Algo/JavaScript – USAL34

TP1 : Introduction au langage C

Objectifs du TP

— Apprendre à compiler un programme avec deux fichiers (dont un déjà fourni).
— Ecrire des procédures simples sans puis avec paramètres.
— Tester les affichage produits par ces procédures.

Les programmes et les commandes ci-dessous peuvent être copiée/collées dans l'éditeur ou le terminal. Pour cela il peut être nécessaire d'activer la sélection de texte dans le visionneur de pdf (`okular`) : menu `Outil/Mode Sélection de texte`.

1 Prise en main et compilation

Effectuez les opérations suivantes :

— Démarrez une session linux.
— Téléchargez les fichiers inout.h et inout.c et mettez les dans un répertoire `tp1`. Pour cela vous pouvez faire les commandes suivantes dans un terminal (menu pricipal : `terminal/Konsole`) :

```
mkdir tp1
cd tp1
wget "{https://cedric.cnam.fr/~lamberta/enseignements/DSP/C/sources/inout.h}"
wget "{https://cedric.cnam.fr/~lamberta/enseignements/DSP/C/sources/inout.c}"
```

— Dans le même terminal, effectuez la commande suivante pour compiler le fichier `inout.c` :

```
gcc -c inout.c
```

— Vérifiez le contenu du répertoire : `ls`
— Ouvrez l'éditeur `kwrite` (menu principal `editeur/kwrite`) ou tout autre éditeur de votre choix, SAUF LES IDEs de type Eclipse/Netbeans.
— Ouvrez un nouveau fichier `priseenmain.c` dans le même répertoire `tp1`. Et copiez-collez le programme suivant dedans :

```
#include"inout.h"

void f () {
  ecrireString("Hello World\n");
}

void main(void) {
  f(); // Appel a la procedure f
}
```

— Compilez ce nouveau fichier :

```
gcc -c priseenmain.c
```

— Fabriquez un exécutable :

```
gcc inout.o priseenmain.o -o priseenmain
```

— Exécutez le le pogramme obtenu :

```
./priseenmain
```

Modifiez le programme pour ajouter une procédure `g` et faire appel à `g` (après l'appel à `f`) dans le `main` :

```
#include"inout.h"

void f () {
  ecrireString("Hello World\n");
}

void g() {
  ecrireInt(lireInt());
}

void main(void) {
  f(); // Appel a la procedure f
  g(); // Appel a la procedure g
}
```

1. Que va faire ce programme ?
2. Testez votre réponse en compilant et exécutant le fichier `priseenmain.c` modifié.

2 Premières procédures en C

Avant de commencer

— Ouvrez un nouveau fichier `tp1.c` dans le même répertoire `tp1`. **Vous écrirez toutes vos procédures dans ce fichier, chaque procédure devra être testée. N'effacez pas vos procédures une fois finies et testées.**

— Lorsque vous voulez compilez ce fichier :

```
gcc -c tp1.c
```

— Fabriquez un exécutable :

```
gcc inout.o tp1.o -o tp1
```

— Exécutez le le pogramme obtenu :

```
./tp1
```

Comment testez vos procédures ?

Programmez et tester les procédures *une par une* (testez une procédures dès que vous pensez qu'elle est finie). Les procédures de ce TP n'ont aucun effet à part les affichages, donc il faut vérifier sur l'écran que les messages qui s'affichent sont corrects. TESTEZ PLUSIEURS CAS, ce n'est pas parce-qu'un appel semble fonctionner que votre procédure est correcte.

Pour tester une procédure `proc(...)` écrivez une autre procédure `testproc()` dans laquelle vous effectuez un ou plusieurs appels à `proc`. Lorsque vous désirez lancer le test, ajoutez l'instruction `testproc();` dans la fonction `main`, compilez, lancez l'exécutable, vérifiez les affichages.

N'effacez pas vos procédures de test ! Elle peuvent servir à nouveau si vous devez corriger/améliorer vos procédures. En revanche vous pouvez effacer l'appel aux tests dans le `main`, il sera facile de le remettre plus tard. Dans l'exemple ci-dessous la procédure `testecritSommeDeuxInt` commence par afficher un message permettant de savoir quelle procédure est testée, puis chaque appel de la méthode est annoncé avant d'être appelée.

Exemple :

```
#include"inout.h"

void ecritSommeDeuxInt(int x, int y){
   ...
}

void testecritSommeDeuxInt(){
  ecrireString("** Début test ecritSommeDeuxInt:");
  ecrireString("** (2,5): ");
  ecritSommeDeuxInt(2,5); // devrait afficher 7
  ecrireSautDeLigne();
  ecrireString("** (-2,2): ");
  ecritSommeDeuxInt(-2,2); // devrait afficher 0
  ecrireString("** Fin test ecritSommeDeuxInt:");
}

void main(void){
  testecritSommeDeuxInt();
}
```

Exercice 1 — *Lecture au clavier et affichage à l'écran*

Programmez et tester ces fonctions *une par une* (testez une procédure dès que vous pensez qu'elle est finie). Pour tester une procédure, il faut programmer un ou plusieurs appels à cette procédure dans le `main` et on observe les affichage.

1. `void ecritBonjour() { ... }` qui écrit `"Bonjour"` à l'écran.
2. `void ecrit3Bonjour() { ... }` qui écrit à l'écran le texte ci-dessous :

   ```
   Bonjour
   Bonjour
   Bonjour
   ```

   Vous pouvez faire appel à la procédure précédente.
3. `void ecritSommeDeuxlireInt() { ... }` qui écrit à l'écran la somme de deux entiers lus au clavier (`lireInt()`).
4. `void ecritDeuxFoislireInt(){ ...` qui écrit deux fois à l'écran un entiers lu au clavier (avec un seul appel à `lireInt()`). Il est nécessaire d'utiliser une variable.
5. `void ecritDeuxInt(int x, int y) { ... }` qui écrit les deux entiers passés en paramètres.
6. `void ecritSommeDeuxInt(int x, int y) { ... }` qui écrit à l'écran la somme de deux entiers passés en paramètres.
7. `void ecritSommeDeuxIntBavard(int x, int y) { ... }` qui écrit les deux entiers passés en paramètres puis leur somme.
8. `void ecritSommeTroisInt(int x, int y, int z){ ... }` qui écrit à l'écran la somme des trois entiers passés en paramètres.
9. `void ecritSommeDifflireInt(){ ... }` qui écrit la somme puis la différence (sur deux lignes consécutives) de deux entier lus au clavier.

Exercice 2 — *Fonctions (si vous avez fini avant la fin)*

Si vous connaissez l'instruction `return` et la notion de fonction, reprogrammez les procédures de l'exercice 1 sous forme de fonctions retournant le résultat au lieu de l'afficher. Programmez des tests pour ces fonctions.