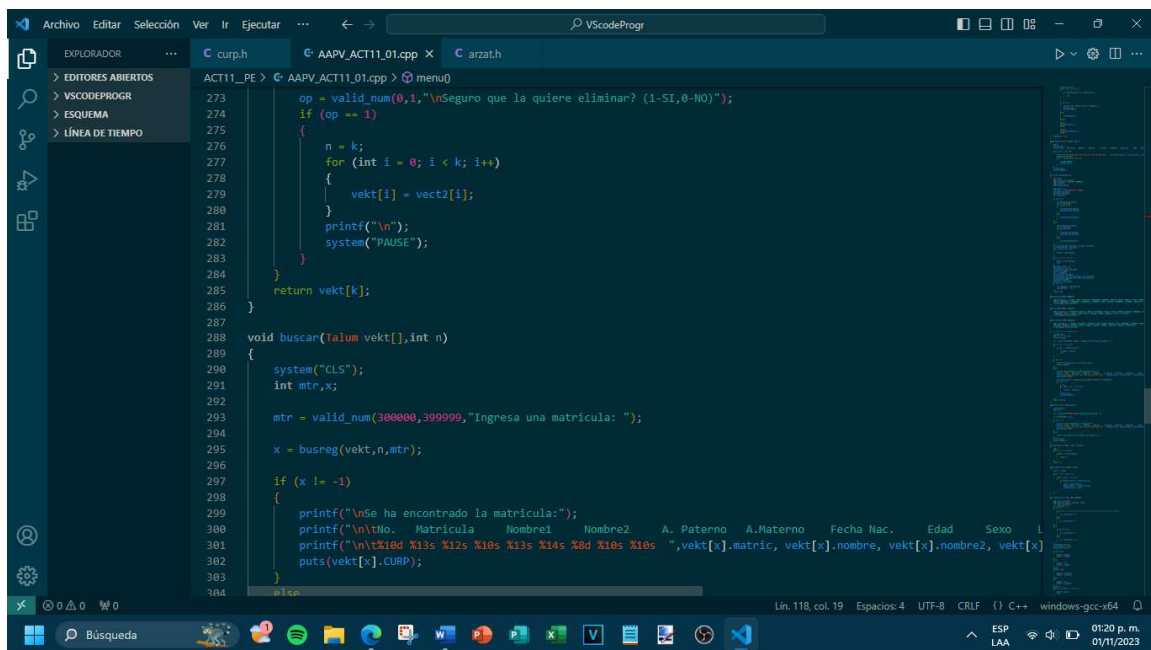
```
189 {
190     nom_mujer(reg.nombre);
191 }
192
193 iniciales(reg.apat, reg.amat, reg.nombre, reg.CURP);
194 a = fechainac(reg, reg.fecha);
195 for (i = 0; i < 2; i++)
196 {
197     dia[i] = reg.fecha[i];
198 }
199
200 for (j = 3; j < 5; j++)
201 {
202     mes[k] = reg.fecha[j];
203     k++;
204 }
205 reg.edad = 2023 - a;
206 sprintf(anio, "%d", a);
207 fecha(anio, mes, dia, reg.CURP);
208 sexo(sx, reg.CURP);
209 est = rand() % 33;
210 letrasEst(est, reg.CURP);
211 cons(reg.apat, reg.amat, reg.nombre, anio, reg.CURP);
212 excep(reg.apat, reg.amat, reg.nombre, reg.CURP, sx);
213 buscanom(reg.CURP);
214 estnac(est, reg.estado);
215 if (jj > 0)
216 {
217     reg.CURP[17] = reg.CURP[18];
218     reg.CURP[18] = '\0';
219 }
220
```

This screenshot shows the implementation of functions to initialize arrays of names in 'curp.h'. The functions 'apellidos', 'nom_mujer', and 'nom_hombre' are defined, each taking a character array and a random seed as input. The 'eliminar' function is also shown, which takes a vector of 'Talun' objects and an integer 'n' as input.

