Programmation II

SMI S4

WWW.TALIB24.COM

L'OBJECTIF

Étudier les concepts avancés de la programmation avec le langage C.

WWW.TALIB24.COM

Pointeurs et tableaux

Tout tableau en C est en fait un pointeur constant. Dans la déclaration :

```
int tab[10];
```

tab est un pointeur constant (non modifiable) dont la valeur est l'adresse du premier élément du tableau. Autrement dit, `tab` a pour valeur `&tab[0]`. On peut donc utiliser un pointeur initialisé à `tab` pour parcourir les éléments du tableau.

WWW.TALIB24.COM

Pointeurs et tableaux 2

On accède à l'élément d'indice `i` du tableau tab grâce à l'opérateur d'indexation `[ ]`, par l'expression `tab[i]`. Cet opérateur d'indexation peut en fait s'appliquer à tout objet p de type pointeur. Il est lié à l'opérateur d'indirection * par la formule

```
p[i] = *(p + i)
```

Pointeurs et tableaux se manipulent donc exactement de même manière

WWW.TALIB24.COM

Pointeurs et tableaux 3

La manipulation de tableaux, et non de pointeurs, possède certains inconvénients du fait qu'un tableau est un pointeur constant. Ainsi

La création de tableaux dont la taille est une variable du programme n’est pas possible.

De même les tableaux bi-dimensionnels dont les lignes n'ont pas toutes le même nombre d'éléments.

Ces opérations deviennent possibles dès que l'on **manipule des pointeurs alloués dynamiquement**. Ainsi, pour créer un tableau d'entiers à n éléments où n est une variable du programme,

WWW.TALIB24.COM

Tableaux dynamiques

```
#include <stdlib.h>
main( )
{
   int n;
   int *tab;
      ...
   tab = (int*)malloc(n * sizeof(int));
   ...
   free(tab);
}
```

WWW.TALIB24.COM

ALLOCATION DYNAMIQUE

Les fonctions `malloc` et `calloc` de la bibliothèque <*stdlib.h*> nous aide à localiser et à réserver de la mémoire au cours d'un programme.

`malloc( <N> )` réserve une zone mémoire contiguë de <N> octets et retourne un pointeur sur le début du bloc réservé ou le pointeur `NULL` (la valeur zéro) s'il n'y a pas assez de mémoire.

Avec N un entier positif < 65535

WWW.TALIB24.COM

ALLOCATION DYNAMIQUE 2

Exemple:

Si on a besoin d'un bloc en mémoire pour un texte de 4000 caractères,

. Nous disposons d'un pointeur T sur **char** (**char *T**). Alors l'instruction:

T = malloc(4000); // ou bien **calloc(1000, 4)**

fournit l'adresse d'un bloc de 4000 octets libres et l'affecte à T. S'il n'y a plus assez de mémoire, T obtient la valeur zéro ou NULL.

Pour réserver de la mémoire à des données d'un type nous avons besoin de l'opérateur **sizeof**

WWW.TALIB24.COM

LA FONCTION calloc

La fonction `calloc` de la librairie stdlib.h a le même rôle que la fonction malloc mais elle initialise en plus l'objet pointé `*p` à `0` Sa syntaxe est : `calloc(nb-objets, taille-objets)`

Ainsi, si `p` est de type `int*`, l'instruction :

`p = (int*)calloc(N, sizeof(int));` est strict équivalente à:

```
p = (int*)malloc(N * sizeof(int));
for (i = 0; i < N; i++)
  *(p + i) = 0;
```

L'emploi de `calloc` est simplement plus rapide.

WWW.TALIB24.COM

TABLEAUX et CHAINES DE CARACTERES

En langage C, les chaînes de caractères sont des **tableaux de caractères**. Leur manipulation est donc analogue à celle d'un tableau à une dimension:

Déclaration: `char nom[dim];`

Exemple: `char texte[350];`

`Char Nom[20]= "Titane " , Prenom[25]= "Alfred";`

`Char information[ ]=" Il fait tres beau ";`

Le compilateur réserve (dim-1) places en mémoire pour la chaîne de caractères: En effet, il ajoute toujours le caractère NULL ('\0') à la fin de la chaîne en mémoire.

