



Universidad Autónoma de Baja California

Facultad de Ingeniería, Arquitectura y Diseño



**FUNCIONES Y METODOS DE ORDENACION Y
BUSQUEDA ESTRUCTURAS Y LIBRERÍAS**

PROGRAMACIÓN ESTRUCTURADA

PRACTICA 11 - ANEXO

Ingeniero en Software y Tecnologías Emergentes

Brayan Ivan Perez Ventura

372781

22 de Octubre del 2023 en Ensenada, Baja California

```
75 void menu()
76 {
77     int op;
78     int position = 0;
79     int flagOrd = 0;
80     int tempPosition = 0;
81     Tstdnt students[N];
82     do
83     {
84         system("CLS");
85         op = msge_menu();
86         system("CLS");
87         switch (op)
88         {
89             case 1:
90                 position = menuAdd(position, students);
91                 if (tempPosition != position)
92                 {
93                     flagOrd = 0;
94                 }
95                 break;
96             case 2:
97                 if (position != 0)
98                 {
99                     deleteStdnt(students, position);
100                 }
101                 else
102                 {
103                     printf("No hay nada para eliminar:\n");
104                 }
105                 break;
106             case 3:
107                 if (position != 0)
108                 {
109                     searchStdnt(students, position, flagOrd);
110                 }
111                 else
112                 {
113                     printf("No hay nada para buscar:\n");
114                 }
115                 break;
```

```

116     case 4:
117         if (position != 0)
118         {
119             orderStdnts(students, position, flagOrd);
120             flagOrd = 1;
121         }
122         else
123         {
124             printf("No hay nada para ordenar:\n");
125         }
126         break;
127     case 5:
128         if (position != 0)
129         {
130             displayReg(students, position);
131         }
132         else
133         {
134             printf("No hay nada que imprimir:\n");
135         }
136         break;
137     case 6:
138         if (position != 0)
139         {
140             getTXT(students, position);
141             position = 0;
142             printf("El archivo ha sido generado correctamente y se ha eliminado el registro.\n");
143         }
144         else
145         {
146             printf("No hay nada para almacenar:\n");
147         }
148         break;
149     }
150     if (op != 0 || op != 6)
151     {
152         system("PAUSE");
153     }
154     tempPosition = position;
155 } while (op != 0);
156 printf("Saliendo del programa");
157 }

```

```

159 int msge_menu()
160 {
161     printf("-----MENU DE REGISTROS-----\n");
162     printf("1.- Agregar\n");
163     printf("2.- Eliminar registro\n");
164     printf("3.- Buscar\n");
165     printf("4.- Ordenar\n");
166     printf("5.- Imprimir\n");
167     printf("6.- Archivo de texto\n");
168     printf("0.- Salir\n");
169
170     return valid("Por favor, selecciona una opcion: ", 0, 6);
171 }

```

PROBLEMS

TERMINAL

OUTPUT

PORTS

GITLENS

COMMENTS

-----MENU DE REGISTROS-----

1.- Agregar

2.- Eliminar registro

3.- Buscar

4.- Ordenar

5.- Imprimir

6.- Archivo de texto

0.- Salir

Por favor, selecciona una opcion: █

```

173 int menuAdd(int position, Tstdnt students[])
174 {
175     int op;
176     int i;
177     do
178     {
179         system("CLS");
180         op = msge_menuAdd();
181         system("CLS");
182         switch (op)
183         {
184             case 1:
185                 if (N > position + 1)
186                 {
187                     students[position] = addManual(students, position);
188                     students[position].status = 1;
189                     position += 1;
190                     printf("Estudiante anadido correctamente\n");
191                 }
192                 else
193                 {
194                     printf("Tamano maximo alcanzado.\n");
195                 }
196                 break;
197             case 2:
198                 if (N > position + 100)
199                 {
200                     for (i = 0; i < 100; i++)
201                     {
202                         students[position + i] = addOneStdnt(students, position);
203                         students[position + i].status = 1;
204                     }
205                     position += 100;
206                     printf("100 estudiantes han sido añadidos correctamente\n");
207                 }
208                 else
209                 {
210                     printf("Tamano maximo alcanzado.\n");
211                 }
212                 break;
213             }
214             if (op != 3)
215             {
216                 system("PAUSE");
217             }
218         } while (op != 3);
219         return position;
220     }

