PLATYPUS

esigetel PARIS SUD efrei PARIS SUD Ingénieurs en mouvement

Techniques et outils de programmation
Cahier de TD et de début de TP
L1 2020 (2015-16)
TD n°1

Entrées/ sorties, variables, types, déclarations, affectations, expressions

Exo 1 Affichage

Concevoir un algorithme qui affiche le message 'Hello World !'

Exo 2 Déclaration d'une variable, affectation initiale (initilisation)

Concevoir un algorithme qui initialise une variable entière à 5.

Exo 3 Affichage et saisie

Concevoir un algorithme qui demande son nom à l'utilisateur, par exemple 'John', puis lui affiche le message 'Bonjour John, comment allez-vous ?'

Exo 4 Évaluation d'une expression et affectation du résultat

Concevoir un algorithme qui initialise deux variables entières *x* et *y* à l'aide de la saisie utilisateur puis qui affiche le résultat du produit sous la forme, par exemple avec x = 2 et y = 3, '2 x 3 = 6'. Dans une première version, utiliser une variable intermédiaire *z* pour stocker le résultat avant de l'afficher. Dans une seconde, se passer de la variable *z*.

Exo 5 Affectation successives (changements d'état d'une variable)

Concevoir un algorithme qui initialise deux variables entières *n* et *p*2 puis affiche les 6 premières puissances de 2 successives sous la forme suivante :

0) 1
1) 2
2) 4
3) 8
4) 16
5) 32

A chaque étape de votre algorithme, vous devez réutilisez les valeurs précédentes de *n* et *p*2 pour déterminer leurs valeurs suivantes.

Ex 6 Transposition

Explorer toutes les manières de transposer les valeurs de deux variables (de même type). Concevoir une technique de rotation des valeurs sur une série de 5 variables.

Ex 7 Synthèse

Concevoir un algorithme qui affiche les 10 premiers termes de la suite de Fibonacci en n'utilisant pour cela que deux variables.

PLATYPUS

esigetel PARIS SUD efrei PARIS SUD Ingénieurs en mouvement

Techniques et outils de programmation en C
Cahier de TD et de début de TP
L1 2020 (2015-16)
TD n°2

Branchements conditionnels, expression de conditions

Objectifs pédagogiques :

- Branchement simples
- Branchements imbriqués
- Conditions complexes
- Synthèse

Ce TD est à compléter chez soi en langage algorithmique.
La traduction en C sera effectuée au début de la séance de TP suivante.

Branchements simples

Exo 1 Pair / impair

Concevoir un algorithme qui prend en paramètre un entier et qui affiche la valeur de cet entier suivie du texte ' est pair' si l'entier est pair et du texte ' est impair' sinon.

Exo 2 Supérieur / inférieur

Concevoir un algorithme qui prend en paramètre un entier *x*, demande à l'utilisateur de saisir un entier *y* puis indique par un message si *y* est supérieur ou inférieur à *x*.

Branchements imbriqués

Exo 3 Intervalle

Concevoir un algorithme qui prend en paramètre deux entiers *a* et *b*, s'arrête avec un message d'erreur si $a > b$, et demande sinon à l'utilisateur de saisir un entier *y* puis indique par un message si *y* est dans l'intervalle [*a*, *b*] ou non.

Exo 4 Âge et sexe

Concevoir un algorithme qui demande à l'utilisateur son nom, son année de naissance et son sexe (0 : M ou 1 : F) puis qui affiche un message qui salue l'utilisateur, indique son âge, en utilisant la première personne si l'utilisateur a moins de 16 ans, la troisième sinon, enfin s'il est 'homme' ('garçon' si moins de 16 ans) ou 'femme' ('fille' si moins de 16 ans).
Exemples :

- Bonjour Bertand, tu as 5 ans et tu es un garçon.
- Bonjour Jasmine, tu as 15 ans et tu es une fille.
- Bonjour Pierre, vous avez 50 ans et vous êtes un homme.
- Bonjour sophie, vous avez 16 ans et vous êtes une femme.

