|  |
| --- |
|  |
| Programming With Persistent Data Assignment |
|  |
|  |
| **Brian Willis – C12331591** |
|  |

|  |
| --- |
|  |

Psuedocode Version

PROGRAM assignment

main

IF create()=1

PRINT employees.dat was created

ELSE

PRINT employees.dat was found

ENDIF

WHILE exit=0

decision=0

PRINT Please select an operation from 1 to 5:

1. Add Employee

2. Delete Employee

3. List Employees

4. Compact Records

5. Exit

READ decision

IF decision=1

add()

ELSE IF decision=2

delete()

ELSE IF decision=3

list()

ELSE IF decision=4

compact()

ELSE IF decision=5

exit=1

ELSE

PRINT Invalid Input

ENDIF

ENDWHILE

END main

add()

PRINT Enter employee number

READ employeenumber

WHILE end of file is not reached

read next record

IF addition.employeenumber=current.employeenumber

found=1

ENDIF

ENDWHILE

IF found=1

PRINT This number already exists. Overwrite the record?

WHILE overwrite=0

READ answer

IF answer=y

overwrite=1

ELSE IF answer=0

return

ELSE

PRINT invalid input

ENDWHILE

ENDIF

PRINT Enter first name

READ addition.firstname

PRINT Enter surname

READ addition.surname

PRINT Enter address

READ addition.address

PRINT Enter department code

READ addition.departmentcode

PRINT Enter duration

READ addition.duration

write to file

update(1, 0)

END add

del()

PRINT Enter employee number

READ employeenumber

WHILE found=0

IF end of file is not reached

read next record

IF addition.employeenumber=current.employeenumber

found=1

ENDIF

ELSE

found=2

ENDIF

ENDWHILE

IF found<>1

PRINT Number does not exist in file

ELSE

employeenumber[0]='\*'

write to file

update(0, 1)

PRINT Record was marked for deletion

ENDIF

END del

list()

WHILE end of file is not reached

read next record

IF current.employeenumber[0]!='\*'

PRINT current.employeenumber

PRINT current.firstname

PRINT current.surname

PRINT current.address

PRINT current.departmentcode

PRINT current.duration

ENDIF

ENDWHILE

rewind file pointer

IF header.totalrecords=0

PRINT No records were found in the file

ELSE

PRINT Active employees:

PRINT activeemployees

PRINT Total records:

PRINT totalrecords

PRINT Records marked for deletion:

PRINT deletedrecords

ENDIF

END list

compact()

read first record

htransfer.totalrecords=htransfer.totalrecords-htransfer.deletedrecords

htransfer.deletedrecs=0

WHILE end of file is not reached

read next record

IF rtransfer.employeenumber[0]<>'\*'

write to file

ENDIF

ENDWHILE

PRINT The compacted file was created

END compact

create()

IF checkpointer=null

addheader.totalrecords=0

addheader.deletedrecords=0

write to file

return 1

ELSE

return 0

ENDIF

END create

update(added, marked)

IF added=1

access.totalrecords=access.totalrecords+1

ENDIF

IF marked=1

access.deletedrecords=access.deletedrecords+1

ENDIF

write to file

END update

This design separates each different operation into a function. This should modularise the program, allowing each operation to be easily tested without the entire program having to be finished. There are also functions to create or detect the file when the program opens and to update the header record when a record is added or deleted. This allows for more straightforward code in the functions for adding and deleting. Because the main function is only used for the main menu and calling the other functions, it can be quickly implemented fully and then used as a test bed for the other functions as they are written.

C Version

/\*Program to manage a database of employee records

Due 23 April 2013

Brian Willis

\*/

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

struct employee

{

char emp\_num[6];

char firstname[16];

char surname[16];

char address[26];

char dep\_code[10];

char duration[3];

};

struct header

{

int totalrecs;

int delrecs;

char filler[16];

};

#define RECSIZE sizeof(struct employee)

#define HEADERSIZE sizeof(struct header)

void add(void);

void del(void);

void list(void);

void compact(void);

int create(void);

void update(int, int);

main()

{

if (create()==1)

{

printf ("employees.dat was created.");

}

else

{

printf ("employees.dat was found.");

}

int decision;

int exit=0;

while (exit==0)

{

decision=0;

flushall();//flushing anything left over from previous operations

printf ( "\n\nPlease select an operation from 1 to 5:\n"

"1. Add Employee\n"

"2. Delete Employee\n"

"3. List Employees\n"

"4. Compact Records\n"

"5. Exit\n"

);

scanf ("%d", &decision);

flushall();

switch (decision)

{

case 1:

{

add();

break;

}

case 2:

{

del();

break;

}

case 3:

{

list();

break;

}

case 4:

{

compact();

break;

}

case 5:

{

exit=1;

break;

}

default:

{

printf ("\nInvalid input. Please enter a number between 1 and 5 to select an operation.");

}

}//end switch

}//end while

}//end main()

void add()

{

struct employee addition;

struct employee check;

FILE \*add;

add=fopen("employees.dat", "rb+");

fseek(add, HEADERSIZE, SEEK\_SET);

int found=0;

int overwrite=0;

char answer;

if (add==NULL)//error checking

{

printf ("\nERROR: File could not be opened.\n");

}

else

{

printf ("\nPlease enter the number of the employee to be added.\n");

scanf ("%5s", &addition.emp\_num);

while(found==0)

{

if(fread(&check, RECSIZE, 1, add)!=NULL)

{

if(strcmp(check.emp\_num, addition.emp\_num)==0)

{

found=1;

}//end inner if

}

else

{

found=2;