```

```
173 > int menuAdd(int position, Tstdnt students[]) You,
221
222 int msge_menuAdd()
223 {
224     printf("1.- Manual (1)\n");
225     printf("2.- Automatico (100)\n");
226     printf("3.- Regresar al inicio\n");
227
228     return valid("Selecciona una opcion: ", 1, 3);
229 }
```

PROBLEMS

TERMINAL

OUTPUT

PORTS

GITLENS

```
1.- Manual (1)
2.- Automatico (100)
3.- Regresar al inicio
Selecciona una opcion: █
```

```

295 void deleteStdnt(Tstdnt students[], int position)
296 {
297     int num, index;
298     num = valid("Ingrese la matricula del estudiante que desea eliminar: ", 300000, 399999);
299     index = existElem(students, position, num);
300     if (index != -1)
301     {
302         if (students[index].status != 0)
303         {
304             displayOneStdnt(students[index]);
305             if (valid("Deseas eliminar el registro (0.- No / 1.- Si): ", 0, 1))
306             {
307                 students[index].status = 0;
308                 printf("El estudiante ha sido dado de baja correctamente.\n");
309             }
310         }
311         else
312         {
313             printf("El estudiante ya sido dado de baja con anterioridad.\n");
314         }
315     }
316     else
317     {
318         printf("La matricula ingresada no pertecene a ningun estudiante\n");
319     }
320 }

```

PROBLEMS

TERMINAL

OUTPUT

PORTS

GITLENS

COMMENTS

DEBUG CONSOLE

Ingrese la matricula del estudiante que desea eliminar: 372782

```

-----
MATRICULA:    372782
NOMBRE:       BRAYAN IVAN
AP. PAT:
AP. MAT:
FECHA DE NAC: 01/04/2003
EDAD:         20
SEXO:         Hombre
LUGAR NAC:    Baja California
CURP:         XXXB030401HBCXXRA5
-----

```

```

Deseas eliminar el registro (0.- No / 1.- Si): 1
El estudiante ha sido dado de baja correctamente.
Presione una tecla para continuar . . .

```

```

322 void searchStdnt(Tstdnt students[], int position, int flag)
323 {
324     int num, index;
325     num = valid("Ingrese la matricula del estudiante que desea buscar: ", 300000, 399999);
326     if (flag == 0)
327     {
328         index = existElem(students, position, num);
329     }
330     else
331     {
332         index = binarySearch(students, 0, position, num);
333     }
334
335     if (index != -1)
336     {
337         printf("El alumno ha sido encontrado: \n");
338         displayOneStdnt(students[index]);
339     }
340     else
341     {
342         printf("La matricula no ha sido registrada\n");
343     }
344 }

```

PROBLEMS TERMINAL OUTPUT PORTS GITLENS COMMENTS DEBUG CONSOLE

Ingrese la matricula del estudiante que desea buscar: 372781
El alumno ha sido encontrado:

```

-----
MATRICULA:    372781
NOMBRE:       BRAYAN IVAN
AP. PAT:      PEREZ
AP. MAT:      VENTURA
FECHA DE NAC: 01/04/2003
EDAD:         20
SEXO:         Hombre
LUGAR NAC:    Baja California
CURP:         PEVB030401HBCRNRA3

```

Presione una tecla para continuar . . .


```
346 void orderStdnts(Tstdnt students[], int position, int flag)
347 {
348     if (flag == 1)
349     {
350         printf("Ya ha sido ordenado de menor a mayor\n");
351     }
352     else
353     {
354         if (position >= 200)
355         {
356             quicksort(students, 0, position);
357         }
358         else
359         {
360             bubbleSort(students, position);
361         }
362         printf("Los estudiantes han sido ordenados correctamente\n");
363     }
364 }
```

