در این برنامه سعی شده تا بیشتر مفاهیم آموزش داده شده در طول ترم توسط استاد محترم و همچنین یک سری مفاهیم دیگر که اینجانب در کتاب معرفی شده یاد گرفته ام، مرور شود. مواردی همچون: دستورات شرطی، دستورات کنترلی، حلقه­ها، آرایه ها، اشاره گرها، توابع، فایل، تعریف نوع جدبد و ...

در اینجا header مربوط به تابع malloc اصطلاحا include شده تا بدین ترتیب تابع malloc توسط برنامه قابل استفاده شود.

#include <memory>

در این قسمت یک سری مقادیر که هر کدام معنی خاصی دارد تعریف شده اند.

#define INCORRECTFILE 1 فایل غیر درست

#define BLANKFILE 2 فایل خالی

#define BAD 1 نمره بد

#define MIDDLE 2 نمره متوسط

#define GOOD 3 نمره خوب

#define EXCELENT 4 نمره عالی

#define NAMELENGTH 50 طول یک نام شخص

ساختار مربوط به اطلاعات یک دانشجو به صورت یک نوع جدید تعریف شده است.

typedef struct \_Student

{

char Name[NAMELENGTH];

char Family[NAMELENGTH];

float grade;

}Student;

در قسمت زیر اعلان توابعی که بعد از تابع \_tmain تعریف شده اند آورده شده تا این توابع توسط تابع \_tmain قابل شناسایی و استفاده باشند.

int EnterGradesOfStudents();

int ComputeAverageOfStudents();

float ComputeAverage(float\* grades, int gradesCount);

void PrintGrades(Student\* students, int studentCount);

void ReadGradesFromRecords(Student\* students, float\* grades, int gradesCount);

تابع \_tmain تابع اصلی می باشد که در آن یک حلقه تعریف شده که سه گزینه در اختیار کاربر قرار می دهد. 1 برای وارد کردن نمرات و ذخیره در یک فایل، 2 برای محاسبه معدل و در نهایت 0 برای خروج.

int \_tmain(int argc, \_TCHAR\* argv[])

{

while(true)

{

int todo = -1;

printf("\n\n\nif you want to enter grades please enter 1\n");

printf("else if you want to compute average please enter 2\n");

printf("else to exit please enter 0\n");

scanf("%d", &todo);

switch(todo)

{

case 1 : EnterGradesOfStudents();

break;

case 2 : ComputeAverageOfStudents();

break;

default :

break;

}

}

return 0;

}

در تابع زیر نمرات وارد شده و در یک فایل ذخیره میشود. در این فایل ابتدا تعداد دانشجویان ذخیره و سپس رکورد مربوط به تک تک دانشجویان وارد می­شود.

int EnterGradesOfStudents()

{

int studentCount = 0;

Student students = {};در اینجا ساختار مربوط به دانشجو مقداردهی اولیه میشود

printf("\nPlease Enter Number Of Student\n");

scanf("%d", &studentCount);

printf("\nPlease Enter Path of output file\n");

char filePath [256];آرایه کاراکتری برای مسیر فایل متشکل از حداکثر 256 کاراکتر

scanf("%s", filePath);

FILE\* file = fopen((const char\*)filePath, "a");

if(file)

{

در اینجا ابتدا تعداد دانشجویان ذخیره در فایل ذخیره میشود.

fwrite(&studentCount, sizeof(int), 1, file);

printf("\nPlease, First Enter Name then Family and finally enter grade of each Student:\n");

در این حلقه به ترتیب نام، نام خانوادگی و نمره فرد را گرفته و در آخرین دستور حلقه رکورد مربوط به دانشجو ذخیره میگردد.

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

{

printf("\nName:");

scanf("%s", students.Name);

printf("\tFamily:");

scanf("%s", students.Family);

printf("\tGrade:");

scanf("%f", &students.grade);

fwrite(&students, sizeof(Student), 1, file);

}

fclose(file);

}

else

{

اگر تعداد دانشجویان صفر باشد ادامه کار بی معناست، لذا پیغام زیر چاپ میشود.

printf("\noutput file path is incorrect or this file is using by another program\n");

return -1;

}

printf("\nTo Continue press any key");

getchar();

return 0;

}

تابع زیر معدل دانشجویان را محاسبه میکند.

int ComputeAverageOfStudents()

{

char filePath [256];

printf("please, enter address of input file as grades file\n");

scanf("%s", filePath);

FILE\* file = fopen((const char\*)filePath, "r");

if(file)

{

Student\* students = NULL;

float \*grades = NULL;

int studentCount = 0;

int gradesCount = 0;

