Université Hassan 1er
Faculté des Sciences et Techniques
Département Mathématiques et Informatique

Algorithmique et Programmation en langage C

Solution des Travaux Pratiques

Professeur BALOUKI Youssef
Département Mathématiques et Informatique

Université Hassan 1er
Faculté des Sciences et Techniques
Département Mathématiques et Informatique

Travaux Pratiques 1 - Solution

Exercice 1 :

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

int main()
{
   int N,M,P, i, j, k;
   int **A, **B, **C;
   printf("Entrer le nombre de lignes de la Matrice A:");
   scanf("%d",&N);
   printf("Entrer le nombre de colonne de la Matrice A (qui est en meme temps le nombre de ligne de B):");
   scanf("%d",&M);
   printf("Entrer le nombre de colonne de la Matrice B:");
   scanf("%d",&P);
   // allocation de la mémoire
   A = (int**)malloc(N*sizeof(int*));
   B = (int**)malloc(M*sizeof(int*));
   C = (int**)malloc(N*sizeof(int*));
   for(i=0;i<N;i++)
      A[i]=(int*)malloc(M*sizeof(int));
   for(i=0;i<M;i++)
      B[i]=(int*)malloc(P*sizeof(int));
   for(i=0;i<N;i++)
      C[i]=(int*)calloc(P,sizeof(int));
   //Lecture de A et B
   printf("Saisir les elements de A:\n");
   for(i=0;i<N;i++)
      for(j=0;j<M;j++)
      {
         printf("A[%d][%d]=",i,j);
         scanf("%d",*(A+i)+j);
      }
   printf("Saisir les elements de B:\n");
   for(i=0;i<M;i++)
      for(j=0;j<P;j++)
      {
         printf("B[%d][%d]=",i,j);
         scanf("%d",B[i]+j);
      }
   //calcul du produit
   for(i=0;i<N;i++)
      for(j=0;j<P;j++)
         for(k=0;k<M;k++)
            C[i][j]+=A[i][k]*B[k][j];
   //Affichage de la valeur de C
   printf("C:\n");
   for(i=0;i<N;i++)
   {
```

```
        for(j=0;j<P;j++)
        {
           printf("%d\t",C[i][j]);

        }
        printf("\n");
    }
    return 0;
}
```

Exercice 2 :

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

int main()
{
    char TABCH[5][50], c;
    int i, j, N;
    printf("Saisir 5 mots:\n");
    for(i=0;i<5;i++)
        gets(TABCH[i]);
    for(i=0;i<5;i++)
    {
        N = strlen(TABCH[i]);
        for(j=0;j<N/2;j++)
        {
            c = TABCH[i][j];
            TABCH[i][j]= TABCH[i][N-1-j];
            TABCH[i][N-1-j]=c;
        }
    }
    printf("La liste des mots inverses:\n");
    for(i=0;i<5;i++)
        puts(TABCH[i]);
    return 0;
}
```

Exercice 3 :

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

int main()
{
    char ph[200];
    char **tab=NULL;
    int i, nbr_mot, debut_mot;
    printf("Entrer une phrase : ");
    gets(ph);
    debut_mot=0;
    nbr_mot=0;
    for(i=0;i<strlen(ph)+1;i++)
    {
        if(ph[i]==' ' || ph[i]=='\0')
        {
```

```
            if(tab==NULL)
            {
               tab=(char**)malloc(sizeof(char*));
            }
            else
            {
               tab=(char**)realloc(tab,(nbr_mot+1)*sizeof(char*));
            }
            tab[nbr_mot]=(char*)malloc((i-debut_mot+1)*sizeof(char));
            strncpy(tab[nbr_mot],ph+debut_mot,i-debut_mot);
            tab[nbr_mot][i-debut_mot]='\0';
            nbr_mot++;
            debut_mot=i+1;
         }
      }
      printf("les mots de la phrase sont:\n");
      for(i=0;i<nbr_mot;i++)
      {
         printf("mot %d :",i);
         puts(tab[i]);
      }
      return 0;
}
```

Exercice 4 :

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

int main()
{
   char A[50][50], B[50][50], FUS[100][50], aux[50];
   int N, M, i, j;
   printf("Veuillez saisir le nombre de chaine à inserer dans le tableau A:");
   scanf("%d",&N);
   scanf("%c");
   printf("Saisir les elements du tableau A:\n");
   for(i=0;i<N;i++)
   {
      gets(A[i]);
   }
   printf("Veuillez saisir le nombre de chaine à inserer dans le tableau B:");
   scanf("%d",&M);
   scanf("%c");
   printf("Saisir les elements du tableau B:\n");
   for(i=0;i<M;i++)
   {
      gets(B[i]);
   }
   for(i=0;i<N+M;i++)
      strcpy(FUS[i],(i<N)?A[i]:B[i-N]);
   for(i=0;i<N+M;i++)
   {
      for(j=i+1;j<N+M;j++)
      {
```

```
            if(strcmp(FUS[i],FUS[j])>0)
            {
               strcpy(aux,FUS[i]);
               strcpy(FUS[i],FUS[j]);
               strcpy(FUS[j],aux);
            }
         }
      }
      printf("Les elements de FUS:\n");
      for(i=0;i<N+M;i++)
         puts(FUS[i]);
      return 0;
}
```