PROBLEMS

TERMINAL

OUTPUT

PORTS

GITLENS

COMMENTS

DEBUG CONSOLE

```
Los estudiantes han sido ordenados correctamente
Presione una tecla para continuar . . . █
```



```

390 void getTXT(Tstdnt students[], int position)
391 {
392     int i;
393     FILE *fa;
394     fa = fopen("C:/PVBI_Programacion_Estructurada_932/PVBI_Act11_0_932/Registers.txt", "w");
395     fprintf(fa, "%-6s %-10s %-15s %-15s %-10s %-18s %-5s %-5s %s\n", "No.", "Matricula", "Ape Paterno", "Ape Materno", "Nombre",
396         "Fecha de Nacimiento", "Edad", "Sexo", "CURP");
397
398     for (i = 0; i < position; i++)
399     {
400         fprintf(fa, "%-6d %-10d %-15s %-15s %-10s %02d-%02d-%04d %11d %8s %10s\n", i + 1, students[i].matricula,
401             students[i].personalName.lastName1,
402             students[i].personalName.lastName2, students[i].personalName.name,
403             students[i].date.day, students[i].date.month, students[i].date.year, students[i].age, students[i].sex, students[i].curp);
404     }
405     fclose(fa);
406 }

```

PROBLEMS TERMINAL OUTPUT PORTS GITLENS COMMENTS DEBUG CONSOLE

El archivo ha sido generado correctamente y se ha eliminado el registro.
Presione una tecla para continuar . . .

