Université des Antilles et de la Guyane

L2-MI

UEO35 Algorithmique et structure de données avancées

TD / TP 2011-2012

UEO35 Algorithmique et structure de données avancées

TD 1
Variables et structures de contrôle

Pour chaque exercice, on écrira successivement le pseudo-code et le code C du programme. On prendra soin de bien commenter les programmes.

Exercice 1
Ecrire un programme qui lit le prix HT d'un article, le nombre d'articles et le taux de TVA (5,5% par exemple) et qui affiche le prix total TTC correspondant, puis le prix soldé (-30% par exemple).

Exercice 2
Ecrire un programme qui demande deux nombres à l'utilisateur et détermine ensuite si leur produit est négatif ou positif (on laisse de côté le cas où le produit est nul) en calculant le produit des deux nombres puis, sans le calculer.

Exercice 3
Nombre secret : écrire un programme qui demande à l'utilisateur 1 d'entrer un nombre et à l'utilisateur 2 de le trouver en affichant, à chaque tentative, « trop grand » si le nombre entré est plus grand que le nombre secret, « trop petit » sinon. Le programme s'arrête quand l'utilisateur 2 a trouvé le nombre secret.

Exercice 4
Ecrire le code C du programme qui affiche le texte suivant pour les chiffres de 1 à 10 :

```
1
2       2
3       3       3
…
```

Exercice 5
Ecrire un algorithme qui demande un nombre de départ et qui ensuite affiche les cinq nombres impairs suivants. Par exemple, si l'utilisateur entre le nombre 17, le programme affichera les nombres de 19 à 27.

Exercice 6
Ecrire un programme qui demande le numéro du mois (entre 1 et 12) à l'utilisateur et affiche le nombre de jours du mois correspondant (on laisse de côté le cas des années bissextiles).

Exercice 7
Ecrire un programme qui demande les coefficients d'une équation du second degré et affiche les racines réelles si elles existent (on utilisera la fonction *sqrt* de la bibliothèque *math.h*).

Exercice 8
Soit une fonction f sur [a, b] et telle que f(a) f(b) < 0.
La méthode de recherche par dichotomie permet de trouver la racine de la fonction f, c'est à dire la valeur r telle que f(r) = 0.
Pour cela, on répète les étapes suivantes, jusqu'à convergence :

1. déterminer m = (a + b) / 2 et calculer v = f(m);
2. si f(m) f(a) > 0, remplacer a par m, sinon remplacer b par m.

Coder cet algorithme en C.

UEO35 Algorithmique et structure de données avancées

TD 2
Tableaux

Pour chaque exercice, on écrira successivement le pseudo-code et le code C du programme. On prendra soin de bien commenter les programmes.

Exercice 1
Ecrire un programme permettant à l'utilisateur de saisir un nombre N (inférieur à 100) d'entiers qui devront être stockées dans un tableau T. L'utilisateur doit donc commencer par entrer le nombre de valeurs qu'il compte saisir. Il effectuera ensuite cette saisie. Enfin, une fois la saisie terminée, le programme affichera le nombre de valeurs négatives et le nombre de valeurs positives.

Exercice 2
Ecrivez un algorithme qui inverse l'ordre des éléments du tableau T dont on suppose qu'il a été préalablement saisi (« les premiers seront les derniers… »). Utiliser une seule variable de type tableau.

Exercice 3
Ecrire un programme qui recherche un entier X dans le tableau T et affiche soit son index s'il est présent, soit « absent ». Traiter aussi le cas où plusieurs occurrences de X sont présentes dans le tableau.

Exercice 4
Le tri par insertion est une méthode utilisée pour trier des tableaux. L'algorithme, pour un tableau T de taille n est donné ci-dessous. Expliquer en traitant l'exemple T[5]={4, 2, 0, 5, 3}.

```
pour i de 1 à n - 1
    x <- T[i]
    j <- i
    tant que j > 0 et T[j - 1] > x
        T[j] <- T[j - 1]
        j <- j – 1
    fin tant que
    T[j] <- x
fin pour
```

Ecrire le code C correspondant.