Conditions complexes

Exo 5 Conjonction

Concevoir un algorithme qui demande à l'utilisateur trois entiers *x*, *y* et *z*, et qui retourne vrai si *x* = *y* = *z* et faux sinon.

Ex 6 Disjonction

Concevoir un algorithme qui demande à l'utilisateur trois entiers *x*, *y* et *z*, et qui retourne vrai si deux au moins des trois variables sont égales, et faux sinon.

Ex 7 Exclusivité

Concevoir un algorithme qui demande à l'utilisateur trois entiers *x*, *y* et *z*, et qui retourne vrai si deux est seulement deux des trois variables sont égales, et faux sinon.

Synthèse

Exo 8 Calculatrice

Concevoir un algorithme qui demande à l'utilisateur deux entiers *x* et *y*, les opérandes, et un caractère *op*, l'opérateur à choisir parmi 'x', '/', '+', '-'. Suivant l'opérateur effectuer puis afficher l'opération, par exemple '2 x 3 = 6'. Si l'opérateur est '/' et y vaut 0, ne pas effectuer l'opération et afficher le message d'erreur 'Division par 0!'.

Exo 9 Jeu de la vie

Soit un carré composé de 3 x 3 cellules qui peuvent être vivantes ou mortes. On nomme les 8 cellules de bord par les points cardinaux. Par exemple, la cellule *sw* est la cellule située au sud-ouest, la cellule *ee* celle qui est plein est. La cellule centrale est nommée *c*. La règle du jeu de la vie stipule qu'à la génération suivante :

- si *c* est morte et entourée par 3 cellules vivantes, alors elle devient vivante.
- si *c* est vivante elle survit ssi elle est entourée de 2 ou 3 cellules vivantes.

Concevoir un algorithme qui prend en paramètre *nw*, *nn*, *ne*, *ee*, *se*, *ss*, *sw*, *ww, c*, neuf booléens, chacun vrai si la cellule correspondante est vivante, faux sinon, et qui retourne un booléen indiquant si la cellule centrale est vivante ou morte à la génération suivante.

Exo 10 Résolution d'équations du second degré

Concevoir un algorithme qui saisit des coefficients réels *a*, *b* et *c* d'un polynôme $P[X] = aX^2 + bX + c$ puis qui affiche les racines réelles ou complexes de l'équation $P[X] = 0$.
Les plus motivés pourront répéter l'exercice avec les équations de degré 3.

PLATYPUS

esigetel PARIS SUD efrei PARIS SUD Ingénieurs en mouvement

Techniques et outils de programmation en C
Cahier de TD et de début de TP
L1 2020 (2015-16)
TD n°3

Répétitions avec les boucles ttq et ftq

Objectifs pédagogiques :

- Applications des boucles simples
- Synthèse

Ce TD est à compléter chez soi en langage algorithmique.
La traduction en C sera effectuée au début de la séance de TP suivante.

Applications des boucles simples

Boucle principale d'un programme

Ex 1 Menu principal d'un programme

Concevoir un algorithme qui :

1. affiche un menu comportant les choix a, b, c proposés à l'utilisateur
2. suivant le choix de l'utilisateur :
    a. affiche 'Hello World !' et repart en 1
    b. affiche 'How are you ?' et repart en 1
    c. termine le programme
    d. choix incorrect : affiche un message d'erreur et repart en 1

Le menu à afficher est le suivant :

a. Execute HelloWorld
b. Execute HowAreYou
c. Quit

Calcul d'une suite de valeurs

Ex 2 Suite de Fibonacci

Reprendre l'exercice de synthèse 7 du TD 1 pour calculer et afficher les premiers termes de la *suite de Fibonacci* jusqu'à un rang saisi par l'utilisateur.

Calcul itératif avec critère de convergence

Ex 3 Calcul de PI

$$\sum_{n=0}^{\infty} \frac{(-1)^n}{2n+1} = \frac{1}{1} - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \cdots = \frac{\pi}{4}.$$

Suivant la formule de Leibniz, ci-dessus, concevoir un algorithme itératif qui permet (en théorie, le TP étudiera ce qu'il se passe en pratique) de calculer *pi* avec une précision *e* fixée par l'utilisateur.

