**Correction Exercice 1 :**

L’intégration de nouvelles opérations applicables sur une SD exige :

Modification apporté à l’interface

Modification apporté à l’implémentation

**application : cadre objet abstrait**

objet abstrait :pile.h

pile.c

**modification apporté au pile.h :**

/\* on garde inchangé la déclaration des opérations fondamentales : creer\_pile, vide…\*/

unsigned nb\_elements(void) ;

void effacer(void) ;

void change\_sommet(int) ;

**modification apporté au pile.c :**

unsigned nb\_elemV1(void){

unsigned nb ;

nb=0;

while(!vide())

{

nb++;

depiler();

}

return(nb);

}

static unsigned nb;/\*nombre d’éléments dans la pile\*/

unsigned nb\_elements(void)

{

return nb ;

}

void creer pile(void)

{...

nb=0;

}

void empiler(int info)

{…

nb++;

}

void depiler(void)

{…

nb--;

}

void change\_sommet(int nouvelle\_valeur)

{

assert( !vide()) ;

sommet🡪 cle=nouvelle\_valeur ;

}

void effacer(void)

{

while( !vide())

depiler() ;

}

**Remarque :**

La réalisation de l’opération effacer est basée sur des opérations fondamentales (ici vide et dépiler).Il s’agit d’une opération dite élaborée obtenue par composition des opérations fondamentales. Par contre, change\_sommet manipule des éléments liés à la représentation physique de la pile. Si la structure de donnée pile est vide alors (implication) nb\_element=0. De même, si le nombre d’éléments de la pile est nul alors cette pile est vide.

**Correction Exercice 2 :**

Idée : une parenthèse fermante est associée à la dernière parenthèse ouvrante rencontrée d’où LIFO = >PILE.

#include<stdio.h>

#include<string.h>

#include<stdlib.h>

#include ’’pile.h’’ /\*objet abstrait \*/

char \*sous\_chaine(char ch[], unsigned debut, unsigned fin) {

/\* rend la sous\_chaine comprise entre ch[debut et ch[fin] \*/

unsigned i ; /\* entre debut et fin \*/

char \*sch ;

char \*p ;

sch=(char \*) malloc (fin – debut +2) ;

p=sch ;

for(i=debut ;i<=fin ;i++) {

\*p=ch[i] ;

p++ ;

}

\*p= ‘\0’ ;

return(sch) ;}

void main( ) {

char ch[80] ; /\* expression proposée \*/

char \*sch ; /\* expression courant entre parenthèses \*/

unsigned i ;

printf(‘‘exp=’’) ;

gets(ch) ;

for(i=0, creepile() ;i<strlen(ch) ;i++) {

if(ch[i]= = ‘(’)

empiler(i) ;

else if(ch[i]= = ‘)’){

sch=sous\_chaine(ch,dernier(),i) ;

puts(sch) ;

depiler() ;

}

} /\* fin for \*/

} /\* fin main \*/

**Correction Exercice 3 :**

#include <stdio.h>

#include « pile.h »/\*objet abstrait\*/

void main(void)

{

unsigned nb ;

unsigned chiffre ;/\*0…9\*/

printf(“nb=”) ;

scanf(“%u 2 ”, &nb) ;

creer\_pile() ;

do

{

chiffre=nb%10 ;/\* reste de la division entière de nb par 10\*/

empiler(chiffre) ;

nb=nb/10 ;

}

while(nb !=0) ;

/\*utilisation\*/

while( !vide())

{ printf(’’%u\n ” ,dernier()) ;

depiler() ;

}

}/\*fin main\*/