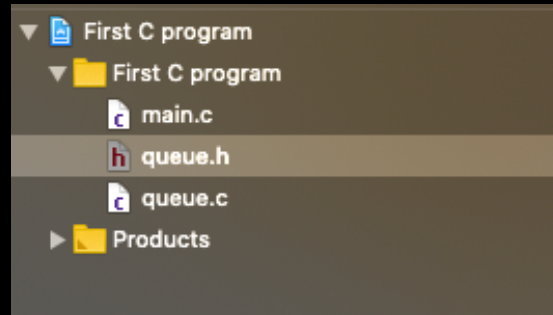


Les Files
20 | MAR | 2020
Dorian.H Mekni

L'architecture d'une file



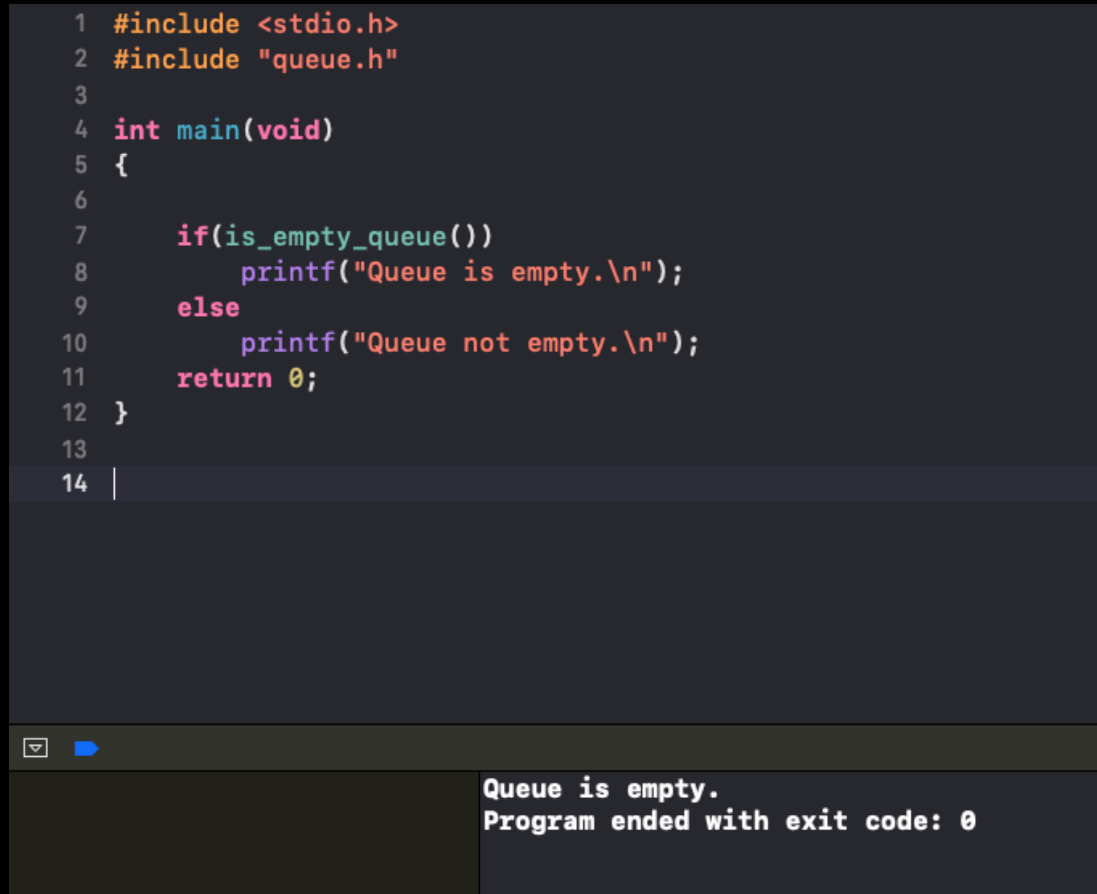
- L'architecture d'une file se fait directement sur le fichier en tête.h
- Tous les paramètres de la file ainsi que les fonctions prototypes y sont répertoriés.