```
222 void apellidos(char cadena[])
223 {
224     char ap[20][12] = {"LOPEZ", "RUIZ", "GONZALEZ", "HERNANDEZ", "RAMIREZ", "GARCIA", "NUÑEZ", "OROZCO", "TAPIA", "PARRA",
225     "REYES", "RODRIGUEZ", "FERNANDEZ", "VALENZUELA", "ALVARADO", "PEREZ", "MULLER", "DOMINGUEZ", "ESTRADA", "CASTILLO"};
226     strcpy(cadena, ap[rand() % 20]);
227 }
228
229 void nom_mujer(char cadena[])
230 {
231     char nom[20][12] = {"MARIANA", "DANIELA", "MARIA", "ALEJANDRA", "DARYA", "DIANA", "JIMENA", "KARINA", "FERNANDA", "SARA",
232     "CAROLINA", "JULIA", "LAURA", "DANNA", "KARLA", "KSENIA", "ELENA", "GABRIELA", "SOFIA", "ALICIA"};
233     strcpy(cadena, nom[rand() % 20]);
234 }
235
236 void nom_hombre(char cadena[])
237 {
238     char nom[20][12] = {"AARON", "ALEJANDRO", "ALBERTO", "JUAN", "PEDRO", "DAVID", "LUIS", "FERNANDO", "ESTEBAN", "RODOLFO",
239     "RONALDO", "RODRIGO", "ANTONIO", "ANGEL", "MIGUEL", "DANIEL", "ARTURO", "CRISTIAN", "CARLOS", "FRANCISCO"};
240     strcpy(cadena, nom[rand() % 20]);
241 }
242
243
244 Talun eliminar(Talun vekt[], int n)
245 {
246     system("CLS");
247     int mtr, k = 0, i, op;
248     Talun vect2[N];
249
250     mtr = valid_num(300000, 399999, "Ingresa una matricula a eliminar: ");
251     for (i = 0; i < n; i++)
252     {
253
```



```
259 }
260
261
262 x = busreg(vekt,n,mtr);
263 if (x != -1)
264 {
265     printf("\nSe eliminara la siguiente matricula:");
266     printf("\n\tMo.   Matricula   Nombre1   Nombre2   A. Paterno   A. Materno   Fecha Mac.   Edad   Sexo   L
267     printf("\n\t%10d %13s %12s %10s %13s %14s %8d %10s %10s   ",vekt[x].matric, vekt[x].nombre, vekt[x].nombre2, vekt[x]
268     puts(vekt[x].CURP);
269     op = valid_num(0,1,"\nSeguro que la quiere eliminar? (1-SI,0-NO)");
270     if (op == 1)
271     {
272         n = k;
273         for (int i = 0; i < k; i++)
274         {
275             vekt[i] = vekt2[i];
276         }
277         printf("\n");
278         system("PAUSE");
279     }
280 }
281 else
282 {
283     printf("\nLa matricula no existe");
284     printf("\n");
285     system("PAUSE");
286 }
287 return vekt[k];
288 }
289
290 void buscar(Talun vekt[],int n)
```



```
273 op = valid_num(0,1,"\nSeguro que la quiere eliminar? (1-SI,0-NO)");
274 if (op == 1)
275 {
276     n = k;
277     for (int i = 0; i < k; i++)
278     {
279         vekt[i] = vekt2[i];
280     }
281     printf("\n");
282     system("PAUSE");
283 }
284 return vekt[k];
285 }
286
287 void buscar(Talun vekt[],int n)
288 {
289     system("CLS");
290     int mtr,x;
291
292     mtr = valid_num(300000,399999,"Ingresa una matricula: ");
293
294     x = busreg(vekt,n,mtr);
295
296     if (x != -1)
297     {
298         printf("\nSe ha encontrado la matricula:");
299         printf("\n\tMo.   Matricula   Nombre1   Nombre2   A. Paterno   A. Materno   Fecha Mac.   Edad   Sexo   L
300         printf("\n\t%10d %13s %12s %10s %13s %14s %8d %10s %10s   ",vekt[x].matric, vekt[x].nombre, vekt[x].nombre2, vekt[x]
301         puts(vekt[x].CURP);
302     }
303 }
304
```



```
ACT11_PE > AAPV_ACT11_01.cpp > menu0
302     puts(vekt[k].CURP);
303 }
304 else
305 {
306     printf("\nLa matricula no existe en el registro.");
307 }
308 printf("\n");
309 system("PAUSE");
310 }
311
312 int busreg(Talun vekt[], int n, int mtr)
313 {
314     int i;
315     for(i = 0; i < n; i++)
316     {
317         if(mtr == vekt[i].matric)
318         {
319             return i;
320         }
321     }
322     return -1;
323 }
324
325 void ordenar(Talun vekt[], int m)
326 {
327     int i, j, temp;
328
329     for(i = 0; i < m-1; i++)
330     {
331         for(j = i+1; j < m; j++)
332         {
333             if (vekt[j].matric < vekt[i].matric)
```

```
324
325 void ordenar(Talun vekt[], int m)
326 {
327     int i, j, temp;
328
