***Lab 2: File access***

Lab 2-1:

Write information of the employees to “Salary.txt” file randomly.

* Source Code: lab2-1/randomWriteEmployee.c

*#include* <stdio.h>

*#include* <stdlib.h>

*#include* <string.h>

*#include* <time.h>

*// struct to save employee salary data.*

struct EmployeeSalary {

*// name*

    char name[20];

*// department in NULL-terminated string*

    char department[11];

*// salary*

    int salary[3];

};

*// name array*

char names[10][20] = {

    "CHOI",

    "HWANG",

    "KIM",

    "KANG",

    "LEE",

    "PARK",

    "WANG",

    "OK",

    "AHN",

    "CHRIS"

};

*// department array*

char departments[3][11] = {

    "Technical",

    "Management",

    "Production"

};

*/\*\**

*\* construct new EmployeeStruct Struct with random data.*

*\**

*\* @return constructed EmployeeSalary struct.*

*\*\*/*

struct EmployeeSalary employeeSalaryConstructor() {

*// create struct*

    struct EmployeeSalary emp = { "", "", { 1 + (rand() % 20), 1 + (rand() % 20), 1 + (rand() % 20)} };

*// get random name from name array*

    strcpy(emp.name, names[(int)rand() % 10]);

*// get random department from departments array*

    strcpy(emp.department, departments[(int)rand() % 3]);

*// return struct*

*return* emp;

}

*// main function*

int main() {

*// set random seed as current time epoch*

    srand(time(NULL));

*// variable for howMany.*

    int howMany = 0;

*// ask user howMany*

    printf("How many data should I generate? ");

    scanf("%d", &howMany);

*// create file pointer - write*

    FILE \*salaryRandom = fopen("Salary.txt","w");

*// File I/O Exception Handling*

*if* (!salaryRandom) {

*// user-friendly errormsh*

        printf("[ERROR] Unable to write the file.\nPlease check write permission or Disk's S.M.A.R.T.\n");

*// errorcode 1*

*return* 1;

    }

*// print howmany to file*

    fprintf(salaryRandom, "%d\n", howMany);

*// for every employee*

*for* (int i = 0; i < howMany; i++) {

*// create new struct using random constructor defined above*

        struct EmployeeSalary empl = employeeSalaryConstructor();

*// and write to the file*

        fprintf(salaryRandom, "%s %s %d %d %d\n", empl.name, empl.department, empl.salary[0], empl.salary[1], empl.salary[2]);

    }

*// close the file pointer.*

    fclose(salaryRandom);

}

The following c code utilizes struct EmployeeSalary, and using predefined name and departments array, it can randomly generate employee, due to limitations of OOP capabilities available in C language, I created pseudo-constructor function called employeeSalaryConstructor.

This program will get how many employees should be generated and generate EmployeeSalary struct data by calling employeeSalaryConstructor and write to the “Salary.txt”.

모니터, 스크린샷, 검은색, 화면이(가) 표시된 사진

자동 생성된 설명

Read information of the employees from “Salary.txt” file.

* Source Code: lab2-1/readEmployee.c

The following c code utilizes struct EmployeeSalary, and reads the employee salary data from Salary.txt and outputs to the terminal.

모니터, 컴퓨터, 스크린샷, 화면이(가) 표시된 사진

자동 생성된 설명

Write the output file “Salary\_technical.txt” whose dept is Technical.

* Source Code: lab2-1/readSalaryTechnical.c

*#include* <stdio.h>

*#include* <string.h>

*#include* <stdlib.h>

*// struct to save employee salary data.*

struct EmployeeSalary {

*// name*

    char name[20];

*// department in NULL-terminated string*

    char department[11];

*// salary*

    int salary[3];

};

*// main function*

int main() {

*// definition of FILE pointer - read.*

    FILE \*empFile = fopen("Salary.txt", "r");

*// exception handling: I/O Error*

*if* (!empFile) {

*// print user-friendly error message.*

        printf("[ERROR] Unable to read the file.\nPlease check read permission or Disk's S.M.A.R.T.\n");

*// finish program with errorcode 1*

*return* 1;

    }

*// variable to save how many employees should be saved into empFile*

    int howMany = 0;

*// read from the file*

    fscanf(empFile, "%d",&howMany);

*// definition of FILE pointer - to save technical people's salary only*

    FILE \*techFile = fopen("Salary\_technical.txt", "w");

*// exception handling again. : I/O error*

*if* (!techFile) {

*// print user-friendly error message*

        printf("[ERROR] Unable to write the file.\nPlease check read permission or Disk's S.M.A.R.T.\n");

*// finish program with errorcode 1*

*return* 1;

    }

*// dynamically allocate heap area size of pre-defined header.*

    struct EmployeeSalary \*empList = malloc(sizeof(struct EmployeeSalary)\*howMany);

*// exception handling for memory I/O*

*if* (!empList) {

*// user-friednly errormsg*

        printf("[ERROR] Unable to allocate memory.\nPlease check your computer is currently out of memory.\n");

*// errorcode 1*

*return* 1;

    }

*// variable to store how many technical people in the file?*

    int count = 0;

*// loop until end of file or previously defined howmany people, in case file was damaged.*