queue.h

```
5  #include <stdio.h>
6
7  // Définition du type Booléen
8  typedef enum
9  {
10     false,
11     true
12 }Bool;
13
14 // Def d'une file
15 typedef struct QueueElement
16 {
17     int value;
18     struct QueueElement *next;
19 }QueueElement, *Queue;
20
21
22 // Paramètres d'une file
23 static QueueElement *first = NULL; // élément qui sera en tête de file
24 static QueueElement *last = NULL; // élément qui sera en tete de file
25 static int nb_elements = 0; // definit le nombre d'éléments
26
27 // Prototypes
28 Bool is_empty_queue(void);
29 int queue_length(void);
30 int queue_first(void);
31 int queue_last(void);
32 void push_queue(int);
33 void pop_queue(void);
34 void clear_queue(void);
35
36
37
38 #endif /* queue_h */
```

is_empty_queue()

```
1 #include <stdio.h>
2 #include "queue.h"
3
4 int main(void)
5 {
6
7     if(is_empty_queue())
8         printf("Queue is empty.\n");
9     else
10         printf("Queue not empty.\n");
11     return 0;
12 }
13
14
```

The image shows a code editor window with a C program. The code includes `<stdio.h>` and `"queue.h"`, and defines a `main` function that calls `is_empty_queue()` and prints the result. Below the code editor, there is a terminal window showing the output of the program. The terminal output is "Queue is empty." followed by "Program ended with exit code: 0".

Queue is empty.
Program ended with exit code: 0

cette fonction permet d'identifier le contenu de la file en énonçant un algorithme au conditionnel.

La liste des protocoles

Cette dernier nous permettra de manipuler les donner au sein d'une file

```
// Prototypes
Bool is_empty_queue(void);
int queue_length(void);
int queue_first(void);
int queue_last(void);
void push_queue(int);
void pop_queue(void);
void clear_queue(void);
```

2 paramètres

- `queue_first()` -> identifier la 1er donnée de la file
- `queue_last()` -> identifier la 2eme donnée de la file

first() | last()

```
/*-----*/  
  
int queue_first(void)  
{  
    if(is_empty_queue())  
        exit(1);  
  
    return first->value;  
}  
  
/*-----*/  
  
int queue_last(void)  
{  
    if(is_empty_queue())  
        exit(1);  
  
    return last->value;  
}
```

queue_length()

cette fonction calcule la longueur de la file

```
int main(void)
{

    printf("Queue size is %d.\n", queue_length());

    return 0;
}
```

```
Queue size is 0.
Program ended with exit code: 0
```


push_queue()

fonction ajoutant une donnée à la file

```
void push_queue()
{
    int x = 0;
    QueueElement *element;

    element = malloc(sizeof(*element));

    if(element == NULL)
    {
        fprintf(stderr, "Error : allocation issue.\n");
        exit(EXIT_FAILURE);
    }

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

    if(is_empty_queue())
    {
        first = element;
        last = element;
    } else {
        last->next = element;
        last = element;
    }

    nb_elements++;
}
```

pop_queue()

Fonction retire une donnée de la file.

```
void pop_queue(void)
{
    if(is_empty_queue())
    {
        printf("nothing to remove, the queue is already empty.\n");
        return;
    }

    QueueElement *temp = first;

    if(first == last)
    {
        first = NULL;
        last = NULL;
    } else
        first = first->next;

    free(temp);
    nb_elements--;
}
```

clear_queue()

Fonction qui vide une file de ses données

```
void clear_queue(void)
{
    if(is_empty_queue())
    {
        printf("nothing to remove, the queue is already empty.\n");
        return;
    }

    while(!is_empty_queue())
        pop_queue();
}
```

Compilation finale

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include "queue.h"
4
5  int main(void)
6  {
7      printf("Queue size is %d.\n", queue_length());
8
9      push_queue(11);
10     push_queue(21);
11     push_queue(33);
12     printf("Queue size is %d.\n", queue_length());
13
14     pop_queue();
15     printf("Queue size is %d.\n", queue_length());
16     clear_queue();
17     printf("Queue size is %d.\n", queue_length());
18     return 0;
19 }
20
21
```

Compilation finale -> Console

```
Queue size is 0.  
Queue size is 3.  
Queue size is 2.  
Queue size is 0.  
Program ended with exit code: 0
```

