Programmation en C/C++

Série 9
Chaînes de caractères
Types complexes
(structures, énumérés)

Cette série est longue et propose des exercices corrigés
Plusieurs exercices (facultatifs) ne sont pas corrigés vous pouvez néanmoins les chercher et nous publierons éventuellement en ligne les corrections.

 Important !

 Difficile !

Objectifs
Maîtriser les types "caractère" et "chaîne de caractères".
Utiliser des fonctions travaillant sur les chaînes de caractères.
Créer de nouveaux types complexes : des structures ou des énumérés.
Combiner les différents types complexes : tableaux d'énumérés, tableaux de structures, structures de tableaux…

Introduction

Le type caractère permet de stocker un seul caractère (symbole, lettre).
Le type chaîne de caractères permet de stocker un ensemble de caractères composé de symboles ou lettres (mot, nom, phrase…).
En C, il existe des fonctions prédéfinies qui permettent de manipuler les chaînes de caractères. Il faudra utiliser la librairie *<string.h>.*
Nous verrons enfin qu'il est possible de créer de nouveaux types de données à ceux déjà existant et prédéfinis.

1- Caractères et chaînes de caractères

Notion de caractère

Nous avions parlé du type *char* (caractère) lors du chapitre sur les variables et leurs types…

Revenons sur le programme suivant qui utilise le type *char* :

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

int main() {
char VAR1 = 'A';

printf("Lettre stockee : %c\n", VAR1);
printf("Code ASCII associe a la lettre : %d\n", VAR1);

return 0;
}
```

La variable *VAR1* a le type *char* (caractères), on lui affecte la lettre "*A*". Mais *A* correspond à un caractère ASCII (numéro 65, pour les cas).
On peut donc utiliser les formats suivants :

- *%c* pour afficher la lettre
- *%d* pour afficher le n° ASCII correspond à cette lettre.

On aurait très bien pu écrire le programme suivant :

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

int main() {
int VAR1 = 'A';

printf("Lettre stockee : %c\n", VAR1);
printf("Code ASCII associe a la lettre : %d\n", VAR1);

return 0;
}
```

Donc en résumé,

- Le type *char* permet de stocker des nombres compris entre -128 et 127.
- Si l'on utilise le format *%c* il s'affichera le caractère associé (lettre ou symbole)
- Si l'on utilise le format *%d* il s'affichera le code ASCII correspondant à ce caractère.
- Les apostrophes ' …' sont utilisées pour les caractères.

Exercice d'application
Écrire le programme qui affiche les caractères associés aux codes ASCII entre 0 et 127.

Corrigé
```
#include<stdio.h>
#include<stdlib.h>
int main() {
int i;

for (i=0;i<127;i++)
        printf ("ASCII %d : %c\t",i,i);
return 0;
}
```

Notion de purge du flux d'entrée
Voici un programme tout simple :

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


int main() {
char L1 ;
char L2 ;

printf("Saisissez une lettre \n");
scanf("%c",&L1);

printf("Saisissez une deuxième lettre \n");
scanf("%c",&L2);

printf("lettre 1 : %c\n",L1);
printf("lettre 2 : %c\n",L2);

return 0;
}
```

Rq
Le programme n'attend pas la saisie de la deuxième lettre et a placé le restant de la première saisie (le *\n* lorsqu'on appuie sur la touche return pour valider) dans *L2* et affiche donc *L2* comme étant vide.

Corrigeons le programme
```
#include <stdlib.h>
#include <string.h>
#include<stdio.h>


int main() {
```

```
char L1 ;
char L2 ;

char buffer; // pour l'élimination de la fin de chaine

printf("Saisissez une lettre \n");
scanf("%c",&L1);

scanf("%c", &buffer); // on met le caractère résiduel du flux dans la variable buffer

printf("Saisissez une deuxième lettre \n");
scanf("%c",&L2);

scanf("%c", &buffer);

printf("lettre 1 : %c\n",L1);
printf("lettre 2 : %c\n",L2);

return 0;
}
```

Rq
Là, le programme demande de saisir un caractère dans L1 puis un autre caractère dans L2 et les affiche ensuite. Le caractère *n* a été placé dans la variable buffer.

Notion de chaîne de caractères

On l'avait entrevu sans le dire au chapitre sur les tableaux :

- **Une chaîne de caractères est un tableau de caractères,**
- La chaîne de caractères est composé de symbole et se termine par le symbole de fin de chaîne "\0",
- Les guillemets " " sont utilisés pour les chaînes de caractères.

Soit un mot de 7 lettres ("Langage") stocké dans une variable nommée *Chaine*.

Voici sa déclaration suivie de son affectation :
char Chaine[] = "Langage";
Il est à noter que le caractère de fin de chaîne est ajouté automatiquement.

Attention ! On ne peut affecter de cette façon qu'en début de programme, lors de la déclaration de la chaîne de caractères. Si l'affectation se réalise plus tard il faudra utiliser une boucle et remplir "les cases du tableau".

On peut représenter cette chaîne comme un tableau de 8 cases :

L	*a*	*n*	*g*	*a*	*g*	*e*	\0

Pour afficher une chaîne de caractères on peut soit utiliser :

- Le format %s pour afficher la chaîne d'un seul coup,
- Le format %c avec une boucle for qui va afficher tous les caractères du tableau.

Exercice d'application

Créer un programme C qui affecte "Essai" dans une variable nommée MOT. Afficher la variable d'un seul coup puis afficher, d'une deuxième façon, la variable à l'aide d'une boucle qui balaie les éléments de MOT.

Corrigé

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

int main() {
char MOT[] = "Essai";
int i;

printf("Votre mot : %s\n ", MOT);

for (i=0;i<6;i++)
        printf ("MOT [%d] : %c\n", (i+1),MOT[i]);

return 0;
}
```

