Assignment 4

Generated by Doxygen 1.8.6

Tue Nov 30 2021 12:50:18

Contents

1 Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

2 File Index

2.1 File List

Here is a list of all documented files with brief descriptions:

3 Data Structure Documentation

3.1 DATE Struct Reference

This Structure contains three data fields, month, day, year.

`#include <work.h>`

Data Fields

- int month
- int day
- int year

3.1.1 Detailed Description

This Structure contains three data fields, month, day, year.

Definition at line 38 of file work.h.

3.1.2 Field Documentation

3.1.2.1 int day

The day of the date.

Definition at line 41 of file work.h.

Referenced by checkNotNull(), fillArr(), output(), and structInit().

3.1.2.2 int month

The month of the date.

Definition at line 40 of file work.h.

Referenced by checkNotNull(), fillArr(), output(), and structInit().

3.1.2.3 int year

The year of the date.

Definition at line 42 of file work.h.

Referenced by checkNotNull(), fillArr(), output(), and structInit().

The documentation for this struct was generated from the following file:

- work.h

3.2 PERSON Struct Reference

This Structure contains siz data fields, last_name, date_of_birth, ssn, street_address, city, and age.

`#include <work.h>`

Data Fields

- char last_name [20]
- struct DATE date_of_birth
- int ssn
- char street_address [30]
- char city [20]
- int age

3.2.1 Detailed Description

This Structure contains siz data fields, last_name, date_of_birth, ssn, street_address, city, and age.

Definition at line 53 of file work.h.

3.2.2 Field Documentation

3.2.2.1 int age

The age of the person.

Definition at line 60 of file work.h.

Referenced by checkNotNull(), fillArr(), output(), and structInit().

3.2.2.2 char city[20]

The city the person lives in.

Definition at line 59 of file work.h.

Referenced by checkNotNull(), fillArr(), output(), and structInit().

3.2.2.3 struct DATE date_of_birth

The person's date of birth.

Definition at line 56 of file work.h.

Referenced by checkNotNull(), fillArr(), output(), and structInit().

3.2.2.4 char last_name[20]

The last name of the person.

Definition at line 55 of file work.h.

Referenced by checkNotNull(), fillArr(), output(), sort(), and structInit().

3.2.2.5 int ssn

The person's social security number.

Definition at line 57 of file work.h.

Referenced by checkNotNull(), fillArr(), output(), and structInit().

3.2.2.6 char street_address[30]

The person's street address.

Definition at line 58 of file work.h.

Referenced by checkNotNull(), fillArr(), output(), and structInit().

The documentation for this struct was generated from the following file:

- work.h

4 File Documentation

4.1 main.c File Reference

This file contains the main function for the 4th assignment.

```
#include "work.h"
```

Macros

- #define ARR_SIZE 20

Functions

- int main ()

  This is the main function for the program, it will allocate memory for an array, open three files, will call all the proper functions, and will close the files that were opened.

4.1.1 Detailed Description

This file contains the main function for the 4th assignment.

Author

Paul Huffman

email: huffmanp4@nku.edu

Course: CSC362

Section: 002

Assignment: 4

Date

11/30/2021 +main

Hours spent on this assignment: 6

Specific portions that gave you the most trouble: fillArr

Definition in file main.c.

4.1.2 Macro Definition Documentation

4.1.2.1 #define ARR_SIZE 20

This is the number of elements in an array

Definition at line 22 of file main.c.

Referenced by main().

4.1.3 Function Documentation

4.1.3.1 int main ( )

This is the main function for the program, it will allocate memory for an array, open three files, will call all the proper functions, and will close the files that were opened.

Returns

This function will return 0.

Definition at line 34 of file main.c.

