מטלה 4

מטע קלוסקי

ארבל צדוק

**Git link :**

https://github.com/matatheking/hm4.git

**Code**:

#define \_CRT\_SECURE\_NO\_WARNINGS

/\*#define \_CRTDBG\_MAP\_ALLOC

#include <stdlib.h>

#include <crtdbg.h>\*/ //uncomment this block to check for heap memory allocation leaks.

// Read https://docs.microsoft.com/en-us/visualstudio/debugger/finding-memory-leaks-using-the-crt-library?view=vs-2019

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

typedef struct StudentCourseGrade

{

char courseName[35];

int grade;

}StudentCourseGrade;

typedef struct Student

{

char name[35];

StudentCourseGrade\* grades; //dynamic array of courses

int numberOfCourses;

}Student;

//Part A

void countStudentsAndCourses(const char\* fileName, int\*\* coursesPerStudent, int\* numberOfStudents);

int countPipes(const char\* lineBuffer, int maxCount);

char\*\*\* makeStudentArrayFromFile(const char\* fileName, int\*\* coursesPerStudent, int\* numberOfStudents);

void printStudentArray(const char\* const\* const\* students, const int\* coursesPerStudent, int numberOfStudents);

void factorGivenCourse(char\*\* const\* students, const int\* coursesPerStudent, int numberOfStudents, const char\* courseName, int factor);

void studentsToFile(char\*\*\* students, int\* coursesPerStudent, int numberOfStudents);

//Part B

Student\* transformStudentArray(char\*\*\* students, const int\* coursesPerStudent, int numberOfStudents);

void writeToBinFile(const char\* fileName, Student\* students, int numberOfStudents);

Student\* readFromBinFile(const char\* fileName);

int main()

{

//Part A

int\* coursesPerStudent = NULL;

int numberOfStudents = 0;

//studentsToFile(students, coursesPerStudent, numberOfStudents); //this frees all memory. Part B fails if this line runs. uncomment for testing (and comment out Part B)

//Part B

//Student\* transformedStudents = transformStudentArray(students, coursesPerStudent, numberOfStudents);

//writeToBinFile("students.bin", transformedStudents, numberOfStudents);

// Student\* testReadStudents = readFromBinFile("students.bin");

//add code to free all arrays of struct Student

/\*\_CrtDumpMemoryLeaks();\*/ //uncomment this block to check for heap memory allocation leaks.

// Read https://docs.microsoft.com/en-us/visualstudio/debugger/finding-memory-leaks-using-the-crt-library?view=vs-2019

char pipes[] = { "a||b||||" };//chack if countpipes works

int maxCount = 8;

char c = 't';

printf("the amount of '|' is %d", countPipes(&pipes, maxCount, c));

char name[] = "studentList.txt";

countStudentsAndCourses("studentList.txt", &coursesPerStudent, &numberOfStudents);

printf("%d", numberOfStudents);

printf("\n");

for (int i = 0; i < 9; i++)

{

printf("%d %p\n",coursesPerStudent[i], &coursesPerStudent[i]);

}

char\*\*\* students = makeStudentArrayFromFile("studentList.txt", &coursesPerStudent, &numberOfStudents);

// factorGivenCourse(students, coursesPerStudent, numberOfStudents, "Advanced Topics in C", +5);

studentsToFile(students, coursesPerStudent, numberOfStudents);

//Student\* stu = transformStudentArray(students, coursesPerStudent, numberOfStudents);

//writeToBinFile("students.bin", stu, numberOfStudents);

//readFromBinFile("students.bin");

return 0;

}

void countStudentsAndCourses(const char\* fileName, int\*\* coursesPerStudent, int\* numberOfStudents){

char line[1023];//stirng array which we will use later to count how many pipes it has.

int counter = 0;

FILE \*file = fopen(fileName, "r");//file pointer

if (!file) {

puts("cannot open file\n");

exit(1);

}

while (fgets(line,1023,file)!= NULL)

counter++;

\*numberOfStudents = counter;

rewind(file);

int\* array = (int\*)malloc(counter \* sizeof(int));//array that will contain the courses

if (!array) {

puts("alocation failed\n");

exit(1);

}

counter = 0;

while (fgets(line, 1023, file) != NULL) {//go over each line of the text file,count how many pipes it has, and promote the courses counter

\*(array+counter) = countPipes(line, 1023);

counter++;

}

\*coursesPerStudent = array;

fclose(file);

}

int countPipes(const char\* lineBuffer, int maxCount)

{

int i = 0;//loop counter

int sum\_of\_pipe = 0;//pipes counter

if (lineBuffer == NULL)

return -1;

if (maxCount <= 0)

return 0;

while (lineBuffer[0] != '\0' && i != maxCount)//go over the string and count pipes

{

if (lineBuffer[0] == '|')

sum\_of\_pipe++;

i++;

lineBuffer++;

}

return sum\_of\_pipe;

}

char\*\*\* makeStudentArrayFromFile(const char\* fileName, int\*\* coursesPerStudent, int\* numberOfStudents)

{

char pipe\_psik[3] = "|,";//array that will help us decide to whrere we wanna put the string, the order of the text is : name | course , grade | and ext.

char\* str;

char line[1023];

FILE\* file = fopen(fileName, "r");

if (!file) {

puts("cannot open file\n");

exit(1);

}

int size = \*numberOfStudents;

char\*\*\* triple\_array = (char\*\*\*)malloc(size \* sizeof(char\*\*));//triple pointer array

if (!triple\_array) {

puts("alocation failed\n");

exit(1);

}

for (int i = 0; i < size; i++) {// allocate memory acording to the number of courses.

int numofcourse = \*(\*(coursesPerStudent)+i);

