CNAM
DSP Informatique - Réseau de la Réussite Vincent Merle

Algo/JavaScript – USAL34

TP2 Fonctions et conditionnelle en C

Objectifs du TP

- Apprendre à utiliser la conditionnelle.
- Ecrire des fonctions simples sans puis avec paramètres.
- Tester les retours produits par ces fonctions.

1 Rappels : bibliothèque et compilation

La bibliothèque d'entrée-sortie : inout.h et inout.c mettez les dans un répertoire `tp2`. Pour cela vous pouvez faire les commandes suivantes (une seule fois) dans un terminal (menu pricipal : `terminal/Konsole`) :

```
mkdir tp2
cd tp2
wget "https://cedric.cnam.fr/~lamberta/enseignements/DSP/C/sources/inout.h"
wget "https://cedric.cnam.fr/~lamberta/enseignements/DSP/C/sources/inout.c"
```

Programmez et tester ces fonctions *une par une* (testez une fonction dès que vous pensez qu'elle est finie). Pour tester une fonction `f` on programme un ou plusieurs appels à cette procédure dans une procédure `test_f` : on test le résultat obtenu par rapport au résultat attendu et on affiche un message d'erreur si ils ne sont pas égaux.

2 La conditionelle `if`

Il est possible dans un programme de définir deux séquences d'instructions différentes en fonction du résultat d'un test à l'exécution. L'instruction conditionnelle comporte trois morceaux : le test, la partie à exécuter si le test est vrai à l'exécution (« then ») et la partie à exécuter si le test est faux à l'exécution (« `else` »). Cette dernière partie (`else`) est facultative.

Voici la syntaxe de la conditionnelle en C (et java).

```
if (test) {
  ... // instructions si test vrai
}
else {
  ... // instructions si test faux
}
```

Exemple d'utilisation

```
1 int x;
2 x = lireInt();
3 if (x >=0) {
4   ecrireString("nombre positif ou nul.");
5 }
6 else {
7   ecrireString("nombre strictement négatif.");
8 }
9 ecrireString("\n");
```

À l'exécution ce programme se comporte comme suit :

— *ligne 2.* attend que l'utilisateur tape un entier au clavier et le stocke dans la variable `x` ; puis :
— *lignes 3 à 8.* si `x` est plus grand ou égal à zéro (`x>=0`)
  — *ligne 4.* alors le texte « `nombre positif ou nul.` » est affiché à l'écran ;
  — *ligne 7.* sinon le texte « `nombre strictement négatif.` » est affiché ;
— *ligne 9.* : Enfin quoiqu'il arrive un saut de ligne est affiché.

Dans ce TP, nous utiliserons les expressions de tests suivantes, où `e1` et `e2` doivent être des expressions entière (`12`, `x`, `x+3`, etc) :

— `e1 == e2` et `e1 != e2`
— `e1 <= e2` et `e1 >= e2`
— `e1 < e2` et `e1 > e2`

Exercice 1 — *Procédures avec conditionnelles*

Programmez et tester ces procédures *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 ecritMax(int x, int y)` qui écrit à l'écran le plus grand de deux entiers passés en paramètres (si ils sont égaux, on en écrit un).
2. `void ecritTestPlusGrandEq(int x, int y)` qui affiche le text « <x> `plus grand que` <y> » si le premier argument est plus grand ou égal que le deuxième et « <valeur de l'arg 1> `pas plus grand que` <valeur de l'arg 2> » sinon. <x> et <y> étant remplacés par les valeur réelles des arguments à l'exécution.

   Par exemple : `ecritTestPlusGrandEq(12,13)` doit afficher « `12 n'est pas plus grand que 13` ».
3. `void ecritTestPlusPetit(int x, int y)` qui se comporte comme `ecritTestPlusGrandEq` mais en inversant le test (premier argument strictement plus petit).
4. `void ecritTrie3(int x, int y, int z)` qui affiche « `<x>, <y>, <z> triés` » si les trois arguments sont ordonnés par ordre croissant. On tolère les arguments consécutifs égaux. Écrire « `<x>, <y>, <z> pas triés` »sinon.
5. `void ecritMax3(int x, int y, int z)` qui écrit à l'écran le plus grand des trois arguments.
6. `void ecritTestDeuxEgaux(int x, int y, int z, int t)` qui qui affiche « `2 paramètres égaux` » si parmi les 4 arguments au moins deux sont égaux.
7. `void ecritPlusSommeProd(int z, int t)` écrit à l'écran le plus grand entier entre la somme et le produit des deux arguments.