Exercice 5 :

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

int main()
{
   char terminasons[6][4]={ "e","es","e","ons","ez","ent" },
   sujets[6][11]={ "Je ", "Tu ", "Il/Elle ", "Nous ", "Vous ", "Ils/Elles " },
   verbe[20], radical[20], verbe_conjugue[6][20];
   int i, N;
   printf("Veuillez saisir le verbe que vous voulez conjuguer:");
   gets(verbe);
   N = strlen(verbe);
   if(verbe[N-2]=='e' && verbe[N-1]=='r')
   {
      strncpy(radical,verbe,N-2);
      for(i=0;i<6;i++)
      {
         strcpy(verbe_conjugue[i],sujets[i]);
         strcat(verbe_conjugue[i],radical);
         strcat(verbe_conjugue[i],terminasons[i]);
      }
      printf("Le verbe %s au present de indicatif:\n",verbe);
      for(i=0;i<6;i++)
      {
         puts(verbe_conjugue[i]);
      }
   }
   else
      printf("Le verbe que vous avez entree ne se termine pas par 'er' !!!");
   return 0;
}
```

Université Hassan 1er
Faculté des Sciences et Techniques
Département Mathématiques et Informatique

Travaux Pratiques 2 - Solution

Exercice 1 :

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
float distance(float,float,float,float);
int main()
{
   printf("%f",distance(0,0,0,1));
   return 0;
}

float distance(float xa, float ya, float xb, float yb)
{
   return sqrt(pow(xa-xb,2)+pow(ya-yb,2));
}
```

Exercice 2 :

```
#include <stdio.h>
#include <stdlib.h>
#define L 5
#define C 5

void afficher_matrice(int**,int,int);

int main()
{
   int **iMat;
   int i, j;
   iMat = (int**)malloc(L*sizeof(int*));
   for(i=0;i<L;i++)
      iMat[i]=(int*)malloc(C*sizeof(int));
   printf("Saisir les elements de la Matrice [5,5]:\n");
   for(i=0;i<L;i++)
   {
      for(j=0;j<C;j++)
      {
         /*
         printf("iMat[%d][%d]=",i,j);
         scanf("%d",iMat[i]+j);
         //*/
         iMat[i][j]=i+j;
      }
   }
   afficher_matrice(iMat,L,C);
   return 0;
}

void afficher_matrice(int **M, int l, int c)
{
   int i, j;
   printf("Affichage de la matrice sous forme de tableau:\niMat");
   for(j=0;j<c;j++)
```

```
        printf("\t| Col %d",j);
    printf("\t|\n");
    for(j=0;j<c;j++)
        printf("----------");
    printf("\n");
    for(i=0;i<l;i++)
    {
        printf("Ligne %d\t|",i);
        for(j=0;j<c;j++)
            printf("  %d\t|",*(M[i]+j));
        printf("\n");
    }

}
```

Exercice 3 :

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

int iHeures, iMinutes, iSecondes;
void affiche_heure();
void saisir_heure(int,int,int);
void tick();

int main()
{
    int iH, iM, iS;
    printf("Veuillez saisir l'heure actuel:\n");
    printf("heure = ");
    scanf("%d",&iH);
    printf("Minute = ");
    scanf("%d",&iM);
    printf("Seconde = ");
    scanf("%d",&iS);
    saisir_heure(iH,iM,iS);
    system("color a");
    while(1)
    {
        system("cls");
        affiche_heure();
        tick();
        sleep(1);
    }
    return 0;
}