329     for(i = 0; i < m-1; i++)
330     {
331         for(j = i+1; j < m; j++)
332         {
333             if (vekt[j].matric < vekt[i].matric)
334             {
335                 temp = vekt[i].matric;
336                 vekt[i].matric = vekt[j].matric;
337                 vekt[j].matric = temp;
338             }
339         }
340     }
341 }
342
343 int fechanac(Talun reg, char cadena[])
344 {
345     int d,m,a,i,j=0,k=0;
346     char dob[12], day[3], month[3], an[5];
347     a = 1960 + rand()%51;
348     m = rand()%12 + 1;
349     if (m != 2)
350     {
351         if (m == 1 || m == 3 || m == 5 || m == 7 || m == 8 || m == 10 || m == 12)
352         {
353             d = (rand()%31) + 1;
354         }
355         else
```

```
345 int d,m,a,i,j=0,k=0;
346 char dob[12], day[3], month[3], an[5];
347 a = 1960 + rand()%51;
348 m = rand()%12 + 1;
349 if (m != 2)
350 {
351     if (m == 1 || m == 3 || m == 5 || m == 7 || m == 8 || m == 10 || m == 12)
352     {
353         d = (rand()%31) + 1;
354     }
355     else
356     {
357         d = (rand()%30) + 1;
358     }
359 }
360 else
361 {
362     if (a % 4 == 0)
363     {
364         d = (rand()%29) + 1;
365     }
366     else
367     {
368         d = (rand()%28) + 1;
369     }
370 }
371 printf(day,"%d",d);
372 printf(month,"%d",m);
373 printf(an,"%d",a);
374
375 if (d >= 10)
376
```

```
374
375 if (d >= 10)
376 {
377     dob[0] = day[0];
378     dob[1] = day[1];
379 }
380 else
381 {
382     dob[0] = '0';
383     dob[1] = day[0];
384 }
385 dob[2] = '-';
386 if (m >= 10)
387 {
388     dob[3] = month[0];
389     dob[4] = month[1];
390 }
391 else
392 {
393     dob[3] = '0';
394     dob[4] = month[0];
395 }
396 dob[5] = '-';
397 for (i = 0; an[i] != '\0'; i++)
398 {
399     dob[i] = an[i];
400     i++;
401 }
402 strcpy(cadena,dob);
403 return a;
404
```



```
ACT11_PE > AAPV_ACT11_01.cpp > menu0
404 }
405
406 void estnac(int est, char cadena[])
407 {
408     char estados[35][20] = {"EXTRANJERO", "AGUASCALIENTES", "BAJA CALIFORNIA", "BAJA CALIFORNIA SUR", "CAMPECHE", "CHIAPAS",
409                             "CHIHUAHUA", "CDMX", "COAHUILA", "COLIMA", "DURANGO", "GUANAJUATO", "GUERRERO", "HIDALGO", "JALISCO", "EDOMEX",
410                             "MICHOACAN", "MORELOS", "NAYARIT", "NUEVO LEON", "OAXACA", "PUEBLA", "QUERETARO", "QUINTANA ROO", "SAN LUIS POTOSI",
411                             "SINALOA", "SONORA", "TABASCO", "TAMAULIPAS", "TLAXCALA", "VERACRUZ", "YUCATAN", "ZACATECAS"};
412
413     strcpy(cadena, estados[est]);
414 }
415
416 void createtxtfile(Talun vekt[], int n)
417 {
418     FILE *fa;
419     int i;
420     fa = fopen("registro.txt", "w");
421     fprintf(fa, "\tNo.   Matricula   Nombre1   Nombre2   A. Paterno   A. Materno   Fecha Nac.   Edad   Sexo   Lug
422     for(i = 0; i < n; i++)
423     {
424         fprintf(fa, "\n\t%2d %10d %13s %12s %10s %13s %14s %8d %10s %10s   %10s ", i+1, vekt[i].matric, vekt[i].nombre, vekt[i]
425     }
426     fclose(fa);
427     printf("\nArchivo .txt generado.");
428 }
```

| | | | | | | | | | | |
|----|--------|---------|---------|-----------|----------|------------|----|--------|---------------------|--------------------|
| 36 | 314038 | ESTEBAN | RODOLFO | ALVARADO | REYES | 26-02-1960 | 63 | HOMBRE | BAJA CALIFORNIA SUR | AARE600226HBSLYS88 |
| 37 | 314096 | KARLA | DARYA | TAPIA | PEREZ | 26-08-1988 | 35 | MUJER | TABASCO | TAPK880826MTCPRR01 |
| 38 | 318640 | SOFIGA | DARYA | GARCIA | ALVARADO | 11-10-1999 | 24 | MUJER | EDOMEX | GAAS991011MTCRVF03 |
| 39 | 316074 | CARLOS | RONALDO | RODRIGUEZ | ALVARADO | 14-11-1994 | 29 | HOMBRE | EDOMEX | ROAC941114HMCQVR06 |
| 40 | 321573 | PEDRO | RONALDO | DOMINGUEZ | REYES | 21-12-1991 | 32 | HOMBRE | YUCATAN | DORP911221HYNMYD07 |

Presione una tecla para continuar . . .