Université Sultan Moulay Slimane

Faculté Polydisciplinaire **Khouribga**

Sciences Mathématiques et Informatique

Structures de Données

Chapitre 1 : Types de Données Abstraits

Pr. Ibtissam Bakkouri

i.bakkouri@usms.ma

Année Universitaire : **2022/2023**

Plan

Introduction

Le langage C a été créé en 1972 par Denis Ritchie avec un objectif relativement limité : écrire un système d'exploitation (UNIX). Mais ses qualités opérationnelles l'ont très vite fait adopter par une large communauté de programmeurs.

C est un langage de programmation procédural et généraliste. Il est qualifié de langage de bas niveau dans le sens où chaque instruction du langage est conçue pour être compilée en un nombre d'instructions machine assez prévisible en termes d'occupation mémoire et de charge de calcul.

Types de base

Le C est un langage typé statiquement : chaque variable, chaque constante et chaque expression, a un type défini à la compilation.
Les types de base du langage C sont :

- **Entiers :** int, long int, short int.
- **Réels :** float, double, long double.
- **Caractères :** char.
- **Booléens :** bool.
- **Vide :** void.

Opérateurs arithmétiques

Les opérateurs arithmétiques effectuent des opérations mathématiques, telles que l'addition et la soustraction avec des opérandes.

Opérateur	**Définition**
+	Addition.
−	Soustraction.
*	Multiplication.
/	Division.
%	Modulo (reste d'une division euclidienne).

Opérateurs d'affectation

L'opérateur d'affectation affecte la valeur de son opérande droit à une variable, une propriété ou un élément indexeur donné par son opérande gauche. Le résultat d'une expression d'assignation est la valeur assignée à l'opérande de gauche.

Opérateur	**Définition**
+=	Additionne puis affecte le résultat.
-=	Soustrait puis affecte le résultat.
*=	Multiplie puis affecte le résultat.
/=	Divise puis affecte le résultat.
%=	Calcule le modulo puis affecte le résultat.

Opérateurs de comparaison

Les opérateurs de comparaison comparent deux valeurs et renvoient Vrai ou Faux. (De telles expressions sont parfois appelées expressions booléennes.) D'un point de vue mathématique, le résultat est Vrai (1) ou Faux (0).

Opérateur	**Définition**
==	Permet de tester l'égalité sur les valeurs.
!=	Permet de tester la différence en valeurs.
<>	Permet également de tester la différence en valeurs.
<	Permet de tester si une valeur est strictement inférieure à une autre.
>	Permet de tester si une valeur est strictement supérieure à une autre.
<=	Permet de tester si une valeur est inférieure ou égale à une autre.
>=	Permet de tester si une valeur est supérieure ou égale à une autre.

Opérateurs d'incrémentation et de décrémentation

Opérateur	**Définition**
++x	Pré-incrémentation : incrémente la valeur contenue dans la variable x, puis retourne la valeur incrémentée.
x++	Post-incrémentation : retourne la valeur contenue dans x avant incrémentation, puis incrémente la valeur de x.
- -x	Pré-décrémentation : décrémente la valeur contenue dans la variable x, puis retourne la valeur décrémentée.
x- -	Post-décrémentation : retourne la valeur contenue dans x avant décrémentation, puis décrémente la valeur de x

Entrées-sorties conversationnelles

Le langage C dispose d'un grand nombre de fonctions, fournies dans une bibliothèque standard, qui sont destinées à afficher des informations à l'écran ou à lire des données tapées au clavier : ces activités sont appelées entrées/sorties conversationnelles.

Les fonctions d'entrées/sorties servent à écrire et récupérer des données dans les fichiers ouverts et les flux standards déclarées dans l'entête <**stdio.h**>.

- La lecture sur l'entrée standard: La fonction scanf().
- L'écriture sur la sortie standard: La fonction printf().

Conditions en C

Les structures de contrôle conditionnelles vont nous permettre d'exécuter une série d'instructions si une condition donnée est vérifiée ou une autre série d'instructions si elle ne l'est pas.

- La condition if (si).
- La condition if... else (si... sinon).
- La condition if... elseif... else (si... sinon si... sinon).

Instruction switch en C

Switch

L'instruction switch évalue une expression et, selon le résultat obtenu et le cas associé, exécute les instructions correspondantes. Elle va nous permettre d'exécuter un code en fonction de la valeur d'une variable.

Syntaxe

```
switch (expression) {
        case valeur1:
        instructions1;
        [break;]
        case valeurN:
        instructionsN;
        [break;]
        [default:
        instructions_def;
        [break;]]
}
```

Instruction switch en C

- **expression :** Une expression à comparer avec chacune des clause case.
- **case valeurN :** Une clause qu'on compare avec expression.
- **default :** Une clause exécutée si aucune correspondance n'est trouvée avec les clause case (et/ou s'il n'y a pas de break pour les clauses case précédentes).
- **instructionsN :** Les instructions à exécuter lorsque l'expression correspond au cas présenté pour cette clause.
- **instructions_def :** Les instructions à exécuter si l'expression ne correspond à aucun cas de figure précédemment décrit.

Boucles en C

Les boucles permettent de répéter des actions simplement et rapidement.
Voici les différentes boucles fournies par C :

- La boucle while.
- La boucle do... while.
- La boucle for.

Fonctions en C

Le principe de modularité dans les langages de programmation permet qu'un groupe d'instructions regroupé dans un module puisse être accédé de façon répétitive de différents endroits dans un programme. Dans le langage C, un programme source peut être constitué d'une ou plusieurs unités de programmes appelés fonctions.

Une fonction correspond à un bloc de code nommé et réutilisable et dont le but est d'effectuer une tâche précise.

Fonctions en C

En plus des nombreuses fonctions C prédéfinies et immédiatement utilisables, nous allons pouvoir créer nos propres fonctions en C lorsque nous voudrons effectuer une tâche très précise.

Exemple :

```
float carré(float nombre)
{
float val;
val=nombre * nombre;
return val;
}
```

Récursivité

La récursivité c'est quand une fonction s'appelle elle-même jusqu'à atteindre une condition d'arrêt. Elle arrête alors de s'appeler elle-même. Le résultat de chaque fonction enfant est retourné dans les fonctions parent, jusqu'à retourner à la fonction originale.

Exemple :

```
int Factorielle (int N) {
if (N<=1) return 1;
return N*Factorielle(N-1);
}
```

Tableaux

On appelle tableau une variable composée de données de même type, stockée de manière contiguë en mémoire (les unes à la suite des autres).

Un tableau est donc une suite de cases (espace mémoire) de même taille. La taille de chacune des cases est conditionnée par le type de donnée que le tableau contient.

Tableaux à un indice

Pour accéder à un élément du tableau, il suffit donc de donner le nom du tableau, suivi de l'indice de l'élément entre crochets :

Nom_du_tableau[indice]

- L'indice du premier élément du tableau est 0.
- Un indice est toujours positif.
- L'indice du dernier élément du tableau est égal au nombre d'éléments - 1.

Définition du tableau	**Taille du tableau (en octets)**
char T[12]	1 * 12 = 12
int T[10]	2 * 10 = 20
float T[8]	4 * 8 = 32
double T[15]	8 * 15 = 120

Tableaux à deux indices

En C, nous pouvons définir des tableaux multidimensionnels en termes simples comme tableau de tableaux. Les données des tableaux multidimensionnels sont stockées sous forme de tableaux.

Le nombre total d'éléments pouvant être stockés dans un tableau à deux dimensions peut être calculé en multipliant la taille de toutes les dimensions.

Un tableau à deux dimensions est la forme la plus simple d'un tableau multidimensionnel. Nous pouvons voir un tableau à deux dimensions comme un tableau de tableau à une dimension pour faciliter la compréhension.

Tableaux à deux indices

Les éléments des tableaux à deux dimensions sont communément désignés par Tab[i][j], où "i" est le numéro de la ligne et "j" le numéro de la colonne.

Un tableau à deux dimensions peut être vu sous forme de tableau avec "n" lignes et "m" colonnes où le numéro de ligne est compris entre 0 et (n-1) et le numéro de colonne est compris entre 0 et (m-1).

Pointeurs

Un pointeur est une variable spéciale qui peut contenir l'adresse d'une autre variable.

Chaque pointeur est limité à un type de données.

Si un pointeur P contient l'adresse d'une variable A, on dit que 'P pointe sur A'.

Les pointeurs et les noms de variables ont le même rôle: Ils donnent accès à un emplacement dans la mémoire interne de l'ordinateur.

Déclaration d'un pointeur

La déclaration d'un pointeur se fait par : **Type *NomPointeur**.

Exemple :

int *P;

Signifie que :

- *P est du type int.
- P est un pointeur sur int.
- P peut contenir l'adresse d'une variable du type int.

Lors du travail avec des pointeurs, nous avons besoin d'un opérateur adresse **&** pour obtenir l'adresse d'une variable.

Alors, **&NomVariable** fournit l'adresse de la variable **NomVariable**.

Types complexes

Le langage C permet la définition de types personnalisés construits à partir des types de base du langage. Outre les tableaux, que l'on a déjà présentés, il est possible de définir différents types de données évolués, principalement à l'aide de la notion de structure.

En dehors des types de variables simples, le C permet de créer des types plus complexes. Ces types comprennent essentiellement les structures, les unions et les énumérations, mais il est également possible de définir de nouveaux types à partir de ces types complexes.

Structures

Les types complexes peuvent se construire à l'aide de structures. Pour cela, on utilise le mot clé struct. Sa syntaxe est la suivante :

```
struct [nom_structure]
{
type champ;
...
};
```

Alias de types

Le C dispose d'un mécanisme de création d'alias, ou de synonymes, des types complexes. Le mot clé à utiliser est typedef. Sa syntaxe est la suivante : **typedef définition alias;**
Où alias est le nom que doit avoir le synonyme du type et définition est sa définition.

Énumérations

Les énumérations sont des types intégraux (c'est-à-dire qu'ils sont basés sur les entiers), pour lesquels chaque valeur dispose d'un nom unique. Leur utilisation permet de définir les constantes entières dans un programme et de les nommer.

La syntaxe des énumérations est la suivante :

```
enum enumeration
{
nom1=valeur1,
nom2=valeur2,
...
};
```

Unions

Les unions constituent un autre type de structure. Elles sont déclarées avec le mot clé union, qui a la même syntaxe que struct. La différence entre les structures et les unions est que les différents champs d'une union occupent le même espace mémoire. On ne peut donc, à tout instant, n'utiliser qu'un des champs de l'union.

Exemple :

```
union entier_ou_reel
{
int entier;
float reel;
};
union entier_ou_reel x;
```

Présentation

Un TDA est un ensemble de données organisé de sorte que les spécifications des objets et des opérations sur ces objets (interface) soient séparées de la représentation interne des objets et de de la mise en oeuvre des opérations.

Une mise en oeuvre d'un TDA est la structure de données particulière et la définition des opérations primitives dans un langage particulier.

Avantages des TDA

Les avantages des TDA sont :

- Les programmes sont plus clairs car ils se focalisent sur la manière de résoudre un problème. On ne mélange pas les choix d'implantation et l'étude des algorithmes.
- Le codage des algorithmes étant complètement indépendant de l'implantation des données, les programmes restent valident même si on décide de changer complètement l'implantation.
- L'utilisateur d'un TDA n'a pas besoin de connaitre les détails du codage.
- Écriture de programmes modulaires.

Types abstraits standard

La plupart des langages modernes fournissent quelques nombre de TDA de base, soit dans le langage lui même, soit dans des bibliothèques :

- **Piles :** Une pile est un ensemble de données auquelles on accède dans l'ordre inverse ou on les a insérées (Last Input, First Ouput).
- **Files :** Dans une file, les éléments sont ajoutés en queue et supprimés en tête (First Input First Ouput).
- **Listes :** Une liste est un ensemble fini d'éléments. Les listes servent à gérer un ensemble de données, un peu comme les tableaux.
- **Arbres binaires :** Un arbre binaire est un arbre tel que les noeuds ont au plus deux fils (gauche et droit).

Travail à Rendre

Exercice 1 :

Ecrire un programme C qui lit un ensemble de personnes avec leurs âges, dans un tableau de structures, et supprime ensuite toutes celles qui sont âgées de vingt ans et plus.

Exercice 2 :

Ecrire un programme C qui définit une structure point qui contiendra les deux coordonnées d'un point du plan. Puis lit deux points et affiche la distance entre ces deux derniers.