



Universidad Autónoma de Baja California

Facultad de Ingeniería, Arquitectura y Diseño



ARCHIVOS INDEXADOS

PROGRAMACIÓN ESTRUCTURADA

ANEXO 14

Ingeniero en Software y Tecnologías Emergentes

Brayan Ivan Perez Ventura

372781

MENU:

```
87 void menu()
88 {
89     int op;
90     int max_registers, position;
91     position = LoadBinaryFile();
92     getIndexFile(position);
93     getTempFiles(position);
94     max_registers = position * 1.25;
95     int ord = 0;
96     do
97     {
98         system("CLS");
99         // printf("Registros: %d\n", max_registers);
100         op = msge_menu();
101         system("CLS");
102
103         switch (op)
104         {
105             case 1:
106                 if (position + 1 < max_registers)
107                 {
108                     addRegister(position, max_registers);
109                     ord = 0;
110                     position++;
111                 }
112                 else
113                 {
114                     printf("Vector lleno\n");
115                 }
116                 break;
117             case 2:
118                 deleteRegister(position, max_registers, ord);
119                 break;
120             case 3:
121                 findRegister(position, ord);
122                 break;
123             case 4:
124                 ord = OrderIndex(position, ord);
125                 break;
126             case 5:
127                 ord = displayReg(position, ord);
128                 break;
129             case 6:
130                 genTxT(position, ord);
131                 break;
132             case 7:
133                 packageFile(position);
134                 break;
135             default:
136                 break;
137         }
138         if (op != 0)
139         {
140             system("PAUSE");
141         }
142     } while (op != 0);
143 }
144 }
```

```
73 int msge_menu()
74 {
75     printf("----- M E N U ----- \n");
76     printf("1.- Agregar\n");
77     printf("2.- Eliminar\n");
78     printf("3.- Buscar\n");
79     printf("4.- Ordenar\n");
80     printf("5.- Mostrar\n");
81     printf("6.- Generar archivo de texto\n");
82     printf("7.- Empaquetar\n");
83     printf("0.- Salir\n");
84     return valid("Selecciona una opcion: ", 0, 7);
85 }
```

PROBLEMS TERMINAL OUTPUT PORTS

```
----- M E N U -----
1.- Agregar
2.- Eliminar
3.- Buscar
4.- Ordenar
5.- Mostrar
6.- Generar archivo de texto
7.- Empaquetar
0.- Salir
Selecciona una opcion: █
```

AÑADIR REGISTRO:

```
191 void addRegister(int position, int max_registers)
192 {
193     TIndexStrct employees[max_registers];
194     Twrkr tempEmployee;
195     TIndexStrct temp;
196     tempEmployee = generateRegister();
197     fillIndexRegister(employees);
198     do
199     {
200         tempEmployee.enrollment = numRandom(300000, 399999);
201     } while (existElem(employees, position, tempEmployee.enrollment) != -1);
202
203     addRegisterData(tempEmployee);
204     temp.enrollment = tempEmployee.enrollment;
205     temp.index = position;
206     addRegisterIndex(temp);
207     displayRegEmp(tempEmployee);
208 }
```

PROBLEMS TERMINAL OUTPUT PORTS GITLENS

Añadido correctamente

Presione una tecla para continuar . . .

ENCONTRAR REGISTRO:

```
210 void findRegister(int position, int ord)
211 {
212     int index;
213     TIndexStrct indexFile[position];
214     TWkr regTemp;
215     FILE *fa;
216     index = valid("Ingrese la matricula del estudiante a eliminar: ", 300000, 399999);
217     fillIndexRegister(indexFile);
218     if (ord)
219     {
220         index = binarySearch(indexFile, 0, position, index);
221     }
222     else
223     {
224         index = existElem(indexFile, position, index);
225     }
226
227     if (index != -1)
228     {
229         fa = fopen("datos.dat", "rb");
230         fseek(fa, index * sizeof(TWkr), SEEK_SET);
231         fread(&regTemp, sizeof(TWkr), 1, fa);
232         fclose(fa);
233
234         if (regTemp.status)
235         {
236             displayRegEmp(regTemp);
237         }
238         else
239         {
240             printf("Alumno eliminado con anterioridad\n");
241         }
242     }
243 }
```

PROBLEMS TERMINAL OUTPUT PORTS GITLENS COMMENTS

```
Ingrese la matricula del estudiante a eliminar: 314318
Matricula:   314318
Nombre:      PILAR
Ap. Paterno: ROMERO
Ap. Materno: SOTO
Sexo:        MUJER
Edad:        26
Posicion:    Enferm
Num. Cel:    646-1020168
Lug. Nacim:  CC
Presione una tecla para continuar . . .
```