Exercice 5
Programmer la recherche dichotomique d'un élément X du tableau T supposé trié.

Exercice 6
Ecrire un programme qui transfère un tableau M à deux dimensions L et C (dimensions maximales: 10 lignes et 10 colonnes) dans un tableau V à une dimension L*C.

Exercice 7
Ecrire un programme qui range les index de toutes les occurrences du tableau M (à deux dimensions) telles que M[l][c] > X dans deux tableaux indexl et indexc.

UEO35 Algorithmique et structure de données avancées

TD 3
Structures de Données

On pourra utiliser des fonctions dans les deux exercices.

Exercice 1

Soit le modèle de structure suivante, définissant un nombre complexe en représentation cartésienne :

```
struct Complex
{
        float re ;
        float im ;
}
```

1) Ecrire un programme permettant de saisir deux nombres complexes et de calculer leur somme et leur produit.
2) Compléter le programme afin de déterminer la représentation polaire de ces nombres (on pourra éventuellement modifier le modèle de structure). Programmer de même le passage de la représentation polaire à la représentation cartésienne.

Exercice 2

Soit le modèle de structure suivante :

```
struct Point
{
        char c ;
        int x,y ;
}
```

1) Ecrire un programme qui permette de lire le nom et les coordonnées d'un point au clavier.
2) Définir un modèle de structure **segment** qui permette de représenter un segment du plan à l'aide de deux points.Compléter le programme afin d'initialiser le segment en saisissant les coordonnées de ses deux extrémités et de calculer sa longueur.
3) Définir un modèle de structure **rectangle**.
4) Compléter le programme afin de saisir les coordonnées des coins du rectangle, de calculer son aire et de vérifier s'il s'agit ou non d'un carré.
5) Définir un modèle de structure **ligne_brisee** contenant un nombre quelconque de segments.
6) Compléter le programme afin de saisir les coordonnées des extrémités d'une ligne brisée et de calculer sa longueur. prévoir d'arrêter la saisie à la demande de l'utilisateur.

UEO35 Algorithmique et structure de données avancées

TD 4
Pointeurs (1)

Exercice 1
Préciser les valeurs de A, B, P1 et P2 après chacune des instructions suivantes.

```
main()
{
   int A = 1, B = 2, C = 3, *P1, *P2 ;

   P1 = &A ;
   P2 = &C ;
   *P1 = (*P2)++ ;
   P1 = P2 ;
   P2 = &B ;
   *P1 -= *P2 ;
}
```

Exercice 2
Dans le programme ci-dessous, que réalise chacune des instructions ?

```
int main(void)
{
   int *px, y, x[20];

   px = &x[0];
   px = x;
   px = px + 1;
   px = &x[0] + 1;
   px = x + 1;
   y  =*px + 1 ;
   *px = *px + 10 ;
   *px += 2 ;
}
```

Exercice 3
Soit P un pointeur sur un tableau A:

```
   int A[] = {12, 23, 34, 45, 56, 67, 78, 89, 90}, *P ;
   P = A ;
```

Quelles valeurs ou adresses fournissent ces expressions ?

a)	*P+2	e)	A+3
b)	*(P+2)	f)	&A[7]-P
c)	&P+1	g)	P+(*P-10)
d)	&A[4]-3	h)	*(P+*(P+8)-A[7])

Exercice 4
Traiter l'ensemble du TD2 en utilisant le formalisme pointeur.

UEO35 Algorithmique et structure de données avancées

TD 5
Pointeurs (2)

On utilisera des pointeurs dans tous les exercices.

Exercice 1
Un tableau A de dimension N+1 contient N valeurs entières triées par ordre croissant; la $(N+1)^{ième}$ valeur est indéfinie. Insérer une valeur VAL donnée au clavier dans le tableau A de manière à obtenir un tableau de N+1 valeurs triées.

Exercice 2
Ecrire un programme qui calcule le produit scalaire de deux vecteurs U et V (tableaux d'entiers de même dimension).