WWW.TALIB24.COM

Les chaînes de caractères

Affichage à l'écran:

On peut utiliser la fonction `printf` et le format `%s`:

```
char texte[10] = "BONJOUR";
printf("VOICI LE TEXTE: %s\n",texte);
```

On utilisera si possible la fonction <u>`puts`</u> non formatée:

`puts(texte);` est équivalent à `printf("%s\n",texte);`

Saisie:

On peut utiliser la fonction `scanf` et le format `%s`. Une chaîne étant un pointeur, on n'écrit pas le symbole `&`. On utilisera de préférence la fonction <u>`gets`</u> non formatée.

WWW.TALIB24.COM

Les chaînes de caractères 2

```
char texte[10];
printf("ENTRER UN TEXTE: ");
scanf("%s",texte);
```

est équivalent à `gets(texte);`

Remarque: `scanf` ne permet pas la saisie d'une chaîne comportant des espaces: les caractères saisis à partir de l'espace ne sont pas pris en compte (l'espace est un délimiteur)

Pour saisir une chaîne de type "il fait beau", il faut utiliser `gets`.

WWW.TALIB24.COM

Manipulation de chaînes

Pour utiliser les fonctions prédéfinies qui manipulent les chaines il faut charger la bibliothèque **string.h**

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

Parmi ces fonction on trouve:

-Mettre 2 chaines bout à bout

```
strcat( chaine1 , chaine2)
```

(le résultat sera dans chaine1)

-Mesurer la longueur d'une chaine

```
strlen(chaine)
```

('\0' non comptabilisé).

WWW.TALIB24.COM

La manipulation des chaînes 2

-Comparer 2 chaines:

`strcmp(chaine1, chaine2)` le résultat est:

0 si les chaînes sont identiques.

Positif si la chaîne1 est > à la chaine2 (l'ordre alphabétique)

Négatif si la chaîne1 est < à la chaîne2

-Copier une chaine dans une autre:

`strcpy(chaine1, chaine2)`

recopie chaine2 dans chaine1 et renvoie l'adresse de chaîne1.

(l'affectation = n'est pas possible entre les chaines)

WWW.TALIB24.COM

La manipulation des chaînes 3

-Recherche un caractère dans une chaîne :

```
strchr(chaine, caractère)
```

-Recherche une sous chaine dans une chaîne :

```
strstr(chaîne, sous-chaîne)
```

Ces fonctions renvoient l'adresse de l'information recherchée en cas de succès, sinon le pointeur NULL

WWW.TALIB24.COM

La manipulation des chaînes 3

Conversions Majuscules-minuscules:

```
strlwr(chaîne)
```

convertit tous les Majuscules en minuscules sans changer les autres caractères.

```
strupr(chaine)
```

convertit tous les minuscules en Majuscules

WWW.TALIB24.COM

Modularisation de programmes

- Pour des problèmes plus complexes, si on se limite à la fonction principale **main()** et les fonctions prédéfinies,
- on va se trouver avec de longues listes d'instructions, peu structurées et par conséquent peu compréhensibles. En plus souvent répéter les mêmes suites de commandes dans le programme.
- Gaspillage de mémoire et d'énergie.

WWW.TALIB24.COM

Modulisation de programmes 2

- La plupart des langages de programmation nous permettent de subdiviser nos programmes en sous-programmes, fonctions ou procédures plus simples et plus compacts.
- A l'aide de ces structures nous pouvons ***modulariser*** nos programmes pour obtenir des solutions plus élégantes et plus efficaces.

WWW.TALIB24.COM

Modularisation de programmes 3

Avantages:

- ***Meilleure lisibilité***
- ***Diminution du risque d'erreurs***
- ***Possibilité de tests sélectifs***
- ***Dissimulation des méthodes***
- ***Réutilisation de modules déjà existants***
- ***Simplicité de l'entretien***
- ***Favoriser le travail en équipe***

WWW.TALIB24.COM

LES FONCTIONS

En langage C les sous-programmes s'appellent des fonctions.

L'imbrication de fonctions n'est pas autorisée en C: une fonction ne peut pas être déclarée à l'intérieur d'une autre fonction. Par contre, une fonction peut appeler une autre fonction. Cette dernière doit être déclarée avant celle qui l'appelle.