References ARR_SIZE, fileOpenAndCheck(), fillArr(), freeMem(), output(), sort(), and structInit().

```
35 {
36     /* Creates 3 pointers to files that will be used in the program. */
37   FILE *inFile1;
38   FILE *inFile2;
39   FILE *outFile;
40     /* Allocates memory for an array of PERSON. */
41   struct PERSON *arr_emp = (struct PERSON *) calloc(ARR_SIZE,
42     sizeof(struct PERSON));
43
44     /* Checks if the memory was properly allocated. */
45   if(arr_emp != NULL)
46   {
47       /* Initializes all data within the array. */
48     structInit(arr_emp, ARR_SIZE);
49
50       /* Opens all the files, and points the pointers to them. */
51     inFile1 = fileOpenAndCheck("person.txt", "rb");
52     inFile2 = fileOpenAndCheck("dob.txt", "rb");
53     outFile = fileOpenAndCheck("output.txt", "wb");
54
55       /* Fills the array, then sorts the array, then putputs the array, and
56          finally frees the memory of the array.                          */
57     fillArr(inFile1, inFile2, arr_emp);
58     sort(arr_emp, ARR_SIZE);
59     output(arr_emp, outFile);
```

```
60      freeMem(arr_emp);
61
62        /* Closes the file pointers. */
63      fclose(inFile1);
64      fclose(inFile2);
65      fclose(outFile);
66    }
67      /* This only happens if the memory is not able to be allocated. */
68    else
69    {
70      printf("Failed to allocate memory!\n");
71      printf("Quitting now...\n");
72    }
73
74    return 0;
75  }
```

4.2 work.c File Reference

This file contains the implementation of several functions for the 4th assignment.

```
#include "work.h"
```

Macros

- #define MAX_LINE_LENGTH 80

Functions

- FILE * fileOpenAndCheck (char *filename, char *openMode)

  This function will open a file, check if it is opened, and will return a pointer to the file.
- void fillArr (FILE *infile1, FILE *infile2, struct PERSON *arr_emp)

  This function will take in the two file pointers, then will fill the array of structs with their data.
- void sort (struct PERSON *arr_emp, int arr_size)

  This function implements an inseertion sort for the array of structs based on alphabetical order of the datafield last_name.
- void output (struct PERSON *arr_emp, FILE *outFile)

  This function will output an array of structs to the output file.
- void freeMem (struct PERSON *arr_emp)

  This function will free the memory that had been previously allocated.
- void structInit (struct PERSON *arr_emp, int num)

  This function will initialize all data within the array of structs to either ' ' or 0.
- int checkNotNull (struct PERSON emp)

  This function will check all data points in a PERSON to see if it is empty or not.

4.2.1 Detailed Description

This file contains the implementation of several functions for the 4th assignment.

Author

Paul Huffman

email: huffmanp4@nku.edu

Course: CSC362

Section: 002

Assignment: 4

Date

11/30/2021 +fileOpenAndCheck +fillArr +sort +output +freeMem +structInit +checkNotNull

Hours spent on this assignment: 6

Specific portions that gave you the most trouble: fillArr

Definition in file work.c.

4.2.2 Macro Definition Documentation

4.2.2.1 #define MAX_LINE_LENGTH 80

This is the max line length for a fgets call.

Definition at line 30 of file work.c.

Referenced by fillArr().

4.2.3 Function Documentation

4.2.3.1 int checkNotNull (struct **PERSON** *emp*)

This function will check all data points in a PERSON to see if it is empty or not.

Parameters

emp	This is a person that will be checked to see if it is empty.

Returns

This function will return 1 if the PERSON is not empty, and 0 otherwise.

Definition at line 249 of file work.c.

References PERSON::age, PERSON::city, PERSON::date_of_birth, DATE::day, PERSON::last_name, DATE::month, PERSON::ssn, PERSON::street_address, and DATE::year.

Referenced by output(), and sort().

```
250 {
251     /* Checks if all of the data in the struct is not set to initial values. */
252   if(emp.last_name[0] != ' ')
253   {
254     if(emp.date_of_birth.month != 0)
255     {
256       if(emp.date_of_birth.day != 0)
257       {
258         if(emp.date_of_birth.year != 0)
259         {
260           if(emp.ssn != 0)
261           {
262             if(emp.street_address[0] != ' ')
263             {
264               if(emp.city[0] != ' ')
265               {
266                 if(emp.age != 0)
267                 {
268                   return 1;
269                 }
270               }
271             }
272           }
273         }
274       }
275     }
276   }
277   return 0;
278 }
```

4.2.3.2 FILE* fileOpenAndCheck (char * *filename*, char * *openMode*)

This function will open a file, check if it is opened, and will return a pointer to the file.

Parameters

filename	This is a pointer to the string that contains name of the file that will be opened.
openMode	This is a pointer to the string that contains the mode in which the file will be opened.

Returns

This is a pointer to the file that has been opened.

Definition at line 49 of file work.c.

