NFO5A C project :

The Smart Parking

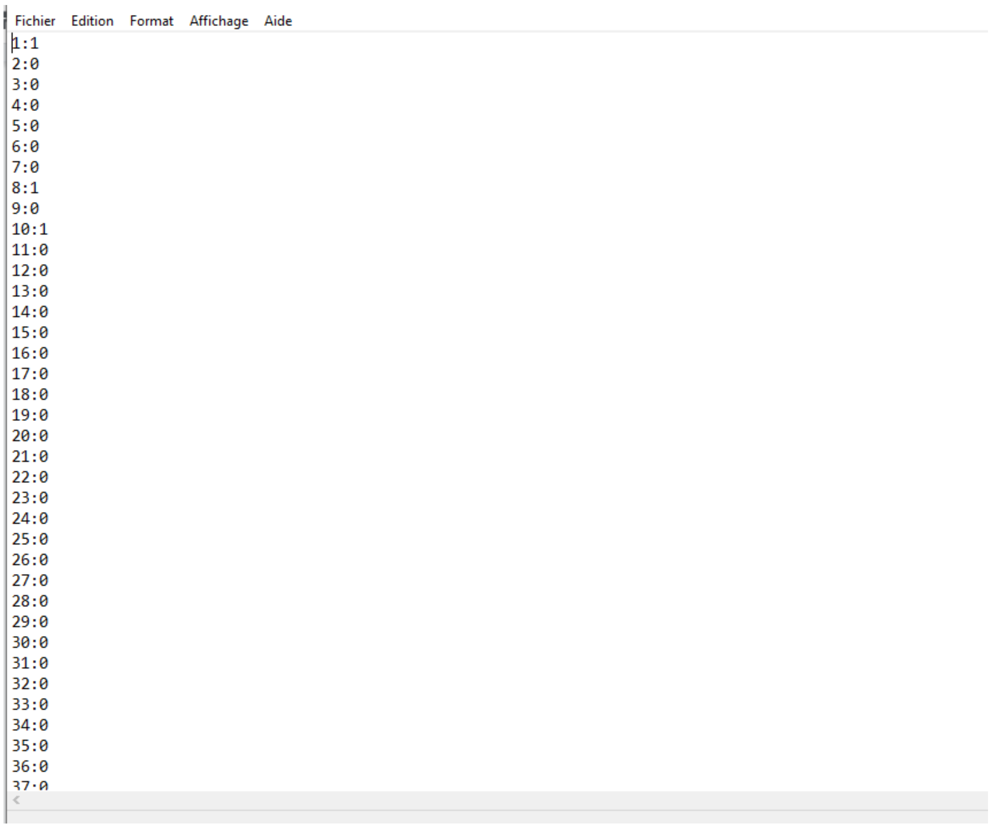
Summary :

1. Introduction
2. Schema of The Algorithm and Functions
3. Encountered Problems
4. Instruction Manual
5. Conclusion
6. Appendix

Introduction

For this project we were asked to code a smart parking reservation system in the C language. Smart parking is a place where a person can park his car for a period (In a mall, supermarket, . . .). Each car slot is equipped with a sensor in order to detect if there is a parked car or not. We suppose that the parking has the possibility to contain a maximum of 100 cars. The reservation system should offer different possibilities :
-Check the availability of a specific car slot
-Display all available car slots
-Display the number of available car slots
-Make a reservation

