



Universidad Autónoma de Baja California

Facultad de Ingeniería Arquitectura y Diseño

Ingeniero en Software y tecnologías emergentes

Materia: Programación Estructurada / Clave 36276

Alumno: Antonio Ramos González

Matrícula: 372576

Maestro: Pedro Núñez Yépiz

Actividad No. 10: ACTIVIDAD 10

Tema - Unidad 7: Estructuras

Ensenada Baja California a 14 de octubre del 2023

```

1 // Antonio Ramos Gonzalez Mt: 372576
2 // 10/9/2023 || 10/13/2023
3 // En esta Practica se definira un tipo de variable como un struct, para ingresar datos de distintos alumnos con un menu
4 // RGA_Act10_932
5 #include "Babilonia.h" //llamar mi biblioteca
6 #define N 500
7 typedef struct _alumn
8 {
9     char name[30];
10    char lasP[30]; // apellido paterno
11    char lasM[30]; // apellido materno
12    int mt;        // matricula
13    int age;        // edad
14    char sex[10];   // sexo
15    int status;     // dado de baja o no
16 } Talum;          // defino mi tipo de variable de tipo struct
17
18 int msg();
19 void nombAl(char nombre[], int sex);
20 void apellidoAl(char apellido[]);
21 void menu();
22 void printReg(Talum alumn[], int n);
23 int order_reg(Talum alumn[], int n, int band);
24 int searchSec(Talum alumn[], int n, int mt);
25 int searchBin(Talum alumn[], int inf, int sup, int mt);
26 Talum genAl(Talum alumn[], int i);
27 Talum llenarManual(Talum alumn[], int i);
28 Talum eliminar(Talum alumn);
29 void busq(int ind);
30 int verMt(Talum alumn[], int n, int mt);
31
32 int main()
33 {
34     srand(time(NULL));
35     menu();
36     return 0;
37 }
38
39 // Muestra mensaje
40 int msg()
41 {
42     system("CLS");
43     printf("MENU\n");
44     printf("1.-Agrega alumnar(AUTOM 10 alumnos)\n");
45     printf("2.-Agrega alumno manual\n");
46     printf("3.-Eliminar alumno(logico)\n");
47     printf("4.-Buscar\n");
48     printf("5.-Ordenar\n");
49     printf("6.-Imprimir\n");
50     printf("0.-Salir\n");
51     return valid("Elige una opcion: ", 0, 6);
52 }
53

```

```

54 // Meno de elecciones
55 void menu()
56 {
57     Talum reg[N]; // genera vector del tipo Talum
58     int opc, el, i = 0, j;
59     int mt, bus;
60     bool band;
61
62     do
63     {
64         opc = msg(); // lee el valor recibido de msg
65         switch (opc)
66         {
67             case 1:
68                 if (i < N) // valida que no se supere el limite maximo
69                 {
70                     for (j = 0; j < 10; j++)
71                     {
72                         if (i < N)
73                         {
74                             reg[i] = genAl(reg, i);
75                             i++;
76                         }
77                     }
78                     band = 1; // se usa para los tipos de busqueda
79                 }
80                 else
81                 {
82                     printf("BASE DE DATOS LLENA\n");
83                     system("PAUSE");
84                 }
85                 break;
86
87             case 2:
88                 if (i < N) // valida que no se supere el limite maximo
89                 {
90                     reg[i] = llenarManual(reg, i);
91                     i++; // aumenta el indice del vector
92                     band = 1;
93                 }
94                 else
95                 {
96                     printf("BASE DE DATOS LLENA\n");
97                     system("PAUSE");
98                 }
99                 break;
100
101             case 3:
102                 el = valid("Ingrese matricula: ", 300000, 399999);
103                 if (band == 1) // verifica si el vectro ya fue ordenado
104                 {
105                     bus = searchSec(reg, i, el);
106                 }
107                 else
108                 {
109                     bus = searchBin(reg, 0, i, el);
110                 }
111                 reg[bus].status=0;
112                 break;

```

```

114         case 4:
115             mt = valid("Ingresa Matricula: ", 300000, 399999);
116
117             if (band == 1) // verifica si el vector ya fue ordenado
118             {
119                 bus = searchSec(reg, i, mt);
120             }
121             else
122             {
123                 bus = searchBin(reg, 0, i, mt);
124             }
125
126             busq(bus);
127
128             break;
129
130         case 5:
131
132             if (band == 1)
133             {
134                 band = order_reg(reg, i, band); // ordena registro
135             }
136
137             break;
138
139         case 6:
140             printReg(reg, i); // imprime vector
141             break;
142     }
143     } while (opc != 0);
144 }