PVBI_Act11_0_932.c M Registers.txt U X									
	No.	Matricula	Ape Paterno	Ape Materno	Nombre	Fecha de Nacimiento	Edad	Sexo	CURP
1									
2	1	300256	MOLINA	ORTIZ	OSCAR	31-01-1993	30	Hombre	M000930131HMCLRS00
3	2	300649	ROMERO	SANDOVAL	ANTONIO	17-12-1972	50	Hombre	ROSA721217HYNNMN00
4	3	301084	VALENCIA	VARELA	ENRIQUE	11-06-1954	69	Hombre	VAVE540611HSPLRN06
5	4	301501	BELTRAN	VALENZUELA	ENRIQUE	20-08-1954	69	Hombre	BEVE540820HOCCLN04
6	5	301761	GUERRERO	MENDEZ	PEDRO	07-04-1967	56	Hombre	GUMP670407HTSRND00
7	6	301965	SOLIS	PACHECO	MARIO	17-03-1982	41	Hombre	SOPM820317HNTLCR04
8	7	302152	CERVANTES	ORTIZ	ENRIQUE	11-09-1940	83	Hombre	CEOE400911HTSRRN07
9	8	302178	CERVANTES	MUNGUIA	ENRIQUE	29-06-1979	44	Hombre	CEME790629HDFRNN09
10	9	302339	SALAZAR	MERCADO	JUAN	01-07-1986	37	Hombre	SAMJ860701HNTLRN00
11	10	302638	MENENDEZ	CRUZ	LUIS	18-07-1974	49	Hombre	MECL740718QTNRS02
12	11	304124	PENA	CERVANTES	GLORIA	19-07-1911	112	Mujer	PEG6110719MGTNRL06
13	12	304635	PACHECO	SALAZAR	CARLOS	29-07-1960	63	Hombre	PASC600729HQRCLR09
14	13	305788	VARELA	ESPINOSA	MARIA	25-03-1921	102	Mujer	VAEM210325MQTRSR05
15	14	306058	ROMERO	GOMEZ	EDUARDO	16-09-1900	123	Hombre	ROGE000916HCCMD03
16	15	306253	VALENZUELA	CONTRERAS	CARLOS	07-02-2000	23	Hombre	VACC000207HAGLNRA2
17	16	306514	BAUTISTA	VALENCIA	FRANCISCO	11-06-1961	62	Hombre	BAVF610611HBCTLR03
18	17	306942	SOTO	FLORES	CLARA	21-04-1920	103	Mujer	SOFC200421MMCTLL07
19	18	307128	CASILLAS	CONTRERAS	ALBERTO	27-02-1974	49	Hombre	CXCA740227HBSSNL01
20	19	307355	CONTRERAS	ESCOBAR	ISRAEL	25-03-1928	95	Hombre	COEI280325HMSNSS01
21	20	307455	SOTO	BARRIOS	ARTURO	06-05-2014	9	Hombre	SOBA140506HOCTRRB8
22	21	307665	ROMERO	CABRERA	GABRIELA	23-06-1954	69	Mujer	ROCG540623MGRMBB02
23	22	308248	VALENZUELA	SALAZAR	DANIEL	07-10-1955	68	Hombre	VASD551007HNECLN09
24	23	308388	BELTRAN	MERCADO	RAQUEL	02-10-2009	14	Mujer	BEMR091002MGRLRQA4
25	24	308516	CERVANTES	MERCADO	PAULA	02-12-1907	115	Mujer	CEMP071202MCHRRLO2
26	25	309767	ROMAN	CONTRERAS	EMILIO	05-09-1925	98	Hombre	ROCE250905HQRMMN03
27	26	310717	JIMENEZ	RIVERA	LUIS	24-07-1944	79	Hombre	JIRL440724HYNMVS09
28	27	310856	MENDEZ	MORENO	ISRAEL	12-02-1989	34	Hombre	MEMI890212HOCNRS04
29	28	311336	ROLDAN	SUAAREZ	MARIA	20-07-1947	76	Mujer	ROSM470720MCCLRR03
30	29	311852	GONZALES	ESPINOZA	VICTORIA	13-08-1973	50	Mujer	GOEV730813MDGNSC01
31	30	312085	CASTANEDA	ALVAREZ	ADRIANA	19-10-2014	9	Mujer	CAAA141019MTLSLDB5
32	31	312545	ESCOBAR	URIBE	FRANCISCO	13-09-1966	57	Hombre	EOUF660913HTLSRR05
33	32	313263	ALVAREZ	MALDONADO	GLORIA	30-08-1962	61	Mujer	AAMG620830MBCLLL00
34	33	313294	BAUTISTA	MEDINA	MIGUEL	16-04-1954	69	Hombre	BAMM540416HPLTDG06
35	34	313354	VEGA	URIBE	MARIO	27-01-1947	76	Hombre	VEUM470127HCMGRR06
36	35	313534	MENDOZA	ESTRELLA	MARIA	15-07-1923	100	Mujer	MEEM230715MGTNRSR06
37	36	313745	SOLANO	ZAMORA	MIGUEL	25-09-1905	118	Hombre	SOZM050925HVZLMG00