Conséquences

Une chaîne de caractères étant un tableau donc le dernier élément étant le symbole de fin de chaîne "\0" ; il ne sera pas nécessaire d'envoyer en argument à une fonction la longueur du tableau : le tableau seul suffira.

Saisie d'une chaîne de caractères

L'instruction *scanf()* avec le format *%s* permettra de saisir une chaîne de caractères et de l'affecter à une variable (tableau de caractères dont on précisera la taille entre crochets).

Exemple d'application

Écrire un programme qui demande de saisir son prénom (chaîne de 30 caractères) puis l'affiche.

Corrigé

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

int main() {
char Prenom[30] ;
```

```
printf("Quel est votre prenom \n?");
scanf("%s",&Prenom);
printf("Votre prenom : %s \n ", Prenom);

return 0;
}
```

Remarque
Si, lors de la saisie d'une chaîne de caractères à l'aide d'un *scanf()*, on saisit une espace (nom féminin !), les lettres saisies après l'espace ne seront pas enregistrées dans la variable.

Fonctions de base manipulant les chaînes de caractères

En C, il existe des fonctions prédéfinies qui permettent de manipuler les chaînes de caractères. Il faudra utiliser la librairie <*string.h*> et ajouter, en début de programme, la directive de précompilation suivante # *include <string.h>.*

Remarque Par la suite, *Ch1* et *Ch2* seront des chaînes de caractères.

Fonction *strlen()* Longueur de chaîne

La fonction *strlen()* reçoit une chaîne en argument et renvoie un entier correspondant à la longueur réelle de la chaîne stockée. *strlen()* ne tient pas compte du symbole de fin de chaîne.

Prototype de la fonction
int strlen(const char);*

La fonction reçoit une chaîne de caractères en argument et renvoie un entier.

Exemple
```
char Ch1[20] ="Hello !";
return strlen(Ch1);
```

La fonction retournera 7.
Testez.

Fonction *strchr()* Recherche d'un caractère dans une chaîne

La fonction *strchr()* reçoit une chaîne et un caractère à chercher et renvoie un pointeur. La fonction retourne un pointeur sur la première occurrence du caractère trouvé et NULL (pointeur vide) si le caractère n'a pas été trouvé.

Prototype de la fonction

char strchr(char* ch1, int Car);*

La fonction reçoit une chaîne de caractères et un caractère en arguments et renvoie une adresse (pointeur).

Exemple
char strchr ("Bonjour",'o');*
Renvoie l'adresse de la sous-chaîne où est trouvée le premier 'o'.

Fonction *strstr()* Recherche d'une chaîne dans une autre chaîne

La fonction *strstr()* reçoit 2 chaînes et renvoie un pointeur sur la première occurrence de la chaîne 2 trouvée dans la chaîne 1 et NULL si la chaîne 2 n'a pas été trouvée.

Prototype de la fonction
char strstr(char* , char*);*
La fonction reçoit deux chaînes de caractères en arguments et renvoie une adresse (pointeur).

Exemple
char strstr ("Bonjour","jour");*
Renvoie l'adresse où est trouvée la chaîne "jour".

Fonction *strcmp ()* Comparaison de 2 chaînes

La fonction *strcmp()* compare 2 chaînes de caractères : renvoie un entier positif si la chaîne 1 arrive avant la chaîne 2 par ordre alphabétique. Retourne 0 si les 2 chaînes sont de taille égale.

Prototype de la fonction
int strcmp(const char , const char*);*
La fonction reçoit deux chaînes de caractères en arguments et renvoie un entier.

Exemple
char Ch1[20] ="lo!";
char Ch2[5] ="lo!";
strcmp(Ch1, Ch2);

La fonction retournera 0 : les deux chaînes sont identiques.

Fonction *strcpy ()* copie une chaîne dans une chaîne

La fonction *strcpy()* renvoie la chaîne 1 dans laquelle on a recopié la chaîne 2.
Le contenu de la chaîne 1 est donc écrasé.

Prototype de la fonction
char strcpy(char*, const char*);*
La fonction reçoit deux chaînes de caractères en arguments et renvoie une chaîne de caractères (pointeur).

Exemple
char Ch1[20] ="Hel";
char Ch2[5] ="lo !";
printf ("%s",strcpy(Ch1, Ch2));
lo ! s'affichera.

Fonction *strcat ()* Concaténation de 2 chaînes

La fonction *strcat()* concatène une chaîne 2 après une chaîne 1 et renvoie le résultat dans la chaîne 1.

Prototype de la fonction
char strcat(char* , char*);*

La fonction reçoit deux chaînes de caractères en arguments et renvoie une chaîne de caractères.

Exemple
char Ch1[20] ="Hel";
char Ch2[5] ="lo !";
strcat(Ch1, Ch2);

Ch1 contiendra "Hello !".

Attention à la taille minimale de *Ch1* !

2- Les types complexes

Jusqu'à présent nous avons toujours utilisé des types prédéfinis (*int, float, char, long*...). Le langage C autorise la création de nouveaux types personnalisés ; on parle de types complexes. Nous allons aborder ici les structures puis les types énumérés.

Les structures

Exemple
Créons ici la structure "*Date*" composé de 3 entiers correspondant respectivement aux jour, mois, année.

// Création de la structure complexe nommée *Date* composée de 3 entiers : jour mois année

struct Date {
int jour ;
int mois ;
int annee ;
};

//Déclaration d'une variable nommée *date1* de type *date*
struct Date date1;

Pour accéder au jour de la variable *date1* on fera appel à *date1.jour*

- Pour créer une structure, on utilisera l'instruction *struct*
- La structure sera placée entre accolades.
- Attention : il faut placer un point virgule après l'accolade fermante.
- Lorsqu'on déclarera une variable de ce type il faudra ajouter le mot clé *struct* avant sa déclaration.
- Ne pas confondre création du type et déclaration de la variable.

Exemple de cours
Créer un programme qui demande de saisir une date au clavier puis l'affiche.

Corrigé

