Faculté des Sciences de Gabès

Cours Atelier de Programmation

Chapitre 7 : Les sous Programmes

Faîçal Felhi

felhi_fayssal@yahoo.fr

Problème

- Dès qu'on commence à écrire des programmes sophistiqués, il devient difficile d'avoir une vision globale sur son fonctionnement
- Difficulté de trouver des erreurs
- Solution : décomposer le problème en sous problèmes
  - Trouver une solution à chacun
  - La solution partielle donne lieu à un sous-programme

Faîçal Felhi - felhi_fayssal@yahoo.fr

Programmation procédurale

- Principe:
  - Il s'agit d'écrire des programmes en utilisant des sous-programmes
- Forme générale d'un programme

Programme P

Sous-programme SP_1

…

Sous-programme SP_n

FinP

Faîçal Felhi - felhi_fayssal@yahoo.fr

Particularités des fonctions en C

- **Types de modules**: les « fonctions » et « procédures » ne sont pas distinguées
- **Mode de transmission des arguments**: uniquement par valeur
- **Variables globales**: accessibles à toutes les fonctions

Types de modules

- Les fonctions fournissent un résultat (*valeur de retour*), calculé à partir des valeurs de ses paramètres, qui peut ensuite apparaître dans une expression.
- Typiquement, les procédures des autres langages réalisent des *actions*, mais en pratique rien n'empêche les fonctions d'en réaliser également.
- En C, il n'existe que la notion de *fonction*:
  - la valeur de retour d'une fonction peut être ignorée (ex: **printf**)
  - une fonction peut ne retourner aucune valeur

Définition d'une fonction en c

- Syntaxe générale:

  <type-de-retour> <nom>(<liste-paramètres>)

  <instructions>

- Ex:

```
/* définition d'une fonction « abs » retournant un entier et prenant
   deux entiers comme paramètres */
int abs(int a, int b) {
   /* corps de la fonction */
   if (a > b)
   return (b – a);
}
```

Définition d'une procédure en c

- Syntaxe générale:

  <type-de-retour> <nom>(<liste-paramètres>)

  <instructions>

- Ex:

  /* définition d'une procédure « abs » retournant un entier et prenant deux entiers comme paramètres */

```
void abs(int a, int b) {
    /* corps de la procedure */
    int s;
    if (a > b)
        s=a-b;
    Printf (s);
}
```

Paramètres d'une fonction

- Tout type d'objet peut être passé comme paramètre d'une fonction:
  - types de base (variantes de **int**, **float**, **double**, **char**)
  - Tableaux
  - ....
- Les déclarations des paramètres, séparées par des virgules, associent un spécificateur de type à un déclarateur (nom de variable)
- Un liste de paramètres vide est possible

Exemples de portée des déclarations de fonctions (1/2)

```
/* déclaration de maFonction */
int maFonction(int val);
int main() {
        /* maFonction est accessible ici */
}
void autreFonction() {
        /* maFonction est accessible ici */
}
```

Exemples de portée des déclarations de fonctions (2/2)

```
int main() {
        /* maFonction n'est pas connue ici */
}

/* déclaration de maFonction */
int maFonction(int val);
void autreFonction() {
        /* maFonction est accessible ici */
}
```

Variables locales

- Les variables locales appartiennent au bloc dans lequel elles sont déclarées (par exemple, le bloc de définition d'une fonction).
- Les paramètres formels se comportent comme des variables locales.
- Portée d'une variable locale:
  - elle est visible depuis sa déclaration jusqu'à la fin du bloc où elle est déclarée
  - elle peut masquer des variables issues des contextes englobants (cf. exemple suivant)

Exemple de portée des variables locales

```
int a, b; /* variables globales */
void main() {
    int b, c; /* variables locales à main */
    /* ici b se réfère à la variable locale à main */
    {
        long a, c;
        /* ici a et c se réfèrent aux variables locales au bloc  */
        /* b se réfère à la variable locale à main */
    }
    /* ici b et c se réfèrent aux variables locales à main */
    /* a se réfère à la variable globale */
}
```

Bibliographie

- « Cours d'algorithmique » de McGraw-Hill
- « Initiation à l'algorithmique et à la programmation C » de Rémy Malgouyres, Rita Zrour et Fabien Feschet
- « Introduction à l'algorithmique » de Charles E. Leiserson, Clifford Stein, Ronald Rivest et Thomas H. Cormen.

Faîçal Felhi - felhi_fayssal@yahoo.fr