Algo/Prog – DUT 1

Notes

Notes

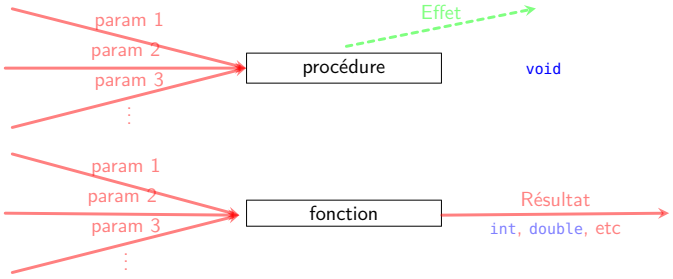
Sous-programmes

- Un « bout de programme »
- qui utilise des paramètres (ou arguments)
- qu'on peut appeler autant qu'on veut avec des paramètres différents
- qui peut avoir des effets (modif. écran ou mémoire, etc)
- qui peut avoir un résultat (valeur de « retour »)

- écrit par soi-même
- ou par un autre programmeur
- code source connu/inconnu
- mieux si documenté (API)
- changement de versions

Notes

Sous-programme : procédures VS fonction



procédure = instruction/expression ?

fonction = instruction/expression ?

Notes

Appel de procédure

```
ecrireDate();
```

Appel de la procédure `ecrireDate`

1. Séparez les feuilles de la salade
2. Lavez les feuilles
3. **Faites une vinaigrette, page 12.**
4. égouttez les feuilles
5. ...

Appel de la procédure `vinaigrette`

Notes

Appel de procédure I – exécution

Salade verte :

1. Séparez les feuilles
2. Lavez les feuilles
3. **Faites une vinaigrette**
9. égouttez les feuilles
10. ...

Vinaigrette :

4. versez l'huile dans un bol
5. versez le vinaigre
6. versez la moutarde
7. battre jusqu'à émulsion
8. salez poivrez

- arrête l'exécution de la recette actuelle
- démarre l'autre recette
- autre recette finie ⇒ redémarre l'actuelle

Notes

Deuxième programme (procédure `pause`)

procédure `pause()` attend une pression sur « entrée ».
procédure `ecrireDate()` écrit la date à l'écran.

```c
#include "inout.h"

int main(int nargs, char **args)
{
  ecrireString("Hello world! Please press enter\n");
  pause();
  ecrireDate();
  ecrireString("Bye world!\n");
}
```

Notes

Appel de procédure II – Paramètres

```
ecrireString("Hello world\n");
ecrireInt(12);
```

paramètres

Omelette salade

1. Séparez les feuilles de la salade
2. Lavez les feuilles
3. Faites une vinaigrette (p.12)
4. **Faites une omelette pour 4 (p.23)**
5. égouttez les feuilles
6. ...

Notes

Paramètre de procédures

Omelette pour 1 personne(s)

1. Pelez et hachez finement 1/4 oignon(s)
2. Dans un bol :
   1. battez 2 oeufs entiers
   2. ajoutez 1 branche(s) de basilic ciselé
   3. ajoutez 1/4 cube(s) de bouillon de volaille.
   4. ajoutez sel et poivre à votre goût
   5. mélangez bien au fouet.
3. Dans une poêle :
   1. faites chauffer l'huile d'olive
   2. faites revenir les oignons hachés pendant 2 min, en remuant
4. Versez dans la poêle les oeufs et le fromage râpé laissez cuire 7 min.

Notes

Paramètre de procédures

⇒ Paramètre *n* (*formel*)

Omelette pour n personne(s)

1. Pelez et hachez finement n/4 oignon(s)
2. Dans un bol :
   1. battez 2×n oeufs entiers
   2. ajoutez n branche(s) de basilic ciselé
   3. ajoutez n/4 cube(s) de bouillon de volaille.
   4. ajoutez sel et poivre à votre goût
   5. mélangez bien au fouet.
3. Dans une poêle :
   1. faites chauffer l'huile d'olive
   2. faites revenir les oignons hachés pendant 2 min, en remuant
4. Versez dans la poêle les oeufs et le fromage râpé laissez cuire 7 min.

Notes

Paramètre de procédures

Omelette pour 3 ← paramètre *réel*

Omelette pour 3 personne(s)

1. Pelez et hachez finement 3/4 oignon(s)
2. Dans un bol :
   1. battez 2×3 oeufs entiers
   2. ajoutez 3 branche(s) de basilic ciselé
   3. ajoutez 3/4 cube(s) de bouillon de volaille.
   4. ajoutez sel et poivre à votre goût
   5. mélangez bien au fouet.
3. Dans une poêle :
   1. faites chauffer l'huile d'olive
   2. faites revenir les oignons hachés pendant 2 min, en remuant
4. Versez dans la poêle les oeufs et le fromage râpé laissez cuire 7 min.

