Département d'Informatique

Solution TD-TP N° 2

Problème traitant de la validité d'une expression mathématique contenant trois délimiteurs '{,}', '(,)' et '[,]' en utilisant le concept de la pile statique

```
/* directives de préprocesseur : inclure les bibliothèques standards */
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <conio.h>
/* directives de préprocesseur : définir une constante*/
#define MAX 20
*/ définition de la structure pile statique */
struct Pile {
        char tab[MAX];
        int sommet;
};
Typedef struct Pile pile;
/* création et initialisation d'une pile vide*/
pile creer(){
   pile p;
   p.sommet = -1;
   return p;
}
```

```
/*fonction qui teste si une pile est vide*/
int vide(pile * p){
   return(p->sommet==-1);
}
/*fonction qui teste si une pile est pleine*/
int pleine(pile * p){
   return(p->sommet==MAX-1);
}
/*fonction qui empile un symbole ouvrant dans la pile*/
void empile(char a, pile * p){
   p->sommet=p->sommet+1;
   p->tab[p->sommet]=a;
}
/*fonction qui dépile de la pile un symbole ouvrant et le retourne*/
char depiler(pile * p){
   char a = p->tab[p->sommet];
   p->sommet-=1;
   return a;
}
/* fonction qui reçoit un délimiteur fermant et retourne son équivalent ouvrant*/
/* default : return a ; la fonction retournera le même caractère passé en argument*/
/*résultat de cette fonction qui est un délimiteur ouvrant est comparé avec le sommet de la pile si non vide*/
char ouvrant(char a){
   switch(a){
      case '}': return'{';break;
```

```
    case ')': return '('; break;
    case ']': return '['; break;
    default: return a;
      }
}
/*programme principal traitant si une expression mathématique entrée au clavier, avec trois délimiteurs '{,}', '(,)' et '[,]', est correcte ou pas*/
int main(){
int valide=1 ;  /*Expression saisie au clavier est supposée valide */
int pos=0; /*compteur qui va parcourir l'expression mathématique initialisée à 0*/
  char symb, s; /* variables intermédiaires*/
  char expr[80]; /*déclarations de l'expression mathématique comme une chaine de caractères*/
  pile p; /*déclaration d'une variable de type pile (enregistrement : tableau statique et sommet*/
  p = creer(); /* création d'une pile vide*/
  printf("Donner une expression mathématique:\n");
  scanf("%s",expr);
  symb=expr[pos]; /* la variable, symb, reçoit le premier caractère de l'expression*/
  do{  /* parcourir l'expression mathématique en utilisant la boucle do{} while ; */
    if(symb=='[' || symb=='{' || symb=='('){  /*tester si symb est un délimiteur ouvert*/
      if(pleine(&p)){ /*Si la pile est pleine, on ne peut pas insérer et on sort du programme */
        printf("Pile est pleine!!\n");
        break;}
      else /*Si la pile n'est toujours pleine, on y insère le délimiteur ouvrant*/
        empile(symb, &p);
    }
    else if (symb==']' || symb=='}' || symb==')'){ /*tester si symb est un délimiteur fermer*/
```

```
        /*dans ce cas on dépile le sommet de la pile si non vide et on le compare au délimiteur ouvert correspondant au délimiteur ouvert rencontré dans l'espression*/

        if(vide(&p)){
/* si pile vide, on déclare expression non valide car un fermant qui ne possède pas son ouvrant auparavant*/

            valide=0;

       }

        Else /* si pile non vide, on compare le délimiteur ouvrant dépilé de délimiteur ouvrant retourné par la fonction ouvrant*/

            s=depiler(&p);

            if(ouvrant(symb)!=s){ /*on déclare l'expression non valide si c'est deux délimiteurs sont différents*/

                valide=0;

 }

      }

      pos+=1; /* incrémenter le compteur pour passer au caractère suivant de l'expression*/

      symb=expr[pos]; /*caractère suivant de l'expression à traiter*/

   }

   While (valide==1 && pos<strlen(expr)); /*la boucle tourne tant que l'expression est valide et n'est pas toute parcourue*/

   if(valide==0) /*Si valide est à 0, l'expression est fausse*/

      printf("expression incorrecte\n");

   else /* sinon, l'expression est vraie*/

      printf("expression correcte\n");

} /*fin du programme principale*/
```