Referenced by main().

```
50 {
51   FILE *temp; /* This is a temporary pointer to a file. */
52
53     /* This opens a file and points it to temp. */
54   temp = fopen(filename, openMode);
55
56     /* This checks to make sure that temp is not NULL, and if it is, then **
57     ** it will print an error.                                             */
58   if(temp == NULL)
59   {
60     perror("Cant open file");
61   }
62
63   return temp;
64 }
```

4.2.3.3 void fillArr (FILE * *infile1*, FILE * *infile2*, struct PERSON * *arr_emp*)

This function will take in the two file pointers, then will fill the array of structs with their data.

Parameters

infile1	This is a pointer to the file that contains the name, address, ssn, and age of the people.
infile2	This is a pointer to the file that contains the date of birth of the people.
arr_emp	This is a pointer to an array of structs of type PERSON.

Definition at line 83 of file work.c.

References PERSON::age, PERSON::city, PERSON::date_of_birth, DATE::day, PERSON::last_name, MAX_LINE_LENGTH, DATE::month, PERSON::ssn, PERSON::street_address, and DATE::year.

Referenced by main().

```
84 {
85     /* This creates 2 strings used to hold data from the inFiles. */
86   char dString[MAX_LINE_LENGTH] = {0};
87   char eString[MAX_LINE_LENGTH] = {0};
88   int i = 0; /* A counting integer. */
89
90     /* Loops while not at the end of the file. */
91   while(!feof(infile2))
92   {
93       /* Copies a string from the files to the strings created earlier. */
94     fgets(eString, MAX_LINE_LENGTH, infile1);
95     fgets(dString, MAX_LINE_LENGTH, infile2);
96
97       /* Sets the data from infile2 to the proper data fields.          */
98     arr_emp[i].date_of_birth.month = atoi(strtok(dString, ","));
99     arr_emp[i].date_of_birth.day = atoi(strtok(NULL, ","));
100     arr_emp[i].date_of_birth.year = atoi(strtok(NULL, ","));
101
102       /* Sets the data from infile1 to the proper data fields.          */
103     strcpy(arr_emp[i].last_name, strtok(eString, ","));
104     arr_emp[i].ssn = atoi(strtok(NULL, ","));
105     strcpy(arr_emp[i].street_address, strtok(NULL, ","));
106     strcpy(arr_emp[i].city, strtok(NULL, ","));
107     arr_emp[i].age = atoi(strtok(NULL, ","));
108
109     i++; /* Increments i. */
110   }
111 }
```

4.2.3.4 void freeMem (struct PERSON * *arr_emp*)

This function will free the memory that had been previously allocated.

Parameters

arr_emp	This is a pointer to the array that will be freed.

Definition at line 199 of file work.c.

Referenced by main().

```
200 {
201     /* Frees arr_emp, then sets it to null for safty. */
202   free(arr_emp);
203   arr_emp = NULL;
204 }
```

4.2.3.5 void output (struct PERSON * *arr_emp*, FILE * *outFile*)

This function will output an array of structs to the output file.

Parameters

arr_emp	This is a pointer to the array that will be outputted.
outFile	This is a pointer to the file that wil be output to.

Definition at line 171 of file work.c.

References PERSON::age, checkNotNull(), PERSON::city, PERSON::date_of_birth, DATE::day, PERSON::last_name, DATE::month, PERSON::ssn, PERSON::street_address, and DATE::year.

Referenced by main().

```
172 {
173   int i = 0; /* creates an int for iterating through the array. */
174
175     /* Loops while arr_emp[i] is not null. */
176   while(checkNotNull(arr_emp[i]))
177   {
178       /* Prints all of the necissary information to the outFile. */
179     fprintf(outFile, "%-15s%-9d%-28s", arr_emp[i].last_name, arr_emp[i].
    ssn,
180       arr_emp[i].street_address);
181     fprintf(outFile, "%-15s%-7d\n", arr_emp[i].city,arr_emp[i].age);
182     fprintf(outFile, "Date of Birth: %d/%d/%d\n\n",
183       arr_emp[i].date_of_birth.month, arr_emp[i].date_of_birth.
    day,
184       arr_emp[i].date_of_birth.year);
185     i++; /* Increments i. */
186   }
187
188 }
```

4.2.3.6 void sort (struct PERSON * *arr_emp*, int *arr_size*)

This function implements an inseertion sort for the array of structs based on alphabetical order of the datafield last_name.