ELIMINAR ESTUDIANTE:

```
245 void deleteRegister(int position, int max_registers, int ord)
246 {
247     int index;
248     TIndexStrct indexFile[max_registers];
249     TWrkr regTemp;
250     FILE *fa;
251     index = valid("Ingrese la matricula del estudiante a eliminar: ", 300000, 399999);
252     fillIndexRegister(indexFile);
253     if (ord)
254     {
255         index = binarySearch(indexFile, 0, position, index);
256     }
257     else
258     {
259         index = existElem(indexFile, position, index);
260     }
261
262     if (index != -1)
263     {
264         fa = fopen("datos.dat", "rb");
265         fseek(fa, index * sizeof(TWrkr), SEEK_SET);
266         fread(&regTemp, sizeof(TWrkr), 1, fa);
267         fclose(fa);
268
269         if (regTemp.status)
270         {
271             displayRegEmp(regTemp);
272             if (valid("Desea eliminarlo? (1.- Si, 0.- No): ", 0, 1))
273             {
274                 regTemp.status = 0;
275                 fa = fopen("datos.dat", "rb+");
276                 fseek(fa, index * sizeof(TWrkr), SEEK_SET);
277                 fwrite(&regTemp, sizeof(TWrkr), 1, fa);
278                 fclose(fa);
279                 printf("Eliminado con exito\n");
280             }
281         }
282         else
283         {
284             printf("Alumno eliminado con anterioridad\n");
285         }
286     }
287     else
288     {
289         printf("El alumno no existe\n");
290     }
291 }
```

PROBLEMS TERMINAL OUTPUT PORTS GITLENS COMMENTS DEBUG CONSOLE

```
Ingrese la matricula del estudiante a eliminar: 319601
Matricula: 319601
Nombre: LUIS
Ap. Paterno: ROJAS
Ap. Materno: ESTRELLA
Sexo: HOMBRE
Edad: 34
Posicion: Abogado
Num. Cel: 646-1012697
Lug. Nacim: MS
Desea eliminarlo? (1.- Si, 0.- No): 1
Eliminado con exito
Presione una tecla para continuar . . .
```

ORDENAR

```
293 | int OrderIndex(int position, int flag)
294 | {
295 |     TIndexStrct Index[position];
296 |     FILE *fa;
297 |
298 |     if (flag)
299 |     {
300 |         printf("El vector ya ha sido ordenado con anterioridad");
301 |     }
302 |     else
303 |     {
304 |         fillIndexRegister(Index);
305 |         bubbleSort(Index, position);
306 |         fa = fopen("datos_index.dat", "wb");
307 |         fwrite(Index, sizeof(TWrkr), 1, fa);
308 |         fclose(fa);
309 |         printf("El vector ha sido ordenado\n");
310 |         return 1;
311 |     }
312 |     return flag;
313 | }
```

PROBLEMS

TERMINAL

OUTPUT

PORTS

GITLENS

C

```
El vector ha sido ordenado
Presione una tecla para continuar . . .
```

IMPRIMIR REGISTRO

```
315 int displayReg(int position, int flag)
316 {
317     FILE *fa;
318     TWkrkr temp;
319     int i = 0;
320     fa = fopen("datos.dat", "rb");
321     if (valid("Imprimir datos\n 1.- Ordenado\n 0.- Normal\n Selecciona una opcion: ", 0, 1))
322     {
323         TIndexStrct Indexs[position];
324         fillIndexRegister(Indexs);
325         if (!flag)
326         {
327             bubbleSort(Indexs, position);
328             flag = 1;
329         }
330
331         for (i = 0; i < position; i++)
332         {
333             fseek(fa, Indexs[i].index * sizeof(TWkrkr), SEEK_SET);
334             fread(&temp, sizeof(TWkrkr), 1, fa);
335             if (temp.status)
336             {
337                 displayListEmp(temp, i);
338             }
339         }
340         fclose(fa);
341     }
342     else
343     {
344         system("CLS");
345         if (fa)
346         {
347             while (fread(&temp, sizeof(TWkrkr), 1, fa))
348             {
349                 if (temp.status)
350                 {
351                     displayListEmp(temp, i);
352                     i++;
353                 }
354             }
355             fclose(fa);
356         }
357     }
358     return flag;
359 }
```

PROBLEMS TERMINAL OUTPUT PORTS GIT

```
Imprimir datos
 1.- Ordenado
 0.- Normal
 Selecciona una opcion: 
```

