DOC TECHNIQUE

src1.c

permet d'écrire '*', qui permet de mettre fin a la saisie.

correspond a la taille maximale des tableaux dans le programme.

type tableau de taille TAILLE_MAX+1, Tout les tableaux du programme partent de ce type, et sont des caractères.

DESCRIPTION

Ce programme nous demande des caractères et les stocks dans un tableaux. Qu'il comparent avec d'autres caractères dans un autre tableau. Il renvoie aussi le nombre

de caractères dans le tableau. Il permet aussi
de vérifier la présence d'un caractère dans le tableau de
base. Il peut aussi comparer 2 tableau pour vérifier
si les 2 tableaux sont identiques ou non.

FONCTIONS ET PROCEDURES

void lireMot(tMot m):
\detail deux paramètres char sont pris en compte,
le fonction utilise un scanf dans une boucle qui
s'arrête uniquement si on rentre une étoile au lieu
d'un caractère
sinon la boucle peut s'arrêter si le mot est trop grand
notez qu'on ne peut pas revenir en arrière, une faute
de frappe et c'est tant pis, plus qu'a relancer.
\param m c'est le tableau utilisé

void affiche(tMot m):
\brief Elle permet de d'afficher les caractères du tableau
\detail Elle affiche les mots représenter par les tableaux,
puisqu'elle affiche a la suite sans utiliser de backslash n (sauf
a la fin).
\param tMot m c'est le tableau qu'on affiche.

int longueur(tMot m):
\brief Elle permet de renvoyer la longueur du tableau
\param m c'est le tableau utilisé
\return renvoie la longueur du tableau
\detail Elle affiche le nombre de caractères présent dans le
tableau, 15 étant son max,
la fonction arrête de lire quand elle croise une étoile ,
puisqu'aucun caractère ne figure après.

void testelongueur() // va nous permettre de tester la
fonction longueur:
\brief Teste la fonction longueur avec plusieurs tableaux. Elle
sert uniquement a tester.

bool estDans(tMot m, char c):
\brief Permet de vérifier la présence d'un caractère dans le
tableau.
\detail possède 3 variable (dont 2 de boucles)
la boucle s'arrête quand l'un des variants de boucle
atteint la taille maximum, ou qu'on trouve le mot.
Auquel cas, la dernière variable res deviendra true.
\param m C'est le tableau utilisé.
\param c C'est le caractère que l'on recherche dans le
tableau.
\return true si le caractère est présent, sinon false.

int compare(tMot m1, tMot m2):
\brief Compare deux tableaux afin de savoir s'ils sont

identiques.
\detail possède 2 variants de boucle, fonctionne un peu
comme estDans, mais prends 2 paramètres, n'utilise
pas estDans et compare 2 tableaux.
res reverra true si les 2 tableaux sont identiques,
Sinon, renvoie false.
\param m1 Premier tableau.
\param m2 Deuxième tableau.
\return true si les tableaux sont identiques, sinon false.

int main():
\brief Fonction principale du programme.
\detail le programme vérifie en sois toutes les fonctions.
Des tableaux sont fournis avec des mots pour vérifier
La fonction compare, mais on remplis aussi un tableau
grâce a la fonction LireMot, qui s'affiche avec
affiche, et nous renvoie la longueur du mot, et la
présence d'un caractère ou non dans le tableau

DEFINE

TAILLE_MAX 15 : correspond a la taille maximale des
tableaux dans le programme.

src2.c

Taille maximale d'une chaine quand on souhaite ordonner
ses caractères.

caractère de la chaîne

pointeur sur l'élément suivant

Structure d'un élément d'une liste chaînée.

Un élément est composé d'un caractère et d'un pointeur sur
le caractère suivant
(ou qui vaut NULL si c'est le dernier). Cette structure de

données est récursive. Tandis que le type chaine permet de déclarer des chaines de caractères en tant que liste chaînée.

Type tableau de TMAX caractères.

la j'avoue je sais pas

DESCRIPTION

Ce programme propose plusieurs opérations de manipulation
de chaînes de caractères où les chaînes sont implémentées par des listes chainées de caractères.Programme de manipulation de chaînes de caractères.

FONCTIONS ET PROCEDURES

int main():
\brief Programme principal.
\detail Le programme principal traite un exemple qui teste quelques fonctions :
il crée deux chaines, les affiche, puis ordonne la deuxième et l'affiche.
\return Code de sortie du programme (0 : sortie normale).

void init(chaine * ch):
\brief Fonction qui initialise la chaine.
\detail Met à NULL la chaine passée en paramètre.
Cette fonction permet d'initialiser une chaîne. Elle doit être appelée à chaque utilisation de chaîne.
\param ch : paramètre de sortie qui représente la chaine à initialiser.

bool estV(chaine ch):
\brief Fonction qui indique si une chaine est vide.
\param ch : paramètre d'entrée qui représente la chaine à tester.
\return true si la chaine est vide, false sinon.

void ajT(chaine * ch, char c):
\brief Ajout d'un caractère en tête.
\detail Après une allocation dynamique, ajoute le nouvel élément au début de la liste chaînée.
\param ch : paramètre d'entrée/sortie qui représente la chaine à modifier.
\param c: paramètre d'entrée qui représente le caractère à ajouter.

void ajQ(chaine * ch, char c):
\brief Ajout d'un caractère en queue.
\detail Après une allocation dynamique, ajoute le nouvel élément à la fin de la liste chaînée.