Une fonction possède un et un seul point d'entrée, mais éventuellement plusieurs points de sortie (à l'aide de **return**).

Une variable connue uniquement d'une fonction ou de main( ) est une variable locale.

Une variable connue de tout le programme est une variable globale.

WWW.TALIB24.COM

FONCTIONS sans passage d'arguments et ne renvoyant rien au programme

Une fonction ne renvoyant rien au programme est une fonction de type **void.**

La déclaration a pour syntaxe:

void Nom_de_la_fonction ( )

ou bien

void Nom_de_la_fonction (void)

WWW.TALIB24.COM

```
#include <stdio.h>
#include <conio.h>
void BJR( void)        /* declaration de la fonction */
{
char NOM[20];
 printf("\n\n\n Introduisez votre nom : ");
 gets(NOM);
 printf("\n\n Bonjour %s !\n\n\n", NOM);
 printf("Merci de votre attention\n\n\n");
}
void main( )              /* programme principal */
{
BJR( );   /* appel de la fonction */
printf("Au revoir ");
getch( );
}
```

WWW.TALIB24.COM

LES FONCTIONS (variables locales)

```
#include <stdio.h>
#include <conio.h>
void carre( )        /* declaration de la fonction */
{    int  n , n2;   /* variables locales à carre */
printf("ENTRER UN NOMBRE: ");
scanf("%d",&n);
n2 = n*n ;
printf("VOICI  SON CARRE: %d\n",n2);   }
void main( )              /* programme principal */
{  carre( );   /* appel de la fonction */
printf("POUR CONTINUER FRAPPER UNE TOUCHE: ");
getch( );    }
```

Les variables n et n2 ne sont connues que de la fonction carré.

WWW.TALIB24.COM

LES FONCTIONS (variables globales)

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

int N;                  /* variable globale, connue de tout le programme */
void carre( )           /* declaration de la fonction */
{  int n2;   /* variable locale */
printf("ENTRER UN NOMBRE: ");
scanf("%d",&N);
n2 = N*N;
printf("VOICI SON CARRE: %d\n",n2);  }
void cube( )            /* declaration de la fonction */
{  int n3;  /* variable locale */
printf("ENTRER UN NOMBRE: ");
scanf("%d",&N);
n3 = N*N*N;
printf("VOICI SON CUBE: %d\n",n3);  }
```

WWW.TALIB24.COM

LES FONCTIONS (variables globales)2

```
void main( )            /* programme principal */
{
char choix;                     /* variable locale a main() */
printf("CALCUL DU CARRE TAPER 2\n");
printf("CALCUL DU CUBE TAPER 3\n");
printf("\nVOTRE CHOIX: ");
scanf("%c",&choix);
switch(choix)
          {   case '2':carre();break;
              case '3':cube();break;
               default : puts(“ Choix incorrect”);   }
printf("\n POUR CONTINUER FRAPPER UNE TOUCHE: ");
getch( );
}
```

La variable globale **N** est connue de tout le programme (fonctions et main( )).

La variable locale **choix** n'est connue que de main( ).

WWW.TALIB24.COM

FONCTION renvoyant une valeur au programme et sans passage d'arguments

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include <time.h>
int lance_de()  /* declaration de la fonction */
{  int test;  /* variable locale */
    test = random(6) + 1;
   return(test);   }
void main( )
{  int  resultat;
   randomize( );
   resultat = lance_de( );       /* resultat prend la valeur retournee par      la
fonction  (ou le sous-programme) */
   printf("Vous avez obtenu le nombre: %d\n",resultat);
getch( );
}
```

WWW.TALIB24.COM

FONCTIONS avec passage d'arguments

Type Nom_de_la fonction (liste paramètres)

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

int  puissance( int x , int y )
{  int i , p = 1 ;
for ( i = 1 ; i<=y ; i++) p = x*p ;
return( p ) ;    }
void   main( )
{  int a , b , res ;
printf("\n ENTRER la valeur a: ") ; scanf(" %d", &a ) ;
printf("\n ENTRER la valeur b: ") ; scanf(" %d" , &b )  ;
res = puissance(a , b ) ;
printf("\nA PUISS B = %d \n", res) ;    }
```

WWW.TALIB24.COM

FONCTIONS avec passage d'arguments
(par valeur)

Exemple: Fonction permettant d'échanger la valeur de deux variables