PROBLEMS	TERMINAL	OUTPUT	PORTS	GITLENS	COMMENTS	DEBUG CONSOLE					
1009	309459	HECTOR		CERVANTES		IGLESIAS	HOMBRE	AnlVent	CM	46	6461030224
1010	309465	EDUARDO		MACIEL		ROMAN	HOMBRE	AnMktDg	NT	21	6461030704
1011	309466	RAQUEL		MACIEL		DURAN	MUJER	EspERen	MN	50	6461028287
1012	309467	LUIS		REYES		DELGADO	HOMBRE	Traduct	SR	46	6461026132
1013	309476	ARMANDO		CASILLAS		CAMACHO	HOMBRE	TecSup	AG	39	6461013823
1014	309485	JORGE		RAMOS		ROJAS	HOMBRE	GteProj	OC	36	6461010533
1015	309495	GUILLERMO		ESTRELLA		CAMACHO	HOMBRE	DisUX	NE	46	6461024759
1016	309498	DANIEL		ALVARADO		RUBIO	HOMBRE	EdCont	YN	33	6461013563
1017	309499	ERNESTO		GUERRERO		URIBE	HOMBRE	IngRed	YN	28	6461029439
1018	309510	JULIA		VARELA		CERVANTES	MUJER	ChefEje	SL	50	6461015169
1019	309520	HECTOR		GUERRERO		VALENZUELA	HOMBRE	MedGen	NL	29	6461007931
1020	309532	MARIA		ROJAS		ESPINOSA	MUJER	GteProj	CH	35	6461001447
1021	309540	GLORIA		AGUILAR		RUBIO	MUJER	IngCivil	VZ	48	6461015660
1022	309549	LAURA		MERCADO		BARRIOS	MUJER	ChefEje	SL	44	6461012076
1023	309552	PILAR		CALZADA		VALENZUELA	MUJER	AsistAdm	VZ	24	6461028689
1024	309576	ERNESTO		LARA		CERVANTES	HOMBRE	Abogado	MS	47	6461021959
1025	309610	GABRIELA		SOLIS		CORDERO	MUJER	AnMktDg	VZ	38	6461005544
1026	309627	ANA		CRUZ		VALENZUELA	MUJER	DevSoft	MC	28	6461010491
1027	309637	MARTA		ROJAS		BELTRAN	MUJER	Contado	TS	28	6461014655
1028	309638	RAQUEL		RAMIREZ		MORA	MUJER	AsServ	NL	20	6461027396
1029	309655	ANTONIA		SILVA		LARA	MUJER	Arquitect	DG	44	6461019876
1030	309656	ROSA		MORENO		ARIAS	MUJER	ChefEje	NE	38	6461014881
1031	309657	MARTA		ARIAS		TORRES	MUJER	TrabSoc	TL	36	6461012917
1032	309660	MIGUEL		CASTANEDA		MACIEL	HOMBRE	PsOrg	HG	28	6461002085
1033	309661	ANTONIA		CASILLAS		PENA	MUJER	IngRed	PL	32	6461019665
1034	309670	JULIA		SOTO		BARRIOS	MUJER	RepVInt	NE	21	6461022133
1035	309678	ALEJANDRO		LOPEZ		OCHOA	HOMBRE	Arquitect	CH	49	6461014007
1036	309684	ROBERTO		ESPINOSA		MACIAS	HOMBRE	ChefEje	BC	45	6461026190
1037	309686	ALBERTO		BAUTISTA		DIAZ	HOMBRE	DisGraf	SL	38	6461010124
1038	309688	NATALIA		ALVAREZ		CERVANTES	MUJER	Contado	NE	24	6461021312
1039	309695	PAULA		LEAL		FERNANDEZ	MUJER	EspERen	QT	36	6461024448
1040	309709	GLORIA		CASTILLO		ESTRELLA	MUJER	EspLog	NT	39	6461017370
1041	309741	HECTOR		ROJAS		VALENCIA	HOMBRE	ProfPrim	NL	24	6461010552