\param ch : paramètre d'entrée/sortie qui représente la chaine à modifier.
\param c: paramètre d'entrée qui représente le caractère à ajouter.

int nbC(chaine ch):
\brief Calcule la taille d'une chaîne.
\detail Parcours complet de la liste chaînée pour compter le nombre de ses éléments.
\param ch : paramètre d'entrée qui représente la chaine dont on veut connaître la taille.
\return le nombre de caractères présents dans la chaîne.

void aff(chaine ch):
\brief Affiche une chaîne à l'écran.
\detail Parcours complet de la liste chaînée pour afficher chacun de ses éléments (le champ
'lettre').
\param ch : paramètre d'entrée qui représente la chaine à afficher.

bool ide(chaine ch1, chaine ch2):
\brief Indique si deux chaînes sont identiques.
\param ch1 : paramètre d'entrée qui représente la première chaîne.
\param ch2 : paramètre d'entrée qui représente la deuxième chaîne.
\return true si les deux chaînes sont identiques, false sinon.
\detail Parcourt parallèlement les deux listes chaînées, caractère par caractère,
tant que les caractères correspondants sont les mêmes.

void inv(chaine ch1, chaine * ch2):
\brief Inverse l'ordre des caractères dans une chaîne.
\param ch1 : paramètre d'entrée qui représente la chaîne à inverser.
\param ch2 : paramètre de sortie qui représente la nouvelle chaîne construite.
\detail Parcourt la première liste et ajoute chaque élément en tête de la deuxième.
ATTENTION : ch2 doit être initialisée par la procédure/fonction appelante.

bool pal(chaine ch1):
\brief Indique si une chaîne est un palindrome.
\param ch1 : paramètre d'entrée qui représente la chaîne à tester.
\return true si les deux chaînes sont identiques, false sinon.
\detail Une chaîne est un palindrome si elle est identique à son inverse.

bool app(chaine ch, char c):
\brief Indique si un caractère fait partie d'une chaîne.
\param ch : paramètre d'entrée qui représente la chaîne à tester.
\param c : paramètre d'entrée qui représente le caractère à tester.
\return true si le caractère est présent dans la chaîne, false sinon.
\detail Effectue une recherche séquentielle de c dans la liste chaînée ch.

bool supprimer(chaine * ch, char c):
\brief Supprime un caractère d'une chaîne.
\param ch : paramètre d'entrée/sortie qui représente la chaîne à modifier.
\param c : paramètre d'entrée qui représente le caractère à enlever.
\return true si la suppression a eu lieu (ie si le caractère est présent), false sinon.
\detail ATTENTION : cette fonction ne supprime que la première occurrence du caractère.

bool ana(chaine ch1, chaine ch2):
\brief Indique si deux chaînes sont anagrammes l'une de l'autre.
\param ch1 : paramètre d'entrée qui représente la première chaîne.
\param ch2 : paramètre d'entrée qui représente le deuxième chaîne.
\return true si les deux chaînes sont anagrammes l'une de l'autre, false sinon.
\detail La fonction teste si les deux chaînes sont composées des mêmes caractères. Elle parcourt la liste ch1.
A chaque element, elle enlève dans ch2 le caractère courant correspondant * de ch1 s'il existe.
S'il n'existe pas, le résultat est faux, arrêt du parcours. Mais si à la fin du parcours * de ch1,
ch2 est vide alors c'est OK.
ATTENTION, dans cette fonction la chaine ch2 est modifiée, donc plus utilisable à la fin.

int LvT(chaine ch, typTab t):
\brief Recopie les caractères d'une chaîne dans un tableau.
\param ch : paramètre d'entrée qui représente la chaîne à copier.
\param t : paramètre d'entrée/sortie qui représente le tableau.
\return La taille de la chaîne ou -1 si cette taille dépasse TMAX.
\detail La fonction est utilisée par la fonction ord(). Elle copie les caractères d'une Liste

dans un tableau A CONDITION que la taille de la chaîne soit inférieure ou égale à TMAX.

void TvL(typTab t, chaine * ch, int N):
\brief Recopie les caractères d'une chaîne dans un tableau.
\param t : paramètre d'entrée/sortie qui représente le tableau à recopier.
\param ch : paramètre de sortie qui représente la chaîne résultat.
\param N: paramètre d'entrée qui représente le nombre de caractères à recopier.
\see ord()
\detail La fonction est utilisée par la fonction ord(). Elle ajoute un à un les caractères du tableau
à une liste (ajout en queue).
ATTENTION : ch doit être initialisée par la procédure/fonction appelante.

void trT(typTab t, int N):
\brief Trie les éléments d'un tableau de caractères.
\param t : paramètre d'entrée/sortie qui représente le tableau à trier.
\param N: paramètre d'entrée qui représente le nombre de caractères du tableau.
\detail La fonction utilise le tri par insertion. Elle ordonne les éléments de manière croissante.

void ord(chaine * ch):
\brief Ordonne les caractères d'une chaîne par ordre croissant.
\param t : paramètre d'entrée/sortie qui représente la chaîne à ordonner.
\detail La fonction utilise les fonctions LvT, trT et TvL pour respectivement :
- copier les caractères de la liste dans un tableau,
- trier le tableau,
- et recopier les caractères du tableau dans la liste.
ATTENTION : la chaine doit faire moins de TMAX caractères.