Dessiner en 2D

Ex 4 Dessins 2D

Dessiner un damier de taille *n* x *n* dont chaque cellule est constituée de *p* x *p* pixels représentés par un caractère, le caractère blanc pour le noir, et le caractère '*' pour le blanc.

Dessiner, en blanc sur fond noir :

1. un damier simple (chaque pixel alterne en couleur avec ses voisins directs)
2. une série de lignes diagonales montantes à raison d'une sur trois
3. une série de lignes horizontales, à raison d'une sur d'une sur trois
4. une série cadres concentriques, en partant de la rangée des pixels de bord, et à raison d'un cadre toutes les 3 rangées internes
5. défi : tracer une spirale

Exercices de synthèse

Ex 5 Interaction utilisateur

Demander à l'utilisateur de saisir des nombres positifs ou nuls et les sommer à mesure qu'il les saisit. Quand l'utilisateur saisit un nombre négatif, l'algorithme se termine après avoir affiché la somme des nombres positifs précédents. Si la première saisie de l'utilisateur est un nombre négatif, le programme retourne 0.

Ex 6 Suite de Syracuse

Pour rappel, la *suite de Syracuse* est la suite qui calcule chaque (nouveau) terme à partir du précédent, en divisant ce dernier par 2 s'il est pair, et en le multipliant par 3 et en lui ajoutant 1 s'il est impair.

Concevoir et réaliser un programme qui génère la suite de Syracuse de terme initial *un entier strictement positif* saisi par l'utilisateur, et qui s'arrête quand est généré le premier terme de valeur 1. Chaque terme de la suite est affiché sur une nouvelle ligne, sous la forme suivante 's(' + *n* + ') = ' + s_n, où *n* est le rang d'itération et s_n le terme de rang *n* (le terme initial saisi par l'utilisateur est s_0 et doit être le premier terme affiché, de même que le dernier terme valant 1).

En option, vous pouvez permettre à l'utilisateur de générer autant de suites de Syracuse qu'il le souhaite. La suite de Syracuse de terme initial 27 est spécialement intéressante.

PLATYPUS

esigetel PARIS SUD efrei PARIS SUD Ingénieurs en mouvement

Techniques et outils de programmation en C
Cahier de TD et de début de TP
L1 2020 (2015-16)
TD n°4

Emboîtements de structures de contrôle

Objectifs pédagogiques :

- Contructions à base de boucles et de branchements conditionnels

Ce TD est à compléter chez soi en langage algorithmique.
La traduction en C sera effectuée au début de la séance de TP suivante.

Ex 2 Calcul du nombre d'occurrences

Écrire un programme qui demande à l'utilisateur de donner 10 nombres entiers, puis de calculer le nombre fois qu'il a donné un nombre négatif (ou un nombre donné en paramètre).

Exemple : Nombres : 10, -3, 5, 7, 9, -17, 45, 6, -2, 34,

Résultat: vous avez entrez 3 nombres négatifs

Ex 3 Calcul de moyenne sur plusieurs séries de nombres

Écrire un programme qui demande à l'utilisateur le nombre de séries (pour lesquelles la moyenne doit être calculée) puis les n (le nombre n doit être saisi également par l'utilisateur) nombres correspondants à chaque série.

Exemple :

Nombre de séries : 3

Serie 1: n= 4, N1=10, N2=5, N3=20, N4=2, Moyenne = 9.25

Serie 2: n= 2, N1=7, N2=25, Moyenne = 16

Serie 3: n= 5, N1=12, N2=15, N3=30, N4=8, N5=6, Moyenne = 14.2

Ex 3 Triangle de Pascal

Écrire un programme qui prend comme paramètre un entier n et affiche les n premières lignes du triangle de Pascal de la façon suivante

Exemple : n= 5

```
          1
        1   1
      1   2   1
    1   3   3   1
  1   4   6   4   1
```