FUNCIONES ADICIONALES:

```
409 Tbirthday getBirthday()
410 {
411     Tbirthday tempBirthday;
412     int days[] = {0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};
413     int daysBis[] = {0, 31, 29, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};
414
415     tempBirthday.year = valid("Ingresa tu año de nacimiento: ", 1900, 2023);
416     tempBirthday.month = valid("Ingresa tu mes de nacimiento: ", 1, 12);
417
418     if (isLapYear(tempBirthday.year))
419     {
420         tempBirthday.day = valid("Ingresa tu día de nacimiento: ", 1, daysBis[tempBirthday.month]);
421     }
422     else
423     {
424         tempBirthday.day = valid("Ingresa tu día de nacimiento: ", 1, days[tempBirthday.month]);
425     }
426
427     return tempBirthday;
428 }
```

```
430 int existElem(Tstdnt students[], int longi, int num)
431 {
432     int i;
433
434     for (i = 0; i < longi; i++)
435     {
436         if (students[i].matricula == num)
437         {
438             return i;
439         }
440     }
441     return -1;
442 }
```

```
444 int getAge(Tbirthday birth)
445 {
446     int age;
447     age = 2023 - birth.year;
448     if (birth.month > 11)
449     {
450         age = age - 1;
451     }
452     return age;
453 }
```

```

455 void getCURP(Tstdnt stdntData, char CURP[])
456 {
457     char firstFourLetters[5];
458     char temp[2];
459     int startPosition, startPosition2, startPosition3;
460     startPosition = nameCompound(stdntData.personalName.lastName1);
461     startPosition2 = nameCompound(stdntData.personalName.LastName2);
462     startPosition3 = nameCompound(stdntData.personalName.name);
463
464     if (stdntData.personalName.lastName1[0] != '\0')
465     {
466         CURP[0] = stdntData.personalName.lastName1[startPosition];
467         CURP[1] = noVowelsApComp(stdntData.personalName.lastName1, startPosition);
468         CURP[13] = getConsonant(stdntData.personalName.lastName1, startPosition);
469     }
470     else
471     {
472         CURP[0] = 'X';
473         CURP[1] = 'X';
474         CURP[13] = 'X';
475     }
476
477     if (stdntData.personalName.LastName2[0] != '\0')
478     {
479         CURP[2] = stdntData.personalName.LastName2[startPosition2];
480         CURP[14] = getConsonant(stdntData.personalName.LastName2, startPosition2);
481     }
482     else
483     {
484         CURP[2] = 'X';
485         CURP[14] = 'X';
486     }
487
488     CURP[3] = stdntData.personalName.name[startPosition3];
489     CURP[15] = getConsonant(stdntData.personalName.name, startPosition3);
490
491     firstFourLetters[0] = CURP[0];
492     firstFourLetters[1] = CURP[1];
493     firstFourLetters[2] = CURP[2];
494     firstFourLetters[3] = CURP[3];
495     firstFourLetters[4] = '\0';

```

```
497     if (antiSonant(firstFourLetters))
498     {
499         CURP[1] = 'X';
500     }
501
502     convertNumber(stdntData, CURP);
503
504     CURP[10] = stdntData.sex[0];
505
506     for (int i = 0; i < 2; i++)
507     {
508         CURP[11 + i] = stdntData.placeBirth[i];
509     }
510
511     if (stdntData.date.year < 2000)
512     {
513         CURP[16] = '0';
514     }
515     else
516     {
517         if (stdntData.date.year <= 2009)
518         {
519             CURP[16] = 'A';
520         }
521         else
522         {
523             if (stdntData.date.year <= 2019)
524             {
525                 CURP[16] = 'B';
526             }
527             else
528             {
529                 CURP[16] = 'C';
530             }
531         }
532     }
533     sprintf(temp, "%d", numRandom(0, 9));
534     CURP[17] = temp[0];
535     CURP[18] = '\\0';
536
537     strcpy(stdntData.curp, CURP);
538 }
```

```

540 void convertNumber(Tstdnt studnt, char CURP[18])
541 {
542     char date[6];
543     studnt.date.year = studnt.date.year % 100;
544
545     sprintf(date, "%02d%02d%02d", studnt.date.year, studnt.date.month, studnt.date.day);
546
547     for (int i = 0; i < 6; i++)
548     {
549         CURP[4 + i] = date[i];
550     }
551 }

```

```

553 Tstdnt addOneStdnt(Tstdnt studentArray[], int position)
554 {
555     Tstdnt tempStudentArray;
556     char curp[18];
557
558     do
559     {
560         tempStudentArray.matricula = numRandom(300000, 399999);
561     } while (existElem(studentArray, position, tempStudentArray.matricula) != -1);
562
563     // Name
564     LastName(tempStudentArray.personalName.lastName1);
565     LastName(tempStudentArray.personalName.lastName2);
566
567     if (numRandom(0, 1) == 1)
568     {
569         nameMen(tempStudentArray.personalName.name);
570         strcpy(tempStudentArray.sex, "Hombre");
571     }
572     else
573     {
574         nameWomen(tempStudentArray.personalName.name);
575         strcpy(tempStudentArray.sex, "Mujer");
576     }
577
578     // Get random birthday and age
579     tempStudentArray.date = randomBirthday();
580     tempStudentArray.age = getAge(tempStudentArray.date);
581
582     getState(tempStudentArray.state, tempStudentArray.placeBirth);
583
584     getCURP(tempStudentArray, curp);
585     strcpy(tempStudentArray.curp, curp);
586
587     return tempStudentArray;
588 }

