Projet école

CRUD COURS:

crud.h

#ifndef COURS\_H

#define COURS\_H

#define MAX\_COURS 100

// Structure pour représenter un cours

typedef struct {

int id;

char nom[50];

} Cours;

// Variables globales pour stocker les cours

extern Cours cours[MAX\_COURS];

extern int nbCours;

// Déclarations des fonctions

void ajouterCours(int id, const char\* nom);

void modifierCours(int id, const char\* nouveauNom);

void supprimerCours(int id);

void afficherCours();

#endif // COURS\_H

Crud.c

#ifndef COURS\_C

#define COURS\_C

#define MAX\_COURS 100

// Structure pour représenter un cours

typedef struct {

int id;

char nom[50];

} Cours;

// Variables globales pour stocker les cours

extern Cours cours[MAX\_COURS];

extern int nbCours;

// Déclarations des fonctions

void ajouterCours(int id, const char\* nom);

void modifierCours(int id, const char\* nouveauNom);

void supprimerCours(int id);

void afficherCours();

Code.c::

#endif // COURS\_C

#include <stdio.h>

#include <string.h>

#define MAX\_COURS 100

typedef struct {

int id;

char nom[50];

} Cours;

Cours cours[MAX\_COURS];

int nbCours = 0;

// Fonction pour ajouter un cours

void ajouterCours(int id, const char\* nom) {

if (nbCours >= MAX\_COURS) {

printf("Erreur : La liste des cours est pleine.\n");

return;

}

cours[nbCours].id = id;

strncpy(cours[nbCours].nom, nom, sizeof(cours[nbCours].nom) - 1);

nbCours++;

printf("Cours ajouté avec succès.\n");

}

// Fonction pour modifier un cours

void modifierCours(int id, const char\* nouveauNom) {

int i;

for ( i = 0; i < nbCours; i++) {

if (cours[i].id == id) {

strncpy(cours[i].nom, nouveauNom, sizeof(cours[i].nom) - 1);

printf("Cours modifié avec succès.\n");

Return;

}

}

printf("Erreur : Cours avec ID %d introuvable.\n", id);

}

// Fonction pour supprimer un cours

void supprimerCours(int id) {

int i;

int j;

for ( i = 0; i < nbCours; i++) {

if (cours[i].id == id) {

// Décaler les éléments pour combler l'espace

for ( j = i; j < nbCours - 1; j++) {

cours[j] = cours[j + 1];

}

nbCours--;

printf("Cours supprimé avec succès.\n");

return;

}

}

printf("Erreur : Cours avec ID %d introuvable.\n", id);

}

// Fonction pour afficher tous les cours

void afficherCours() {

if (nbCours == 0) {

printf("Aucun cours enregistré.\n");

return;

}

printf("\n--- Liste des cours ---\n");

int i;

for ( i = 0; i < nbCours; i++) {

printf("ID : %d, Nom : %s\n", cours[i].id, cours[i].nom);

}

}

int main() {

int choix, id;

char nom[50];

while (1) {

printf("\n--- Gestion des cours ---\n");

printf("1. Ajouter un cours\n");

printf("2. Modifier un cours\n");

printf("3. Supprimer un cours\n");

printf("4. Afficher tous les cours\n");

printf("5. Quitter\n");

printf("Entrez votre choix : ");

scanf("%d", &choix);

switch (choix) {

case 1:

printf("Entrez l'ID du cours : ");

scanf("%d", &id);

printf("Entrez le nom du cours : ");

scanf(" %[^\n]", nom);

ajouterCours(id, nom);

break;

case 2:

printf("Entrez l'ID du cours à modifier : ");

scanf("%d", &id);

printf("Entrez le nouveau nom du cours : ");

scanf(" %[^\n]", nom);

modifierCours(id, nom);

break;

case 3:

printf("Entrez l'ID du cours à supprimer : ");

scanf("%d", &id);

supprimerCours(id);

break;

case 4:

afficherCours();

break;

case 5:

printf("Au revoir !\n");

return 0;

default:

printf("Choix invalide. Veuillez réessayer.\n");