```
#include <stdio.h>
void ech(int x,int y)
{  int tampon;
    tampon = x;
    x = y;
    y = tampon;   }
 void main( )
{    int a = 5 , b = 8;
     ech(a,b);
     printf(“ a=%d\n ”, a) ;
     printf(“ b=%d\n ”, b) ;          }
```

PASSAGE DES PARAMETRES PAR VALEUR

WWW.TALIB24.COM

FONCTIONS avec passage d'arguments
(par adresse)

Fonction permettant d'échanger la valeur de deux variables

```
#include <stdio.h>
void  ech(int *x , int* y)
{ int tampon;
   tampon = *x;
   *x = *y;
   *y = tampon;  }

void main( )
{  int a = 5 , b = 8 ;
    ech(&a , &b ) ;
    printf(“ a=%d\n ”, a) ;
    printf(“ b=%d\n ”, b) ;      }
```

PASSAGE DES PARAMETRES PAR ADRESSE

WWW.TALIB24.COM

FONCTIONS avec passage d'un tableau à une dimension (par adresse)

- il est impossible de passer 'la valeur' de tout un tableau à une fonction, on fournit ***l'adresse du 1er élément du tableau***.

```
int Longueur(char *S)
{ int N;
for (N=0; *S != '\0'; S++)   N++;
return N; }
```

- Après les instructions,

```
char CH[ ] = "Bonjour !";
char *P;
P = CH;
```

- nous pouvons appeler la fonction **Longueur** définie ci-dessus par:

```
Longueur(CH)       Longueur(P)          /* résultat: 9 */
Longueur(&CH[4])   Longueur (P+2)   Longueur(CH+2)
```

WWW.TALIB24.COM

FONCTIONS avec passage d'un tableau à deux dimensions (par adresse)

- Pour passer une matrice float de type NxM en paramètre d'une fonction il existe:
- Une solution praticable consiste à faire en sorte que la fonction reçoive un pointeur (de type float*) sur le début de la matrice et de parcourir tous les éléments comme s'il s'agissait d'un tableau à une dimension N*M.

WWW.TALIB24.COM

FONCTIONS RECURSIVES

Le langage C autorise la récursivité des appels de fonctions. Il a deux aspects:

-**récursivité directe**: fonction fait appel à elle même,

-**récursivité croisée**: fonction fait appel à une autre fonction qui, à son tour appelle la fonction initiale (le cycle pouvant d'ailleurs faire intervenir plus de deux fonctions).

Exemple classique:

```
int fact(int n)
{  if (n>1) return(fac(n-1)*n);
                    else return(1);  }
```

WWW.TALIB24.COM

Structures

Le langage C autorise la déclaration de types particuliers: **les structures .**

Une structure est constituée de plusieurs objets de types différent.

Syntaxe:

```
struct nom {
        déclarations des champs ;
        } liste_variables ;
```

Les champs peuvent être des vecteurs, des variables d'autres structures, …. qui doivent être déclarés.

WWW.TALIB24.COM

Structures 2

Exemple:

```
1    struct identite
        {
2         int CIN;
3        char nom[25];
4         char prenom[20];

5         } Etudiant, Prof, liste[60];

6    Etudiant.nom= "Essaid" ;
```

Pour déclarer un type structuré, il faut utiliser le schéma suivant:

```
7    struct  identite  Agent, Comptable;
8    struct   identite    Adminstrateur[50];
```

WWW.TALIB24.COM

Structures 3

Initialisation des structures

Les structures s'initialisent de la même manière que les tableaux.