\*(triple\_array+i) = (char\*\*)malloc((numofcourse \* 2 + 1)\*sizeof(char\*));

if (!\*(triple\_array + i)) {

puts("alocation failed\n");

exit(1);

}

}

int i = 0,length;

int j = 0;

while (fgets(line, 1023, file) != NULL)

{

j = 0;

str = strtok(line, pipe\_psik);

while (str != NULL) {

length = strlen(str);

triple\_array[i][j] = (char\*)malloc((length + 1));

if (!triple\_array[i][j]) {

puts("alocation failed");

exit(1);

}

strcpy(triple\_array[i][j],str);

str = strtok(NULL, pipe\_psik);

j ++;

}

i++;

}

fclose(file);

return(triple\_array);

}

void factorGivenCourse(char\*\* const\* students, const int\* coursesPerStudent, int numberOfStudents, const char\* courseName, int factor)

{

printf("\n");

int i = 0, grade;

for (int i = 0; i < numberOfStudents; i++)//loop that goes over the student list

{

for (int j= 0; j < \*(coursesPerStudent + i) \* 2 + 1; j ++)//go over the courses of the current student, if it is matching to the factor course, upgrade its grade

{

char check[1023];

strcpy(check, students[i][j]);

if (strcmp(check,courseName)==0)

{

grade = atoi(students[i][j+1]);

grade += factor;

if (grade > 100) grade = 100;

if (grade < 0)grade = 0;

\_itoa(grade, (students[i][j+1]), 10);

}

}

}

printStudentArray(students, coursesPerStudent, numberOfStudents);

}

void printStudentArray(const char\* const\* const\* students, const int\* coursesPerStudent, int numberOfStudents)

{

for (int i = 0; i < numberOfStudents; i++)

{

printf("name: %s\n\*\*\*\*\*\*\*\*\*\n", students[i][0]);

for (int j = 1; j <= 2 \* coursesPerStudent[i]; j += 2)

{

printf("course: %s\n", students[i][j]);

printf("grade: %s\n", students[i][j + 1]);

}

printf("\n");

}

}

void studentsToFile(char\*\*\* students, int\* coursesPerStudent, int numberOfStudents)

{

FILE\* file = fopen("studentList.txt", "w");//file pointer

for (int i = 0; i < numberOfStudents; i++)//goes over the student list

{

for (int j = 0; j < \*(coursesPerStudent+i)\*2+1; j++)//write to the text - if the index % 2 is 0 put | else put ,

{

fprintf(file, students[i][j]);

free(students[i][j]);

if (j%2 == 0)

fprintf(file, "|");

else

fprintf(file, ",");

}

fprintf(file, "\n");

free(students[i]);

}

free(coursesPerStudent);

free(students);

fclose(file);

}

void writeToBinFile(const char\* fileName, Student\* students, int numberOfStudents){

FILE\* binfile = fopen(fileName, "wb");//binary file pointer

if (!binfile) {

puts("cannot open file\n"); exit(1);

}

fwrite(&numberOfStudents, sizeof(int), 1, binfile);//write first the number of students

for (int i = 0; i < numberOfStudents; i++) {//goes over the student list

fwrite(&students[i].name, 35,1, binfile);//writee the name of the student

fwrite(&students[i].numberOfCourses, sizeof(int),1, binfile);//number of courses

for (int j = 0; j < students[i].numberOfCourses; j++) {

fwrite(&students[i].grades[j].courseName, 35,1, binfile);//puts its name

fwrite(&students[i].grades[j].grade, sizeof(int), 1, binfile);//puts its grade

}

}

fclose(binfile);

}

Student\* readFromBinFile(const char\* fileName)

{

FILE\* binfile = fopen(fileName, "rb");//binary file pointer

if (!binfile) {

puts("cannot open file\n"); exit(1);

}

int numofstudents;

fread(&numofstudents, sizeof(int), 1, binfile);//copy from the binary text the number of student

Student\* students = (Student\*)malloc(numofstudents \* sizeof(Student));//struct array - for each student

if (!students) {

puts("allocation failled\n"); exit(1);

}

for (int i = 0; i < numofstudents; i++) {

fread(&(students + i)->name, 35,1, binfile);//copy the bame of the student

fread(&(students + i)->numberOfCourses, sizeof(int), 1, binfile);//copy number of coursrs

students[i].grades = (StudentCourseGrade\*)malloc(students[i].numberOfCourses \* sizeof(StudentCourseGrade));

if (!students[i].grades) {

puts("allocation failled\n"); exit(1);

}

for (int j = 0; j < students[i].numberOfCourses; j++) {

fread(&((students + i)->grades + j)->courseName, 35,1, binfile);//copy course name

fread(&((students + i)->grades + j)->grade, sizeof(int), 1, binfile);//copy grade

}

}

fclose(binfile);

}

Student\* transformStudentArray(char\*\*\* students, const int\* coursesPerStudent, int numberOfStudents)

{

Student\* stu = (Student\*)malloc(numberOfStudents \* sizeof(Student));//struct array - for each student

for (int i = 0; i < numberOfStudents; i++)

{

strcpy((stu + i)->name, students[i][0]);//copy name

(stu + i)->numberOfCourses = coursesPerStudent[i];//copy the number of courses of the current student

(stu + i)->grades = (StudentCourseGrade\*)malloc(coursesPerStudent[i] \* sizeof(StudentCourseGrade));

for (int j = 1,k=0; j < (coursesPerStudent[i]\*2)+1; j++) {//i - students counter,k - index for whats inside grade, j - inner lines loop index

if (j % 2 == 1) {

strcpy(((stu + i)->grades +k ), students[i][j]);

}

else {

((stu + i)->grades+ k)->grade = atoi(students[i][j]);

k++;

}

}

}

return stu;

}

**Picture of the txt file after we wrote it with the function**

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי