

3 Variables, types et types structurés

© J. Morinet Lambert , M. Cadot, L. Pierron
version :13/10/04

Notion de variable

- Définition
 - c'est une **place** en mémoire pour stocker de l'information
 - on y accède par son adresse repérée par un identificateur
 - unique, selon règles de syntaxe : un nom de variable significatif
 - association d'une **adresse**, d'un **identificateur** et d'une **valeur**
- Déclarer une variable
 - nommer et **réserver** une place en mémoire (contenant) pour y placer la valeur (contenu) : **obligatoire** avant toute utilisation
 - différent des mathématiques : connotation **temporelle**
 - à un instant cette valeur est unique
 - change lors de calcul ou traitement effectués par un programme

Valeur d'une variable

- Initialisation
 - ATTENTION : espace mémoire réutilisé contenant une configuration binaire précédente pas toujours compatible parfois 00, parfois ff ou toute autre valeur
 - valeur par défaut dans certains langages : s'en assurer sinon **initialiser**
 - dans le programme au moment de la déclaration
 - exemples `int i=5; float x=3.42;`
- Saisie
 - depuis un périphérique ou un fichier on transmet la valeur
 - `printf("explication sur la saisie"); scanf("%i %f", &var1, &var2);`
- Calcul
 - affecter le résultat d'un calcul d'expression ou de fonction : `x=pqr(16);`

Notion de type

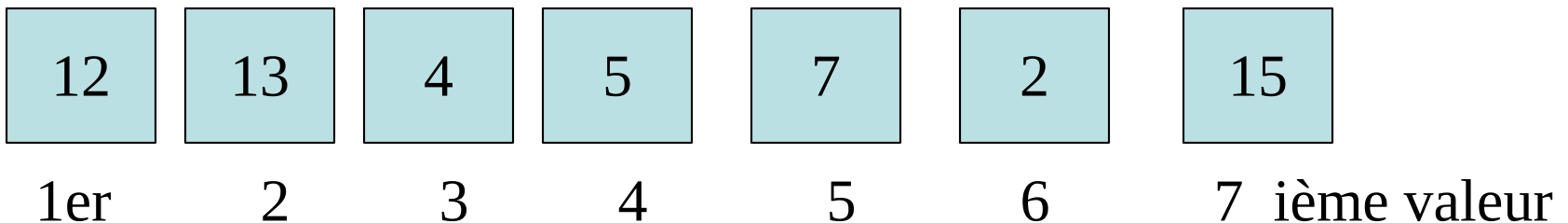
- caractérise une catégorie de variables
 - la place nécessaire pour stocker l'information
 - le mode de codage
 - un ensemble de fonctions et procédures pour les gérer
- déclaration de type de variable
 - catégorie d'objets de même taille
 - qui nécessiteront une organisation et des fonctions identiques
 - ne correspond pas à la réservation de place en mémoire
 - pas d'existence informatique tant que pas de variable déclarée de ce type
- le transtypage
 - possibilité de forcer le changement de type d'une variable en un autre type
 - compatibilité de taille
 - automatique lors de certains calculs (entier *réel -> réel)

Types prédéfinis

- Simples : une seule information
 - char,
 - int, float
- Agrégats :
 - regroupe plusieurs informations
 - tableau :
 - des informations semblables de même type en nombre fini
 - structure ou enregistrement :
 - des informations disparates de types différents
 - définis par l'utilisateur
 - fixe la taille, le type de l'élément de tableau..
 - précise les noms des champs et leur nature

Type structuré : le tableau

- **Exemples**
- tableau de 7 entiers
- tableaux de 20 caractères
- tableaux de 10 réels....



- accéder à un élément : par un indice i de type entier
- valeur d'un élément du tableau : $T[i]$

Définition

<TypeSimple> <NomTableau>[<Dimension>];

- Formelle
 - fonction qui associe une valeur à un indice
- Déclaration en C
 - il faut connaître le nombre d'éléments et le type des éléments

```
#define taille 20                /* nombre d'éléments */  
typedef int tableau[taille];     /* type tableau */  
tableau T1,T2;                  /* déclaration de variables de type  
    tableau */
```

la place réservée pour T1 est de : 4octects (pour un entier)*20

T1[0] représente le premier élément du tableau

T1 représente l'adresse du premier élément du tableau (&T1[0])

Problèmes en C

- Attention en C
 - premier élément d'indice 0 : T[0]
 - dernier élément d'indice taille -1
 - déclaration : `int T[5]={5, 8, 3 , 4 , 2 } ;`
 - T[0] vaut 5, T[1] vaut 8, T[2] vaut 3, T[3] vaut 4, T[4] vaut 2,
- Que se passe-t-il si on tape à côté ?
 - que vaut T[7] ??? quels contrôles ?
- Calcul automatique de la taille à la déclaration
 - `int T1 [ ] = {3,5,7,6} ;` /* 4 éléments indice 0 à 3 */
 - `int T2[ 5 ]={ 4,2,3} ;` /* complète avec 0 , 0 */
 - `int T3[4]={3, 6 ,9 ,5 , 2, 5} ;` /* plantage */



Utilisation de tableaux

- Initialisation
 - déclaration : `int T[6]={5, 8, 3 , 4 , 2 } ;`
 - saisie : saisir chaque élément du tableau
`for (i=deb; i<taille; i++) { /* saisir le ième élément */ }`
- Modification
 - d'un élément `T[i] = valeur;` ou `T[i] = expression;`
 - de tous les éléments : avec une itération pour chaque élément...
- Parcours
 - calcul du maximum, calcul du minimum
 - calcul de la somme, calcul de la moyenne
 - trier etc.

Parcours de tableau

exemple une chaîne : char S[10];

- **Deux méthodes**
 - Utiliser un **indice de parcours** : i entier
 - Utiliser un **pointeur** de parcours : pc pointeur sur l'élément (caractère) char *pc;
- Algorithme- Programmation

```
/* 1 Init :se placer sur le premier élément*/
/* i=0 ;*/
/* 2 Iter: Examiner chaque élément du tableau */
for (i=0; i<10; i++)

    printf("%c ",S[i]);

/* i++ ;*/

printf("\n");
```

```
char S[10];

/* 1 Init :se placer sur le premier élément*/
/* 2 Iter: Examiner chaque élément du tableau */
/* 3 Traiter l'élément courant : */
/* 4 Passer au suivant : */
/* 5 Fin :*/
```

```
pc= S;
while (*pc != '\0') {
    printf("%c ",*pc );
    pc++; }

printf("\n");
```

Tableau à plusieurs dimensions

- Tableaux à plusieurs "entrées"
 - Tableau de tableaux, Matrice
 - Exemple : emploi du temps EPT[jour, heure] -> le cours
- Déclaration
 - `int T[3][4]; /*matrice 3 lignes 4 colonnes */`
 - initialisation : `int T[3][4]={1,2,3,4, 2,4,6,8, 3,6,9,12};`

T[0][0]			T[0][3]
T[2][0]			T[2][3]

$T[i][j]$
i : indice de ligne de 0 à 2
j indice de colonne de 0 à 3