```
1    struct  X{
2            float Note; int Coef;
3            char Matiere[10];
4          } A = { 16.0 , 5 , " Math" };
```

Utilisation:

On accède à une composante par:

NOM_VARIABLE.NOM_CHAMPS

gets (Etudiant.nom) ;

printf (" Le professeur: %c . %s\n " , Prof.prenom[0] , Prof.nom) ;

for(i = 0 ; i < 60 ; i++) printf ("%d \t %s\n" , liste[i].CIN , liste[i].nom) ;

WWW.TALIB24.COM

Déclaration de type

Le langage C permet de définir de nouveaux types pour déclarer des variables dans la suite du programme, il faut utiliser ***typedef***

Déclaration: typedef type_de_base nouveau_nom;

Ceci permet de donner un nom à un type donné, mais ne crée aucune variable.

Exemple:

```
typedef   int   entier ;
typedef   float   mes_reels ;
 typedef   struct   personne {
                char nom[25] ;
                float  longueur ;
                } OUVRIER ;
entier  i , j ;
mes_reels  X , delta, Y2 ;
OUVRIER  masson ;
 entier Tab[60] ;
```

WWW.TALIB24.COM

Branchement inconditionnel

Instruction **break**:

Permet de quitter une boucle ou de sortir d'une instruction de test comme dans switch. Elle renvoie à la première instruction qui suit la boucle ou le test.

Instruction **continue**:

Elle permet de passer à l'itération suivante dans la boucle.

```
for( i = 0 ; i < 10 ; i++ )
{
   if ( i == j )    continue ;
   ……..   ;
}
```

WWW.TALIB24.COM

Branchement inconditionnel 2

Instruction **goto :**

Permet le branchement en un point quelconque du programme.

syntaxe : go to étiquette;

Où étiquette fait référence à une instruction étiquetée

Exemple:

```
for( i=0 ; i<10 ; i++)
{
 scanf ( " %c" , &T[i]) ;
if  T[i]== 'N'  goto   sortie;
}
sortie :  puts ( "FIN de SAISIE") ;
```

WWW.TALIB24.COM

Branchement inconditionnel 3

Instruction **exit :**

Un programme C peut être interrompu par l'instruction exit.

Syntaxe: exit(expression);

Où expression est un entier.

Instruction **return :**

Permet de sortir de la fonction actuelle. Elle permet aussi de rendre la valeur résultat de la fonction.

WWW.TALIB24.COM

LES FICHIERS

Un fichier est un ensemble d'informations stockées sur une mémoire de masse (disque dur, clé USB , CD-ROM, DVD, …).

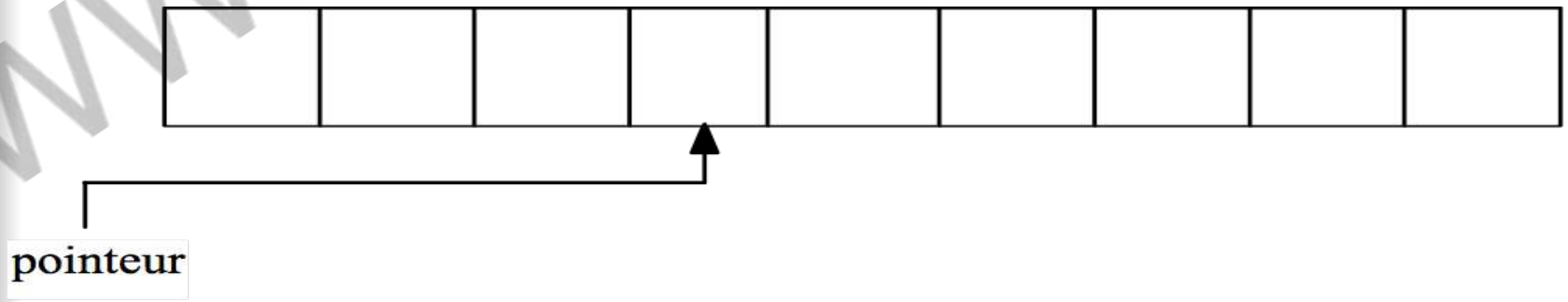
En C, un fichier est une suite d'octets. Les informations contenues dans le fichier ne sont pas forcément de même type (un char, un int, une structure ...)

On distingue généralement deux types d'accès:

Accès séquentiel

Accès direct

Un **pointeur** fournit l'adresse d'une information quelconque.



WWW.TALIB24.COM

LES FICHIERS 2

Il existe d'autre part deux façons de coder les informations stockées dans un fichier :

1- En BINAIRE :

Fichier dit "binaire", les informations sont codées telles que. Ce sont en général des fichiers de nombres. Ils ne sont pas listables.

2- En ASCII :

Fichier dit "texte", les informations sont codées en ASCII. Ces fichiers sont listables. Le dernier octet de ces fichiers est EOF (caractère ASCII spécifique).

WWW.TALIB24.COM

MANIPULATION DES FICHIERS

Le langage C ne distingue pas les fichiers à accès séquentiel des fichiers à accès direct, certaines fonctions de la bibliothèque livrée avec le compilateur permettent l'accès direct. Les fonctions standards sont des fonctions d'accès séquentiel.

1 - Déclaration:

FILE *fichier ;

/* majuscules obligatoires pour FILE */

2 - Ouverture:

FILE *fopen(char *nom , char *mode) ;

WWW.TALIB24.COM

Ouverture des fichiers

fopen prend en paramètres deux chaînes de caractères

nom: celui figurant sur le disque, exemple: " C:\essai.txt "

mode (pour les fichiers TEXTES) :

" **r** " lecture seule

" **w** " écriture seule (destruction de l'ancienne version si elle ∃)

" **w+** " écriture (en autorisant la modification)

" **r+** " lecture/écriture d'un fichier existant (mise à jour), pas de création d'une nouvelle version.

" **a+** " lecture/écriture d'un fichier existant (mise à jour), pas de création d'une nouvelle version, le pointeur est positionné à la fin du fichier.

WWW.TALIB24.COM

Ouverture des fichiers 2

mode (pour les fichiers BINAIRES) :

" **rb** " lecture seule

" **wb** " écriture seule (destruction de l'ancienne version si elle ∃)

" **wb+** " lecture/écriture (destruction ancienne version si elle ∃)

" **rb+** "lecture/écriture d'un fichier existant (mise à jour), pas de création d'une nouvelle version.

" **ab+** " lecture/écriture d'un fichier existant (mise à jour), pas de création d'une nouvelle version, le pointeur est positionné à la fin du fichier.

A l'ouverture, le pointeur est positionné au début du fichier (sauf "a+ " et " ab+")

Exemple : **FILE *fichier ;**

fichier = fopen(" c :\essai.dat ", " rb ") ;

WWW.TALIB24.COM

Fermeture, suppression… des fichiers

3 - Fermeture:

int fclose(FILE *fichier);

Il faut toujours fermer un fichier à la fin d'une session.

Exemple : **FILE *fichier ;**

