

Algorithmie en langage C

Semestre 3

1 Paradigmes de programmation

Un paradigme est une manière de programmer, il en existe plusieurs :

1.1 Programmation fonctionnelles

Type de langage ¹ ou interprétés². Ce paradigme

Entité de base Appel de fonction

Structure de contrôle Approche récursive.

Elle est utilisée pour des systèmes critiques³. Elle a une approche très mathématiques, ce qui permet d'avoir des outils de preuves générique.

Elle possède une abstraction de l'environnement d'exécution, approche détachée de la machine, pas de notion de mémoire.

Ex Le Caml est un langage de programmation fonctionnelle

1.2 Programmation déclarative

Type de langage Interprété

Entité de base Règles de déduction logique.

Structure de contrôle Possède une abstraction de la machine cible.

Ex Le prolog est un langage de programmation déclarative

1.3 Programmation Impérative

La programmation est directement liée à la machine d'exécution.

Type de langage Compilé ou Interprété

Entité de base Affectation d'une valeur à une variable, qui est une place en mémoire.

Structure de contrôle Séquence, sélection, répétition.

Ex C, Python, Ada ...

1. Traduction du langage source vers le langage cible(compilation) + une édition de liens, qui est une instanciation sur la machine d'exécution (Recherche d'adresse, mémoire, résolution de fonctions) Elle peut être statique ou dynamique. Ex : C, Adda

2. Le langage source est traduit en langage cible à la volée par un interpréteur. Il est ainsi possible de modifier le programme pendant le fonctionnement du programme.

3. Besoin d'une sûreté de fonctionnement

2 Programmation impérative en C

Énormement de langage sont fondés sur la syntaxe du langage C.

Il a été développé dans les années 1960 par Denis Ritchie.

On trouvera toujours une partie description de l'organisation des données en mémoire⁴, nous aurons donc une déclaration de variables et un type de données.

```
1 type nomVariable;
```

Listing 1 – Syntaxe de déclaration de variable

2.1 Description de l'organisation des données en mémoire

Le C possède différents type de données :

int Entiers signés

unsigned int Entiers non signés

float Nombre réel sur 32bits.

double Nombre réel sur 64bits.

char Entier signé sur 8bits.

pointeur type* ptr ; La case mémoire contient une adresse.

2.2 Code syntaxe

2.2.1 Blocs

```
1 bloc { // début du bloc  
2 } //fin du bloc
```

Listing 2 – Syntaxe de déclaration de variable

Toute variable est visible dans son bloc de déclaration et ses blocs imbriqués.

Un bloc transforme une séquence en action.

2.2.2 Séquence

```
1 action 1;  
2 action 2;  
3 action 3;
```

Listing 3 – Syntaxe de déclaration de variable

4. C'est un grand tableau découpé en cases mémoire.

2.2.3 Sélection

```
1 if(condition) {  
2     action 1;  
3 } else {  
4     action 2;  
5 }
```

Listing 4 – Syntaxe de déclaration de variable

Condition est une expression booléenne⁵

2.2.4 Répétition

```
1 while(condition) {  
2     action;  
3 }
```

Listing 5 – Syntaxe de déclaration de variable

Condition est une expression booléenne, tant que la condition est vrai, les actions se repètent.

2.2.5 Affectation

```
1 variable = expression;  
2 \end{itemize}
```

Listing 6 – Syntaxe de déclaration de variable

2.2.6 Opérateurs de base sur les types

= Affectation

+, -, /, * Opérateurs arithmétiques.

&&, ||, ! Opérateurs logiques

==, !=, <, >, <=, >= Opérateurs booléens

++i, i++, -i, i- Opérateur unaires d'incrément.

2.2.7 Opérateurs d'entrées / sorties

```
Ecriture  
1 printf('format', var1, var2);
```

Listing 7 – Syntaxe de déclaration de variable

La chaîne format peut contenir une chaîne de caractères, avec des caractères spéciaux :

5. Expression renvoyant vrai(! = 0 ou faux(= 0))


'%d' Entier sous forme décimale
'%ox' Entier sous forme hexadécimale
'%f' Flottant
'%c' Caractère
'%s' Chaîne de caractères
'n' Vide le buffer et fait un retour chariot
't' Tabulation
'r' Revient en début de ligne.
'...' RTFM

Les différents formats doivent être dans l'ordre des variables passés en paramètres.

Lecture

```
1 scanf('format', &var1);
```

Listing 8 – Syntaxe de déclaration de variable

 Ne jamais utiliser scanf