void affiche_heure()
{
    printf("Il est %d heure%c %d minute%c %d seconde%c\n",iHeures,(iHeures>1)?'s':'
',iMinutes,(iMinutes>1)?'s':' ',iSecondes,(iSecondes>1)?'s':' ');
}

void saisir_heure(int iH, int iM,int iS)
{
```

```
   if(iH<24 && iM < 60 && iS < 60)
   {
      iHeures = iH;
      iMinutes = iM;
      iSecondes = iS;
   }
   else
   {
      printf("l'heure qui vous avez saisi est incorrecte!!");
      exit(0);
   }
}

void tick()
{
   iSecondes++;
   if(iSecondes>=60)
   {
      iSecondes=0;
      iMinutes++;
   }
   if(iMinutes>=60)
   {
      iMinutes=0;
      iHeures++;
   }
   if(iHeures>=24)
   {
      iHeures=0;
   }
}
```

Exercice 4 : *(Algorithme de cryptage)*

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

char* crypt(char* message, int cle);
char* decrypt(char* message, int cle);

int main()
{
   char *Message = "Smi S4";
   int cle = 5;
   char* Message_c = crypt(Message,cle);
   printf("le message crypte est : %s\n",Message_c);
   char* Message_d = decrypt(Message_c,cle);
   printf("le message decrypte est : %s\n",Message_d);
   return 0;
}

char* crypt(char* message, int cle)
{
   char *message_c = (char*)malloc((strlen(message)+1)*sizeof(char));
   strcpy(message_c,message);
```

```
    int i;
    for(i=0;i<strlen(message);i++)
    {
        if(message_c[i]>='a' && message_c[i]<='z')
            message_c[i]=(message_c[i]+cle-'a')%('z'-'a'+1)+'a';
        if(message_c[i]>='A' && message_c[i]<='Z')
            message_c[i]=(message_c[i]+cle-'A')%('Z'-'A'+1)+'A';
        printf("%c devient %c\n",message[i],message_c[i]);
    }
    return message_c;
}

char* decrypt(char* message, int cle)
{
    char *message_d = (char*)malloc((strlen(message)+1)*sizeof(char));
    strcpy(message_d,message);
    int i;
    for(i=0;i<strlen(message);i++)
    {
        if(message_d[i]>='a' && message_d[i]<='z')
            message_d[i]=(message_d[i]-cle-'a'+((message_d[i]-cle-'a'<0)?26:0))%('z'-'a'+1)+'a';
        if(message_d[i]>='A' && message_d[i]<='Z')
            message_d[i]=(message_d[i]-cle-'A'+((message_d[i]-cle-'a'<0)?26:0))%('Z'-'A'+1)+'A';
        printf("%c devient %c\n",message[i],message_d[i]);
    }
    return message_d;
}
```

Université Hassan 1er
Faculté des Sciences et Techniques
Département Mathématiques et Informatique

Travaux Pratiques 3 - solution

Exercice 1 :

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
// Definition des types
enum Identifiant { CIN, CNE, APOGEE };
enum Type_Utilisateur { ETUDIANT, ADMINISTRATEUR};
union Nom_Utilisateur
{
   char CIN[8];
   char CNE[11];
   int APOGEE;
};
struct User
{
   enum Identifiant id;
   union Nom_Utilisateur login;
   char password[20];
   enum Type_Utilisateur type;
   char nom[20];
   char prenom[20];
   char date_N[10];
   struct user* suivant;
};
typedef struct User User;
struct Livre
{
   char ISBN[13];
   char titre[50];
   char auteur[50];
   struct Livre* suivant;
};
typedef struct Livre Livre;
// prototypes des fonctions
User* Initialiser_Admin();
Livre* Initialiser_Livres();
void afficher_utilisateurs();
char* crypt(char* message, int cle);
char* decrypt(char* message, int cle);
User* authentification();
User* deconnexion();
void afficher_menu_etudiant();
void afficher_menu_admin();
void lister_choix_admin();
void lister_choix_etudiant();
void ajouter_utilisateur();
void afficher_livres();
//fonctions
User* liste_u=NULL;
Livre* liste_l=NULL;
```