```

```

146 // Funcion para nombres
147 void nombAl(char nombre[], int sex)
148 {
149     int fil;
150     char nombM[10][10] =
151     {
152         "ANA",
153         "JULIETA",
154         "JOSEFINA",
155         "DANIELA",
156         "CARMEN",
157         "SOFIA",
158         "LAURA",
159         "ANDREA",
160         "ELENA",
161         "ISABEL"}; // Nombres de mujeres
162
163     char nombH[10][10] =
164     {
165         "JUAN",
166         "CARLOS",
167         "ROBERTO",
168         "DAMIAN",
169         "ANDRES",
170         "DAVID",
171         "ALEJANDR",
172         "MIGUEL",
173         "DENZO"

```

```

163 char nombH[10][10] =
164 {
165     "JUAN",
166     "CARLOS",
167     "ROBERTO",
168     "DAMIAN",
169     "ANDRES",
170     "DAVID",
171     "ALEJANDR",
172     "MIGUEL",
173     "PEDRO",
174     "FERNANDO"}; // Nombres de hombre
175
176 if (sex == 1) // Se encarga de leer el sexo para decidir si el nombre sera de hombre
177 {
178     fil = rand() % 10; // elige un nombre de manera aleatoria
179     strcpy(nombre, nombH[fil]); // copia el valor del nombre el la variable
180 }
181 if (sex == 2) // Se encarga de leer el sexo para decidir si el nombre sera de mujer
182 {
183     fil = rand() % 10; // elige un nombre de manera aleatoria
184     strcpy(nombre, nombM[fil]); // copia el valor del nombre el la variable
185 }
186 }
187
188 // Funcio para apellidos
189 void apellidoAl(char apellido[])
190 {
191     int fil;
192     char ap[10][15] =
193     {
194         "GARCIA",
195         "RODRIGUEZ",
196         "PEREZ",
197         "LOPEZ",
198         "MARTINEZ",
199         "GONZALEZ",
200         "SANCHEZ",
201         "ROMERO",
202         "FRENANDEZ",
203         "TORRES"}; // Apellidos generales
204
205     fil = rand() % 10; // elige un apellido de manera aleatoria
206     strcpy(apellido, ap[fil]); // copia el valor del apellido en la variable
207 }
208
209 // Funcion que verifica que no se repitan matriculas
210 int verMt(Talum alumn[], int n, int mt)
211 {
212     int i, j, cont; // variables locales
213     cont = 0;
214     for (i = 0; i < n; i++) // ciclo que aumenta el valor de i
215     {
216         for (j = 0; j < i; j++) // ciclo que aumenta el valor de j
217         {
218             if (alumn[j].mt == mt) // valida que no existan
219             {
220                 cont = 1;

```

```

220         cont = 1;
221     }
222 }
223 }
224 if (cont == 1) // verifica el valor de cont
225 {
226     return 1; // si se repiten matriculas retorna un 1
227 }
228 else
229 {
230     return 0; // si no se repiten retorna un 0
231 }
232 }
233
234 // Funcion que genera datos de alumnos aleatorios
235 Talum genAl(Talum alumn[], int i)
236 {
237
238     int sex, val; // variables locales
239     char nombre[10], apellido[15]; // variables que guardaran nombres y apellidos
240
241     sex = rand() % 2 + 1; // genera un sexo aleatorio
242
243     nombAl(nombre, sex); // ingresa a la funcion para decidir un nombre
244     strcpy(alumn[i].name, nombre); // copia nombre en el struct
245
246     apellidoAl(apellido); // ingresa a la funcion para decidir un apellido
247     strcpy(alumn[i].lasP, apellido); // copia apellido en el struct
248
249     apellidoAl(apellido); // ingresa a la funcion para decidir un apellido
250     strcpy(alumn[i].lasM, apellido); // copia apellido en el struct
251
252     alumn[i].age = rand() % 13 + 18; // genera una edad aleatoria entre 18 y 30
253
254     alumn[i].status = 1; // aplica un estatus al alumno
255
256     do
257     {
258         val = 0;
259         alumn[i].mt = rand() % 100000 + 300000; // genera una matricula entre 300k y 399k
260         val = verMt(alumn, i, alumn[i].mt); // ingresa a la funcion para validar
261     } while (val != 0); // ciclo que se repite si 2 matriculas son iguales
262
263     if (sex == 1) // valida el sexo del usuario
264     {
265         strcpy(alumn[i].sex, "H"); // si sex es 1 entonces sera hombre
266     }
267     else
268     {
269         strcpy(alumn[i].sex, "M"); // si no sera mujer
270     }
271
272     return alumn[i]; // retorna alumn
273 }
274
275
276 // Funcion para ingresar a un alumno de manera manual
277 Talum llenarManual(Talum alumn[], int i)
278 {