Notes

Paramètre de procédures

Omelette pour 4

Omelette pour 4 personne(s)

1. Pelez et hachez finement 4/4 oignon(s)
2. Dans un bol :
   1. battez 2×4 oeufs entiers
   2. ajoutez 4 branche(s) de basilic ciselé
   3. ajoutez 4/4 cube(s) de bouillon de volaille.
   4. ajoutez sel et poivre à votre goût
   5. mélangez bien au fouet.
3. Dans une poêle :
   1. faites chauffer l'huile d'olive
   2. faites revenir les oignons hachés pendant 2 min, en remuant
4. Versez dans la poêle les oeufs et le fromage râpé laissez cuire 7 min.

Notes

Troisième programme

```c
#include "inout.h"

int main(int nargs, char **args) {
  ecrireString("\nHello world!\n");
  pause();
  ecrireInt(12);
  ecrireString("\nBye world!\n");
}
```

Écrit un entier à l'écran

Notes

Quatrième programme

```c
#include "inout.h"

int main(int nargs, char **args) {
  ecrireString("\nHello world!\n");
  pause();
  ecrireInt(12+5);
  ecrireString("\nBye world!\n");
}
```

entier

Notes

Appel de procédure I

Salade verte pour 3 :

1. Séparez les feuilles de la salade
2. Lavez les feuilles
3. **Faites une vinaigrette pour 3**
9. égouttez les feuilles
10. ...

Vinaigrette pour n3 :

4. versez 3 cuillère d'huile dans un bol
5. versez 3/3 cuillères de vinaigre
6. versez 3/3 cuillères de moutarde
7. battre jusqu'à émulsion
8. salez poivrez

- arrête l'exécution de la recette actuelle
- démarre l'autre recette **avec le paramètre réèl**
- autre recette finie ⇒ redémarre l'actuelle

Notes

Fonctions

```c
heureActuelle();
```

- « Retourne » le numéro de l'heure actuelle (entre 0 et 23)
- `heureActuelle()` **équivalent à un entier**
- `heureActuelle() + 2` bloque l'addition jusqu'à son retour (démo)
- expression : retourne un résultat mais n'a pas d'effet
- utiliser = l'inclure dans une instruction.
- ex : `ecrireInt(heureActuelle());` (ecrireInt attends le retour de la fonction (démo)

Notes

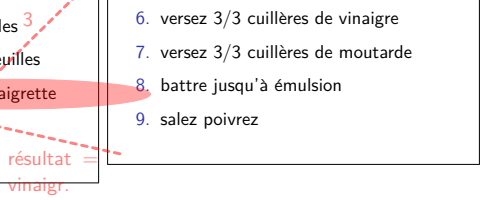
Appel de fonction

Salade verte pour 3 :

1. Séparez les feuilles de la salade
2. Lavez les feuilles
3. égouttez les feuilles
4. Faites une vinaigrette pour 3 et versez la
10. ...

Vinaigrette pour 3 :

5. versez 3 cuillère d'huile dans un bol
6. versez 3/3 cuillères de vinaigre
7. versez 3/3 cuillères de moutarde
8. battre jusqu'à émulsion
9. salez poivrez



- arrête l'exécution de la recette actuelle
- instruction 4. pas finie !
- démarre l'autre recette **avec le paramètre réèl**
- finie ⇒ instruction 4. redémarre **en utilisant la valeur retournée**

Notes

Variables

```
lireInt();
```

- « retourne » l'entier tapé
- `ecrireInt(lireInt()+lireInt())` fait 2 lectures clavier.
- comment faire pour utiliser plusieurs fois le résultat du premier `lireInt()` ?
- Exemple : lire un entier n au clavier, puis afficher la valeur de $n + n^2$.

⇒ Stockage du résultat de (= valeur retournée par) `lireInt()` pour l'utiliser plus tard.