```
int main()
{
   liste_u=Initialiser_Admin();
   liste_l=Initialiser_Livres();
   User* utilisateur=NULL;
   while(1)
   {
         while(utilisateur == NULL)
            utilisateur = authentification();
         if(utilisateur!=NULL)
         {
            system("cls");
            printf("                Bienvenue  %s\n", utilisateur->nom);
            if(utilisateur->type == ADMINISTRATEUR)
               afficher_menu_admin();
            else
               afficher_menu_etudiant();
            scanf("%c");
            system("cls");
            utilisateur = deconnexion();
         }
   }
   return 0;
}

void afficher_menu_etudiant()
{
   int choix;
   lister_choix_etudiant();
   do
   {
      printf("Tapez votre choix : ");
      scanf("%d",&choix);
      if(choix<1 || choix > 2)
         printf("Votre choix est invalide !!!\n");
      else
      {
         switch(choix)
         {
         case 1:
            afficher_livres();
            system("pause");
            lister_choix_etudiant();
            break;
         case 2:
            printf("Au revoir");
            break;
         }
      }
   }while(choix != 2);
}

void lister_choix_etudiant()
{
```

```
    system("cls");
    printf("========== Menu ==========\n");
    printf("1- Afficher la liste des livres\n");
    printf("2- Se deconnecter\n");
}

void afficher_menu_admin()
{
    int choix;
    lister_choix_admin();
    do
    {
        printf("Tapez votre choix : ");
        scanf("%d",&choix);
        if(choix<1 || choix > 4)
            printf("Votre choix est invalide !!!\n");
        else
        {
            switch(choix)
            {
            case 1:
                afficher_utilisateurs();
                system("pause");
                lister_choix_admin();
                break;
            case 2:
                ajouter_utilisateur();
                system("pause");
                lister_choix_admin();
                break;
            case 3:
                afficher_livres();
                system("pause");
                lister_choix_admin();
                break;
            case 4:
                printf("Au revoir");
                break;
            }
        }
    }while(choix != 4);
}

void afficher_livres()
{
    Livre* l = liste_l;
    system("cls");
    printf("==================Liste des Livres:==================\n");
    int cle;
    while(l!=NULL)
    {
        printf("ISBN : %s\n",l->ISBN);
        printf("Titre : %s\n",l->titre);
        printf("Auteur : %s\n",l->auteur);
```

```
        printf("============================================================\n");
        l = l->suivant;
    }
}

void ajouter_utilisateur()
{
    int choix;
    User* utilisateur;
    utilisateur = (User*)malloc(sizeof(User));
    system("cls");
    printf("          === Ajout d'un nouveau utilisateur ===\n");
    do
    {
        printf("Voulez vous identifier l'utilisateur par CIN(1), CNE(2) ou bien par code apogee(3) ?");
        scanf("%d",&choix);
        if(choix>3 || choix <1)
            printf("Votre choix est invalide !!\n");
    }while(choix>3 || choix <1);
    scanf("%c");
    int cle;
    switch(choix)
    {
        case 1:
            utilisateur->id = CIN;
            printf("CIN :");
            gets((utilisateur->login).CIN);
            cle = strlen((utilisateur->login).CIN);
            break;
        case 2:
            utilisateur->id = CNE;
            printf("CNE :");
            gets((utilisateur->login).CNE);
            cle = strlen((utilisateur->login).CNE);
            break;
        case 3:
            utilisateur->id = APOGEE;
            printf("APOGEE :");
            scanf("%d",&((utilisateur->login).APOGEE));
            cle = (utilisateur->login).APOGEE%('z'-'a');
            break;
    }
    char password[20];
    printf("Mot de passe :");
    scanf("%s",password);
    strcpy(utilisateur->password, crypt(password,cle));
    do
    {
        printf("Le type d'utilisateur ? Etudiant(1) ou Administrateur(2):");
        scanf("%d",&choix);
        if(choix>2 || choix <1)
            printf("Votre choix est invalide !!\n");
    }while(choix>2 || choix <1);
    scanf("%c");
```

```
    if(choix == 1)
        utilisateur->type = ETUDIANT;
    else
        utilisateur->type = ADMINISTRATEUR;
    printf("Nom :");
    gets(utilisateur->nom);
    printf("Prenom :");
    gets(utilisateur->prenom);
    printf("Date de Naissance :");
    gets(utilisateur->date_N);
    utilisateur->suivant = liste_u;
    liste_u = utilisateur;
}

Livre* Initialiser_Livres()
{
    int i, Nb_L = 10;
    Livre* l;
    for(i=0;i<Nb_L;i++)
    {
        l = (Livre*)malloc(sizeof(struct Livre));
        char snum[5];
        itoa(i,snum,10);    // convertir un entier en chaine de caractère
        strcpy(l->ISBN,snum);
        strcpy(l->titre,snum);
        strcpy(l->auteur,snum);
        l->suivant = liste_l;
        liste_l = l;
    }
    return liste_l;
}

void lister_choix_admin()
{
    system("cls");
    printf("========== Menu ==========\n");
    printf("1- Afficher la liste d'utilisateurs\n");
    printf("2- Ajouter un utilisateur\n");
    printf("3- Afficher la liste des livres\n");
    printf("4- Se deconnecter\n");
}