► Correction

```
#include "inout.h"

void ecritMax(int x, int y){
  if (x> y)
    {
      ecrireInt(x);
      ecrireString("\n");
    }
  else
     {
      ecrireInt(y);
      ecrireString("\n");
    }
}
void ecritTestPlusGrandEq(int x, int y){
```

```
  if (x> y)
    {
      ecrireInt(x);
      ecrireString(" plus grand que ");
      ecrireInt(y);
      ecrireString("\n");
    }
  else
    {
      ecrireInt(y);
      ecrireString(" plus grand que ");
      ecrireInt(x);
      ecrireString("\n");
    }
}
void ecritTestPlusPetit(int x, int y){
  if (x< y)
    {
      ecrireInt(x);
      ecrireString(" plus petit que ");
      ecrireInt(y);
      ecrireString("\n");
    }
  else
    {
      ecrireInt(y);
      ecrireString(" plus petit que ");
      ecrireInt(x);
      ecrireString("\n");
    }
}

void ecritTrie3(int x, int y, int z){
  if (x<= y && y <= z)
    ecrireString("<X>, <Y>, <Z> triÃ©s\n");
  else
    ecrireString("<X>, <Y>, <Z> pas triÃ©s\n");
}

void ecritMax3(int x, int y,int z){
  if (x< y)
    {
      if (y<z)
        ecrireInt(z);
      else
        ecrireInt(y);
    }
  else
    {
      if (x<z)
        ecrireInt(z);
      else
        ecrireInt(x);
    }
}

void ecritTestDeuxEgaux(int x, int y,int z, int t){
  if (x== y || x==z || x== t || y==z || y==t || z== t)
    ecrireString("2 paramÃ¨tres Ã©gaux\n");
  else
    ecrireString("aucun paramÃ¨tres Ã©gaux\n");
```

```
}
void ecritPlusSommeProd(int z, int t){
   if (z+ t>=z*t)
        ecrireInt(z+t);
      else
        ecrireInt(z*t);
}
void main ( void ) {
  ecritMax(2,3);
  ecritTestPlusGrandEq(2,3);
  ecritTestPlusPetit(2,3);
  ecritTrie3(2,3,4);
  ecritMax3(2,3,4);
  ecritTestDeuxEgaux(2,3,4,3);
  ecritPlusSommeProd(2,3);


}
```

Exercice 2 — *Menu*

Écrivez un programme dans la fonction `main` qui propose les différentes fonctionnalités des procédures de la section précédente.
Le déroulement du programme doit être le suivant :

1. Affichage des opérations disponibles avec un numéro pour chaque.
2. l'utilisateur tape un entier (+ « entrée »)
3. invitation à taper le premier argument de la procédure
4. invitation à taper le deuxième argument de la procédure
5. etc
6. lancement de la procédure
7. Saut de ligne et fin de programme.

► **Correction**

```
#include "inout.h"
void main ( void ) {
 ecrireString("Voici les procedures disponibles, tapez le numero de
celle que vous voulez executer:\n
1. ecritMax\n
2. ecritTestPlusGrandEq \n
3. ecritTestPlusPetit\n
4. ecritTrie3\n
5. ecritMax3\n
6. ecritTestDeuxEgaux\n
7. ecritPlusSommeProd\n");
  int numFonction = lireInt();
  ecrireString("Entrez le premier parametre\n");
  int x = lireInt();
  ecrireString("Entrez le deuxieme parametre\n");
  int y = lireInt();
  int z, t;
  if  (numFonction == 1)
    ecritMax(x,y);
  if  (numFonction == 2)
  ecritTestPlusGrandEq(x,y);
  if (numFonction == 3)
  ecritTestPlusPetit(x,y);
```

```
  if (numFonction <=0 || numFonction >7)
    {
       ecrireString("Voici les procedures disponibles, tapez le numero de
celle que vous voulez executer:\n
1. ecritMax\n
2. ecritTestPlusGrandEq \n
3. ecritTestPlusPetit\n
4. ecritTrie3\n
5. ecritMax3\n
6. ecritTestDeuxEgaux\n
7. ecritPlusSommeProd\n");
      numFonction = lireInt();
    }
  if (numFonction == 4 || numFonction == 5 || numFonction == 6)
    {
      ecrireString("Entrez le troisieme parametre\n");
      z = lireInt();
      if (numFonction == 4)
        ecritTrie3(x,y,z);

      if (numFonction == 5)
        ecritMax3(x,y,z);

      if (numFonction == 6)
        {
          ecrireString("Entrez le quatrieme parametre\n");
          z = lireInt();
          ecritTestDeuxEgaux(x,y,z,t);
        }
    }
  if  (numFonction == 7)
    ecritPlusSommeProd(x,y);

}
```

3 Les fonctions

Exercice 3 — *Fonctions retournant une valeur entière*

Programmez et tester ces fonctions *une par une* (testez une fonction dès que vous pensez qu'elle est finie).