GENERAR ARCHIVO DE TEXTO

```
361 void genTxT(int position, int flag)
362 {
363     TWrkr temp;
364     if (valid("Generar archivo de texto\n 1.- Ordenado\n 0.- Normal\n Selecciona una opcion: ", 0, 1))
365     {
366         TIndexStrct Indexs[position];
367         fillIndexRegister(Indexs);
368         if (!flag)
369         {
370             bubbleSort(Indexs, position);
371             flag = 1;
372         }
373         getTXI(Indexs, position, "datos_activos.txt", 1);
374         getTXI(Indexs, position, "datos_inactivos.txt", 0);
375         printf("Archivos generados correctamente\n");
376     }
377     else
378     {
379         FILE *fa, *pa, *ma;
380         int i = 0, j = 0;
381         fa = fopen("datos.dat", "rb");
382         pa = fopen("datos_activos.txt", "w");
383         ma = fopen("datos_inactivos.txt", "w");
384         if (fa)
385         {
386             while (fread(&temp, sizeof(TWrkr), 1, fa))
387             {
388                 if (temp.status)
389                 {
390                     addOneEmployeeTxT(temp, i, pa);
391                     i++;
392                 }
393                 else
394                 {
395                     addOneEmployeeTxT(temp, j, ma);
396                     j++;
397                 }
398             }
399             fclose(fa);
400             printf("Archivos generados correctamente\n");
401         }
402         fclose(pa);
403         fclose(ma);
404     }
405 }
```

PROBLEMS TERMINAL OUTPUT PORTS GITLENS C

```
Generar archivo de texto
 1.- Ordenado
 0.- Normal
 Selecciona una opcion: 1
Archivos generados correctamente
Presione una tecla para continuar . . .
```

```
▼ PVBI_Act14_0_932
  ▼ output
    datos_activos.txt   U
    datos_backup.dat   U
    datos_inactivos.txt   U
    datos_index.dat   M
```

EMPAQUETAR

```
407 void packageFile(int position)
408 {
409     FILE *fa;
410     Twrkr vectTemp[position];
411     Twrkr vectTempActive[position];
412     int j = 0;
413     fillRegister(vectTemp);
414     rename("datos.dat", "datos.bak");
415     fa = fopen("datos.dat", "wb");
416     if (fa)
417     {
418         for (int i = 0; i < position; i++)
419         {
420             if (vectTemp[i].status)
421             {
422                 vectTempActive[j++] = vectTemp[i];
423             }
424         }
425         fwrite(vectTempActive, sizeof(Twrkr), j, fa);
426         fclose(fa);
427         printf("Empaquetado correctamente\n");
428     }
429 }
```

```
218 fillIndexRegister(indexFile);
```

PROBLEMS TERMINAL OUTPUT PORTS GITLENS CO

Empaquetado correctamente
Presione una tecla para continuar . . .

output	
datos_activos.txt	U
datos_backup.dat	U
datos_inactivos.txt	U
datos_index.dat	M
datos.bak	U
datos.dat	M

FUNCIONES EXTRAS:

```
432 Twrkr generateRegister()
433 {
434     Twrkr temp;
435     temp.age = numRandom(18, 40);
436
437     if (numRandom(0, 1))
438     {
439         nameMen(temp.name);
440         strcpy(temp.sex, "HOMBRE");
441     }
442     else
443     {
444         nameWomen(temp.name);
445         strcpy(temp.sex, "MUJER");
446     }
447
448     LastName(temp.LastName1);
449     LastName(temp.LastName2);
450
451     getState(temp.state);
452     getJobPositions(temp.JobPstion);
453
454     temp.cellPhone = numRandom(1000000, 1999999);
455
456     temp.status = 1;
457
458     return temp;
459 }
```

```
461 int fillRegister(TWrkr employess[])
462 {
463     FILE *fa;
464     fa = fopen("datos.dat", "rb");
465     int i = 0;
466     if (fa)
467     {
468         while (fread(&employess[i], sizeof(TWrkr), 1, fa))
469         {
470             i++;
471         }
472         fclose(fa);
473         return i;
474     }
475     else
476     {
477         printf("El archivo no existe.\n");
478     }
479     return 0;
480 }
```

```
482 int fillIndexRegister(TIndexStrct Index[])
483 {
484     FILE *fa;
485
486     int i = 0;
487     fa = fopen("datos_index.dat", "rb");
488     if (fa)
489     {
490         while (fread(&Index[i], sizeof(TIndexStrct), 1, fa))
491         {
492             i++;
493         }
494         fclose(fa);
495         return i;
496     }
497     else
498     {
499         printf("El archivo no existe\n");
500     }
501     return 0;
502 }
```

```

504 void displayListEmp(TWrkr employee, int i)
505 {
506     printf("%-10d %-12d %-15s %-20s %-15s %-10s %-10s %-10s %-5d 646%-10d\n",
507           i + 1,
508           employee.enrollment,
509           employee.name,
510           employee.LastName1,
511           employee.LastName2,
512           employee.sex,
513           employee.JobPstion,
514           employee.state,
515           employee.age,
516           employee.cellPhone);
517 }