```
#include<stdio.h>
int main () {
// Création de la structure complexe nommée Date composée de 3 entiers : jour mois année.

struct Date  {
int jour;
int mois;
int annee;
};

//Déclaration de la variable date1

struct Date date1;

printf ( "saisir le jour:\n" );
scanf("%d",&date1.jour);
printf ("saisir le mois:\n");
scanf ("%d",&date1.mois);
printf("saisir l'annee:\n");
scanf("%d",&date1.annee );

printf("Votre jour est: %d/%d/%d\n ", date1.jour, date1.mois, date1.annee);

return 0;
}
```

Conséquences immédiates

- On peut créer une **structure de tableaux** (Exemple : une structure avec des chaînes de caractères puisque les chaînes de caractères sont des tableaux de *char* !),
- On peut créer un **tableau de structures** (Exemple : un tableau de dates ou un tableau de personnes, de voitures…),
- On peut créer une **structure de structures** (Exemple : une structure "Achat" avec le montant acheté et la date d'achat qui est elle-même une structure).

Remarque
Une structure nouvellement créée pourra être enregistrée dans un fichier d'en-tête *xxxx.h* (*header*). Si on ajoute une directive de compilation (*#include "xxxx.h")* la structure sera connue du programme C. Cela allège le programme mais il y a un inconvénient : on ne voit pas la structure au niveau du code source C ce qui peut être embêtant pour toute personne qui ne l'a pas créée (on peut néanmoins ouvrir le *header* pour la voir !).

Remarque
Les structures nous serviront pour créer des listes chaînées…

Les types énumérés

Le langage C permet la création de types énumérés.
Il s'agit d'un type de données avec une liste finie de valeurs possibles pour celui-ci.
On utilisera l'instruction "*enum*".

Exemple 1
Le type de couleurs de cartes : *enum Couleur_carte {Carreau,Cœur, Pique, Trèfle};*

Exemple 2
Créons un type *Couleurs* qui autorise uniquement les trois couleurs Vert, Blanc, Rouge.
Voici le code source :

```
enum Couleurs
{
Vert , Blanc, Rouge
};
```

Créons ensuite une variable Color1 qui aura la couleur Rouge.
Voici le code source :

```
Couleurs Color1 = Rouge;
```


Chaque valeur de la liste d'énumérés correspond à un nombre. Par défaut, et en l'absence de consigne :

- Vert vaut 0,
- Blanc vaut 1,

- Rouge vaut 2.

Exemple de cours 1
Créer le programme C reprenant l'exemple ci-dessus et qui affiche la valeur de la couleur. Que s'affichera-t-il ?

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

int main () {
enum Couleurs { Vert , Blanc, Rouge };
enum Couleurs Color1 = Rouge;

printf ( "Numéro de couleur : %d\n", Color1 );

return 0;
}
```

Il s'affichera 2.

Exemple de cours 2
Modifier le programme précédent avec pour valeur des couleurs :
- Vert vaut A,
- Blanc vaut B,
- Rouge vaut C.

. Que s'affichera-t-il ?