Ex 4 Table de multiplication

Écrire un algorithme qui permet d'affichet la table de multiplications de la manière suivante :

```
    I   1   2   3   4   5   6   7   8   9  10
-------------------------------------------------
 1  I   1   2   3   4   5   6   7   8   9  10
 2  I   2   4   6   8  10  12  14  16  18  20
 3  I   3   6   9  12  15  18  21  24  27  30
 4  I   4   8  12  16  20  24  28  32  36  40
 5  I   5  10  15  20  25  30  35  40  45  50
 6  I   6  12  18  24  30  36  42  48  54  60
 7  I   7  14  21  28  35  42  49  56  63  70
 8  I   8  16  24  32  40  48  56  64  72  80
 9  I   9  18  27  36  45  54  63  72  81  90
10  I  10  20  30  40  50  60  70  80  90 100
```

Ex 4 Carré magique

a) Ècrire un algorithme qui prend en donnée un entier n et qui demande à l'utilisateur de donner les (n x n) valeurs du carré.

b) Ècrire un algorithme qui permet de vérifier si un carré est magique (la somme des valeurs sur chaque ligne, sur chaque colonne et sur les deux diagonales est la même). Exemple :

14	5	17
15	12	9
7	19	10

c) Ecrire un algorithme qui permet de vérifier si un carré est magique parfait (c'est un carré magique qui contient toutes les valeurs de 1 à n^2).
Exemple :

8	1	6
3	5	7
4	9	2

PLATYPUS

esigetel PARIS SUD efrei PARIS SUD Ingénieurs en mouvement

Techniques et outils de programmation en C
Cahier de TD et de début de TP
L1 2020 (2015-16)
TD n°5 et ½ 6

Fonctions, tableaux, boucle pour

Objectifs pédagogiques :

- Applications des fonctions
- Applications des tableaux et boucles pour
- Synthèse

Ce TD est à compléter chez soi en langage algorithmique.
La traduction en C sera effectuée au début de la séance de TP suivante. Il est recommandé de prendre de l'avance chez soi.

Application des tableaux et boucles pour

Ex 1 Déclaration, taille utile, initialisation du tableau

A) Concevoir un algorithme qui initialise un tableau de taille maximum *N* avec les valeurs saisies par l'utilisateur :

- L'affectation des cellules du tableau se fait suivant l'ordre des index.
- Les cellules sont affectées des valeurs successives saisies par l'utilisateur.

L'algorithme se termine quand le tableau est plein.
B) Même question avec un tableau 2D de taille maximum *N* x *N* affecté suivant le parcours de lecture occidental usuel (chaque ligne de gauche à droite, de la ligne du haut à la ligne du bas).

Ex 2. Nombre d'éléments pairs

Concevoir un algorithme qui retourne le nombre d'éléments pairs d'un tableau d'entiers.

Ex 3. Afficher le contenu d'un tableau 2D

Concevoir un algorithme qui affiche un tableau 2D de taille *N* x *N* ligne par ligne en indiquant par le symbole ↩ qu'il faut revenir à la ligne.

Ex 4. Prettynit

Concevoir un algorithme qui initialise un tableau 2D de taille *N* x *N* en :

1. Affectant la liste des N premiers nombres premiers à la première ligne.
2. En calculant ainsi les valeurs des lignes suivantes :
    a. $a_{i+1,j} \leftarrow |a_{i,j+1} - a_{i,j}|$
    b. $a_{i+1,n-1} \leftarrow |a_{i,0} - a_{i,n-1}|$

En TP, vous pourrez vous amuser à l'afficher avec des couleurs : demandez conio !

Ex 5. Copie

Concevoir un algorithme qui duplique un tableau 2D de taille *N* x *N*.

Ex 6. Valeur maximale d'un tableau

Concevoir deux algorithmes qui retournent respectivement :

1. la valeur maximale d'un tableau d'entiers de taille maximale *N* et de taille utile *sz*,
2. la seconde plus grande valeur du même tableau.

Ex 7. Moyenne et écart type

Concevoir un algorithme qui retourne la moyenne et l'écart type sur l'ensemble des valeurs d'un tableau d'entiers.

Ex 8. Recherche dichotomique

Concevoir un algorithme qui recherche et retourne l'index d'un élément d'un tableau ordonné croissant. Si aucun élément ne correspond à la clé de recherche, l'algorithme renvoie l'index -1. Réaliser un algorithme performant en O(ln(*n*)).

