Doc list en c

Utilisation:

inclure le header list.h et linker la lib liblinklist.so

Fonction membre:

push_back(list *this, void *data) ajoute data au debut de la liste

push_front(list *this, void *data) ajoute data à la fin de la liste

void *pop_back(list *this) retire un element au debut de la liste

void *pop_front(list *this) retire un element à la fin de la liste

int empty(list *this) renvoi true si la liste est vide sinon false

void *front(list *this) renvoie le premier element de la liste

void *back(list *this) renvoie le dernier

unsigned int size(list *this) renvoie le nombre d'element de la liste

t_list_iterator *begin(list *this) renvoie un iterateur sur le debut de la liste

t_list_iterator *end(list *this) renvoie un iterateur sur la fin de la liste

insert(list *this, t_list_iterator *to, void *data) insert l'element data apres l'iterateur to

remove_if(list *this, int (*match)(void *data, void *value), void *value)
supprime et libere en memoire les element de la liste qui matche avec value avec la fonction match

clear(list *this) vide la liste

for_each(list *this, void (*fct)(void *data)) applique fct à tous les menbres de la liste

for_each_param(list *this, void (*fct)(void *data, void *value), void *value)
applique fct à tous les membres de la liste

t_list_iterator *find(struct s_list *this, t_list_iterator *first, int (*test)(void *data, void *value), void *value)
renvoie un iterateur sur la premiere occurrence trouver qui matche avec la fonction test.

get(list *this)

Exemple:

```
#include "list.h"
#include <stdio.h>

typedef struct coord
{
  unsigned int x;
  unsigned int y;
}coord;

coord *new_coord(const unsigned int x, const unsigned int y)
{
  coord *c = malloc(sizeof(coord));

  c->x = x;
  c->y = y;
  return (c);
}

void aff_coord(void *data)
{
  coord *c = (coord *)data;

  if (data != NULL)
    printf("x = %d, y = %d\n", c->x, c->y);
}

int match(void *data, void *value)
{
  coord *c = (coord *)data;
  int *y = (int *)value;

  return (c->y == *y);
}
```

```
void my_free(void *data)
{
  free(data);
}

void put(void *data, void *value)
{
  coord *c = (coord *)data;

  c->x = *((int *)value);
}

//list.push_front(&list, new_coord(1, x));
//list.for_each(&list, &aff_coord);
int main(int ac, char **av)
{
  list            list;
  t_list_iterator *it;
  coord           *c;

if (ac != 2)
  return 1;
  init_list(&list);
  list.free = my_free;
  for (int x = 0; x < 25; ++x){
      list.push_front(&list, new_coord(1, x));
  }
  int i = 2000;
  list.for_each_param(&list, put, &i);
  int y = atoi(av[1]);
  list.for_each(&list, &aff_coord);
  list.remove_if(&list, match, &y);
  y = atoi(av[1]);
  printf("%s", "premier element:");
  aff_coord(list.back(&list));
  printf("%s", "dernier element:");
  aff_coord(list.front(&list));
  ++y;
  printf("%s", "element find:");
  if ((it = list.find(&list, list.begin(&list), match, &y)) != NULL)
    aff_coord(it->data);
```

```
 //on parcoure la liste avec les iterateurs
  for (it = list.begin(&list); it != NULL; it = it->next){
   aff_coord(it->data);
  }

 //on parcoure la liste avec le foreach

 list.clear(&list);
 return (0);
}
```

Résultat:

```
x = 2000, y = 24
x = 2000, y = 23
x = 2000, y = 22
x = 2000, y = 21
x = 2000, y = 20
x = 2000, y = 19
x = 2000, y = 18
x = 2000, y = 17
x = 2000, y = 16
x = 2000, y = 15
x = 2000, y = 14
x = 2000, y = 13
x = 2000, y = 12
x = 2000, y = 11
x = 2000, y = 10
x = 2000, y = 9
x = 2000, y = 8
x = 2000, y = 7
x = 2000, y = 6
x = 2000, y = 5
x = 2000, y = 4
x = 2000, y = 3
x = 2000, y = 2
x = 2000, y = 1
x = 2000, y = 0
premier element:x = 2000, y = 0
dernier element:x = 2000, y = 24
element find:x = 2000, y = 2
x = 2000, y = 24
```

```
x = 2000, y = 23
x = 2000, y = 22
x = 2000, y = 21
x = 2000, y = 20
x = 2000, y = 19
x = 2000, y = 18
x = 2000, y = 17
x = 2000, y = 16
x = 2000, y = 15
x = 2000, y = 14
x = 2000, y = 13
x = 2000, y = 12
x = 2000, y = 11
x = 2000, y = 10
x = 2000, y = 9
x = 2000, y = 8
x = 2000, y = 7
x = 2000, y = 6
x = 2000, y = 5
x = 2000, y = 4
x = 2000, y = 3
x = 2000, y = 2
x = 2000, y = 0
```