```

```

276 // Funcion para ingresar a un alumno de manera manual
277 Talum llenarManual(Talum alumn[], int i)
278 {
279     int sex, val; // variables locales
280     system("CLS");
281     fflush(stdin);
282     do
283     {
284         val = 0;
285         alumn[i].mt = valid("Ingreso matricula: ", 300000, 399999); // lee una matricula
286         val = verMt(alumn, i, alumn[i].mt); // rntra a la funcion de validar
287
288         if (val != 0) // si se repiten imprime mensaje
289         {
290             printf("Matricula ya existente\n");
291         }
292     } while (val != 0); // ciclo que se repite si 2 matriculas son iguales
293
294     validCad("Ingrese nombre: ", alumn[i].name); // ingresa y valida nombre
295     validCad("Ingrese apellido paterno: ", alumn[i].lasP); // ingresa y valida apellido paterno
296     validCad("Ingrese apellido materno: ", alumn[i].lasM); // ingresa y valida apellido materno
297
298     system("CLS");
299     alumn[i].age = valid("Ingrese Edad: ", 18, 30); // ingresa y valida edad
300
301     sex = valid("Ingresa el sexo(1.-H,2.-M): ", 1, 2); // ingresa y valida sexo
302
303     if (sex == 1) // decide si es hombre o mujer
304     {
305         strcpy(alumn[i].sex, "H");
306     }
307     else
308     {
309         strcpy(alumn[i].sex, "M");
310     }
311     alumn[i].status = 1; // ingresa y valida estatus
312
313     return alumn[i];
314 }
315
316 // Funcion de busqueda secuencial
317 int searchSec(Talum alumn[], int n, int mt)
318 {
319     int i; // define contador
320     i = 0;
321     for (i = 0; i <= n; i++)
322     {
323         if (alumn[i].mt == mt) // Busca en el vector el numero buscado
324         {
325             return i; // si encuentra el valor, retorna el valor del indice
326         }
327     }
328     return -1; // si no encuentra el valor, retorna -1
329 }
330
331 // Funcion que ordena el vector
332 int order_reg(Talum alumn[], int n, int band)

```

```

// Funcion que ordena el vector
int order_reg(Talum alumn[], int n, int band)
{
    int i, j;
    Talum temp; // guarda valor de manera temporal

    for (i = 0; i < n - 1; i++) // Busqueda secuencial
    {
        for (j = i + 1; j < n; j++)
        {
            if (alumn[j].mt < alumn[i].mt)
            {
                temp = alumn[i]; // Guarda el valor de alumn[i] en temp
                alumn[i] = alumn[j]; // Guarda el valor de alumn[j] en alumn[i]
                alumn[j] = temp; // Guarda el valor de temp en alumn[j]
            }
        }
    }
    return band = 0;
}

// Funcion para imprimir el vector
void printReg(Talum alumn[], int n)
{
    int i;
    system("CLS");
    printf("%-10s %-10s %-10s %-10s %-4s %-5s\n",
           "Matricula", "Nombre", "ApP", "ApM", "Edad", "Sexo"); // imprime las columnas, como datos
    for (i = 0; i < n; i++) // ciclo que aumenta el valor del vector
    {
        if (alumn[i].status == 1) // verifica que el estatus sea 1
        {
            printf("%-10d %-10s %-10s %-10s %-4d %-5s\n",
                   alumn[i].mt, alumn[i].name, alumn[i].lasP, // imprime el vector
                   alumn[i].lasM, alumn[i].age, alumn[i].sex);
        }
    }
    system("PAUSE");
}

// Funcion que imprime el alumno buscado
void busq(int ind)
{
    if (ind != -1)
    {
        printf("Alumno encontrado en el indice: %d\n", ind+1);
    }
    else
    {
        printf("Alumno no encontrado\n");
    }
    system("PAUSE");
}

// Funcion de busqueda binaria
int searchBin(Talum alumn[], int inf, int sup, int mt)
{
    int med; // variable local