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

int main () {
enum Couleurs { Vert = 'A' , Blanc ='B', Rouge = 'C' };
enum Couleurs Color1 = Rouge;

printf ( "Numéro de couleur : %c\n", Color1 );

return 0;
}
```

Il s'affichera "C". Cela est possible car une lettre est associée à un nombre (code ASCII), comme nous l'avons déjà vu. Attention au format d'affichage %c pour afficher la lettre associée !

Remarque
Les types énumérés n'autorisent pas d'entrées/sorties pour des affectations. Chaque valeur d'énuméré correspond à un nombre que l'on pourra afficher éventuellement.


On peut boucler sur un énuméré ; on pourra écrire, par exemple :
For (Jour1 = Lundi ; Jour1 <=Dimanche; Jour1++)…
Avec le type énuméré
Enum Jour{
Lundi, Mardi, Mercredi, Jeudi, Vendredi, Samedi, Dimanche
};

Utilisation de l'instruction *typedef*

L'instruction *typedef* permet d'alléger le programme en n'utilisant plus le type *struct* ou *enum* pour les déclarations ultérieures. L'instruction *typedef* permet de donner un autre nom à un type existant ou créé.

Exemple 1
Imaginons que l'on ne souhaite plus utiliser le mot "Int" pour le type "Entier" mais utiliser un type nommé "Entier" :

Typedef int Entier;
Entier Var1;
Ici, on a défini un type *Entier* correspondant au type existant *int* ; on déclare ensuite une variable *VAR1* de type *Entier*.

Exemple 2
struct Date {
int jour;
int mois;
int annee;
};

Si l'on veut déclarer une variable *date1* ayant la structure *Date* on devait écrire, jusqu'à présent :
struct Date date1;

On peut ajouter la ligne suivante :
typedef struct Date Date;
Cela signifie que le type *Date* correspondra à la structure *Date*.

Toutes les variables de type *Date* pourront ensuite être déclarées plus simplement avec le seul type *Date* comme suit :
Date jour1, jour2;
On a déclaré ici 2 variables nommées *jour1* et *jour2* de type *Date.*

Exemple 3
Enum Jour*{*
Lundi, Mardi, Mercredi, Jeudi, Vendredi, Samedi, Dimanche
};

Typedef enum Jour Jour;
Jour Aujourdhui;

On déclare une variable de type *Jour* qui correspond au type *enum Jour*.

On aurait pu écrire :
Enum { Lundi, Mardi, Mercredi, Jeudi, Vendredi, Samedi, Dimanche } Jour ;

EXERCICES

Exercice 1

Énoncé

Écrire le programme qui demande de saisir un symbole au clavier puis l'affiche à l'écran suivi de son code ASCII.

Corrigé

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

int main()
{
char Symb;

printf("Saisissez un symbole\n");
scanf("%c",&Symb);
printf("Le symbole saisi est : %c ; son code ASCII est : %d\n",Symb, Symb);

return 0;
 }
```

Exercice 2

Énoncé

Que fait le programme suivant ?
Même question avec Ch1[7] ?
Même question avec Ch1[8] ?

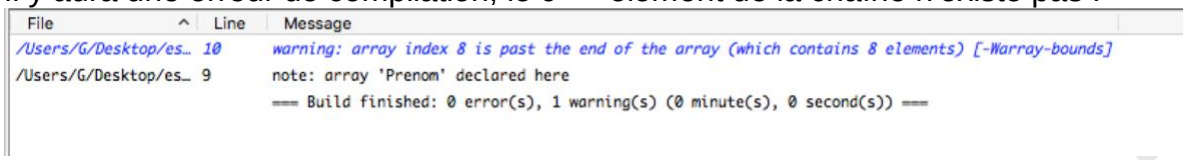
```
#include<stdio.h>
#include<stdlib.h>

int main() {
char Ch1[]="Langage";
printf("%c",Ch1[6] );

return 0;
}
```

Corrigé

Avec Ch1[6] ?
Il s'affichera "e", c'est-à-dire le 7ème élément de la chaîne.

Avec Ch1[7] ?
Il ne s'affichera rien, c'est-à-dire le 8ème élément de la chaîne (caractère de fin de chaîne).

Avec Ch1[8]
Il y aura une erreur de compilation, le 9ème élément de la chaîne n'existe pas !



Exercice 3

Énoncé

Écrire le programme qui demande de saisir un mot clavier et qui effectue les affichages comme spécifiés sur la capture d'écran ci-après.

Conseil : pensez à inclure la librairie *string.h*.

```
Saisissez un mot
FRUIT
Il y a 5 lettre(s) dans votre mot saisi (FRUIT)
```

