Les Piles

19 | 03 | 2020

Dorian.H Mekni

Les piles -> intro

- Une pile est une structure de données
- Créer | Appeler une fonction -> les variables propres à celle-ci s'empile au fur et à mesure qu'elles sont créées. On peut dire qu'elle s'empilent

FILO

- La premiere donnée rentrée est la dernière que l’on sort
- Didactique d’empilage classique
- FILO -> first in - Last out
- Le meilleur moyen d’agencer nos données c’est de faire appel à une pile.

Index des prototypes de fonctions

```c
#ifndef __STACK__H
#define __STACK__H
/*Les prototype sont définis dans le fichier headers dont l'annotation est .h
    puis utilisé par la suite dans les autres fichiers qui permet une
    organisation du code optimal. */

// Type booléen
typedef enum
{
    false,
    true
}Bool;

// Definition d'une pile
typedef struct StackElement
{
    int value;
    struct StackElement *next;
}StackElement, *Stack;

// Prototypes des fonctions
Stack new_stack(void);
Bool is_empty_stack(Stack st);
Stack push_stack(Stack st, int x);
Stack clear_stack(Stack st);
void print_stack(Stack st);
#endif /* Stack_h */
```

new_stack()

- La fonction new_stack() permet la creation d'une pile

```
Stack new_stack(void)
{

    return NULL;
}
/*--------------------------------------------------------*/

Bool is_empty_stack(Stack st){

    if(st == NULL)
        return true;

    return false;
};

/*--------------------------------------------------------*/
```

push_stack()

push_stack() permet d'ajouter une donnée à la pile.

```c
Stack push_stack(Stack st, int x)
{

    StackElement *element;

    element = malloc(sizeof(*element));

    if(element == NULL) {

        fprintf(stderr, "There is problem of dynamic allocation.\n");
        exit(EXIT_FAILURE);
    }

    element->value = x;
    element->next = st;

    return element;
}
/*-----------------------------------------------*/
```

push_stack()->ex:

```c
int main(void)
{
    Stack sta = new_stack();
    print_stack(sta);

    printf("\n/*--------------------------*/\n");

    sta = push_stack(sta, 14);
    sta = push_stack(sta, 47);

    print_stack(sta);

    printf("\n/*--------------------------*/\n");

    sta = clear_stack(sta);

    print_stack(sta);


    return 0;
}
```

```
Nothing to declare, the Stack is empty.

/*--------------------------*/
[47]
[14]

/*--------------------------*/
Nothing to declare, the Stack is empty.
Program ended with exit code: 0
```

pop_stack()

pop_stack permet de retire rune donnée d cela pile.

```
Stack pop_stack(Stack st)
{
    StackElement *element;

    if(is_empty_stack(st))
        return new_stack();

    element = st->next;

    free(st);

    return element;
}

/*-----------------------------------------------------*/
```

pop_stack()->ex:

```c
#include <stdio.h>
#include "Stack.h"

int main(void)
{
    Stack sta = new_stack();
    print_stack(sta);

    printf("\n/*--------------------------*/\n\n");

    sta = push_stack(sta, 14);
    sta = push_stack(sta, 47);
    sta = push_stack(sta, 22);
    sta = push_stack(sta, 12);
    print_stack(sta);

    printf("\n/*--------------------------*/\n\n");

    sta = pop_stack(sta);
    print_stack(sta);

    printf("\n/*--------------------------*/\n\n");

    sta = clear_stack(sta);
    print_stack(sta);


    return 0;
}
```

print_stack()

print_stack() permet l'affichage sur console des valeurs présentes au sein de la pile.

```c
void print_stack (Stack st)
{
    if(is_empty_stack(st))
    {
        printf("Nothing to declare, the Stack is empty.\n");
        return;
    }

    while(!is_empty_stack(st))
    {
        printf("[%d]\n", st->value);
        st = st->next;
    }
}
```

clear_stack()

clear_stack() reinitialise la pile à zéro valeur en vidant cette dernière de toutes ses données.

```
/*-----------------------------------------------*/
Stack clear_stack(Stack st)
{

    while(is_empty_stack(st))
        st = pop_stack(st);


    return new_stack();
}
```

stack_length()

stack_length() mesure la longueur de la pile: combien de données présentes dans cette dernière ?

```c
/*-------------------------------------------------*/

int stack_length(Stack st)
{
    int length = 0;

    while(is_empty_stack(st))
    {
        length++;
        st = st->next;
    }
    return length;
}
/*-------------------------------------------------*/
```

stack_length()->ex:

```c
int main(void)
{
    Stack sta = new_stack();
    print_stack(sta);
    printf("Size of the stack is : %d\n", stack_length(sta));

    printf("\n/*--------------------------*/\n\n");

    sta = push_stack(sta, 14);
    sta = push_stack(sta, 47);
    sta = push_stack(sta, 22);
    sta = push_stack(sta, 12);
    print_stack(sta);
    printf("Size of the stack is : %d\n", stack_length(sta));

    printf("\n/*--------------------------*/\n\n");

    sta = pop_stack(sta);
    print_stack(sta);
    printf("Size of the stack is : %d\n", stack_length(sta));

    printf("\n/*--------------------------*/\n\n");

    sta = clear_stack(sta);
    print_stack(sta);
    printf("Size of the stack is : %d\n", stack_length(sta));


    return 0;
}
```

top_stack()

top_stack() notifie la valeur au sommet de la pile à savoir la dernière donnée ajoutée à la pile

```c
/*-----------------------------------------------*/

int top_stack(Stack st)
{
    if(is_empty_stack(st))
    {
    printf("no value at the top fo the stack.\n");
    return -1;
    }

    return st->value;
}

/*-----------------------------------------------*/
```

top_stack()->ex:

```c
#include <stdio.h>
#include "Stack.h"

int main(void)
{
    Stack sta = new_stack();
    print_stack(sta);
    printf("Pick of the stack is : %d\n", top_stack(sta));

    printf("\n/*-------------------------*/\n\n");

    sta = push_stack(sta, 14);
    sta = push_stack(sta, 47);
    sta = push_stack(sta, 22);
    sta = push_stack(sta, 12);
    print_stack(sta);
    printf("Top of the stack is : %d\n", top_stack(sta));

    printf("\n/*-------------------------*/\n\n");

    sta = pop_stack(sta);
    print_stack(sta);
    printf("Top of the stack is : %d\n", top_stack(sta));

    printf("\n/*-------------------------*/\n\n");

    sta = clear_stack(sta);
    print_stack(sta);
    printf("Top of the stack is : %d\n", top_stack(sta));


    return 0;
}
```

```
Nothing to declare, the Stack is empty.
no value at the top fo the stack.
Top of the stack is : -1

/*-------------------------*/

[12]
[22]
[47]
[14]
Top of the stack is : 12

/*-------------------------*/

[22]
[47]
[14]
Top of the stack is : 22

/*-------------------------*/

Nothing to declare, the Stack is empty.
no value at the top fo the stack.
Top of the stack is : -1
Program ended with exit code: 0
```