Exercice 3
On dispose de deux tableaux A et B (de dimensions respectives N et M), triés par ordre croissant. Fusionner les éléments de A et B dans un troisième tableau FUS trié par ordre croissant.

Méthode: Utiliser trois indices IA, IB et IFUS. Comparer A[IA] et B[IB]; remplacer FUS[IFUS] par le plus petit des deux éléments; avancer dans le tableau FUS et dans le tableau qui a contribué son élément. Lorsque l'un des deux tableaux A ou B est épuisé, il suffit de recopier les éléments restants de l'autre tableau dans le tableau FUS.

Ecrire l'algorithme puis le programme qui utilisera des pointeurs.

Exercice 4
Ecrire un programme qui permet à l'utilisateur de saisir les dimensions de deux tableaux A et B, alloue dynamiquement la mémoire, laisse l'utilisateur saisir le contenus des tableaux et les concatène dans un tableau C en prenant un élément sur deux des tableaux A et B.
C = { A[0] , B[0] , A[2] , A[2] …}

Exercice 5
Ecrire un programme qui lit deux chaînes de caractères (fonction *gets*), et qui indique leur précédence lexicographique dans le code de caractères de la machine (ici: code ASCII).

Exercice 6
Ecrire un programme qui lit deux chaînes de caractères CH1 et CH2 au clavier et élimine toutes les lettres de CH1 qui apparaissent aussi dans CH2. Utiliser deux pointeurs P1 et P2, une variable logique TROUVE et la fonction *strcpy(A,B)* qui copie la chaîne B dans la chaîne A.

UEO35 Algorithmique et structure de données avancées

TP 1 : Préliminaires

1) Prise en main de l'environnement de programmation Visual C

2) Exercice 1

Ecrire un programme qui affiche la différence A – G entre la moyenne arithmétique A et la moyenne géométrique G de deux nombres réels a et b saisis par l'utilisateur.

avec $A = \frac{a+b}{2}$ et $G = \sqrt{a*b}$

On utilisera la fonction racine carrée (sqrt : *square root*) de la bibliothèque math.h.

3) Exercice 2

Ecrire un programme qui, pour une somme donnée en euros (sans centimes) affiche le nombre minimal de billets nécessaires pour la composer.

Exemple : 1949 = 3 × 500 + 2 × 200 + 2 × 20 + 1 × 5 + 2 × 2

4) Exercice 3

Ecrire un programme qui permet à l'utilisateur de saisir une série de nombres positifs dont on calculera la moyenne. Le programme s'arrête quand l'utilisateur saisit un nombre négatif.

5) Exercice 4

Affiche la table des produits pour N variant de 1 à 10 :

```
 X*Y  0   1   2   3   4   5   6   7   8   9  10
-------------------------------------------------
  0   0   0   0   0   0   0   0   0   0   0   0
  1   0   1   2   3   4   5   6   7   8   9  10
  2   0   2   4   6   8  10  12  14  16  18  20
  3   0   3   6   9  12  15  18  21  24  27  30
  4   0   4   8  12  16  20  24  28  32  36  40
  5   0   5  10  15  20  25  30  35  40  45  50
  6   0   6  12  18  24  30  36  42  48  54  60
  7   0   7  14  21  28  35  42  49  56  63  70
  8   0   8  16  24  32  40  48  56  64  72  80
  9   0   9  18  27  36  45  54  63  72  81  90
 10   0  10  20  30  40  50  60  70  80  90 100
```

UE035 Algorithmique et structure de données avancées

TP 2
Tableaux

Exercice 1
Etant donné un tableau T de N nombres positifs ou nuls, écrire le programme qui le tasse, c'est à dire qui détecte les éléments nuls du tableau et qui récupère leur place en décalant vers le début du tableau tous les autres éléments. Tester ce programme avec le tableau :

T = {10 , 2 , 0 , 5 , 4 , 0 , 0 , 7 , 0 , 1} résultat : T = {10 , 2 , 5 , 4 , 7 , 1, 0 , 0 , 0 , 0}

Exercice 2
Ecrire un programme qui génère automatiquement un tableau T1 de taille 16 rempli d'entiers aléatoires compris entre 0 et 20. Pour ce faire, utiliser les fonctions *rand* et *srand* dont on recherchera le fonctionnement sur Internet.