}

}

}

CRUD EVENEMENT/

\*Crud.h :

#ifndef CRUD\_H

#define CRUD\_H

#define MAX\_EVENTS 100

// Structure représentant un événement

typedef struct {

int id;

char titre[50];

char description[200];

} Event;

// Prototypes des fonctions CRUD

void ajouterEvent(); // Ajouter un événement

void afficherEvents(); // Afficher tous les événements

void modifierEvent(); // Modifier un événement

void supprimerEvent(); // Supprimer un événement

\*crud.c

#endif // CRUD\_C

#include <stdio.h>

#include <string.h>

#define MAX\_EVENTS 100

// Structure représentant un événement

typedef struct {

int id;

char titre[50];

char description[200];

} Event;

// Variables globales

Event events[MAX\_EVENTS];

int nbEvents = 0;

// Ajouter un événement

void ajouterEvent() {

if (nbEvents >= MAX\_EVENTS) {

printf("Erreur : Capacité maximale atteinte !\n");

return;

}

Event nouveau;

printf("Entrez l'ID de l'événement : ");

scanf("%d", &nouveau.id);

getchar(); // Supprime le saut de ligne

printf("Entrez le titre de l'événement : ");

fgets(nouveau.titre, sizeof(nouveau.titre), stdin);

nouveau.titre[strcspn(nouveau.titre, "\n")] = '\0'; // Enlève le saut de ligne

printf("Entrez une description pour l'événement : ");

fgets(nouveau.description, sizeof(nouveau.description), stdin);

nouveau.description[strcspn(nouveau.description, "\n")] = '\0';

events[nbEvents++] = nouveau;

printf("Événement ajouté avec succès !\n");

}

// Afficher tous les événements

void afficherEvents() {

if (nbEvents == 0) {

printf("Aucun événement à afficher.\n");

return;

}

printf("\n--- Liste des événements ---\n");

int i;

for ( i = 0; i < nbEvents; i++) {

printf("ID: %d, Titre: %s, Description: %s\n",

events[i].id, events[i].titre, events[i].description);

}

}

// Modifier un événement

void modifierEvent() {

int id, found = 0;

printf("Entrez l'ID de l'événement à modifier : ");

scanf("%d", &id);

int i;

for ( i = 0; i < nbEvents; i++) {

if (events[i].id == id) {

found = 1;

getchar(); // Supprime le saut de ligne

printf("Entrez le nouveau titre de l'événement : ");

fgets(events[i].titre, sizeof(events[i].titre), stdin);

events[i].titre[strcspn(events[i].titre, "\n")] = '\0';

printf("Entrez la nouvelle description de l'événement : ");

fgets(events[i].description, sizeof(events[i].description), stdin);

events[i].description[strcspn(events[i].description, "\n")] = '\0';

printf("Événement modifié avec succès !\n");

break;

}

}

if (!found) {

printf("Erreur : Événement avec ID %d introuvable.\n", id);

}

}

// Supprimer un événement

void supprimerEvent() {

int id, found = 0;

printf("Entrez l'ID de l'événement à supprimer : ");

scanf("%d", &id);

int i;

int j;

for ( i = 0; i < nbEvents; i++) {

if (events[i].id == id) {

found = 1;

for ( j = i; j < nbEvents - 1; j++) {

events[j] = events[j + 1];

}

nbEvents--;

printf("Événement supprimé avec succès !\n");

break;

}

}

if (!found) {

printf("Erreur : Événement avec ID %d introuvable.\n", id);

