IUT "A" Paul SABATIER
Dpt Informatique

AP => compléments du langage C : Pgc2

TP : chaînes de caractères dynamiques

Algorithmes des sous-programmes

---

procédure creerChaineVide (**mise-a-jour** ch <ChaineDyn>)

Glossaire

début

```
ch.nbCar ← 0 ;
ch.ptrCar ← NULL ;
```

fin

procédure libererChaine (**mise-a-jour** ch <ChaineDyn>)

Glossaire

début

```
desallouer(ch.ptrCar ) ;
ch.nbCar ← 0 ;
ch.ptrCar ← NULL ;
```

fin

fonction longueurChaine (**entrée** ch <ChaineDyn>)

retourne <Entier>

Glossaire

début

```
retourner (ch.nbCar);
```

fin

procédure convertirChaine (**mise-a-jour** ch1 <ChaineDyn>, **entrée** ch2 < PointeurCar >)

Glossaire

lgCh2 <Entier> : longueur de la chaîne ch2 ;
ptrAlloc <PointeurCar> : pointeur sur le bloc d'octets alloué dynamiquement
pour stocker la nouvelle chaîne;

début

```
lgCh2 ← longueur(ch2) ;
si ( lgCh2 = 0 ) alors
        ptrAlloc ← NULL ;
sinon
        ptrAlloc ← allouer( lgCh2 ) ;
        si ( ptrAlloc = NULL ) alors
                lever ( ERR_ALLOC ) ;
        finsi ;
        copierBlocOctets( ptrAlloc, ch2, lgCh2 ) ;
finsi ;
si ( ch1.ptrCar != NULL ) alors
        desallouer(ch1.ptrCar ) ;
finsi ;
ch1.nbCar ← lgCh2 ;
ch1.ptrCar ← ptrAlloc ;
```

fin

procédure copierChaine (**mise-a-jour** ch1 <ChaineDyn>, **entrée** ch2 < ChaineDyn >)

Glossaire

lgCh2 <Entier> : longueur de la chaîne ch2 ;
ptrAlloc <PointeurCar> : pointeur sur le bloc d'octets alloué dynamiquement
pour stocker la nouvelle chaîne;

début

```
lgCh2 ← ch2.nbCar ;
si ( lgCh2 = 0 ) alors
        ptrAlloc ← NULL ;
sinon
        ptrAlloc ← allouer( lgCh2 ) ;
        si ( ptrAlloc = NULL ) alors
                lever ( ERR_ALLOC ) ;
        finsi ;
        copierBlocOctets( ptrAlloc, ch2.ptrCar, lgCh2 ) ;
finsi ;
si ( ch1.ptrCar != NULL ) alors
        desallouer(ch1.ptrCar ) ;
finsi ;
ch1.nbCar ← lgCh2 ;
ch1.ptrCar ← ptrAlloc ;
```

fin

procédure collerChaine (**mise-a-jour** ch1 <ChaineDyn>, **entrée** ch2 <ChaineDyn>)

Glossaire

lgCh1 <Entier> : longueur de la chaîne ch1 ;
lgCh2 <Entier> : longueur de la chaîne ch2 ;
ptrAlloc <PointeurCar> : pointeur sur le bloc d'octets alloué dynamiquement
pour stocker la nouvelle chaîne;

début

```
lgCh2 ← ch2.nbCar ;

si ( lgCh2 > 0 ) alors
        lgCh1 ← ch1.nbCar ;
        ptrAlloc ← allouer( lgCh1 + lgCh2 ) ;
        si ( ptrAlloc = NULL ) alors
                lever ( ERR_ALLOC ) ;
        finsi ;
        copierBlocOctets( ptrAlloc, ch1.ptrCar, lgCh1 ) ;
        copierBlocOctets( ptrAlloc+lgCh1, ch2.ptrCar, lgCh2 ) ;

        si ( ch1.ptrCar != NULL ) alors
                desallouer(ch1.ptrCar ) ;
        finsi ;

        ch1.nbCar ← lgCh1+lgCh2 ;
        ch1.ptrCar ← ptrAlloc ;

finsi ;
```

fin

fonction comparerChaine (**entrée** ch1 <ChaineDyn>, **entrée** ch2 <ChaineDyn>)