```
fichier = fopen(" c :\essai.txt ", " rb ") ;
/* Ici instructions de traitement */
fclose(fichier) ;
```

WWW.TALIB24.COM

Fermeture, suppression… des fichiers

4 - Destruction:

int remove(char *nom);

Exemple : **remove(" c :\essai.txt ") ;**

5 - Renommer:

int rename(char *oldname, char *newname);

6-Positionnement du pointeur au début du fichier

void rewind(FILE *fichier);

WWW.TALIB24.COM

Fichier Output

7- Ecriture dans le fichier:

int putc(char c, FILE *fichier);

Retourne EOF en cas d'erreur.

Exemple : **putc('A', fichier) ;**

int putw(int n, FILE *fichier);

de même, n de type int, le pointeur avance du nombre de cases correspondant à la taille d'un entier (4 cases en C standard).

Retourne n si l'écriture s'est bien passée.

WWW.TALIB24.COM

Fichier Output

int fputs(char *chaîne, FILE *fichier);

de même avec une chaîne de caractères, le pointeur avance de la longueur de la chaine ('\0' n'est pas rangé dans le fichier). Retourne EOF en cas d'erreur.

Exemple : **fputs(" BONJOUR ! ", fichier) ;**

WWW.TALIB24.COM

Fichier Output 2

int fwrite (void *p , int tailleBloc , int nbBloc , FILE *fichier) ;

p de type pointeur, écrit à partir de la position courante du pointeur fichier **nbBloc** X **tailleBloc** octets lus à partir de l'adresse p. Le pointeur fichier avance d'autant.

Le pointeur p est vu comme une adresse, son type est sans importance.

Retourne le nombre de blocs écrits.

Exemple: tailleBloc = 4 (taille d'un entier en C),

nbBloc=3 , écriture de 3 valeurs.