}

}

int main() {

int choix;

do {

printf("\n--- Menu Gestion des Événements ---\n");

printf("1. Ajouter un événement\n");

printf("2. Afficher tous les événements\n");

printf("3. Modifier un événement\n");

printf("4. Supprimer un événement\n");

printf("5. Quitter\n");

printf("Votre choix : ");

scanf("%d", &choix);

switch (choix) {

case 1:

ajouterEvent();

break;

case 2:

afficherEvents();

break;

case 3:

modifierEvent();

break;

case 4:

supprimerEvent();

break;

case 5:

printf("Au revoir !\n");

break;

default:

printf("Choix invalide, veuillez réessayer.\n");

}

} while (choix != 5);

return 0;

}

CEUD COURS

Crud.h :

#ifndef CRUD\_H

#define CRUD\_H

#define MAX\_COURS 100

// Structure pour représenter un cours

typedef struct {

int id;

char titre[50];

} Cours;

// Déclaration du tableau global des cours et de la variable de compteur

extern Cours cours[MAX\_COURS];

extern int nbCours;

// Fonctions CRUD pour gérer les cours

void ajouterCours(); // Ajouter un cours

void afficherCours(); // Afficher tous les cours

void modifierCours(); // Modifier un cours

void supprimerCours(); // Supprimer un cours

#endif // CRUD\_H

CRUD.C :

#include <stdio.h>

#include <string.h>

#define MAX\_COURS 100

// Structure pour représenter un cours

typedef struct {

int id;

char titre[50];

} Cours;

// Tableau global des cours

Cours cours[MAX\_COURS];

int nbCours = 0;

// Ajouter un cours

void ajouterCours() {

if (nbCours >= MAX\_COURS) {

printf("Erreur : Capacité maximale atteinte !\n");

return;

}

Cours nouveau;

printf("Entrez l'ID du cours : ");

scanf("%d", &nouveau.id);

getchar(); // Consomme le saut de ligne

printf("Entrez le titre du cours : ");

fgets(nouveau.titre, sizeof(nouveau.titre), stdin);

nouveau.titre[strcspn(nouveau.titre, "\n")] = '\0'; // Supprime le saut de ligne

cours[nbCours++] = nouveau;

printf("Cours ajouté avec succès !\n");

}

// Afficher tous les cours

void afficherCours() {

if (nbCours == 0) {

printf("Aucun cours à afficher.\n");

return;

}

printf("\n--- Liste des cours ---\n");

int i;

for ( i = 0; i < nbCours; i++) {

printf("ID: %d, Titre: %s\n", cours[i].id, cours[i].titre);

}

}

// Modifier un cours

void modifierCours() {

int id, found = 0;

printf("Entrez l'ID du cours à modifier : ");

scanf("%d", &id);

getchar(); // Consomme le saut de ligne

int i;

for ( i = 0; i < nbCours; i++) {

if (cours[i].id == id) {

found = 1;

printf("Entrez le nouveau titre du cours : ");

fgets(cours[i].titre, sizeof(cours[i].titre), stdin);

cours[i].titre[strcspn(cours[i].titre, "\n")] = '\0';

printf("Cours modifié avec succès !\n");

break;

}

}

if (!found) {

printf("Erreur : Cours avec ID %d introuvable.\n", id);

}

}

// Supprimer un cours

void supprimerCours() {

int id, found = 0;

printf("Entrez l'ID du cours à supprimer : ");

scanf("%d", &id);

int i;

for ( i = 0; i < nbCours; i++) {

if (cours[i].id == id) {

found = 1;

int i,j;

// Décaler les éléments après l'élément à supprimer

for (j = i; j < nbCours - 1; j++) {

cours[j] = cours[j + 1];

}

nbCours--;

printf("Cours supprimé avec succès !\n");

break;

}

}

if (!found) {

printf("Erreur : Cours avec ID %d introuvable.\n", id);