```



```
// Funcion de busqueda binaria
int searchBin(Talum alumn[], int inf, int sup, int mt)
{
    int med; // variable local
    while (inf <= sup)
    {
        med = (inf + sup) / 2; // definir la mitad del vector
        if (alumn[med].mt == mt) // verifica si el valor buscado se encuentra en la actual pocicion de med
        {
            return med; // retorna el valor de med
        }
        else
        {
            if (mt < alumn[med].mt) // verifica que el valor buscado sea menor al valor en med
            {
                sup = med--; // resta 1 a med y lo asigna al valor sup para que disminuya asi el valor central
            }
            else
            {
                inf = med++; // suma 1 a med y lo asigna al valor inf para que disminuya asi el valor central
            }
        }
    }
    return -1; // si no encuentra coincidencia retorna -1
}
```

```
70         for (j = 0; j < 10; j++)
71         {
72             if (i < N)
73             {
74                 reg[i] = genAl(reg, i);
75                 i++;

```

PROBLEMS

OUTPUT

DEBUG CONSOLE

TERMINAL

PORTS

MENU

1.-Agrega alumnar(AUTOM 10 alumnos)

2.-Agrega alumno manual

3.-Eliminar alumno(logico)

4.-Buscar

5.-Ordenar

6.-Imprimir

0.-Salir

Elije una opcion: 1

```
67         case 1:
68             if (i < N) // valida que no se supere el limite ma
69             {
70                 for (j = 0; j < 10; j++)
71                 {
72                     if (i < N)
73                     {
74                         reg[i] = genAl(reg, i);
75                         i++;
```

PROBLEMS

OUTPUT

DEBUG CONSOLE

TERMINAL

PORTS

MENU

1.-Agrega alumna(AUTOM 10 alumnos)

2.-Agrega alumno manual

3.-Eliminar alumno(logico)

4.-Buscar

5.-Ordenar

6.-Imprimir

0.-Salir

Elije una opcion: 2

```
63 {
64     opc = msg(); // lee el valor recibido de msg
65     switch (opc)
66     {
67     case 1:
68         if (i < N) // valida que no se supere el límite máximo
69         {
70             for (j = 0; j < 10; j++)
71             {
72                 if (i < N)
73                 {
74                     reg[i] = genAl(reg, i);
75                     i++;
```

PROBLEMS

OUTPUT

DEBUG CONSOLE

TERMINAL

PORTS

Ingrese matricula: 372576

```
66     {
67         case 1:
68             if (i < N) // valida que no se supere el límite máximo
69             {
70                 for (j = 0; j < 10; j++)
71                 {
72                     if (i < N)
73                     {
74                         reg[i] = genAl(reg, i);
75                         i++;
```

PROBLEMS

OUTPUT

DEBUG CONSOLE

TERMINAL

PORTS

Ingrese nombre: ANTONIO

while (true)

do

{

opc = msg(); // lee el valor recibido de msg

switch (opc)

{

case 1:

if (i < N) // valida que no se supere el límite máximo

{

for (j = 0; j < 10; j++)

{

if (i < N)

{

reg[i] = genAl(reg, i);

i++;

PROBLEMS

OUTPUT

DEBUG CONSOLE

TERMINAL

PORTS

Ingrese apellido paterno: RAMOS

```
63     {
64         opc = msg(); // lee el valor recibido de msg
65         switch (opc)
66         {
67             case 1:
68                 if (i < N) // valida que no se supere el limite maximo
69                 {
70                     for (j = 0; j < 10; j++)
71                     {
72                         if (i < N)
73                         {
74                             reg[i] = genAl(reg, i);
75                             i++;
76                         }
77                     }
78                 }
79             }
80         }
81     }
82 }
```

PROBLEMS

OUTPUT

DEBUG CONSOLE

TERMINAL

PORTS

Ingrese apellido materno: GONZALEZ

```
64     opc = msg(), // lee el valor recibido de msg
65     switch (opc)
66     {
67     case 1:
68         if (i < N) // valida que no se supere el límite maxi
69         {
70             for (j = 0; j < 10; j++)
71             {
72                 if (i < N)
73                 {
74                     reg[i] = genAl(reg, i);
75                     i++;
```

