Eléments d'Informatique

Cours 6 – Programme, définition de fonction, appel fonctionnel

Catherine Recanati

UNIVERSITÉ PARIS 13 NORD

Plan général

- **Représentation des nombres. Notion de variable.**
- **Programme. Expressions.**
- **Architecture des ordinateurs: langage machine, langage assembleur, AMIL.**
- **Systèmes d'exploitation : fichiers, processus, compilation.**
- **Instructions de contrôle: boucles et branchements.**
- **Programme, définition de fonction, appel fonctionnel.**
- Tableaux de variables et fonctions d'arguments de type tableau.
- Sens d'un programme, pile d'exécution, compilation.
- Pointeurs et tableaux.
- Chaines de caractères, bibliothèque <string.h>.
- Allocation dynamique, liste chaînées.
- Révisions.

- Cours 6 -
Programme, définition de fonction, appel fonctionnel

- Programmation
- Programmation impérative
- Structure d'un programme
- Définition de fonction
- Appel de fonction

Plan

Programmation

La programmation recouvre l'ensemble des techniques permettant de résoudre des problèmes à l'aide de programmes s'exécutant sur un ordinateur.

L'écriture effective du texte des programmes, bien que fondamentale, est la dernière étape du processus de programmation dans son ensemble, que l'on peut découper en 3 phases.

1. L'analyse et la spécification (du problème à résoudre, de la situation à simuler ou du logiciel attendu). On précise ici la liste exhaustive (=complète) des fonctionnalités fournies par le logiciel.

2. La conception ou modélisation. On conçoit des algorithmes permettant de résoudre le problème et/ou un modèle représentant la réalité simulée. On élabore des structures de données appropriées et on précise *comment* les diverses fonctionnalités seront accessibles à l'utilisateur.

3. L'implantation ou codage. C'est l'écriture du texte des programmes dans un (ou plusieurs) langage de programmation.

Dans le développement d'un logiciel, les deux premières phases prennent environ 80% du temps. Et plus les spécifications sont précises, plus le codage sera rapide.

Mais notre premier objectif étant l'apprentissage du langage C, nous ne traiterons que des exemples simples où les phases d'analyse et de conception sont très réduites, voire inexistantes.

Plan

Consigne: en TD/TP, on s'appliquera néanmoins à rédiger en français quelques remarques ou indications sur l'algorithme, avant de se lancer dans l'écriture en C.

Si l'exercice le permet, on pourra mélanger du C avec du français, et produire un squelette de programme où la plupart des instructions seront remplacées par des commentaires décrivant ce qui sera fait à cet endroit du programme.

Pourquoi avoir choisi le langage C pour ce cours d'initiation à la programmation ?

- Proche du langage machine, Le C a une supériorité de performance indéniable sur les autres langages (avec les architectures classiques des machine actuelles).

- Connaître le C permet par la suite d'apprendre facilement d'autres langages impératifs (Basic, Pascal, Fortran, Cobol), et même de langages d'autre type comme des langages à objet (C++, Perl ou Java), car ces langages dérivent du C, et la syntaxe des expressions est souvent la même.

Plan

Programmation impérative

Les langages dits impératifs, sont basés sur la notion d'**action exécutée**, l'action la plus typique étant **l'affectation de variable**.

Un programme impératif consiste en une suite d'ordres donnés à la machine, appelés instructions qui sont exécutés les uns après les autres. D'où le nom « impératif » donné à ce type de langage.

Structure de programme

Un programme C est une suite de déclarations de variables et de fonctions. Il peut aussi contenir des définitions de types ou de constantes. En fait c'est une suite de définitions et de déclarations (de variable, de type, de constante ou de fonction).

La fonction `main` fournit le bloc d'instructions à exécuter. Sa définition est nécessaire pour que le compilateur puisse produire un code exécutable.

Structure de programme

Un identificateur (de variable, de constante, de fonction, ou de type) **doit toujours avoir été déclaré avant d'être utilisé.**

On peut déclarer une fonction sans pour autant dire ce qu'elle fait (= donner le bloc qui constitue son corps de définition). Mais attention, dans le cas de fonction ou de type, une déclaration n'est pas une définition.

Déclaration de fonction

Une **déclaration de fonction** est une déclaration d'identificateur qui donne :

- le type de la valeur de retour de la fonction
- l'identificateur (=le nom) de la fonction
- le type et le nom des paramètres formels

Exemples :

```
int somme (int a, int b)
void imprime()
```

Définition de fonction

Une **définition de fonction** est constituée de sa déclaration suivie du corps de sa définition (= son bloc d'instructions).

Note: Les paramètres formels peuvent apparaître dans le corps de définition.

Ex :

```
1  int somme (int a, int b)
2  {
3     return (a + b) ;
4  }
```

Portée des variables

```
1  int somme (int x, int b)
2  {
3     int x = 4;/* variable locale */
4    ... que désigne x ici ? ...
5     printf("valeur de x: %d », x);
6     return (x + b);
7  }
```

⇒ Appelée avec `somme(0,5)`, la fonction imprime 4 et retourne 9.

⇒ Ne jamais déclarer de variable locale du même nom qu'un des paramètres formels

Portée des variables

```
1  /* niveau global du fichier */
2  int x = 0; /* variable globale */
3  int somme (int x, int y)
4  { /* x et y paramètres: ce sont
   des variables locales */
5       return (x + y);
6  }
7  int main() {
8     int x = 1;/* variable locale */
9     x = somme (x, x);
10    printf(« x: %d », x);
11    /* imprimera « x: 2 » */
```

Appel de fonction

Pour qu'un appel de fonction soit compris du compilateur, il faut que le type de son identificateur soit connu – donc que la déclaration (ou la définition) de la fonction *précède* l'appel de la fonction dans le texte du programme.

```
1  int somme(int a, int b);
2  int main() {
3      int x = 0, y = 4;
4      x = somme (x, y);
```

Structure d'un programme C

```
        /* inclusion des fichiers d'en-tête de librairies */
#include <stdlib.h>
        /* déclaration des constantes et des types utilisateur */
        /* déclaration des fonctions auxiliaires */
        /* déclaration des variables globales */

/* définition de la fonction principale appelée main */
int main() {
      /* déclarations des variables locales  */
      /* instructions de la fonction main */
}
/* définition des fonctions auxiliaires */
```

Appel de fonction

Un appel de fonction s'effectue:

- **en affectant** aux paramètres formels **les valeurs** des expressions en position d'arguments dans l'appel
- **puis en exécutant le bloc** du corps de définition de la fonction (jusqu'à rencontrer l'instruction `return`).

ex: `somme(3, 2*45);`

Ici la variable locale (paramètre formel a) sera initialisée à 3, et la variable b (deuxième paramètre) à 90.

Appel récursif

Les fonctions qui contiennent dans leur corps de définition un appel à elle-même sont dites « récursives ». Exemple

```
int f(int n){
   if (n==1)
     return n;
   else
     return(n + f(n-1));
}
```

Il y aura empilement des environnements de calcul, puis dépilement, avec retours successifs. Mais nous verrons cela plus loin.

Plan

Merci pour votre attention !

Des questions ?