Notes

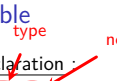
Variable

- Portion de mémoire que l'on nomme (en C : n'importe quel nom commençant par une lettre)
- Suffisamment grande pour contenir la donnée (ex : contenir un `int`)
- Change de valeur au cours de l'exécution du programme
- Déclaration nécessaire avant usage
- usage :
  - Affectation : remplacer la valeur actuelle par une autre
  - Utilisation : utiliser la valeur actuelle

Notes

Variable

- Déclaration :




```
int n ;
int x,m ;
```

- Affectation :

```
n = 12 ;
x = 3+4 ;
m = lireInt()+7 ;
```

  - signe « égal » (`=`) ne signifie pas égalité (voir `!==` !)
  - `x = 12;` signifie « Mettre la valeur 12 dans la variable x ».

- Utilisation :

```
ecrireInt(n+8+g(n+3));
f(n + n*n + x * m);
x = g(n +3 , m - 7);
```

Notes

Variable – Affectation

```
x = g(n +3 , m - 7);
```

≪ Receptacle ≫ ≪ Valeur ≫

pas la même signification à gauche ou à droite de l'affectation

Règle

Le nom d'une variable désigne toujours sa valeur sauf à gauche de l'affectation.

Notes

Variable – Affectation

Déroulement (à l'exécution) de l'affectation :

```
x = g(n +3 , m - 7);
```

- partie de droite (ici `g(n +3 , m - 7)`) exécutée (donne un résultat)
- ensuite résultat de la partie droite écrit dans la variable de gauche (`x`) (efface la valeur précédente)

D'où la possibilité de faire ceci :

```
int n = 4;
ecrireInt(n); // ? qu'affiche ceci
n = n + 3;
ecrireInt(n); // ? qu'affiche ceci
```

Notes

Variable

L'instruction d'affectation :

```
x = f(12) + 7 - g(4)
```

S'exécute de la manière suivante :

1. Évaluation à droite du ≪ = ≫ : `f(12) + 7 - g(4)`
2. Valeur obtenue stockée dans `x` (valeur précédente écrasée).

Question : Que fait l'instruction :

```
x = x + 2;
```

?

Notes

Fonction et procédures utiles

```
lireInt();
```

- Attend que l'utilisateur tape un entier (ex : 123) + ≪ entrée ≫
- ≪ retourne ≫ l'entier tapé
- `lireInt()` **équivalent à un entier**
- `ecrireInt(lireInt())` (démo)
- `ecrireInt(lireInt() + lireInt())` (démo)

Notes

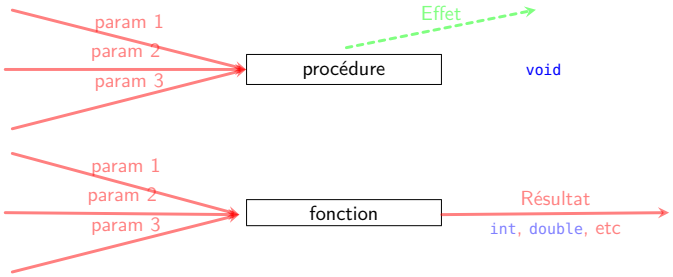
Fonction et procédures utiles

```
ecrireInt( un entier );
ecrireInt(0xFF + 34+67);
ecrireInt(0xFF + 34+lireInt());
```

- calcule l'entier (ex : `(0xFF) + 34+67`)
- Calcule la chaîne de caractère permettant d'afficher cet entier (ex : "356")
- Affiche cette chaîne dans le terminal
- PUIS OUBLIE TOUT ET SE TERMINE 

Notes

Sous-programme : procédures VS fonction



Notes

Erreur extrêmement courante

Répétez en cœur :

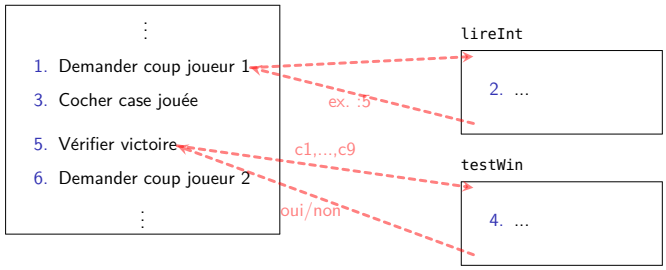
« Prendre en paramètre n'est pas lire au clavier »

« Retourner un résultat n'est pas afficher à l'écran »

Notes

Explications – Morpion

9 variables (`char`) représentant l'état du jeu. Au début : `c1,...,c9 = '_'`



- la ligne 3 a besoin du résultat de `lireInt` pour mettre à jour la variable
- L'affichage de cet entier serait inutile
- De même `testWin` a besoin des valeurs des 9 variables pour pouvoir faire son calcul de victoire.
- `testWin` NE DEMANDE RIEN À L'UTILISATEUR, on l'appelle avec les 9

Notes