```



```

590 Tbirthday randomBirthday()
591 {
592     Tbirthday tempBirthday;
593     int days[] = {0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};
594     int daysBis[] = {0, 31, 29, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};
595
596     tempBirthday.year = numRandom(1900, 2023);
597     tempBirthday.month = numRandom(1, 12);
598
599     if (isLapYear(tempBirthday.year))
600     {
601         tempBirthday.day = numRandom(1, daysBis[tempBirthday.month]);
602     }
603     else
604     {
605         if (tempBirthday.year == 2023)
606         {
607             tempBirthday.month = numRandom(1, 10);
608             tempBirthday.day = numRandom(1, days[tempBirthday.month]);
609         }
610         else
611         {
612             tempBirthday.day = numRandom(1, days[tempBirthday.month]);
613         }
614     }
615     return tempBirthday;
616 }

```

```

618 void displayOneStdntList(Tstdnt students)
619 {
620     printf("%-10d %-10s %-10s %-10s %02d-%02d-%04d  %-4c%-18s\n",
621         students.matricula,
622         students.personalName.lastName1,
623         students.personalName.LastName2,
624         students.personalName.name,
625         students.date.day,
626         students.date.month,
627         students.date.year,
628         students.sex[0],
629         students.curp);
630 }

```

```

632 void displayOneStdnt(Tstdnt students)
633 {
634     printf("-----\n");
635     printf("MATRICULA:    %d\n", students.matricula);
636     printf("NOMBRE:      %s\n", students.personalName.name);
637     printf("AP. PAT:      %s\n", students.personalName.lastName1);
638     printf("AP. MAT:      %s\n", students.personalName.lastName2);
639     printf("FECHA DE NAC: %02d/%02d/%d\n", students.date.day, students.date.month, students.date.year);
640     printf("EDAD:        %02d\n", students.age);
641     printf("SEXO:        %s\n", students.sex);
642     printf("LUGAR NAC:    %s\n", students.state);
643     printf("CURP:        %s", students.curp);
644     printf("\n-----\n");
645 }

```

```

647 int binarySearch(Tstdnt studentArray[], int left, int right, int number)
648 {
649     int medium;
650     while (left <= right)
651     {
652         medium = left + (right - left) / 2;
653
654         if (studentArray[medium].matricula == number)
655         {
656             return medium;
657         }
658
659         if (studentArray[medium].matricula < number)
660         {
661             left = medium + 1;
662         }
663         else
664         {
665             right = medium - 1;
666         }
667     }
668
669     return -1;
670 }

```

```

672 void bubbleSort(Tstdnt studentArray[], int n)
673 {
674     int i, j;
675     Tstdnt temp;
676     for (i = 0; i < n - 1; i++)
677     {
678         for (j = i + 1; j < n; j++)
679         {
680             if (studentArray[j].matricula < studentArray[i].matricula)
681             {
682                 temp = studentArray[i];
683                 studentArray[i] = studentArray[j];
684                 studentArray[j] = temp;
685             }
686         }
687     }
688 }

```

```

691 void swap(Tstdnt students[], int i, int j)
692 {
693     Tstdnt temp = students[i];
694     students[i] = students[j];
695     students[j] = temp;
696 }

```

```

698 int partition(Tstdnt students[], int low, int high)
699 {
700     Tstdnt pivot;
701     pivot.matricula = students[high].matricula;
702     int i = low - 1;
703
704     for (int j = low; j <= high - 1; j++)
705     {
706         if (students[j].matricula <= pivot.matricula)
707         {
708             i++;
709             swap(students, i, j);
710         }
711     }
712     swap(students, i + 1, high);
713     return i + 1;
714 }

```

```
716 void quicksort(Tstdnt students[], int low, int high)
717 {
718     if (low < high)
719     {
720         int pi = partition(students, low, high);
721
722         quicksort(students, low, pi - 1);
723         quicksort(students, pi + 1, high);
724     }
725 }
```