}

}

// Programme principal

int main() {

int choix;

do {

printf("\n--- Menu Gestion des Cours ---\n");

printf("1. Ajouter un cours\n");

printf("2. Afficher tous les cours\n");

printf("3. Modifier un cours\n");

printf("4. Supprimer un cours\n");

printf("5. Quitter\n");

printf("Votre choix : ");

scanf("%d", &choix);

switch (choix) {

case 1:

ajouterCours();

break;

case 2:

afficherCours();

break;

case 3:

modifierCours();

break;

case 4:

supprimerCours();

break;

case 5:

printf("Au revoir !\n");

break;

default:

printf("Choix invalide, veuillez réessayer.\n");

}

} while (choix != 5);

return 0;

}

\*Afficher Notes:

#include <stdio.h>

// Fonction pour afficher les notes de contrôle, d'examen et d'oral

void afficherNotes(char\* nomEleve, int controles[], int tailleControles, int examens[], int tailleExamens, int orales[], int tailleOrales) {

printf("Notes de l'élève %s :\n", nomEleve);

// Afficher les notes de contrôle

printf("Notes de contrôle :\n");

int i;

for ( i = 0; i < tailleControles; i++) {

printf("Contrôle %d : %d\n", i + 1, controles[i]);

}

// Afficher les notes d'examen

printf("Notes d'examen :\n");

for ( i = 0; i < tailleExamens; i++) {

printf("Examen %d : %d\n", i + 1, examens[i]);

}

// Afficher les notes d'oral

printf("Notes d'oral :\n");

for ( i = 0; i < tailleOrales; i++) {

printf("Oral %d : %d\n", i + 1, orales[i]);

}

}

int main() {

// Variables nécessaires

char nom[50]; // Pour stocker le nom de l'élève

int controles[50], examens[50], orales[50]; // Tableaux pour les notes

int tailleControles, tailleExamens, tailleOrales; // Tailles des tableaux

// Saisie du nom de l'élève

printf("Entrez le nom de l'élève : ");

scanf("%s", nom);

// Saisie des notes de contrôle

printf("Entrez le nombre de notes de contrôle : ");

scanf("%d", &tailleControles);

printf("Entrez les %d notes de contrôle :\n", tailleControles);

int i;

for ( i = 0; i < tailleControles; i++) {

printf("Contrôle %d : ", i + 1);

scanf("%d", &controles[i]);

}

// Saisie des notes d'examen

printf("Entrez le nombre de notes d'examen : ");

scanf("%d", &tailleExamens);

printf("Entrez les %d notes d'examen :\n", tailleExamens);

for ( i = 0; i < tailleExamens; i++) {

printf("Examen %d : ", i + 1);

scanf("%d", &examens[i]);

}

// Saisie des notes d'oral

printf("Entrez le nombre de notes d'oral : ");

scanf("%d", &tailleOrales);

printf("Entrez les %d notes d'oral :\n", tailleOrales);

for ( i = 0; i < tailleOrales; i++) {

printf("Oral %d : ", i + 1);

scanf("%d", &orales[i]);

}

// Appel de la fonction pour afficher les notes

afficherNotes(nom, controles, tailleControles, examens, tailleExamens, orales, tailleOrales);

return 0;

}

\*Afficher Absences:

#include <stdio.h>

#define MAX\_ELEVES 100

#define MAX\_ABSENCES 50

typedef struct {

int id;

char nom[50];

int absences[MAX\_ABSENCES];

int nbAbsences;

} Eleve;

Eleve eleves[MAX\_ELEVES];

int nbEleves = 0;

void afficherAbsences(int idEleve) {

int i;

int j;

for ( i = 0; i < nbEleves; i++) {

if (eleves[i].id == idEleve) {

printf("\n--- Absences pour l'élève %s (ID: %d) ---\n", eleves[i].nom, eleves[i].id);

if (eleves[i].nbAbsences == 0) {

printf("Aucune absence enregistrée.\n");

} else {

printf("Jours d'absence : ");

for (j = 0; j < eleves[i].nbAbsences; j++) {

printf("%d ", eleves[i].absences[j]);

}

printf("\n");

}

printf("Erreur : Élève avec ID %d introuvable.\n", idEleve);

}

}

return;

}