Parameters

arr_emp	This is a pointer to the array of structs that will be sorted.
arr_size	This is an int that contains the number of elements in the array.

Definition at line 125 of file work.c.

References checkNotNull(), and PERSON::last_name.

Referenced by main().

```
126 {
127   int i = 0;          /* This int is used to iterate through the array.     */
128   struct PERSON temp; /* This PERSON is used to temporarily hold a PERSON.  */
129   int location = 0;   /* This int is used to track a location in the array. */
130
131     /* This will loop while i is less than arr_size. */
132   for(i = 1; i < arr_size; i++)
133   {
134     temp = arr_emp[i]; /* Sets temp to the data of the current element.    */
135
136       /* This checks that temp is not null.                               */
137     if(!checkNotNull(temp))
138       continue;
139
140     location = i-1;      /* Sets location to i-1.                          */
141
142       /* This will loop while location is greater than or equal to 0, and
143         that the value at the array element at location is greater than the
144         value of temp. */
145     while(location >= 0 && strcmp(arr_emp[location].last_name, temp.last_name)
146           > 0)
147     {
148         /* This will set the value of the element at location plus 1 to
149           the value of the element at location. */
```

```
150       arr_emp[location + 1] = arr_emp[location];
151       location--; /* This decrements the location. */
152     }
153       /* This will set the value of the element at location plus 1 to the
154         value of temp. */
155     arr_emp[location + 1] = temp;
156   }
157
158 }
```

4.2.3.7 void structInit (struct PERSON * *arr_emp*, int *num*)

This function will initialize all data within the array of structs to either ' ' or 0.

Parameters

arr_emp	This is a pointer to the array that will be initialized.
num	This is an int containing the number of elements in the array.

Definition at line 218 of file work.c.

References PERSON::age, PERSON::city, PERSON::date_of_birth, DATE::day, PERSON::last_name, DATE::month, PERSON::ssn, PERSON::street_address, and DATE::year.

Referenced by main().

```
219 {
220   int i = 0; /* Initiaalizes i to 0.   */
221
222     /* Loops while i is less than num. */
223   for(i  = 0; i < num; i++)
224   {
225       /* Initializes all data in the struct. */
226     arr_emp[i].last_name[0] = ' ';
227     arr_emp[i].date_of_birth.month = 0;
228     arr_emp[i].date_of_birth.day = 0;
229     arr_emp[i].date_of_birth.year = 0;
230     arr_emp[i].ssn = 0;
231     arr_emp[i].street_address[0] = ' ';
232     arr_emp[i].city[0] = ' ';
233     arr_emp[i].age = 0;
234   }
235 }
```

4.3 work.h File Reference

This file contains the declaration of several functions for the 4th assignment.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
```

Data Structures

- struct DATE

  This Structure contains three data fields, month, day, year.
- struct PERSON

  This Structure contains siz data fields, last_name, date_of_birth, ssn, street_address, city, and age.

Functions

- FILE * fileOpenAndCheck (char *filename, char *openMode)

This function will open a file, check if it is opened, and will return a pointer to the file.

- void fillArr (FILE *infile1, FILE *infile2, struct PERSON *arr_emp)

  This function will take in the two file pointers, then will fill the array of structs with their data.
- void sort (struct PERSON *arr_emp, int arr_size)

  This function implements an inseertion sort for the array of structs based on alphabetical order of the datafield last_name.
- void output (struct PERSON *arr_emp, FILE *outFile)

  This function will output an array of structs to the output file.
- void freeMem (struct PERSON *arr_emp)

  This function will free the memory that had been previously allocated.
- void structInit (struct PERSON *arr_emp, int num)

  This function will initialize all data within the array of structs to either ' ' or 0.
- int checkNotNull (struct PERSON emp)

  This function will check all data points in a PERSON to see if it is empty or not.

4.3.1 Detailed Description

This file contains the declaration of several functions for the 4th assignment.

Author

Paul Huffman

email: huffmanp4@nku.edu

Course: CSC362

Section: 002

Assignment: 4

Date

11/30/2021 +fileOpenAndCheck +fillArr +sort +output +freeMem +structInit +checkNotNull

Hours spent on this assignment: 6

Specific portions that gave you the most trouble: fillArr

Definition in file work.h.

4.3.2 Function Documentation

4.3.2.1 int checkNotNull (struct PERSON *emp*)

This function will check all data points in a PERSON to see if it is empty or not.

Parameters

emp	This is a person that will be checked to see if it is empty.

Returns

This function will return 1 if the PERSON is not empty, and 0 otherwise.

Definition at line 249 of file work.c.

References PERSON::age, PERSON::city, PERSON::date_of_birth, DATE::day, PERSON::last_name, DATE::month, PERSON::ssn, PERSON::street_address, and DATE::year.

Referenced by output(), and sort().