Ex 9. Motif

Concevoir un algorithme qui recherche et retourne la plus longue séquence de valeurs successives qui se répète au moins une fois dans un tableau. Cette séquence est retournée sous forme d'un tableau.

Ex 10. Intersection

Concevoir un algorithme qui retourne un tableau contenant copie des éléments communs de deux tableaux donnés, y compris les doublons communs.
Par ex. avec {1, 2, 3, 1, 2, 3, 1, 3} et {1, 2, 3, 4, 5, 6, 1, 2} cela retourne par ex. {1, 1, 2, 2, 3}

Ex 11. Partition

Concevoir un algorithme qui retourne, sur un tableau :

1. Le nombre de valeurs distinctes apparaissant dans ce tableau.
2. Pour chaque valeur, son nombre de représentants.

Home & TP

Ex 12. Chiffre de Vigenère !

A l'aide d'un tableau contenant les lettres d'un message (uniquement des lettres alphabétiques majuscules), et d'un second tableau contenant celles d'un clé de chiffrement :

1. concevoir un algorithme qui produit un troisième tableau contenant le cryptogramme,
2. concevoir un algorithme qui prend cryptogramme et clé en argument et déchiffre,
3. pour les plus motivés, il sera intéressant de tenter la cryptanalyse de ce chiffre.

Pour cet exercice, Internet est votre ami.

Ex 13. Transformer un tableau - opérations sur les matrices

Étant donné un tableau 2D de dimension *n* x *n* représentant une matrice (aij) à coefficients réels, concevoir des algorithmes pour :

A. Transposer la matrice :
    a. une première version produit une matrice résultat de sortie distincte de la matrice donnée en entrée.
    b. une seconde version transforme directement la matrice d'entrée pour produire la matrice de sortie. Pour cela, cette version doit utiliser un minimum de variables locales et d'affectations.
B. Retourner la somme, la différence, le produit et le quotient de 2 matrices n x n.

   a. même distinction de versions que pour A.
   b. pour le quotient, s'assurer qu'il est défini, sinon afficher un message d'erreur et interrompre l'opération.
C. Calcul le déterminant de la matrice

Ex 14. Jeu de la vie complet

Vous réutiliserez votre algorithme du T2 Ex 9 Jeu de la vie,
nommé par ex. *generate_cell*(*nw*, *nn*, *ne*, *ee*, *se*, *ss*, *sw*, *ww*) et retournant la valeur de la cellule dont on calcule le statut à la génération suivante.
Construire un algorithme principal *generate_state* qui invoque *generate_cell* pour calculer une nouvelle génération du jeu de la vie à partir de la génération précédente :

A. Une génération du JDV est un état d'une surface de cellules représentée par un tableau 2D de N x N bits, valant 1 si la cellule est vivante, 0 sinon.
B. Soit *s* le tableau représentant l'état courant du jeu de la vie (celui qui vient de s'afficher), soit *next_s* le tableau représentant l'état de jeu à la génération suivante, *generate_state* doit appliquer l'affectation suivante à l'ensemble des positions de la surface de cellules :
   a. $next\_s_{i,j}$ <- *generate_cell*(
      nw(s, i, j), *nn*(s, i, j), *ne*(s, i, j),
      ee(s, i, j), *se*(s, i, j),
      ss(s, i, j), *sw*(s, i, j), *ww*(s, i, j)
)
C. Il faudra définir une politique paramétrable pour la gestion des cellules du bord (index 0 ou n - 1 sur l'une des deux dimensions) :
   a. Soit en considérant qu'au delà du bord, c'est mort.
   b. Soit en transformant la surface en tore (les bords opposés sont mis en correspondance comme si on repliait deux bords pour former un cylindre, puis les deux bords du cylindre pour former un tore).

Construire un nouvel algorithme JDV qui utilise :

1. generate_state pour générer un nombre infini ou paramétrable par l'utilisateur de générations successives du jeu de la vie.
2. une fonction auxiliaire d'affichage print_state qui permet de rafraîchir l'écran à chaque nouvelle génération.