}//end if

}//end while

if (found==1)

{

fseek(add, -RECSIZE, SEEK\_CUR);

printf ("\nThe specified employee number is already listed in the file. Do you wish to overwrite the record? (y/n)\n");

do

{

scanf ("%1s", &answer);

if (answer=='y'||answer=='Y')

{

overwrite=1;

}

else if (answer=='n'||answer=='N')

{

return;

}

else

{

printf ("\nInvalid input.\n");

}//end if

}

while (overwrite==0);

}//end if

flushall();

printf ("\nPlease enter the first name of the employee.\n");

gets(addition.firstname);

addition.firstname[15]='\0';

flushall();

printf ("\nPlease enter the surname of the employee.\n");

gets(addition.surname);

addition.surname[15]='\0';

flushall();

printf ("\nPlease enter the address of the employee.\n");

gets(addition.address);

addition.address[25]='\0';

flushall();

printf ("\nPlease enter the department code of the employee.\n");

gets(addition.dep\_code);

addition.dep\_code[9]='\0';

flushall();

printf ("\nPlease enter the employee's duration (in years) working here.\n");

gets(addition.duration);

addition.duration[2]='\0';

fwrite(&addition, RECSIZE, 1, add);

if (overwrite==0)

{

update(1, 0);

}//end if

printf ("\nThe record was added to the file\n");

}//end error checking if

fclose(add);

}//end add()

void del()

{

struct employee current;

FILE \*mark;

mark=fopen("employees.dat", "rb+");

fseek(mark, HEADERSIZE, SEEK\_SET);

int found=0;

char input[6];

if (mark==NULL)

{

printf ("\nERROR: File could not be opened.\n");

}

else

{

printf ("\nPlease enter the number of the employee to be deleted.\n");

scanf ("%5s", &input);

flushall();

while(found==0)

{

if(fread(&current, RECSIZE, 1, mark)!=NULL)

{

if(strcmp(current.emp\_num, input)==0)

{

found=1;

}//end inner if

}

else

{

found=2;

}//end if

}//end while

if (found!=1)

{

printf ("\nThe specified employee number does exist in the file.\n");

}

else

{

fseek(mark, -RECSIZE, SEEK\_CUR);

current.emp\_num[0]='\*';

fwrite(&current, RECSIZE, 1, mark);

update(0, 1);

printf ("\nThe record was marked for deletion.\n");

}//end if

}//end error checking if

fclose(mark);

}//end del()

void list()

{

struct employee current;

struct header readheader;

FILE \*records;

records=fopen("employees.dat", "rb");

fseek(records, HEADERSIZE, SEEK\_SET);

if (records==NULL)//error checking

{

printf ("\nERROR: File could not be opened.\n");

}

else

{

while (fread(&current, RECSIZE, 1, records)!=NULL)

{

if (current.emp\_num[0]!='\*')

{

printf ("\nEmployee Number: %s", current.emp\_num);

printf ("\nFirst Name: %s", current.firstname);

printf ("\nSurname: %s", current.surname);

printf ("\nAddress: %s", current.address);

printf ("\nDepartment Code: %s", current.dep\_code);

printf ("\nDuration Employed: %s\n", current.duration);

}

}

rewind(records);

fread(&readheader, HEADERSIZE, 1, records);

if (readheader.totalrecs==0)

{

printf ("\nNo records were found in the file.\n");

}

else

{

printf ("\nActive employees: %d", (readheader.totalrecs)-(readheader.delrecs));

printf ("\nTotal records: %d", readheader.totalrecs);

printf ("\nRecords marked for deletion: %d\n", readheader.delrecs);

}//end if

}//end error checking

fclose(records);

}//end list()

void compact()

{

FILE \*oldf;

FILE \*newf;

oldf=fopen("employees.dat", "rb+");

newf=fopen("employeescomp.dat", "wb+");

struct header htransfer;

struct employee rtransfer;

if (oldf==NULL||newf==NULL)//error checking

{

printf ("\nERROR: File could not be opened.\n");

}

else

{

fread(&htransfer, HEADERSIZE, 1, oldf);

htransfer.totalrecs=htransfer.totalrecs-htransfer.delrecs;

htransfer.delrecs=0;

fwrite(&htransfer, HEADERSIZE, 1, newf);

while (fread(&rtransfer, RECSIZE, 1, oldf))

{

if (rtransfer.emp\_num[0]!='\*')

{

fwrite(&rtransfer, RECSIZE, 1, newf);

}

}

printf ("\nThe compacted file was created\n");

}//end error checking if

fclose(oldf);

fclose(newf);

}//end compact()

int create()

{

FILE \*check;

check=fopen("employees.dat", "rb");

int created=1;

if (check==NULL)

{

FILE \*create;

create=fopen("employees.dat", "wb");

struct header addheader;

addheader.totalrecs=0;

addheader.delrecs=0;

fwrite(&addheader, HEADERSIZE, 1, create);

fclose(create);

}

else

{

created=0;

}

fclose(check);

return(created);

}//end create()

void update(int added, int marked)

{

struct header updated;

FILE \*fileaccess;

fileaccess=fopen("employees.dat", "rb+");

fread (&updated, HEADERSIZE, 1, fileaccess);

rewind(fileaccess);

if (fileaccess==NULL)//error checking

{

printf ("\nERROR: File could not be opened.\n");

}

else

{

if (added==1)

{

updated.totalrecs++;

}

if (marked==1)

{

updated.delrecs++;

}

fwrite(&updated, HEADERSIZE, 1, fileaccess);

}//end error checking if

fclose(fileaccess);

}//end update()