retourne <Entier>

Glossaire

lgMin <Entier> : longueur de la chaîne la plus courte ;
result <Entier> : résultat de la comparaison ;
p1 <PointeurCar> : pointeur sur le caractère courant de ch1 ;
p2 <PointeurCar> : pointeur sur le caractère courant de ch2 ;
cpt <Entier> : compteur de caractères ;
fini <Booléen> : indicateur de contrôle de la boucle ;

début

```
si  ( ch1.nbCar <= ch2.nbCar ) alors
        lgMin  ←  ch1.nbCar ;
sinon
        lgMin  ←  ch2.nbCar ;
finsi ;
cpt ← lgMin ;
p1 ←  ch1.ptrCar ;
p2 ←  ch2.ptrCar ;
fini ← FAUX ;
tantque non fini faire
   si ( cpt > 0 ) alors
          si ( p1↑ = p2↑ ) alors
             p1 ←  p1 +  1;
             p2 ←  p2 + 1;
             cpt ← cpt - 1;
       sinon
             fini ← VRAI ;
       finsi ;
   sinon
       fini ← VRAI ;
   finsi ;
fintantque ;
si ( cpt != 0 ) alors
   si ( p1↑ > p2 ↑) alors
         result ← 1;
   sinon
         result ← -1;
   finsi ;
sinon
   si  ( ch1.nbCar = ch2.nbCar ) alors
         result ← 0;
   sinon
          si  ( ch1.nbCar > ch2.nbCar ) alors
         result ← 1;
       sinon
         result ← -1;
       finsi ;
   finsi ;
```

finsi ;

retourner(result) ;

fin

procédure extraireSousChaine (**mise-a-jour** ch1 <ChaineDyn>, **entrée** ch2 <ChaineDyn>,
entrée p <Entier>, **entrée** n <Entier>)

Glossaire

lgCh2 <Entier> : longueur de la chaîne ch2 ;
lgSousCh2 <Entier> : longueur de la sous chaîne à extraire ;
ptrAlloc <PointeurCar> : pointeur sur le bloc d'octets alloué dynamiquement
pour stocker la sous chaîne à extraire ;

début

```
lgCh2 ← ch2.nbCar ;
si ( (p<1) ou (p>lgCh2) ou (n<0) ) alors
        lever( ERR_PARAM ) ;
finsi ;

lg SousCh2 ← lgCh2 – p + 1 ;
si ( lgSousCh2 > n ) alors
        lgSousCh2 ← n ;
finsi ;

si ( lgSousCh2 = 0 ) alors
        ptrAlloc ← NULL ;
sinon
        ptrAlloc ← allouer( lgSousCh2 ) ;
        si ( ptrAlloc = NULL ) alors
                lever ( ERR_ALLOC ) ;
        finsi ;
        copierBlocOctets( ptrAlloc, (ch2.ptrCar + p – 1), lgSousCh2 ) ;
finsi ;

si ( ch1.ptrCar != NULL ) alors
        desallouer(ch1.ptrCar) ;
finsi ;

ch1.nbCar ← lgSousCh2 ;
ch1.ptrCar ← ptrAlloc ;
```

fin

procédure entrerChaine (**mise-a-jour** ch <ChaineDyn>, **entrée** f <FILE *>,
entrée n <Entier>)

Glossaire

ptrAlloc <PointeurCar> : pointeur sur le bloc d'octets alloué dynamiquement
pour stocker la chaîne lue dans le fichier ;
err <PointeurCar> : pointeur retourné par la fonction **fgets** ;

début

si (n<0) **alors**
lever(ERR_PARAM) ;
finsi ;

si (n = 0) **alors**
libererChaine(ch) ;
sinon
ptrAlloc ← allouer(n + 1) ;
si (ptrAlloc = NULL) **alors**
lever (ERR_ALLOC) ;
finsi ;

err ← fgets(ptrAlloc, n+1, f) ;
si (err = NULL) **alors**
desallouer(ptrAlloc) ;
lever(ERR_FIC) ;
finsi ;
convertirChaine(ch, ptrAlloc) ;
desallouer(ptrAlloc) ;
finsi ;
fin

procédure sortirChaine (**entrée** ch <ChaineDyn>, **entrée** f <FILE *>)

Glossaire

nbBloc <Entier> : nombre de blocs écrits dans le fichier ;

début

nbBloc ← fwrite(ch.ptrCar, ch.nbCar, 1, f) ;
si (nbBloc != 1) **alors**
lever(ERR_FIC) ;
finsi ;

fin