```

```

519 int existElem(TIndexStrct employee[], int longi, Tkey num)
520 {
521     int i;
522     for (i = 0; i < longi; i++)
523     {
524         if (employee[i].enrollment == num)
525         {
526             return employee[i].index;
527         }
528     }
529     return -1;
530 }

```

```

532 void getTempFiles(int n)
533 {
534     TWkr tempWrkr[n];
535
536     fillRegister(tempWrkr);
537
538     rename("datos.dat", "datos_backup.dat");
539
540     FILE *fa;
541     fa = fopen("datos.dat", "wb");
542
543     fwrite(tempWrkr, sizeof(TWrkr), n, fa);
544     fclose(fa);
545 }

```

```
547 void addRegisterData(TWrkr employee)
548 {
549     FILE *fa;
550     fa = fopen("datos.dat", "ab");
551     fseek(fa, 0, SEEK_END);
552     fwrite(&employee, sizeof(TWrkr), 1, fa);
553     fclose(fa);
554 }
```

```
556 void addRegisterIndex(TIndexStrct employeeIndx)
557 {
558     FILE *fa;
559     fa = fopen("datos_index.dat", "ab");
560     fseek(fa, 0, SEEK_END);
561     fwrite(&employeeIndx, sizeof(TIndexStrct), 1, fa);
562     fclose(fa);
563 }
```

```
565 int binarySearch(TIndexStrct employee[], int left, int right, Tkey number)
566 {
567     int medium;
568     while (left <= right)
569     {
570         medium = left + (right - left) / 2;
571
572         if (employee[medium].enrollment == number)
573         {
574             return employee[medium].index;
575         }
576
577         if (employee[medium].enrollment < number)
578         {
579             left = medium + 1;
580         }
581         else
582         {
583             right = medium - 1;
584         }
585     }
586
587     return -1;
```

```

603 void getTXT(TIndexStrct Indexs[], int position, char fileName[], int flag)
604 {
605     int i;
606     FILE *fa;
607     FILE *pa;
608     TWkrkr temp;
609     int j;
610     fa = fopen(fileName, "w");
611     pa = fopen("datos.dat", "rb");
612     fprintf(fa, "%-10s %-12s %-15s %-20s %-15s %-10s %-10s %-8s %-5s %-10s\n",
613         "No.",
614         "Matricula",
615         "Nombre",
616         "Ap. Paterno",
617         "Ao. Materno",
618         "Sexo",
619         "Posicion",
620         "Estado",
621         "Edad",
622         "Num. Cel");
623
624     j = 0;
625     for (i = 0; i < position; i++)
626     {
627         fseek(pa, Indexs[i].index * sizeof(TWkrkr), SEEK_SET);
628         fread(&temp, sizeof(TWkrkr), 1, pa);
629         if (temp.status == flag)
630         {
631             addOneEmployeeTxT(temp, j, fa);
632             j++;
633         }
634     }
635     fclose(pa);
636     fclose(fa);
637 }

```

```

654  **** ORDER FUNCTIONS ****
655  void swap(TIndexStrct employees[], int i, int j)
656  {
657      TIndexStrct temp = employees[i];
658      employees[i] = employees[j];
659      employees[j] = temp;
660  }
661
662  int partition(TIndexStrct employees[], int low, int high)
663  {
664      TIndexStrct pivot;
665      pivot.enrollment = employees[high].enrollment;
666      int i = low - 1;
667
668      for (int j = low; j <= high - 1; j++)
669      {
670          if (employees[j].enrollment <= pivot.enrollment)
671          {
672              i++;
673              swap(employees, i, j);
674          }
675      }
676      swap(employees, i + 1, high);
677      return i + 1;
678  }
679
680  void quicksort(TIndexStrct employees[], int low, int high)
681  {
682      if (low < high)
683      {
684          int pi = partition(employees, low, high);
685
686          quicksort(employees, low, pi - 1);
687          quicksort(employees, pi + 1, high);
688      }
689  }
690
691  void bubbleSort(TIndexStrct employees[], int n)
692  {
693      int i, j;
694      TIndexStrct temp;
695      for (i = 0; i < n - 1; i++)
696      {
697          for (j = i + 1; j < n; j++)
698          {
699              if (employees[j].enrollment < employees[i].enrollment)
700              {
701                  temp = employees[i];
702                  employees[i] = employees[j];
703                  employees[j] = temp;
704              }
705          }
706      }
707  }

```