Université Aboubekr BELKAID
كلية العلوم – تيجاني هدام
Faculté des Sciences
قسم الإعلام الآلي
Département d'informatique

Module : Compilation **Année universitaire 2022-2023**
Classe : L3- Informatique
Réalisé par : Mr MERZOUG Mohamed
Mr ETCHIALI Abdelhak

TP N° 02

Exercice 1 : DECOUPAGE D'UNE PHRASE

Ecrire un programme qui permet de découper une phrase et d'enregistrer les mots de la phrase dans une ***Liste Chaînée (ou une File).***

Les mots de la phrase peuvent être séparés par des espaces ou par des séparateurs (, ; . ! ? -)

Indice : Créer une fonction qui teste si un caractère appartient à la liste des caractères séparateurs pour découper une phrase.

Rappel :

1. Structure d'une liste chaînée :

```
typedef struct Mot_De_La_Phrase {
        char     Mon_mot[20];
        struct   Mot_De_La_Phrase*  Adresse_Mot_suivant;
} Mot;
```

2. Insertion à la fin de la liste chaînée :

```
Mot* Inserer_a_la_fin(Mot* les_mots_de_laphrase, char* mot){
        // si la liste chainee est vdie
        if(les_mots_de_laphrase==NULL){
                // cration de la liste / reservation de memoire pour le 1er element
                les_mots_de_laphrase=malloc(sizeof(Mot));
                // c'est le 1er et dernier element (donc le prochain element est NULL)
                les_mots_de_laphrase->Adresse_Mot_suivant=NULL;
                // copier le mot
                strcpy(les_mots_de_laphrase->Mon_mot,mot);
                // retourner l'addresse du 1er element
                return les_mots_de_laphrase;}
        // si la liste chainée n'est pas vide affecter le 1er élément a un pointeur temporaire
        Mot* dernier_mot=les_mots_de_laphrase;
        // tanque le pointeur suivant n'est pas NULL c a d on n'est pas arrivé au dernier
        while(dernier_mot->Adresse_Mot_suivant!=NULL){
                // passer a l'element suivant
```

```
            dernier_mot=dernier_mot->Adresse_Mot_suivant; }
        // créer un nouvel élément a la fin de la liste
        dernier_mot->Adresse_Mot_suivant=malloc(sizeof(Mot));
        // copier le mot
        strcpy(dernier_mot->Adresse_Mot_suivant->Mon_mot,mot);
        // c'est le 1er et dernier element (donc le prochain element est NULL)
        dernier_mot->Adresse_Mot_suivant->Adresse_Mot_suivant=NULL;
        // retourner l'addresse du 1er element
        return les_mots_de_laphrase; }
```

3. Insertion à la fin de la liste chaînée :

```
    Mot* Inserer_au_debut(Mot* les_mots_de_laphrase,char* mot){
        // creer un nouvel element
        Mot* premier_mot=malloc(sizeof(Mot));
        // copier la chaine de caractere
        strcpy(premier_mot->Mon_mot,mot);
        // affecter le pointeur du suivant au debut de la liste principale
        premier_mot->Adresse_Mot_suivant=les_mots_de_laphrase;
        // retourner l'@ du 1er element
        return premier_mot; }
```

Exercice 2 : FILE & PILE ET MANIPULATION DE FICHIER

- Créer un fichier texte nommé ***« etudiant.txt »*** puis insérer le texte suivant :

> Je suis un(e) étudiant(e)
> En 3 Année Licence Informatique
> Département d'Informatique
> Année Universitaire 2022-2023
> Université Abou Bekr Belkaid Tlemcen

- Ecrire un programme qui permet de :

    - Créer une pile.
    - Ouvrir le fichier ***« etudiant.txt »*** en mode lecture.
    - Lire le contenu du fichier ***« etudiant.txt »*** ligne par ligne.
    - Empiler chaque ligne du fichier dans la pile.
    - Afficher la pile.

Fonctions utiles

1- Manipulation de fichiers :

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

int i=0;
int main(){

    FILE *file;     /** Déclaration d'une variable file  de
                        type FILE **/
   char chaine[100]  = {0}; //variable chaine de caractère
   char c;

   /** Créer et Ouvrir un nouveau fichier test.txt en mode
                            Ecriture (w)**/
   file = fopen("test.txt","w");

   /**en cas d'erreur **/
   if(file == NULL){
      printf("Erreur");
      return 0;
   }

   printf("Entrer une chaine: ");

 /** Lire la chaine de taille max 100   à partir de l'écran**/
   fgets(chaine,100,stdin);