ابتدا تعداد دانشجویان از فایل خوانده میشود.

fread(&studentCount, sizeof(int), 1, file);

gradesCount = studentCount;

if(studentCount)

{

در اینجا به تعداد دانشجویان برای دو آرایه داینامیک grades و students به تعداد دانشجویان فضا درخواست میشود.

grades = (float\*)malloc(sizeof(float) \* gradesCount);

students = (Student\*)malloc(sizeof(Student) \* studentCount);

کل رکوردهای مربوط به دانشجویان یکجا به تعداد آنها از فایل خوانده میشود.

fread(students, sizeof(Student), gradesCount, file);

در اینجا تابع PrintGrades که اطلاعات دانشجویان را چاپ میکند فراخوانی میشود.

PrintGrades(students, studentCount);

در اینجا تابع ReadGradesFromRecords که نمرات را از اطلاعات دانشجویان خوانده و در آرایه داینامیک grades که در حقیقت یک اشاره گر است قرار میدهد، فراخوانی میشود.

ReadGradesFromRecords(students,grades, gradesCount);

در اینجا تابع ComputeAverage که میانگین نمرات دانشجویان را محاسبه میکند، فراخوانی میشود.

float average = ComputeAverage(grades, gradesCount);

int level = -1;

در دستورات کنترلی و شرطی بعدی تنها سعی شده تا مروری بر آنها صورت گیرد. ابتدا یک سری دستور if تو در تو و سپس یک switch استفاده شده است.

if(average < 10)

{

level = BAD;

}

else if(average < 15)

{

level = MIDDLE;

}

else if(average < 17)

{

level = GOOD;

}

else

{

level = EXCELENT;

}

switch(level)

{

case BAD:

printf("\n average of whole of students is : %f\n then level of this class is Bad\n", average);

break;

case MIDDLE:

printf("\n average of whole of students is : %f\n then level of this class is Middle\n", average);

break;

case GOOD:

printf("\n average of whole of students is : %f\n then level of this class is Good\n", average);

break;

default:

printf("\n average of whole of students is : %f\n then level of this class is Excelent\n", average);

}

در اینجا فایلی که باز شده بود در انتهای کار بسته شده است.

fclose(file);

در دو دستور زیر فضای گرفته شده از Ram دوباره به آن برگردانده میشود و یا به اصطلاح آزادسازی میگردد.

free(students);

free(grades);

}

else

{

printf("\nnumber of grades is zero, please try a gaint for a true file");

fclose(file);

return BLANKFILE;

}

}

else

{

در اینجا به دلیل نا معتبر بودن فایل پیغام نادرست بودن فایل چاپ میشود.

printf("\ninput file is incorrect or is not exist\n");

getchar();

return INCORRECTFILE;

}

printf("\nTo Continue press any key");

getchar();

return 0;

}

این تابع نمرات را از اطلاعات دانشجویان خوانده و در آرایه داینامیک grades که در حقیقت یک اشاره گر است قرار میدهد.

void ReadGradesFromRecords(Student\* students, float\* grades, int gradesCount)

{

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

{

در اینجا از تعریف آرایه ای برای مراجعه به یک عضو معین از حافظه اختصاص داده شده به اشاره گرها استفاده شده است.

grades[i] = students[i].grade;

}

}

تابع PrintGrades که اطلاعات دانشجویان را چاپ میکند.

void PrintGrades(Student\* students, int studentCount)

{

printf("\nGrades of Strudent are:\n");

حلقه زیر بین اعضای آرایه داینامیک مربوط به رکورد دانشجویان یکی یکی حرکت کرده و اطلاعات مربوط به دانشجویان را چاپ میکند.

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

{

printf("\nName: %s", students->Name);

printf("\t\tFamily: %s", students->Family);

printf("\t\tGrade: %f", students->grade);

در اینجا اشاره گر به اصطلاح یک خانه به جلو حرکت میکند تا به عضو بعدی آرایه داینامیک یا همان اشاره گر، اشاره کند.

students++;

}

}

تابع ComputeAverage که میانگین نمرات دانشجویان را محاسبه میکند.

float ComputeAverage(float\* grades, int gradesCount)

{

float sum = 0;

حلقه زیر بین اعضای آرایه داینامیک مربوط به نمرات دانشجویان یکی یکی حرکت کرده و هر نمره را با حاصل جمع قبلی جمع می کند.

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

{

sum += \*grades;

grades++;

}

در نهایت معدل با تقسیم کردن حاصل جمع بر تعداد نمرات به محل فراخوانی برگردانده میشود.

return sum / gradesCount;

}