1. `int SommeTroisInt(int x, int y, int z)` qui retourne la somme des trois entiers passés en paramètres.
2. `int max2(int x, int y)` qui retourne le plus grand des deux entiers passés en paramètres (si ils sont égaux, on en retourne un).
3. `int max3(int x, int y, int z)` qui retourne le plus grand des trois entiers passés en paramètres (si 2 ou plus sont égaux, on retourne l'un des plus grand).
4. `int PlusSommeProd(int z, int t)` qui retourne le plus grand entier entre la somme et le produit des deux arguments.

► **Correction**

```
#include "inout.h"

int SommeTroisInt(int x, int y, int z){
  return x+y+z;
}
```

```
int max2(int x, int y){
  if (x> y)
    return x;
  else
    return y;
}
int max3(int x, int y,int z){
  if (x< y)
    {
      if (y<z)
        return z;
      else
        return y;
    }
  else
    {
      if (x<z)
        return z;
      else
        return x;
    }
}
int plusSommeProd(int z, int t){
   if (z+ t>=z*t)
     return z+t;
   else
     return z*t;
}

void main ( void ) {
  int res =  SommeTroisInt(2,3,4);
  ecrireInt(res);
  res = max2(3,1);
  ecrireInt(res);
  res = max3(5,6,7);
  ecrireInt(res);
  res = plusSommeProd(3,4);
  ecrireInt(res);
}
```

Exercice 4 — *Fonction retournant une valeur booléenne (test)*

Une fonction de test doit retourner une valeur utilisable comme condition dans un `if (condition) ...`. Elle doit donc retourner l'entier 0 pour signifier que le test est faux, et un entier non nul (1 de préférence) si le test est vrai.

Pour plus de clarté on utilisera les synonymes suivants :

```
#define BOOL int
#define TRUE 1
#define FALSE 0
```

Programmez et tester ces fonctions *une par une* (testez une fonction dès que vous pensez qu'elle est finie).

1. `BOOL testPlusGrandEq(int x, int y)` qui teste si le premier paramètre est plus grand ou égal au deuxième.

   Pour tester la fonction faites :

   ```
   if (testPlusGrandEq(10,8)) {
     ecrireString("test testPlusGrandEq réussi: 10 >= 8");
   ```

```
} else {
  ecrireString("test testPlusGrandEq échoué: 10 < 8");
}

if (testPlusGrandEq(8,8)) {
  ecrireString("test testPlusGrandEq réussi: 8 >= 8");
} else {
  ecrireString("test testPlusGrandEq échoué: 8 < 8");
}
etc
```

2. `BOOL testPlusPetit(int x, int y)` qui se comporte comme `testPlusGrandEq` mais en inversant le test (premier argument strictement plus petit).
3. `BOOL testTrie3(int x, int y, int z)` qui teste si les trois arguments sont ordonnés par ordre croissant. On tolère les arguments consécutifs égaux.
4. `BOOL testDeuxEgaux(int x, int y, int z, int t)` qui teste si parmi les 4 arguments au moins deux sont égaux.

► **Correction**

```
#include "inout.h"
#define BOOL int
#define TRUE 1
#define FALSE 0

BOOL testPlusGrand(int x, int y){
  if (x >= y)
    return TRUE;
  else
    return FALSE;
}
BOOL testPlusPetit(int x, int y){
  if (x <= y)
    return TRUE;
  else
    return FALSE;
}
BOOL testTrie3(int x, int y, int z){
  if (x<= y && y <= z)
    return TRUE;
  else
    return FALSE;
}

BOOL testDeuxEgaux(int x, int y,int z, int t){
  if (x== y || x==z || x== t || y==z || y==t || z== t)
    return TRUE;
  else
     return FALSE;
}
void main ( void ) {
  BOOL res =  testPlusGrand(2,3);
  ecrireInt(res);
  res = testPlusPetit(2,3);
  ecrireInt(res);
  res = testTrie3(5,6,7);
  ecrireInt(res);
  res = testDeuxEgaux(3,5,3,4);
  ecrireInt(res);
}
```