*for* (int i = 0; i < howMany && !feof(empFile); i++) {

*// define EmployeeSalary struct*

        struct EmployeeSalary emp = {"", "", {0,}};

*// save it to new EmployeeSalary.*

        fscanf(empFile, "%s %s %d %d %d", emp.name, emp.department, &emp.salary[0], &emp.salary[1], &emp.salary[2]);

*// check if emp.department is "Technical"*

*if* (!strcmp("Technical", emp.department)) {

*// then add to dynmaically allocated list and add count.*

            empList[count] = emp;

            count++;

        }

    }

*// write to the file of the how many technical people is there*

    fprintf(techFile, "%d\n", count);

*// copy the pointer.*

    struct EmployeeSalary \*empPtr = empList;

*// for all employeeList.*

*for* (int i = 0; i < count; i++, empPtr++) {

*// get sum of the salaries*

        int sum = 0;

*for* (int j = 0; j < 3; j++) {

*// add all salaries.*

            sum += empPtr->salary[j];

        }

*// write to file.*

        fprintf(techFile, "%s %d %.2f\n", empPtr->name, sum, sum/(float)3);

    }

*// free the dynmaically allocated empList*

    free(empList);

*// close File pointer.*

    fclose(empFile);

    fclose(techFile);

}

The following c code utilizes struct EmployeeSalary, and reads the employee data, check whether if he/she is at Technical department, calculate sum of salary, get average. And prints to file.

모니터, 스크린샷이(가) 표시된 사진

자동 생성된 설명

Lab 2-2:

* Source Code: lab2-2/lineDeleter.c

*#include* <stdio.h>

*#include* <string.h>

*/\*\**

*\* fill the buffer to 0 with specified size.*

*\* @param buffer the buffer you want to feel with NULL(0)*

*\* @param bufferSize the size of the buffer*

*\*\*/*

void clearBuffer(char \*buffer, int bufferSize) {

*// for all buffer indexes*

*for* (int i = 0; i < bufferSize; i++) {

*// save as 0 (NULL)*

        buffer[i] = 0;

    }

*return*;

}

*/\*\**

*\* remove provided line-number from the*

*\* input file pointer and output it to output pointer*

*\* @param input input file pointer to remove the line (read mode)*

*\* @param output output file pointer to get file without specified line (write mode)*

*\* @param num line number you want to remove*

*\*\*/*

void lineDeleter(FILE \*input, FILE \*output, int num) {

*// define buffer*

    char buffer[1001] = {0,};

*for* (int i = 1; !feof(input); i++) {

*// clear the buffer.*

        clearBuffer(buffer, 1001);

*// debug purpose.*

*//printf("%d... ", i);*

*// get the string length of 1001 including NULL from input.*

        fgets(buffer, 1001, input);

*// debug purpose*

*//printf("%s, ", buffer);*

*// if this line is user defined one, skip.*

*if* (num == i) { *continue*; }

*// else output it to output file pointer.*

        fputs(buffer, output);

*// debug purpose*

*//printf("writing!\n");*

    }

}

*/\*\**

*\* Checks how many lines are available in the file*

*\* @param input the file pointer to check the line counts*

*\* @return how many lines are available in the file.*

*\*\*/*

int howManyLines(FILE \*input) {

*// loop purpose.*

    int count = 1;

*// define buffer*

    char buffer[1001] = {0,};

*// until eof*

*for* (count = 0; !feof(input); count++) {

*// read file to buffer.*

        fgets(buffer, 1001, input);

    }

*// seek to begin of the file in order to make it easier for later use.*

    fseek(input, 0, SEEK\_SET);

*// return how many lines are available*

*return* count;

}

*// main function*

int main() {

*// open the input.txt file as read mode*

    FILE\* in = fopen("input.txt", "r");

*// exception handling: File I/O*

*if* (!in) {

*// user-friendly errormsg*

        printf("[ERROR] Unable to read the file.\nPlease check read permission or Disk's S.M.A.R.T.\n");

*// errorcode 1*

*return* 1;

    }

*// open the output.txt file as write mode*

    FILE\* out = fopen("output.txt", "w");

*// exception handling: File I/O*

*if* (!out) {

*// user-friendly errormsg*

        printf("[ERROR] Unable to write the file.\nPlease check read permission or Disk's S.M.A.R.T.\n");

*// errorcode 1*

*return* 1;

    }

*// variable to save user requested line to remove*

    int line = 0;

*// variable to save how many lines are in the file.*

    int totalLine = howManyLines(in);

*// infinite loop until break.*

*while*(1) {

*// ask user to line number to delete*

        printf("Enter line number to delete: ");

        scanf("%d", &line);

*// if line number is out of bounds.*

*if* (line <= 0 || line > totalLine) {

*// say its invalid and loop.*

            printf("Invalid line number! Enter again!\n");

        } *else* {

*// else break.*

*break*;

        }

    }

*// call line Deleter*

    lineDeleter(in, out, line);

*// close file pointers.*

    fclose(in);

    fclose(out);

*return* 0;

}

Read input, write to the output but skip the provided line.

스크린샷, 모니터, 검은색, 앉아있는이(가) 표시된 사진

자동 생성된 설명