Corrigé

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

int main() {
char Ch1[20] ;

printf("Saisissez un mot\n");
scanf("%s",&Ch1);
printf("Il y a %ld lettre(s) dans votre mot saisi (%s)\n \n ", strlen(Ch1), Ch1);

return 0;
}
```

Remarque *:* On a utilisé un entier long (*ld*) : cela permet d'éviter un warning signalant qu'il vaut mieux utiliser un entier long qu'un entier.

Exercice 4

Énoncé

Écrire un programme qui demande de saisir une chaîne de caractères puis demande quel caractère on souhaite rechercher dans cette chaîne.

Le programme affiche l'adresse où on le trouve éventuellement sinon, il affiche "caractère non trouvé".

Conseil : Pensez à éliminer le caractère de fin de chaîne avec une instruction du type : *scanf("%c",&buffer);*

Corrigé

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

int main() {
char Ch1[20], Car ;
char* P1;
char buffer; // pour l'elimination de la fin de chaine

printf("Saisissez un mot\n");
scanf("%s",&Ch1);
scanf("%c", &buffer);//elimination de la fin de chaine

printf("Saisissez un caractere a rechercher\n");
scanf("%c",&Car);
scanf("%c",&buffer);//elimination de la fin de chaine

P1= strchr (Ch1,Car);
if (P1==NULL)
        printf ("Caractere non trouve ! \n ");
else
        printf ("caractere present ; adresse ou le  caractere a ete trouve : %p\n ", P1);
return 0;
}
```

Exercice 5

Énoncé Structure de tableaux

Créer une structure *Personne* comportant un nom et un prénom.

Écrire un programme faisant appel à cette structure, demandant de saisir les données d'un utilisateur puis les affichant.

Corrigé

```
#include <stdio.h>
#include <stdlib.h> int main()
{

struct Personne
{
char Nom [15];
char Prenom [15];
};
```

```
typedef struct Personne Personne;

Personne Pers1;
printf("Quel est votre nom de famille ? ");
scanf("%s", Pers1.Nom);
printf("Votre prenom ? ");
scanf("%s", Pers1.Prenom);
printf("Vous vous appelez %s %s", Pers1.prenom, Pers1.nom);
return 0;
}
```

Exercice 6

Énoncé

Qu'affiche le programme ci-après ? Compilez et exécutez le.

```
#include<stdio.h>
#include<stdlib.h>
int main() {

enum Notes {   Do=1, Re=2, Mi=3, Fa=4, Sol=5, La=6, Si=7  };
typedef enum Notes Notes;

Notes Note1, Note2;
Note1=Mi;
Note2=Do;

printf("Position de la note : %d \n\n",Note1);

if (Note1==Note2)
        printf("Notes identiques\n");
else
        printf("Notes differentes\n");

return 0;
}
```

Remarque

La note "do" ne peut pas être utilisée car "do" est un mot clé du langage C.

Corrigé

```
Position de la note : 3

Notes differentes
```

Exercice 7

Énoncé

Écrire un programme qui demande de saisir un mot. Puis le programme demande si on souhaite modifier le mot. Si oui, il est fait appel à une fonction *Modif()* qui modifiera le mot (pointeurs !). Une seconde fonction, *Affiche(),* affichera le mot avant et après modification.

Corrigé

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

void Affiche (char *);
void Modif (char *);

int main() {
char Mot[50] ;
printf("Quel est votre mot ? \n");
scanf("%s",&Mot);

Affiche(Mot);
Modif(Mot);
Affiche(Mot);

return 0;
}

void Affiche (char *Mot) {
printf("Votre mot : %s \n",Mot);
return ;
}

void Modif (char *Mot) {
printf("Quel est votre nouveau mot ?\n");
scanf("%s", Mot);
return ;
}
```

Exercice 8

Énoncé

Créer le type "BilletDeTrain" qui comportera :

- Un numéro de billet
- Prix du billet
- Numéro de train

- Numéro de voiture
- Numéro de place
- Gare de départ
- Jour de départ
- Heure de départ
- Gare d'arrivée
- Jour d'arrivée
- Heure d'arrivée

Créer un programme qui demande de saisir un billet de train et affiche ensuite ses caractéristiques.
Dans un second temps, écrire une fonction qui affiche la durée (en minutes) du voyage associé au billet de train.

Corrigé non finalisé
Rq : le programme calcule les durées de trajets à cheval sur 2 jours et de moins de 24 heures.

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

void dureevoyage(char *stringdep,char *stringarr);
int main()
{
  typedef struct BilletDeTrain BilletDeTrain;
  struct BilletDeTrain
  {
     int numBillet;
     float prixBillet;
     int numTrain;
     int numVoiture;
     int numPlace;
     char gareDepart[50];
     char jourDepart[50];
     char heureDep[10];
     char gareArrivee[50];
     char jourArrivee[50];
     char heureArrivee[10];
  };
BilletDeTrain voyage1;
  printf("saisir le numero de billet\n");
  scanf("%d",&voyage1.numBillet);
```

```
 printf("saisir le prix du billet\n");
 scanf("%f",&voyage1.prixBillet);
 printf("saisir le numero de train\n");
 scanf("%d",&voyage1.numTrain);
 printf("saisir le numero de voiture\n");
 scanf("%d",&voyage1.numVoiture);
 printf("saisir le numero de place\n");
 scanf("%d",&voyage1.numPlace);
 printf("saisir la gare de depart\n");
 scanf("%s",voyage1.gareDepart);
 printf("saisir le jour de depart\n");
 scanf("%s",voyage1.jourDepart);
 printf("saisir l heure de depart **important de saisir sous format xxhxx**\n");
 scanf("%s",voyage1.heureDep);
 printf("saisir la gare d arrivee\n");
 scanf("%s",voyage1.gareArrivee);
 printf("saisir le jour d arrivee\n");
 scanf("%s",voyage1.jourArrivee);
 printf("saisir l heure d arrivee  **important de saisir sous format xxhxx**\n");
 scanf("%s",voyage1.heureArrivee);
 printf("\n\nbillet num: %d \tcoutant: %.2f euros \ntrain num: %d \tvoiture num %d",
voyage1.numBillet, voyage1.prixBillet, voyage1.numTrain, voyage1.numVoiture);
 printf("\tplace num: %d \n\ndepart gare: %s \tle :%s \ta:
%s",voyage1.numPlace,voyage1.gareDepart,voyage1.jourDepart,voyage1.heureDep
);
 printf("\n\narrivee gare: %s \tle :%s \ta:
%s\n\n",voyage1.gareArrivee,voyage1.jourArrivee,voyage1.heureArrivee);
 dureevoyage(voyage1.heureDep,voyage1.heureArrivee);
 return 0;
}
void dureevoyage(char stringdep[],char stringarr[])
{
 char H1=stringdep[0] ; char h1=stringdep[1];
 char H2=stringarr[0] ; char h2=stringarr[1];
 char M1=stringdep[3] ; char m1=stringdep[4];
 char M2=stringarr[3] ; char m2=stringarr[4];
 int C=H1-48 ; int c=h1-48 ; int E=M1-48 ; int e=m1-48 ; //on obtient les heures et
minutes de départ H1h1M1m1

 int D=H2-48 ; int d=h2-48 ; int F=M2-48 ; int f=m2-48 ; //idem pour l heure d arrivee
```

```
  if ((10*H1+h1)>(10*H2+h2)){printf("la duree est: %d min",(((23-(10*C+c))*60)+(60-(10*E+e))+(D*600+d*60+F*10+f)));}
  else printf("la duree est: %d min",((D*600+d*60+F*10+f)-(C*600+c*60+E*10+e)));
}
```

