Informatique MP2I

Table des matières

1 Listes chaînées

1.1 C

1.1.1 Type complexe

Définition d'un type

```c
struct int_list {
    int value;
    struct int_list* next;
};
typedef struct int_list int_list;
// Cette définition de listes chaînées s'adapte également pour les
    autres types
```

Fonction à implémenter : Longueur d'une liste

```c
int length(int_list* lst) {
    if (lst == NULL) {
        return 0;
    }
    return 1 + length(int_list->next);
}
```

Complexité $\Theta(n)$

1.2 OCaml

1.2.1 Le module `List`

Fonction disponible : `List.length`

Description Renvoie la longueur de la liste
Complexité $\Theta(n)$
Signature `'a list -> int`

Implémentation

```ocaml
let rec length = function
    | [] -> 0
    | h::t -> 1 + length t;;
```

Complexité $\Theta(n)$

Fonction disponible : `List.iter`

Description	Applique une fonction à tous les éléments de la liste
Complexité	$\Theta(n)$
Signature	`('a -> unit) -> 'a list -> unit`

Implémentation

```
let rec iter f = function
    | [] -> ()
    | h::t -> f h; iter f t;;
```

Fonction disponible : `List.fold_left`

Description	Applique une fonction successivement à un élément de la liste et au résultat de l'itération précédente en partant de la fin de la liste
Complexité	$\Theta(n)$
Signature	`('a -> 'b -> 'a) -> 'a -> 'b list -> 'a`

Implémentation

```
let rec fold_left f acc = function
    | [] -> acc
    | h::t -> fold_left f (f acc h) t;;
```