DEFINE

TMAX 20 : Taille maximale d'une chaine quand on souhaite ordonner ses caractères.

correspond au nombre maximum de ligne que le tableau peut nous présenter.

correspond au nombre maximum de colonne que le tableau peut nous présenter.

correspond au nombre de ligne & colonne d'un carré sur le tableau du sudoku.

type tableau de taille NB_LIGNE, par NB_COLONNE. C'est ce qui permet de définir la taille de la grille du sudoku.

DESCRIPTION

Ce programme permet de jouer a un jeu de sudoku, Il lit un document avec une grille
et lit les placements de valeur par des commandes. Il y une vérification pour empêcher
de rentrer n'importe quel valeur a n'importe quel endroit, et enfin le jeu se fini
quand la grille est entièrement rempli.

FONCTIONS ET PROCEDURES

void saisir(int* valeur):
\brief Elle permet de vérifier que la valeur est correcte
\param valeur il s'agit de la valeur qui sera vérifié dans la procèdure
\detail Quand on l'utiliser elle va servir de scanf
Puis elle vérifiera que la valeur donnée respecte les contraintes
ne dépasse pas le tableau, est un entier.

void chargerGrille(tGrille g):
\brief Elle permet de récupérer des grilles données de Sudoku
\param g il s'agit de la grille qu'on utilise tout au long du jeu. Et donc celle qu'on charge.
* Cette procèdure permet de rentrer des grilles préchoisies, par des fichiers.

void afficherGrille(tGrille grille):
\param grille il s'agit de la grille qu'on utilise tout au long du jeu. ici elle permet de remplir les cases.

\brief Cette procèdure permet d'afficher la grille.
\detail C'est elle qui fais les colonnes (\see NB_COLONNE).
C'est elle qui fais les lignes. (\see NB_LIGNE)
C'est elle qui affiche les bornes de la grille.
C'est elle qui remplis les cases avec des nombres au 1er tour.
(\see void chargerGrille(tGrille g))
C'est aussi elle qui affiche les points donc.
Sans elle, le programme n'apparaîtrai pas comme un jeu.

bool possible(tGrille grille, int lig, int col, int val):
\brief Vérifie si la valeur et la position rentré en paramètre respectent les règles du sudoku.
* \param grille il s'agit de la grille qu'on utilise tout au long du jeu. Ici elle ne change pas
\param lig La ligne choisie.
\param col La colonne choisie.
\param val La valeur choisie.
\return Renvoie true si le placement ne pose pas problème, sinon renvoie false.
\detail cette fonction récupère la ligne et la colonne de la valeur qu'on veut rentrer dans la grille
Elle vérifie a partir de ces données si la même valeur ne se trouve pas dans la même colonne, ou la même
ligne. Ensuite elle vérifie si elle ne se trouve pas dans la même case, en utilisant parcours_l & parcours_c
Ils permettent de toujours selectionner le début d'une case, en fonction de la colonne et la ligne rentrée.
Si l'une de ces 3 méthodes de recherche trouve la même valeur quand celle rentrée en paramètre, possible
nous renvoie "false" et un message nous expliquant ou est l'erreur.

bool remplie(tGrille grille1):
\fn bool remplie(tGrille grille1)
\brief Vérifie si la grille du sudoku est remplie ou non.
* \param grille1 il s'agit de la grille qu'on utilise tout au long du jeu. Ici elle ne change pas
\return Renvoie true si la grille est remplie, sinon renvoie false.
\detail remplie parcours la grille entière et compte le nombre de case non remplie.
Si elle ne trouve aucune case non remplie, alors ça veut dire que le tableau est plein, elle nous renvoie alors "true".

int main():
\brief Il s'agit du programme principal, sur lequel le jeux se déroule.
* \detail Le programme principal appelle chaque fonction et procèdures. Elle possède une boucle
qui se répéte jusqu'à ce que la procèdure remplie renvoie false

Elle demandera la ligne, la colonne puis la valeur, vérifie si c'est possible puis
met ou non la valeur dans la grille. Quand la grille est remplie, le jeu est fini, et
un message s'envoie.

DEFINE

NB_LIGNE 9 : correspond au nombre maximum de ligne que le tableau peut nous présenter.

NB_COLONNES 9 : correspond au nombre maximum de colonne que le tableau peut nous présenter.

NB_SQUARE 3 : correspond au nombre de ligne & colonne d'un carré sur le tableau du sudoku.

Pied de page

SAE1.03

Groupe 1D1 / équipe FEUR

Mateo MORVAN Victor ROUÉ Yann CHESNEL Baptiste LAÎNÉ