Série d'exercices #9

IFT-2035

November 18, 2024

9.1 Alignement et usage mémoire

Soit le code d'une bibliothèque C pour un *arbre binaire* qui associe des petits entiers de 16bit à une valeur en virgule flottante:

```
typedef struct bt_node bt_node;
struct bt_node {
  short    key;
  double   value;
  char     kind;
  bt_node *left;
  bt_node *right;
};
typedef struct bt bt;
struct bt { bt_node *root; };

bt  *bt_alloc  (void);
void bt_insert (bt *t, short k, double v);
char bt_remove (bt *t, short k);
bt  *bt_below  (bt *t, short k);
...
```

On sait aussi que sur le système qui nous intéresse, un `char` a une taille de 1B (byte), un `short` a une taille de 2B, que les pointeurs ont une taille de 4B, que les `double` ont une taille de 8B, et que tous ces types doivent obéir l'*alignement naturel*.

1. Quelle quantité de mémoire est allouée par un appel à `bt_alloc`?
2. Indiquer tous les endroits dans la structure `bt_node` où le compilateur C va insérer du *padding*.
3. Donner la valeur de `sizeof (bt_node)`.
4. La fonction `bt_below` ne modifie aucune partie de ses arguments et renvoie un tout nouvel arbre qui contient tous les éléments dont la clé est plus petite que `k`. Sachant que `t` a 10 nœuds, combien de bytes de mémoire doit-elle allouer dans le pire des cas?

9.2 Programmation impérative en langage fonctionnel

Re-voici le code d'un évaluateur en Haskell:

```
type Var = String

-- Expressions du code source en forme ASA.
data Exp = Enum Int         -- Une constante
         | Evar Var         -- Une variable
         | Elambda Var Exp  -- Une fonction
         | Ecall Exp Exp    -- Un appel de fonction

data Val = Vnum Int           -- Un nombre entier
         | Vfun (Val → Val)  -- Une fonction

elookup ((x1,v1):env) x =
    if x == x1 then v1 else elookup env x

env0 = [("+", Vfun (λ(Vnum x1) → Vfun (λ(Vnum x2) → Vnum (x1 + x2))))]

eval env (Enum n) = Vnum n
eval env (Evar x) = elookup env x
eval env (Elambda arg body) =
    Vfun (λactual → eval ((arg,actual):env) body)
eval env (Ecall fun actual) =
    case eval env fun of
      Vfun f → f (eval env actual)
```

Changer le code en utilisant le monade `IO` pour y ajouter une fonction primitive *print_hello* dans l'environnement initial qui imprime le mémorable message "`Hello:`" suivi de son argument numérique (qui est aussi renvoyé comme valeur de retour). Pour cela, il utilisera *putStrLn* :: `String -> IO ()`.

9.3 Types et classes de types

Donner le types des expressions Haskell ci-dessous.

1. $(+)$
2. $(+)\ (3 :: Int)$
3. $\lambda x \rightarrow fst\ x + snd\ x$
4. $\lambda x\ y \rightarrow \mathsf{if}\ fst\ x = snd\ x\ \mathsf{then}\ y\ \mathsf{else}\ y + 1$
5. $map\ (\lambda x \rightarrow \mathsf{if}\ fst\ x < 0\ \mathsf{then}\ snd\ x\ \mathsf{else}\ snd\ x + 1)$