Ou
Continuer le TP en définissant T1 = {20 , 16 , 4 , 8 , 1 , 14 , 2 , 5 , 17 , 10 , 12 , 15 , 18 , 3 , 0 , 19}

Exercice 3
Programmer en C l'algorithme du tri par sélection (TD2-Ex4) et l'utiliser pour trier le tableau T1.

Exercice 4
Ecrire un programme pour rechercher un élément X du tableau trié T1 par recherche dichotomique.

Rappel :
La dichotomie (« couper en deux » en grec) est, en algorithmique, un processus itératif ou récursif de recherche où, à chaque étape, on coupe en deux parties égales un espace de recherche qui devient restreint à l'une de ces deux parties.
On suppose bien sûr qu'il existe un test relativement simple permettant à chaque étape de déterminer dans laquelle des deux parties se trouve une solution.
L'algorithme s'applique typiquement à la recherche d'un élément dans un ensemble fini et ordonné.
(source : Wikipedia)

UE023 Algorithmique et structure de données avancées

TP 3
Pointeurs

Exercice 1
Comprendre le fonctionnement du programme suivant. Le compléter pour afficher le tableau.

```
#include <stdio.h>
main()
{
  /* Déclarations */
  int A[10]; /* tableau */
  int *P;    /* pointeur dans A */

  /* Saisie des données */
  printf("Introduire 10 entiers : \n");
  for (P=A; P<A+10; P++)
  {
     printf("Elément %d : ", P-A+1);
     scanf("%d", P);
  }
}
```

On utilisera pour les exercices 2 et 3 le tableau A = {2 , 10 , 0 , 5 , 4 , 2 , 0 , 0 , 1 , 1}

Exercice 2
Compléter le programme pour lire un entier X au clavier et éliminer toutes les occurrences de X dans A en tassant les éléments restants. On utilisera deux pointeurs P1 et P2 pour parcourir le tableau.

Exercice 3
Compléter le programme pour ranger les éléments de A dans l'ordre inverse. On utilisera des pointeurs P1 et P2 et une variable numérique AIDE pour la permutation des éléments.

Exercice 4
Ecrire un programme qui lit (fonction `gets`) une chaîne de caractères CH et détermine la longueur de la chaîne à l'aide d'un pointeur P. Le programme n'utilisera pas de variables numériques.

Exercice 5
Ecrire un programme qui lit une chaîne de caractères CH au clavier et qui compte les occurrences des lettres de l'alphabet en ne distinguant pas les majuscules et les minuscules. Utiliser un tableau ABC de dimension 26 pour mémoriser le résultat et un pointeur PCH pour parcourir la chaîne CH et un pointeur PABC pour parcourir ABC. Afficher seulement le nombre des lettres qui apparaissent au moins une fois dans le texte.

UE035 Algorithmique et structure de données avancées

TP 4
Fonctions

On utilisera dans cette séance des fonctions et des pointeurs sur des tableaux alloués dynamiquement (mais pas de liste chaînée).

Enoncé :
On désire pouvoir disposer d'un espace de stockage (tableau) de réels dont on ne connaît pas la taille à l'avance.

A partir d'un tableau initialement vide, on désire pouvoir :

- ajouter un réel à la fin du tableau,
- rajouter un réel à une position donnée (afin que le tableau soit trié),
- supprimer un réel (puis tasser le tableau),
- vider le tableau.

L'ensemble de ces opérations (fonctions) seront proposées à l'utilisateur à l'aide d'un menu servant d'interface.

Travail à réaliser :
Programmer et tester (dans l'ordre) les fonctions *ajoutfin*, *ajoutpos*, *suppr*, *vider* et *menu* et les utiliser dans une fonction principale appelante.

On montre ci-dessous un exemple d'ajout à la fin, d'ajout avec tri et de suppression. Le réel ajouté (ou supprimé) apparaît en gris.