Information, Calcul et Communication (partie programmation) : Synthèse et Révisions

Jamila Sam

Laboratoire d'Intelligence Artificielle
Faculté I&C

Objectifs du cours d'aujourd'hui

Nous voici donc arrivés au terme de cette première partie du cours.

L'objectif de ces quelques transparents est de vous rafraîchir la mémoire en rappelant les principaux points.

Nous avons abordé :

1. Bases de programmation
2. Approfondissements : E/S, gestion des erreurs

On va se focaliser ici sur les **rappels de programmation**.

Jamila Sam
& Jean-Cédric Chappelier

Qu'avons nous vu en programmation ?

programmer c'est décomposer une **tâche** à automatiser en une séquence d'instructions (**traitements**) et des **données**

Algorithme	S.D.A.
Traitements	Données
	Variables
Expressions & Opérateurs	
Structures de contrôle	
Fonctions	Portée
	Chaînes de caractères
	Tableaux statiques
	Tableaux dynamiques
	Structures
	Pointeurs
Entrées/Sorties	

☞ il faut que vous **soyez au clair** sur chacune de ces notions :

1. qu'est-ce que c'est ?
2. à quoi ça sert ?
3. comment ça s'utilise ?

FONDAMENTAUX

1. **déclarez avant d'utiliser**
   - variables
     ```
     int i;
     vector<double> v;
     ```
   - fonctions ☞ prototype
     ```
     double sin(double x);
     bool cherche_valeur(Listechainee l, Valeur v);
     ```
2. **modularisez** / décomposez / pensez « atomique »
   - 2.1 conception (qu'est ce qu'on veut ?)
   - 2.2 implémentation (comment ça se réalise ?)
   - 2.3 syntaxe (comment ça s'écrit ?)
   - 2.4 tests (où sont mes fautes, comment pourrais-je les tester ?)

Jamila Sam
& Jean-Cédric Chappelier

« Modularisez » : exemple

Exemple : affichage d'une tour de Hanoï

Je veux afficher le jeu

$\rightarrow$ c'est-à-dire afficher 3 « piliers »

$\rightarrow$ c'est-à-dire afficher de haut en bas

- soit vide
- soit un disque

$\Longrightarrow$ c'est-à-dire savoir afficher un disque

afficher un disque $\rightarrow$ c'est-à-dire afficher n caractères similaires

Jamila Sam
& Jean-Cédric Chappelier

Exemple d'approche modulaire (2)

Supposons qu'on ait déjà les objets de base :

```
// le disque -> sa taille (rayon)
typedef unsigned int Disque;
// 0 signifiant pas de disque
Disque const PAS_DE_DISQUE(0);
// un pilier est une pile d'au plus N disques:
typedef array<Disque, N> Pilier;
 // le jeu est constitue de 3 piliers:
typedef array<Pilier, 3> Jeu;
```

On fait alors alors exactement ce qui est écrit plus haut :
on commence par spécifier tour à tour ces fonctions

```
void affiche(Jeu jeu);
void affiche(Disque d);
void affiche(char c, unsigned int n = 1);
```

puis on écrit simplement ce qu'on a conçu.
C'est simple car on a **décomposé** la tâche.

Sinon c'est beaucoup plus compliqué $\Longrightarrow$ on peut plus facilement faire des erreurs

Exemple d'approche modulaire (3)

Afficher le jeu = afficher 3 « piliers » = afficher de haut en bas les 3 disques

```
/* ------------------------------
 * Affiche un jeu
 * Entree : le jeu a afficher
 * ------------------------------ */
void affiche(Jeu jeu)
{ // pour toutes les lignes:
  for (unsigned int i(0); i < N; ++i)
  {
    // affichage d'un disque du 1er pilier:
    affiche(jeu[0][i]);
        // affichage du 2eme:
    affiche(jeu[1][i]);
        // affichage du 3eme:
    affiche(jeu[2][i]);
        cout << endl;
  }
}
```

Exemple d'approche modulaire (4)

Afficher un disque

- soit vide
- soit n caractères identiques ($n =$ taille du disque)

```
/* ------------------------------------
* Affiche un disque
* Entree : le disque a afficher
* ------------------------------------ */
void affiche(Disque d)
{
        if (d == PAS_DE_DISQUE) {
                affiche(' ', N-1);
                cout << '|';
                affiche(' ' , N);
        } else {
                affiche(' ' , N-d);
                affiche('-' , 2*d-1);
                affiche(' ' , N-d+1);
        }
}
```

Exemple d'approche modulaire (5)

« atome » de notre décomposition : afficher *n* caractères

```
/*-----------------------------------
* Affiche n fois un caractere
* Entree : le caractere a afficher
  et le nombre de fois
* ----------------------------------- */
void affiche(char c, unsigned int n) {
 for (unsigned int i(0); i < n; ++i) cout << c;
}
```


Jamila Sam
& Jean-Cédric Chappelier

Faire le point avant l'examen

- prendre le tableau synthétique du transparent 3
- prendre les fiches résumé
- et pour chacun des points, se demander si on sait :
  - de quoi ça parle ?
  - ce que ça veut dire ?
  - l'utiliser ?

☞ se focaliser sur les **concepts**.

Les détails de syntaxe (comment ça s'écrit) peuvent être ensuite rapidement retrouvés dans la fiche résumé, si on sait ce qu'on cherche (c'est-à-dire si on a le concept)

Etude de cas

- Exemple de questions d'examens de programmation

Jamila Sam
& Jean-Cédric Chappelier