Université Paul Sabatier – Toulouse III
IUT A - Toulouse Rangueil

Algorithmie en langage C

Semestre 3

Toulouse, le 13 septembre 2012

1 Paradigmes de programmation

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

1.1 Programmation fonctionnelles

Type de langage [1] ou interprétés [2]. Ce paragradigme

Entité de base Appel de fonction

Structure de contrôle Approche récursive.

Elle est utilisée pour des systèmes critiques [3]. Elle à une approche très mathématiques, ce qui permet d'avoir des outils de preuves générique.

Elle possède une abstraction de l'environnement d'execution, 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 Intérprété

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'execution.

Type de langage Compilé ou Intérprété

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

Structure de contrôle Séquence, séléction, 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'execution (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 intérpreteur. Il est ainsi possible de modifier le programme pendant le fonctionnement du programme.
3. Besoin d'une sureté 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 mémoire[4], 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 réél sur 32bits.

double Nombre réél 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 Séléction

```
if(conditon) {
  action 1;
} else {
  action 2;
}
```

Listing 4 – Syntaxe de déclaration de variable

Condition est une expression booléenne [5]

2.2.4 Répétition

```
while(condition) {
  action;
}
```

Listing 5 – Syntaxe de déclaration de variable

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

2.2.5 Affectation

```
variable = expression;
\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'incrémentation.

2.2.7 Opérateurs d'entrées / sorties

Ecriture

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

Listing 7 – Syntaxe de déclaration de variable

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

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

'%d' Entier sous forme décimale

'%ox' Entier sous forme héxadécimale

'%f' Flottant

'%c' Caractère

'%s' Chaine 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