Exercice 9

Énoncé

Créer un programme qui utilise une variable initialisée à "*true*", de type *Boolean* (*true, false*) à créer. Le programme teste sa valeur (test conditionnel) et affiche sa valeur (0 ou 1).

Corrigé

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

int main () {
enum Boolean {
true =1,
false =0
};

enum Boolean Var1;
Var1 = true;

if (Var1 == true)
        printf ( "Valeur de votre variable : %d\n",Var1 );
else
        printf ( "Valeur de votre variable : %d\n",Var1 );
return 0;
}
```

Exercice 10

Énoncé Structure de structure et d'énumérés

Créer une structure nommée Individu avec, comme champs :

- Nom (chaîne de 15 caractères)
- Prénom (chaîne de 15 caractères)
- Date de naissance (structure Date)
- État (célibataire, marié, pacsé, divorcé) (utiliser un énuméré).

Écrire le programme qui demande de renseigner au clavier les informations concernant un seul individu puis les affiche.

Corrigé

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

int main()
{
  typedef enum Etat Etat;

enum Etat
  {
     celibataire=1 , marie=2, pacse=3, divorce=4
  };

  Etat situation;

 typedef struct Date Date;
  struct Date
  {
     int jour;
     char mois[15];
     int annee;

  };

  typedef struct Individu Individu;
  struct Individu
  {
     char nom[15];
     char prenom[15];
     Date date;
     Etat situation;
  };

  Individu sujet1;

  printf("nom:\t");
  scanf("%s",sujet1.nom);
   printf("\nprenom:\t");
  scanf("%s",sujet1.prenom);
  printf("\njour de naissance:\t");
```

```
  scanf("%d",&sujet1.date.jour);
  printf("\nmois de naissance:\t");
  scanf("%s",sujet1.date.mois);
  printf("\nannee de naissance:\t");
  scanf("%d",&sujet1.date.annee);
  printf("\nstatut(1,2,3 ou 4):\t");
  scanf("%d",&sujet1.situation);
  printf("%s %s ne le %d %s %d Statut : %d\n
",sujet1.nom,sujet1.prenom,sujet1.date.jour,sujet1.date.mois,sujet1.date.annee,sujet
1.situation);

  return 0;
}
```

Exercice 11

Énoncé Tableau de structure et d'énuméré

Modifier le programme précédent en créant un tableau de 4 personnes.

Le programme affichera ensuite les données saisies et concernant les 4 personnes.