```
250 {
251     /* Checks if all of the data in the struct is not set to initial values. */
252   if(emp.last_name[0] != ' ')
253   {
254     if(emp.date_of_birth.month != 0)
255     {
256       if(emp.date_of_birth.day != 0)
257       {
258         if(emp.date_of_birth.year != 0)
259         {
260           if(emp.ssn != 0)
261           {
262             if(emp.street_address[0] != ' ')
263             {
264               if(emp.city[0] != ' ')
265               {
266                 if(emp.age != 0)
267                 {
268                   return 1;
269                 }
270               }
271             }
272           }
273         }
274       }
275     }
276   }
277   return 0;
278 }
```

4.3.2.2 FILE* fileOpenAndCheck (char * *filename*, char * *openMode*)

This function will open a file, check if it is opened, and will return a pointer to the file.

Parameters

filename	This is a pointer to the string that contains name of the file that will be opened.
openMode	This is a pointer to the string that contains the mode in which the file will be opened.

Returns

This is a pointer to the file that has been opened.

Definition at line 49 of file work.c.

Referenced by main().

```
50 {
51   FILE *temp; /* This is a temporary pointer to a file. */
52
53     /* This opens a file and points it to temp. */
54   temp = fopen(filename, openMode);
```

```
55
56     /* This checks to make sure that temp is not NULL, and if it is, then **
57     ** it will print an error.                                             */
58   if(temp == NULL)
59   {
60     perror("Cant open file");
61   }
62
63   return temp;
64 }
```

4.3.2.3 void fillArr (FILE * *infile1*, FILE * *infile2*, struct PERSON * *arr_emp*)

This function will take in the two file pointers, then will fill the array of structs with their data.

Parameters

infile1	This is a pointer to the file that contains the name, address, ssn, and age of the people.
infile2	This is a pointer to the file that contains the date of birth of the people.
arr_emp	This is a pointer to an array of structs of type PERSON.

Definition at line 83 of file work.c.

References PERSON::age, PERSON::city, PERSON::date_of_birth, DATE::day, PERSON::last_name, MAX_LINE_LENGTH, DATE::month, PERSON::ssn, PERSON::street_address, and DATE::year.

Referenced by main().

```
84 {
85     /* This creates 2 strings used to hold data from the inFiles. */
86   char dString[MAX_LINE_LENGTH] = {0};
87   char eString[MAX_LINE_LENGTH] = {0};
88   int i = 0; /* A counting integer. */
89
90     /* Loops while not at the end of the file. */
91   while(!feof(infile2))
92   {
93       /* Copies a string from the files to the strings created earlier. */
94     fgets(eString, MAX_LINE_LENGTH, infile1);
95     fgets(dString, MAX_LINE_LENGTH, infile2);
96
97       /* Sets the data from infile2 to the proper data fields.          */
98     arr_emp[i].date_of_birth.month = atoi(strtok(dString, ","));
99     arr_emp[i].date_of_birth.day = atoi(strtok(NULL, ","));
100     arr_emp[i].date_of_birth.year = atoi(strtok(NULL, ","));
101
102       /* Sets the data from infile1 to the proper data fields.          */
103     strcpy(arr_emp[i].last_name, strtok(eString, ","));
104     arr_emp[i].ssn = atoi(strtok(NULL, ","));
105     strcpy(arr_emp[i].street_address, strtok(NULL, ","));
106     strcpy(arr_emp[i].city, strtok(NULL, ","));
107     arr_emp[i].age = atoi(strtok(NULL, ","));
108
109     i++; /* Increments i. */
110   }
111 }
```

4.3.2.4 void freeMem (struct PERSON * *arr_emp*)

This function will free the memory that had been previously allocated.

Parameters

arr_emp	This is a pointer to the array that will be freed.

Definition at line 199 of file work.c.

Referenced by main().