PROBLEMS

OUTPUT

DEBUG CONSOLE

TERMINAL

PORTS

Ingrese Edad: 19


```
58     int opc, el, i = 0, j;  
59     int mt, bus;  
60     bool band;  
61  
62     do  
63     {  
64         opc = msg(); // lee el valor recibido de msg  
65         switch (opc)  
66         {  
67             case 1:  
68                 if (i < N) // valida que no se supere el limite maximo  
69                 {  
70                     for (j = 0; j < 10; j++)  
71                     {  
72                         if (i < N)  
73                         {  
74                             reg[i] = genAl(reg, i);  
75                             i++;  
76                         }  
77                     }  
78                 }  
79             }  
80         }  
81     } while (band);  
82 }
```

PROBLEMS

OUTPUT

DEBUG CONSOLE

TERMINAL

PORTS

Ingresa el sexo(1.-H,2.-M): 1

```
64     opc = msg(); // lee el valor recibido de msg
65     switch (opc)
66     {
67     case 1:
68         if (i < N) // valida que no se supere el límite máximo
69         {
70             for (j = 0; j < 10; j++)
71             {
72                 if (i < N)
73                 {
74                     reg[i] = genAl(reg, i);
75                     i++;
```

PROBLEMS

OUTPUT

DEBUG CONSOLE

TERMINAL

PORTS

MENU

1.-Agrega alumnar(AUTOM 10 alumnos)

2.-Agrega alumno manual

3.-Eliminar alumno(logico)

4.-Buscar

5.-Ordenar

6.-Imprimir

0.-Salir

Elije una opcion: 6

```

74         reg[i] = genAI(reg, 1);
75         i++;
76     }
77 }
78     band = 1; // se usa para los tipos de busqueda
79 }
80 else
81 {
82     printf("BASE DE DATOS LLENA\n");
83     system("PAUSE");
84 }
85 break;
86
87 case 2:
88     if (i < N) // valida que no se supere el limite maximo

```

PROBLEMS

OUTPUT

DEBUG CONSOLE

TERMINAL

PORTS

Matricula	Nombre	ApP	ApM	Edad	Sexo
302996	MIGUEL	PEREZ	FRENADEZ	22	H
317629	JULIETA	RODRIGUEZ	FRENADEZ	26	M
319636	DANIELA	GARCIA	LOPEZ	28	M
313876	DAMIAN	PEREZ	RODRIGUEZ	24	H
311742	ISABEL	GARCIA	GARCIA	19	M
311124	JOSEFINA	FRENADEZ	TORRES	22	M
325924	JUAN	SANCHEZ	TORRES	23	H
307371	DAVID	GONZALEZ	ROMERO	22	H
303983	CARLOS	LOPEZ	PEREZ	18	H
318562	SOFIA	MARTINEZ	RODRIGUEZ	29	M
372576	ANTONIO	RAMOS	GONZALEZ	19	H

Presione una tecla para continuar . . .

```

75         i++;
76     }
77 }
78     band = 1; // se usa para los tipos de busqueda
79 }
80 else
81 {
82     printf("BASE DE DATOS LLENA\n");
83     system("PAUSE");
84 }
85 break;
86
87 case 2:
88     if (i < N) // valida que no se supere el limite maxi

```

PROBLEMS

OUTPUT

DEBUG CONSOLE

TERMINAL

PORTS

MENU

1.-Agrega alumnar(AUTOM 10 alumnos)

2.-Agrega alumno manual

3.-Eliminar alumno(logico)

4.-Buscar

5.-Ordenar

6.-Imprimir

0.-Salir

Elije una opcion: 3

```
76     }
77     }
78     band = 1; // se usa para los tipos de busqueda
79 }
80 else
81 {
82     printf("BASE DE DATOS LLENA\n");
83     system("PAUSE");
84 }
85 break;
86
87 case 2:
88     if (i < N) // valida que no se supere el límite máximo
```

PROBLEMS

OUTPUT

DEBUG CONSOLE

TERMINAL

PORTS

Ingrese matricula: 372576


```

76         }
77     }
78     band = 1; // se usa para los tipos de busqueda
79 }
80 else
81 {
82     printf("BASE DE DATOS LLENA\n");
83     system("PAUSE");
84 }
85 break;
86
87 case 2:
88     if (i < N) // valida que no se supere el límite máximo
89     {

