

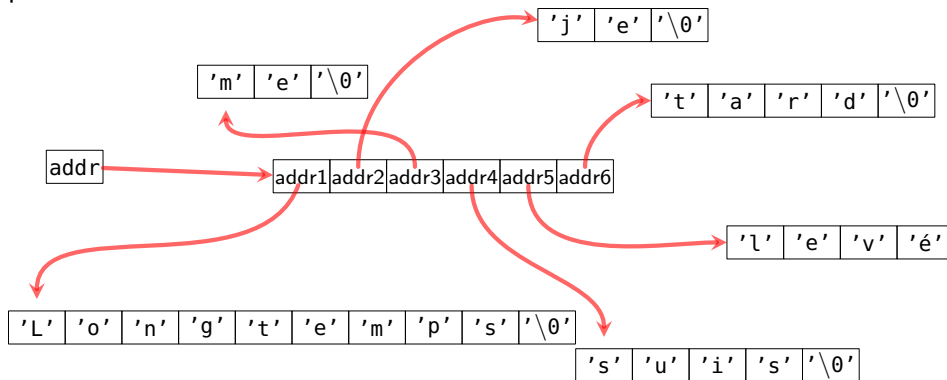
Algo/Prog – DUT 1

- 0 Introduction
- 0 Appeler un sous-programme
- 0 Définition de procédures et fonctions
- 1 Conditionnelle (vu en TP)
- 0 Instructions complexe (boucles)
- 0 Variables, types
- 1 Représentation des données en mémoire
- 0 Boucles imbriquées et tableaux à doubles entrées
- 8 Pointeurs, etc
- 9 `struct`

- 0 Introduction
- 0 Appeler un sous-programme
- 0 Définition de procédures et fonctions
- 1 Conditionnelle (vu en TP)
- 0 Instructions complexe (boucles)
- 0 Variables, types
- 1 Représentation des données en mémoire
- 0 Boucles imbriquées et tableaux à doubles entrées**
- 8 Pointeurs, etc
- 9 `struct`

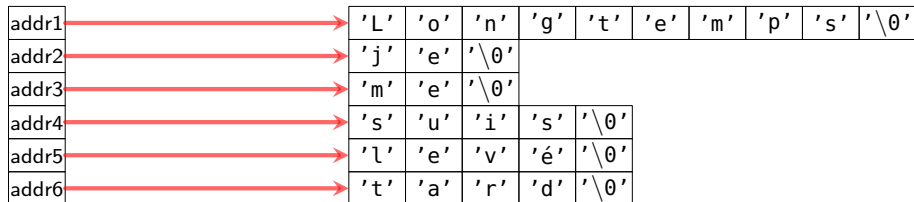
Tableau de tableaux ?

possible :



- voir TP 4 (`char ** t = lireFichierParMots("toto.txt",&x);`).

Illusion : bien « rangé »



- pratique d'y penser comme ça
- mais en fait pas de « localité » mémoire
- en C : tableau à double entrée \neq tableau de tableaux
- on préférera le tableau de tableaux si rectangulaire

Tableau à double entrée I

```
int t[4][3]; // exemple 1: tableau de 4 lignes chacune de longueur 3
int u = // exemple 2: tableaude 3 lignes, chacune de longueur 4.
{
    {3, 5, 6 ,7},
    {7, 4, 3 ,2},
    {1, 9, 45 ,7}
};
```

- toutes les lignes : même longueur
- longueur des lignes connue *statiquement* (à la compilation)
- en fait *un* tableau à une dimension

Tableau à double entrée II

```
int t[4][3]; // exemple 1: tableau de 4 lignes chacune de longueur 3
t[1][2] = 2;
```

- $t[1][2] = 2;$ équivalent à $t[1 * 3 + 2] = 2;$
- $t[i][j]$ équivalent à $t[i * largeur + j]$
- la largeur doit être connue à la compilation

Largeur connue à la compilation

- largeur déclarée dans les signatures de fonctions :

```
void f(int t[][3]){  
    ...  
}
```

- Si plus de 2 dimensions : toutes les dimensions sauf la dernière

```
void f(int t [][][4][5]){  
    ...  
}
```


Morpion grill 3x3

```
/* Affiche les neufs caractères comme une grille dans le terminal:  
   3 lignes de 3 caractères */  
void afficheGrille(char t[][3]){  
    int i,j;  
    for(i=0; i<3; i++){  
        for(j=0; j<3; j++){  
            printf("%c",t[i][j]);  
        }  
        printf("\n");  
    }  
}
```

Pattern typique :

- La boucle imbriquée (démonstration)

Morpion grill 3x3

```
/* Affiche les neufs caractères comme une grille dans le terminal:  
   3 lignes de 3 caractères */  
void afficheGrille(char t[][3]){  
    int i,j;  
    for(i=0; i<3; i++){ ← boucle extérieure  
        for(j=0; j<3; j++){ ← boucle intérieure  
            printf("%c",t[i][j]);  
        }  
        printf("\n");  
    }  
}
```

Pattern typique :

- La boucle imbriquée (démonstration)

Démo boucles imbriquées

DÉMO