```
int tab[10] ;
fwrite(tab , sizeof(int) , 10 , fichier) ;
```

WWW.TALIB24.COM

Fichier Output 3

int fprintf(FILE *fichier, char *format, liste d'expressions) ;

réservée plutôt aux fichiers ASCII.
Retourne EOF en cas d'erreur

Exemples: **fprintf(fichier,"%s","il fait beau") ;**
fprintf(fichier,"%d", n) ;
fprintf(fichier,"%s%d","il fait beau", n) ;

WWW.FALIB24.COM

```
#include<stdio.h>
main()
{ FILE  *fp ;
char filename[20] ;
printf( "Entrer un nom de fichier : ") ;
gets(filename) ;
/*  Ouverture du fichier en écriture seulement  */
fp= fopen(filename , "w") ;
if (fp) == NULL )
  { printf(" impossible d'ouvrir le fichier  %s \n", filename);
   exit(1);}
  /* Ecriture dans le fichier  */
  fprintf(fp , " \nCe fichier se nomme : %s \n" , filename) ;
  /* Fermeture du fichier  */
  if (fclose (fp)  == EOF) printf( "fermeture impossible \n" ) ;
  }
```

WWW.TALIB24.COM

Fichier Input

8 - Lecture du fichier:

int getc (FILE *fichier) ;

- lit un caractère,
- retourne un entier n ou bien EOF en cas fin de fichier atteinte et en cas d'erreur;
- le pointeur avance d'une case.

Exemple: **char c ;**

c = (char)getc(fichier) ;

int getw(FILE *fichier);

- lit et retourne un entier ou EOF
- le pointeur avance de 4 cases (la taille d'un entier).

Exemple: **int n ;**

n = getw(fichier) ;

WWW.TALIB24.COM

Fichier Input 2

char *fgets (char *chaine , int n , FILE *fichier);

lit n-1 caractères à partir de la position du pointeur et les range dans chaine en ajoutant '\0'.

int fread(void *p , int tailleBloc , int nbBloc , FILE *fichier);
analogue à fwrite en lecture.
Retourne le nombre de blocs lus, et 0 à la fin du fichier.

int fscanf(FILE *fichier, char *format, liste d'adresses);
analogue à fprintf en lecture.

WWW.TALIB24.COM

Fichier : les erreurs

9 - Gestion des erreurs:

fopen retourne le pointeur NULL si erreur (Exemple: impossibilité d'ouvrir le fichier).

fgets retourne le pointeur NULL en cas d'erreur ou si la fin du fichier est atteinte.

int feof(FILE *fichier) retourne 0 tant que la fin du fichier n'est pas atteinte.

int ferror(FILE *fichier) retourne 1 si une erreur est apparue lors d'une manipulation de fichier, 0 dans le cas contraire.

WWW.TALIB24.COM

Fichier : déplacement du pointeur

10 - Fonction particulière aux fichiers à acces direct:

int fseek(FILE *fichier , int nb , int direction)
déplace le pointeur de nb cases à partir de direction.
Valeurs possibles pour direction :

0 : à partir du début du fichier.
1 : à partir de la position courante du pointeur.
2 : en arrière, à partir de la fin du fichier.

Retourne 0 si le pointeur a pu être déplacé ;

WWW.TALIB24.COM

Types énumérés

- Dans C on peut décrire complètement la liste de toutes les valeurs possibles que peut prendre une variable.

- On parle dans ce cas de **types énumérés**

Ils sont définis par :

enum nom_du_type {

{ident_symbole [= expr_const_entière],}

} {ident_var [= valeur_initiale],};

WWW.TALIB24.COM

Exemples

– les mois de l'année (janvier, février...);
– les noms des modules enseignés (Math, Anglais, arabe, physique ...);
– les marques de voiture;
– les indications d'état civil (célibataire, marié, divorcé...)

<u>Déclaration:</u>

enum jour { lundi, mardi, mercredi, jeudi, vendredi, samedi, dimanche } ;

enum jour j , min, max ;

WWW.TALIB24.COM

Utilisation d'une variable d'un type énuméré

Ou bien :

enum {dimanche, lundi, mardi, mercredi, jeudi, vendredi, samedi } j , min , max ;

j = dimanche ;

if (j == lundi) printf("Monday") ;

enum jour { lundi, mardi, mercredi = 20, jeudi, vendredi, samedi, dimanche } ;

Alors mardi aura la valeur 1,

Mercredi la valeur 20 et jeudi la valeur 21.

Il n'existe pas de fonctions d'entrés-sorties pour les types énuméré

WWW.TALIB24.COM

Exemple