   /** Ecrire la chaine saisie dans le fichier test.txt **/
   fprintf(file,"%s",chaine);


   /** Fermer le Fichier test.txt **/
   fclose(file);


   /**Lecture d'un fichier Ligne par Ligne et Affichage sur
                            Ecran **/

/** Ouvrir le Fichier existant test.txt en mode Lecture (r) **/

   file = fopen("test.txt","r");

   if(file == NULL){
      printf("Erreur");
      return 0;
   }
```

```
    /** Copier le contenu de chaque ligne du fichier
        Texte.txt dans la variable chaine
        (100 caractères Max)   **/

while( fgets (chaine,100, file)!=NULL ) {
       /**  Affichage sur Ecran de la ligne copiée
            dans la variable chaine **/
                puts(chaine);
                    /**
                    ou bien
                fprintf(stdout,"%s",chaine);
                    ou bien
                printf("%s",chaine);
                    **/
   }
   /** Fermer le Fchier test.txt**/
   fclose(file);

   /**Lecture d'un fichier Carectere par Caractere et
            Affichage sur Ecran **/
 /** Ouvrir le Fichier existant test.txt en mode Lecture
                    (r) **/
   file = fopen("test.txt","r");

   if(file == NULL){
      printf("Erreur");
      return 0;
   }

  /** Copier chaque caractère du fichier Texte.txt dans
            la variable c **/
   while ((c = fgetc(file)) != EOF){
   /**Affichage sur Ecran caracère par caractère **/
                putc(c,stdout);
                    /**
                Ou bien putchar(c);
                    Ou bien
                fprintf(stdout,"%c",c);
                    ou bien
                printf("%c",c);
                    **/

      }
      /** Fermer le Fchier test.txt**/
      fclose(file);
}
```

Les piles

```
typedef struct cellule{
        int valeur;
        struct cellule * next;
}Element;
typedef struct pile{
        Element* sommet;
}Pile;
Pile* InitialiserPile (Pile* p){
        p=malloc(sizeof(Pile));
        p->sommet=NULL;
        return p;}
Pile* Empiler (Pile* p,int v){
   Element* nouveauElement= malloc (sizeof (Element));
   nouveauElement->valeur=v;
   nouveauElement->next=p->sommet;
   p->sommet=nouveauElement;
   return p;
}
Pile* Depiler (Pile* p){
        if (p->sommet==NULL)
                return p;
        Element* e;
        e=p->sommet;
        p->sommet=p->sommet->next;
        free(e);
        return p;
}
int SommetPile (Pile* p){
        if (p->sommet==NULL)
                return -9999; /** code pile vide */
        else
                return p->sommet->valeur;
}
void AfficherPile (Pile* p){
        Element *ep=p->sommet;
        while (ep!=NULL){
                printf("\n |%d|",ep->valeur);
                ep=ep->next;
        }
        printf("\n");
}
Pile* ViderPile (Pile* p){
        while (p->sommet!=NULL){
                Element *e=p->sommet;
                p->sommet=p->sommet->next;
                free(e);
        }
        return p;
}
```

Les files

```
typedef struct cellule{
        int valeur;
        struct cellule * next;
}Element;
typedef struct file{
        Element* sommet;
        Element* queue;
}File;
File* InitialiserFile(File* f){
        f=malloc(sizeof(File));
        f->sommet=NULL;
        f->queue=NULL;
        return f;}
File* Enfiler (File* f,int v){
   Element* nouveauElement= malloc (sizeof (Element));
   nouveauElement->valeur=v;
        nouveauElement->next=NULL;
        if (f->sommet==NULL){
                f->sommet=nouveauElement;
                f->queue=nouveauElement;
                return f;}
        f->queue->next=nouveauElement;
        f->queue=nouveauElement;
        return f;}
File* Defiler (File* f){
        if (f->sommet==NULL)
                return f;
        else if(f->sommet==f->queue){
                free(f->sommet);
                f->sommet=NULL;
                f->queue=NULL;
                return  f;}
        Element* e;
        e=f->sommet;
        f->sommet=f->sommet->next;
        free(e);
        return f;}
int SommetFile (File* f){
  if(f->sommet==NULL) return -9999;  /** code file vide */
 else    return f->sommet->valeur;}
void AfficherFile (File* f){
        Element *ep=f->sommet;
        while (ep!=NULL){
                printf(" |%d| ",ep->valeur);
                ep=ep->next;}}
File* ViderFile (File* f){
        while (f->sommet!=NULL){
                Element *e=f->sommet;
                f->sommet=f->sommet->next;
                free(e);}
        f->sommet=NULL;
        f->queue=NULL;
        return f;}
```