```
200 {
```

```
201     /* Frees arr_emp, then sets it to null for safty. */
202   free(arr_emp);
203   arr_emp = NULL;
204 }
```

4.3.2.5 void output (struct PERSON * *arr_emp*, FILE * *outFile*)

This function will output an array of structs to the output file.

Parameters

arr_emp	This is a pointer to the array that will be outputted.
outFile	This is a pointer to the file that wil be output to.

Definition at line 171 of file work.c.

References PERSON::age, checkNotNull(), PERSON::city, PERSON::date_of_birth, DATE::day, PERSON::last_name, DATE::month, PERSON::ssn, PERSON::street_address, and DATE::year.

Referenced by main().

```
172 {
173   int i = 0; /* creates an int for iterating through the array. */
174 
175     /* Loops while arr_emp[i] is not null. */
176   while(checkNotNull(arr_emp[i]))
177   {
178       /* Prints all of the necissary information to the outFile. */
179     fprintf(outFile, "%-15s%-9d%-28s", arr_emp[i].last_name, arr_emp[i].
    ssn,
180       arr_emp[i].street_address);
181     fprintf(outFile, "%-15s%-7d\n", arr_emp[i].city,arr_emp[i].age);
182     fprintf(outFile, "Date of Birth: %d/%d/%d\n\n",
183       arr_emp[i].date_of_birth.month, arr_emp[i].date_of_birth.
    day,
184       arr_emp[i].date_of_birth.year);
185     i++; /* Increments i. */
186   }
187 
188 }
```

4.3.2.6 void sort (struct PERSON * *arr_emp*, int *arr_size*)

This function implements an inseertion sort for the array of structs based on alphabetical order of the datafield last_-name.

Parameters

arr_emp	This is a pointer to the array of structs that will be sorted.
arr_size	This is an int that contains the number of elements in the array.

Definition at line 125 of file work.c.

References checkNotNull(), and PERSON::last_name.

Referenced by main().

```
126 {
127   int i = 0;          /* This int is used to iterate through the array.     */
128   struct PERSON temp; /* This PERSON is used to temporarily hold a PERSON.  */
129   int location = 0;   /* This int is used to track a location in the array. */
130 
131     /* This will loop while i is less than arr_size. */
132   for(i = 1; i < arr_size; i++)
133   {
134     temp = arr_emp[i]; /* Sets temp to the data of the current element.    */
135 
136       /* This checks that temp is not null.                               */
137     if(!checkNotNull(temp))
```

```
138       continue;
139
140     location = i-1;      /* Sets location to i-1.                          */
141
142       /* This will loop while location is greater than or equal to 0, and
143         that the value at the array element at location is greater than the
144         value of temp. */
145     while(location >= 0 && strcmp(arr_emp[location].last_name, temp.last_name)
146           > 0)
147     {
148         /* This will set the value of the element at location plus 1 to
149           the value of the element at location. */
150       arr_emp[location + 1] = arr_emp[location];
151       location--; /* This decrements the location. */
152     }
153       /* This will set the value of the element at location plus 1 to the
154         value of temp. */
155     arr_emp[location + 1] = temp;
156   }
157
158 }
```

4.3.2.7 void structInit (struct PERSON * *arr_emp*, int *num*)

This function will initialize all data within the array of structs to either ' ' or 0.

Parameters

arr_emp	This is a pointer to the array that will be initialized.
num	This is an int containing the number of elements in the array.

Definition at line 218 of file work.c.

References PERSON::age, PERSON::city, PERSON::date_of_birth, DATE::day, PERSON::last_name, DATE::month, PERSON::ssn, PERSON::street_address, and DATE::year.

Referenced by main().

```
219 {
220   int i = 0; /* Initiaalizes i to 0.   */
221
222     /* Loops while i is less than num. */
223   for(i  = 0; i < num; i++)
224   {
225       /* Initializes all data in the struct. */
226     arr_emp[i].last_name[0] = ' ';
227     arr_emp[i].date_of_birth.month = 0;
228     arr_emp[i].date_of_birth.day = 0;
229     arr_emp[i].date_of_birth.year = 0;
230     arr_emp[i].ssn = 0;
231     arr_emp[i].street_address[0] = ' ';
232     arr_emp[i].city[0] = ' ';
233     arr_emp[i].age = 0;
234   }
235 }
```

Index