User* deconnexion()
{
    return NULL;
}

User* authentification()
{
    char login[11], password[20];
    printf("Login : ");
    gets(login);
    printf("Mot de Passe : ");
    gets(password);
```

```
    User* l = liste_u;
    while(l!=NULL)
    {
        switch(l->id)
        {
        case CIN:
            if(strcmp((l->login).CIN,login)==0 && strcmp(l->password,crypt(password,strlen((l-
>login).CIN)))==0)
                return l;
            break;
        case CNE:
            if(strcmp((l->login).CNE,login)==0 && strcmp(l->password,crypt(password,strlen((l-
>login).CNE)))==0)
                return l;
            break;
        case APOGEE:
            if(atoi(login) == (l->login).APOGEE && strcmp(l->password,crypt(password,(l-
>login).APOGEE%('z'-'a')))==0)
                return l;
            break;
        }
        printf("test avec %s\n",l->nom);
        l = l->suivant;
    }
    printf("le nom d'utilisateur ou le mot de passe est incorrect !!\n");
    system("pause");
    system("cls");
    return NULL;
}

User* Initialiser_Admin()
{
    User* admin;
    admin = (User*)malloc(sizeof(User));
    admin->id = CIN;
    strcpy((admin->login).CIN,"A123456");
    strcpy(admin->password,crypt("root",strlen((admin->login).CIN)));
    admin->type = ADMINISTRATEUR;
    strcpy(admin->nom,"admin");
    strcpy(admin->prenom,"admin");
    strcpy(admin->date_N,"1980-01-01");
    admin->suivant = NULL;
    return admin;
}

void afficher_utilisateurs()
{
    User* l = liste_u;
    system("cls");
    printf("==================Liste d'utilisateurs:==================\n");
    int cle;
    while(l!=NULL)
    {
        switch(l->id)
```

```
        {
            case CIN: printf("CIN : %s\n", (l->login).CIN);
            cle = strlen((l->login).CIN);
            break;
            case CNE: printf("CNE : %s\n", (l->login).CNE);
            cle = strlen((l->login).CNE);
            break;
            case APOGEE: printf("APOGEE : %d\n", (l->login).APOGEE);
            cle = (l->login).APOGEE%('z'-'a');
            break;
        }
        printf("Mot de Passe: %s\n",decrypt(l->password,cle));
        printf("Type d'utilisateur : %s\n", (l->type == ETUDIANT)?"Etudiant":"Administrateur");
        printf("Nom : %s\n",l->nom);
        printf("Prenom : %s\n", l->prenom);
        printf("Date de Naissance : %s\n", l->date_N);
        printf("=============================================================\n");
        l = l->suivant;
    }
}

char* crypt(char* message, int cle)
{
    char *message_c = (char*)malloc((strlen(message)+1)*sizeof(char));
    strcpy(message_c,message);
    int i;
    for(i=0;i<strlen(message);i++)
    {
        if(message_c[i]>='a' && message_c[i]<='z')
            message_c[i]=(message_c[i]+cle-'a')%('z'-'a'+1)+'a';
        if(message_c[i]>='A' && message_c[i]<='Z')
            message_c[i]=(message_c[i]+cle-'A')%('Z'-'A'+1)+'A';
        //printf("%c devient %c\n",message[i],message_c[i]);
    }
    return message_c;
}

char* decrypt(char* message, int cle)
{
    char *message_d = (char*)malloc((strlen(message)+1)*sizeof(char));
    strcpy(message_d,message);
    int i;
    for(i=0;i<strlen(message);i++)
    {
        if(message_d[i]>='a' && message_d[i]<='z')
            message_d[i]=(message_d[i]-cle-'a'+((message_d[i]-cle-'a'<0)?26:0))%('z'-'a'+1)+'a';
        if(message_d[i]>='A' && message_d[i]<='Z')
            message_d[i]=(message_d[i]-cle-'A'+((message_d[i]-cle-'a'<0)?26:0))%('Z'-'A'+1)+'A';
        //printf("%c devient %c\n",message[i],message_d[i]);
    }
    return message_d;
}
```