Corrigé

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

int main()
{  int i ;
  typedef enum Etat Etat;
  enum Etat
  {
     celibataire=1 , marie=2, pacse=3, divorce=4
  };
  Etat situation;
  typedef struct Date Date;
  struct Date
  {
     int jour;
     char mois[15];
     int annee;
```

```
};

typedef struct Individu Individu;
struct Individu
{
    char nom[15];
    char prenom[15];
    Date date;
    Etat situation;
};

//Tableau de 4 individus
Individu T[4];

for (i=0; i<4; i++)
{
printf("Individu numero %d\n" ,  (i+1));
printf("nom:\t");
scanf("%s", T[i].nom);
printf("\nprenom:\t");
scanf("%s", T[i].prenom);
printf("\njour de naissance:\t");
scanf("%d",& T[i].date.jour);
printf("\nmois de naissance:\t");
scanf("%s", T[i].date.mois);
printf("\nannee de naissance:\t");
scanf("%d",& T[i].date.annee);
printf("\nstatut(1,2,3 ou 4):\t");
scanf("%d",& T[i].situation);
 }
for (i=0; i<4;i++)
  {
    printf("%s %s ne le %d %s %d Statut : %d\n ", T[i].nom, T[i].prenom,   T[i].date.jour,
T[i].date.mois, T[i].date.annee, T[i].situation);
   }

return 0;
}
```

Exercice 12

Énoncé

Créer un tableau de billets de train (voir exercice 8). Écrire le programme qui demande de saisir les données concernant 3 billets puis affiche, à l'aide d'une fonction, le prix du billet le plus cher et le trajet concerné (gares de départ et d'arrivée).

Corrigé

Pb pour passer le tableau de structure en parametre à la fonction

```
#include <stdio.h> #include <stdlib.h> typedef struct BilletDeTrain BilletDeTrain; struct BilletDeTrain { int numBillet; float prixBillet; int numTrain; int numVoiture; int numPlace; char gareDepart[50]; char jourDepart[50]; char heureDep[10]; char gareArrivee[50]; char jourArrivee[50]; char heureArrivee[10]; }; void prixLePlusHaut(BilletDeTrain tableau[]); int main() { /* typedef struct BilletDeTrain BilletDeTrain; struct BilletDeTrain { int numBillet; float prixBillet; int numTrain; int numVoiture; int numPlace; char gareDepart[50]; char jourDepart[50]; char heureDep[10]; char gareArrivee[50]; char jourArrivee[50]; char heureArrivee[10]; };*/ BilletDeTrain voyage[5]; int i; for(i=0;i<5;i++) { printf("saisir le numero de billet\n"); scanf("%d",&voyage[i].numBillet); printf("saisir le prix du billet\n"); scanf("%f",&voyage[i].prixBillet); printf("saisir le numero de train\n"); scanf("%d",&voyage[i].numTrain); printf("saisir le numero de voiture\n"); scanf("%d",&voyage[i].numVoiture); printf("saisir le numero de place\n"); scanf("%d",&voyage[i].numPlace); printf("saisir la gare de depart\n"); scanf("%s",voyage[i].gareDepart); printf("saisir le jour de depart\n"); scanf("%s",voyage[i].jourDepart); printf("saisir l heure de depart **important de saisir sous format xxhxx**\n"); scanf("%s",voyage[i].heureDep); printf("saisir la gare d arrivee\n"); scanf("%s",voyage[i].gareArrivee); printf("saisir le jour d arrivee\n"); scanf("%s",voyage[i].jourArrivee); printf("saisir l heure d arrivee **important de saisir sous format xxhxx**\n"); scanf("%s",voyage[i].heureArrivee); return 0; } } void prixLePlusHaut(BilletDeTrain tableau[]) { float prixHaut=tableau[0].prixBillet; int cpt=0; int i; for(i=1;i<5;i++) { if(tableau[i].prixBillet>prixHaut) prixHaut=tableau[i].prixBillet;cpt=i; } printf("le billet le plus cher vaut: %f euros et concerne le voyage entre %s et %s",prixHaut,tableau[cpt].gareDepart,tableau[cpt].gareArrivee); }
```

Exercice 13

Énoncé

Créer ici un programme original utilisant les fonctions manipulant les chaînes de caractères (inclure le header *string.h*).

Corrigé

#

Exercice 14

Énoncé

Écrire le programme qui affiche le nombre de lettres dans un mot saisi au clavier puis le nombre de voyelles (a, e, i, o, u, y) qu'il possède.

Utiliser une fonction *NbVoyelles ()* qui retournera le nombre de voyelles.

Corrigé

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

int NbVoyelles (char *Mot);

int main() {
char Mot[50] ;

printf("Quel est votre mot ?\n");
scanf("%s",&Mot);
printf("Mot avec %d lettre(s) \n ",strlen(Mot));
printf("Mot avec %d voyelle(s) \n ",NbVoyelles(Mot));

return 0;
}

int NbVoyelles (char *Mot) {
int i, compteur = 0;

for (i=0 ; i<strlen(Mot);i++)        {
        if (Mot[i] =='a'||Mot[i] =='e'||Mot[i] =='i'||Mot[i] =='o'||Mot[i] =='u'||Mot[i] =='y')
                compteur++;
}
return compteur;
}
```

Exercice 15

Énoncé

Écrire un programme qui demande de saisir un mot et l'affiche de façon cryptée (toutes les lettres de l'alphabet doivent être décalées d'un caractère).

Exemple : "salut" deviendra : "tbmvu"

Conseil : penser au cas de la lettre z ou de Z… utiliser les codes ASCII.