```

PROBLEMS

OUTPUT

DEBUG CONSOLE

TERMINAL

PORTS

MENU

- 1.-Agregar alumno(AUTOM 10 alumnos)
- 2.-Agregar alumno manual
- 3.-Eliminar alumno(logico)
- 4.-Buscar
- 5.-Ordenar
- 6.-Imprimir
- 0.-Salir

Elije una opcion: 4

```
77     }
78     band = 1; // se usa para los tipos de busqueda
79 }
80 else
81 {
82     printf("BASE DE DATOS LLENA\n");
83     system("PAUSE");
84 }
85 break;
86
87 case 2:
88     if (i < N) // valida que no se supere el limite maximo
89     {
```

PROBLEMS

OUTPUT

DEBUG CONSOLE

TERMINAL

PORTS

Ingresa Matricula: 311124


```
82         printf("BASE DE DATOS LLENA\n");
83         system("PAUSE");
84     }
85     break;
86
87     case 2:
88         if (i < N) // valida que no se supere el límite máximo
89         {
```

PROBLEMS

OUTPUT

DEBUG CONSOLE

TERMINAL

PORTS

Alumno encontrado en el índice: 6

```

72         if (i < N)
73         {
74             reg[i] = genAl(reg, i);
75             i++;
76         }
77     }
78     band = 1; // se usa para los tipos de busqueda
79 }
80 else
81 {
82     printf("BASE DE DATOS LLENA\n");
83     system("PAUSE");
84 }
85 break;
86
87 case 2:
88     if (i < N) // valida que no se supere el limite maximo
89     {

```

PROBLEMS

OUTPUT

DEBUG CONSOLE

TERMINAL

PORTS

MENU

1.-Agrega alumnar(AUTOM 10 alumnos)

2.-Agrega alumno manual

3.-Eliminar alumno(logico)

4.-Buscar

5.-Ordenar

6.-Imprimir

0.-Salir

Elije una opcion: 5

```

76         }
77     }
78     band = 1; // se usa para los tipos de busqueda
79 }
80 else
81 {
82     printf("BASE DE DATOS LLENA\n");
83     system("PAUSE");
84 }
85 break;
86
87 case 2:
88     if (i < N) // valida que no se supere el límite máximo
89     {

```

PROBLEMS	OUTPUT	DEBUG CONSOLE	<u>TERMINAL</u>	PORTS
Matricula	Nombre	ApP	ApM	Edad Sexo
302996	MIGUEL	PEREZ	FRENADEZ	22 H
303983	CARLOS	LOPEZ	PEREZ	18 H
307371	DAVID	GONZALEZ	ROMERO	22 H
311124	JOSEFINA	FRENADEZ	TORRES	22 M
311742	ISABEL	GARCIA	GARCIA	19 M
313876	DAMIAN	PEREZ	RODRIGUEZ	24 H
317629	JULIETA	RODRIGUEZ	FRENADEZ	26 M
318562	SOFIA	MARTINEZ	RODRIGUEZ	29 M
319636	DANIELA	GARCIA	LOPEZ	28 M
325924	JUAN	SANCHEZ	TORRES	23 H
Presione una tecla para continuar . . .				

```
83         system("PAUSE");
84     }
85     break;
86
87     case 2:
88         if (i < N) // valida que no se supere el límite máximo
89         {
```

PROBLEMS

OUTPUT

DEBUG CONSOLE

TERMINAL

PORTS

MENU

- 1.-Agrega alumna(AUTOM 10 alumnos)
- 2.-Agrega alumno manual
- 3.-Eliminar alumno(logico)
- 4.-Buscar
- 5.-Ordenar
- 6.-Imprimir
- 0.-Salir

Elije una opcion: 4

```
76         }
77     }
78     band = 1; // se usa para los tipos de busqueda
79 }
80 else
81 {
82     printf("BASE DE DATOS LLENA\n");
83     system("PAUSE");
84 }
85 break;
86
87 case 2:
88     if (i < N) // valida que no se supere el límite máximo
89     {
```

PROBLEMS

OUTPUT

DEBUG CONSOLE

TERMINAL

PORTS

Ingresa Matricula: 311124

```
74         reg[i] = genAl(reg, i);
75         i++;
76     }
77 }
78     band = 1; // se usa para los tipos de busqueda
79 }
80 else
81 {
82     printf("BASE DE DATOS LLENA\n");
83     system("PAUSE");
84 }
85 break;
86
87 case 2:
88     if (i < N) // valida que no se supere el límite máximo
89     {
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

Alumno encontrado en el índice: 4