```
enum sexes { feminin , masculin } ;
struct personnes {
        char  nom[20] , prenom[20] ;
        int  age ;
        enum sexes   sex ;
        int   note_finale[4];
 } eleve1 = {"Said", "Nachit", 22, masculin, {19,14 , 20 , 13 } };
```

- On a défini trois choses:
  - un type énuméré sexes,
  - une structure personnes et
  - une variable eleve1 du type struct personnes.

WWW.TALIB24.COM

Unions

- Les unions sont de nouvelles structures de données assez proches des structures. Elles permettent de stocker dans une variable des données de **types différents.**

  Par contre, elles ne peuvent contenir qu'**une seule donnée à la fois.**

```
union exemple {
        char car_val; /* contient un caractère */
        int ent_val; /* ou un entier */
        float flo_val; /* ou un flottant */
        char tex_val[10]; /* ou du texte */   }
```

WWW.TALIB24.COM

Exemple

```
union exemple data ;
data.flo_val = 123.425;
printf ("%f", data.flo_val); /* est valide */
printf ("%d", data.ent_val); /* n'a pas de sens car l'union
contient un float.*/
```

Une variable de type union pouvant avoir à priori une valeur de type quelconque, il faut en général associer à des variables un **indicateur (par exemple de type numéré) qui** permette de trouver quel champ est pertinent à un moment donné. En général, on associe cet indicateur et l'union elle-même dans une **structure.**

WWW.TALIB24.COM

```
enum type_valeur {caractere, entier, flottant, chaine};
struct   donnee_complexe
 {  enum    type_valeur   type_val ;
    union {
       char car_val ;
       int ent_val ;
       float flo_val ;
       char tex_val[10] ;
   } valeur ;
 } nd ;
nd.type_val = entier ;
nd.valeur.ent_val = 6 ; /* affectation d'une valeur */
```

WWW.TALIB24.COM

La directive #include

Elle permet d'incorporer dans le fichier source le contenu d'un autre fichier.

Elle possède deux syntaxes voisines :

#include <nom-de-fichier>

nom-de-fichier peut être généralement un fichier en-tête de la librairie standard (stdio.h, math.h,...) qui doit être dans un ou plusieurs répertoires systèmes définis par l'implémentation (typiquement /usr/include/)

#include " nom-de-fichier "

destinée aux fichiers créés par l'utilisateur ces fichiers doivent exister dans le même répertoire que le fichier source

WWW.TALIB24.COM

La directive #define

Elle permet de définir des constantes symboliques ou des macros

Lorsque le processeur lit une ligne du type :

#define nom reste-de-la-ligne

- il remplace dans toute la suite du source toute occurrence de nom par reste-de-la-ligne.

Aucune contrainte sur reste-de-la-ligne.

```
#define Max 100
#define BEGIN {
#define END }
#define NB_LIGNES 24
#define NB_COLONNES 80
#define TAILLE_MATRICE NB_LIGNES * NB_COLONNES
```

WWW.TALIB24.COM

Les macros en langage C

Une macro (avec paramètres) se définit de la manière suivante :

#define nom(liste-de-paramètres) corps-de-la-macro où liste-de-paramètres est une liste d'identificateurs séparés par des virgules.

Par exemple:

#define Max(a,b) (a > b ? a : b)

#define CARRE (a) a*a

WWW.TALIB24.COM

la directive #error

- Elle a la syntaxe:
  #error chaine
  La rencontre de cette commande provoquera l'émission d'un message d'erreur comprenant la chaine.
- Cette commande a pour utilité de capturer à la compilation des conditions qui font que le programme ne peut pas s'exécuter sur cette plate-forme.

WWW.TALIB24.COM