Corrigé

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
int main()
{

   int i;
   int taille;
   char mot[50];
   char motCrypte[50];
   printf("entrez votre mot:");
   scanf("%s",mot);
   taille=strlen(mot);
   for(i=0;i<taille;i++)
   {
    if (mot[i]==122)
       motCrypte[i]=97; //si le code ascii est celui de z(122) on met celui de a(97)
    else if (mot[i]==90)
              motCrypte[i]=65; //si le code ascii est celui de Z (90) on met celui de A(65)
          else
             motCrypte[i]= ++mot[i] ;  //On place la lettre décalée dans motCrypte;
   }

 printf("\nMot crypte : \n");
   for(i=0;i<taille;i++)
   {
printf("%c",motCrypte[i]);
   }
  printf("\n\n ");

   return 0;
}
```

Exercice 16

Énoncé

Programme difficile à comprendre.

On veut afficher le rang où se trouve une sous-chaîne dans une première chaîne (si la chaîne n'est pas trouvée, l'information s'affiche).

Exemple : la sous-chaîne "ER" est au rang 3 de la chaîne "AZERTY".

Créer un programme principal qui demande de saisir une chaîne de caractères nommée *chaine* puis une sous-chaîne de caractères nommée *sous_chaine* à chercher dans la première. Ces chaînes pourront contenir 200 caractères au maximum.

Vous n'avez pas le droit d'utiliser des fonctions prédéfinies. Utiliser une fonction *Cherche* qui envoie *chaine* et *sous_chaine* et affiche le rang où la sous chaîne a été

trouvée (ou affiche chaîne non trouvée). *Cherche* fait appel à une deuxième fonction nommée *Identique* qui vérifie si la sous-chaîne est présente dans la chaîne. Utiliser les pointeurs (nous sommes en présence de chaînes de caractères !).

Bien réfléchir à l'architecture du programme avant de commencer à coder.

Corrigé

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

//prototypes
int Identique(char *chaine,char *sous_chaine);
void Cherche(char *chaine,char *sous_chaine);

int main()  {
int Longueur = 200;
char chaine[Longueur];
char sous_chaine[Longueur];

printf("chaine initiale ? ");
scanf("%s",&chaine);
printf ("sous-chaine a rechercher dans la chaine initiale ?");
scanf("%s",&sous_chaine);

Cherche(chaine,sous_chaine);

return 0;
}


void Cherche(char *chaine,char *sous_chaine)  {
int rang=1;

while (*chaine)
        {
        if (Identique(chaine,sous_chaine))
                {
                        printf("sous-chaine trouve en position %d\n",rang);
                        return;
                }

        chaine++;
        rang++;
        }

printf("Sous chaine non trouvee");
 }


int Identique (char *chaine,char *sous_chaine) {
```

```
while (*sous_chaine && *chaine)
{
        if (*sous_chaine != *chaine)
                return(0);

        else    {
                chaine++;
                sous_chaine++;
                }
}

return (!(*sous_chaine));
}
```

Explications sur l'exercice 16

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

//prototypes
int Identique(char *chaine,char *sous_chaine);
void Cherche(char *chaine,char *sous_chaine);

int main()  {
int Longueur = 200;
char chaine[Longueur];
char sous_chaine[Longueur];
printf("chaine initiale ? ");
scanf("%s",&chaine);
printf ("sous-chaine a rechercher dans la chaine initiale ?");
scanf("%s",&sous_chaine);
Cherche(chaine,sous_chaine);
return 0;
}

//Cherche envoie chaine et sous chaine à Identique et recommence tant que chaine
n'est pas vide en décalant d'un caractère dans chaine

void Cherche(char *chaine,char *sous_chaine)  {
int rang=1;

while (*chaine)    // tant qu'on n'est pas en fin de chaine...
        {
        // On teste si Identique renvoie 1 (sous chaine trouvée) : si oui, on affiche le
        rang et on sort de Cherche

        if (Identique(chaine,sous_chaine))
                {
                        printf("sous-chaine trouve en position %d\n",rang);
                        return;
```

```
            }

        //Identique n'a pas renvoyé 1 donc on n'est pas sorti de cherche donc on
        décale d'un cran dans chaine et on incrémente rang de 1
        chaine++;
        rang++;
        }   //fin de While

// on passe sur cette ligne si on est sorti de la boucle while, donc identique n'a jamais
renvoyé 1  donc la chaine n'a jamais été trouvée.
printf("Sous chaine non trouvee");
 }


int Identique (char *chaine,char *sous_chaine) {
// Identique renvoie 0 et sort immédiatement si les 2 chaines sont différentes
autrement renvoie 1

//tant que le caractère testé de sous_chaine et chaine est identique on boucle
while (*sous_chaine && *chaine)
{
        //si les caractères sont différents à cet endroit on renvoie 0 et on sort
        if (*sous_chaine != *chaine)
                return(0);

        else    {
                 //on se décale d'un cran dans chaine et sous_chaine car les caractères
                étaient identiques
                chaine++;
                sous_chaine++;
                }
} //fin de while

// on renvoie 1 car on est sorti de la boucle donc la sous chaine a été trouvée
return (!(*sous_chaine));
}
```