UNIVERSITÉ DE PAU ET DES PAYS DE L'ADOUR

t.d.

Structures de données et types abstraits

informatique / licence 2

1. Écrire en langage C le code d'une fonction `incremente` prenant en argument l'adresse d'un entier et augmentant de 1 la valeur de cet entier. La signature de la fonction est

```
void incremente(int *a);
```

2. Écrire en langage C le code d'une fonction `swap` prenant en argument les adresses de deux entiers et échangeant les valeurs de ces deux entiers. La signature de la fonction est

```
void swap(int *a, int *b);
```

3. Le fichier en-tête `stdlib.h` comporte la définition de type suivante :

```
typedef struct {
    int quot; /* Quotient */
    int rem;  /* Remainder */
} div_t;
```

ainsi qu'une fonction de calcul de la division euclidienne, de signature

```
div_t div(int numerator, int denominator);
```

Écrire un programme complet et fonctionnel en langage C effectuant la division euclidienne de 25 par 8 et affichant le quotient et le reste de cette division.

4. On modélise un arbre généalogique à l'aide de la structure suivante :

```
struct individu {
    int id;
    char * nom;
    struct individu * pere;
    struct individu * mere;
};
typedef struct individu individu_t;
```

Écrire en langage C le code d'une fonction `meme_fratrie` de signature

```
bool meme_fratrie(individu_t *a, individu_t *b);
```

qui teste si deux individus ont le même père et la même mère.

5. Quel est le résultat de la suite d'instructions suivantes :

```
a = a + b
b = a - b
a = a - b
```

Formuler la réponse de la façon la plus claire et compacte possible.

6. On souhaite créer un tableau de n entiers non signés (`unsigned int`). Le tableau doit s'appeler `data` et la taille de ce tableau est contenue dans la variable entière `n`. L'allocation de la mémoire est ici dynamique.
a) Donner l'instruction permettant de réaliser l'allocation de la mémoire et d'initialiser correctement la variable `data`.
b) Donner les lignes de code initialisant les données du tableau `data` à 0.

7. On modélise la structure hiérarchique d'une entreprise à l'aide de la structure suivante :

```
struct employe {
    int id;
    char * nom;
    int salaire;
    struct employe * responsable;
};
typedef struct employe employe_t;
```

On peut ensuite initialiser des variables du type `employe_t` de la façon suivante :

```
employe_t alice = { 0, "Alice", 4000, NULL };
employe_t bob = { 1, "Bob", 3500, &alice };
employe_t jean = { 2, "Jean", 3300, &alice };
employe_t marie = { 3, "Marie", 2700, &bob };
```

Donner les expressions en langage C permettant d'accéder :

- au salaire de l'individu `x` ;
- au nom du responsable de l'individu `x` ;
- au nom du responsable de l'individu `x` *(sans utiliser la notation `->`)* ;
- au nom du responsable du responsable de l'individu `x`.

Écrire en langage C le code d'une fonction `est_superieur` de signature

```
bool est_superieur(employe_t *a, employe_t *b);
```

qui teste si l'employé `a` est dans la chaîne des supérieurs hiérarchiques de l'employé `b`.

8. On stocke fréquemment des tableaux à deux dimensions sous la forme d'un tableau à deux dimensions en recopiant directement les rangées les unes après les autres.

par exemple

3	8	5
4	1	2
9	7	6

sera stocké sous la forme du tableau suivant :

3	8	5	4	1	2	9	7	6

On représente de cette façon un tableau à deux dimensions comportant `ys` lignes de `xs` colonnes. Les deux variables `ys` et `xs` sont de type `int` (en langage C).

a) Donner l'expression calculant la taille du tableau à une dimension.

b) Donner l'expression calculant l'indice dans le tableau à une dimension de la case de coordonnées (x, y) dans le tableau à deux dimensions.

c) Donner les expressions calculant l'abscisse `x` et l'ordonnée `y` dans le tableau à deux dimensions de la case de coordonnées `n` (de type `int`) dans le tableau à une dimension.

d)

e) On programme un jeu de Sudoku qui recourt à une grille de taille 9×9. Chaque case appartient aussi à un des neuf blocs de taille 3×3. Donner les expressions calculant l'abscisse `x0` et l'ordonnée `y0` de la première case du bloc (la case la plus en haut et la plus à gauche du bloc) auquel appartient une case quelconque de coordonnées (x, y) dans la grille principale.