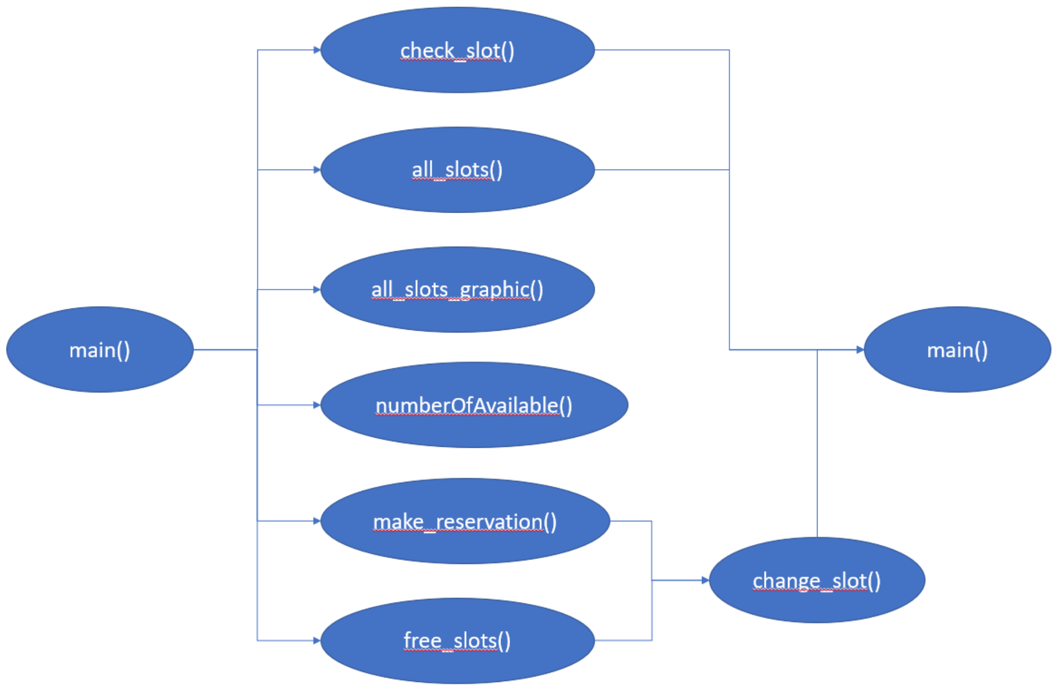
All of the parking slots will be stored in a .txt file and every change too. A reserved slot will be stored as the number of the place XX:1 and an unoccupied place as XX:0, as shown in this file:



```
1:1
2:0
3:0
4:0
5:0
6:0
7:0
8:1
9:0
10:1
11:0
12:0
13:0
14:0
15:0
16:0
17:0
18:0
19:0
20:0
21:0
22:0
23:0
24:0
25:0
26:0
27:0
28:0
29:0
30:0
31:0
32:0
33:0
34:0
35:0
36:0
37:0
```

Schema of The Algorithm and Functions

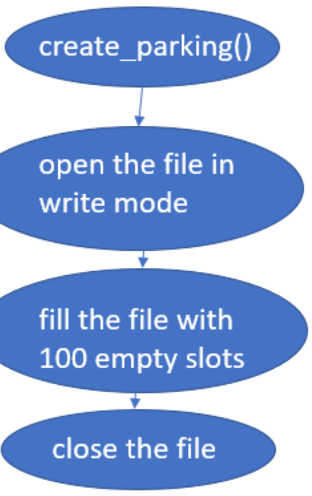
- **main()**



For the whole program we estimated 5hours of coding.

This is how the main function works : when we call it, the user has to make a choice between the different features of the program which are all functions, when it's done the user chooses if he wants to stop the program or choose something else to do, if so, the main function will start over again, because of the "while" in which all the code is contained.
Before doing anything, this function will call the create_parking() function if the file doesn't exist. That function will create it.

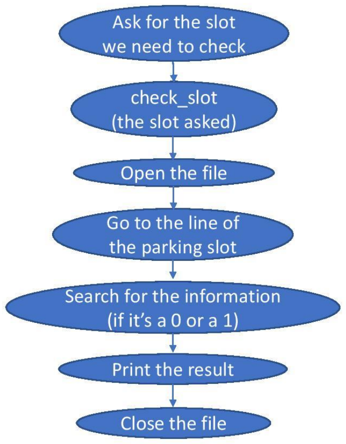
- **create_parking()**


create_parking()
open the file in write mode
fill the file with 100 empty slots
close the file

For this function we estimated 10 minutes of coding.

This function is pretty self-explanatory : it just opens a file and fills it with empty slots (1:0\n2:0\n3:0 and so on…)

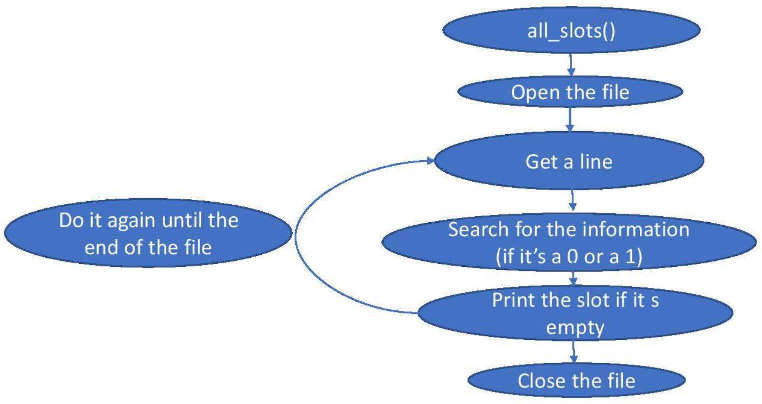
- **check_slot()**


Ask for the slot we need to check
check_slot (the slot asked)
Open the file
Go to the line of the parking slot
Search for the information (if it's a 0 or a 1)
Print the result
Close the file

For this function we estimated 30minutes of coding.

This is the first function, check_slot, When we call it we need to enter the slot we want to check as an argument, the function will return if it's available or not. First this function opens the file where the information are stored, then it goes to the line of the parking slot asked and reads it. We only need to see if the number stored after the place is a 0 or a 1, if it’s available or not.

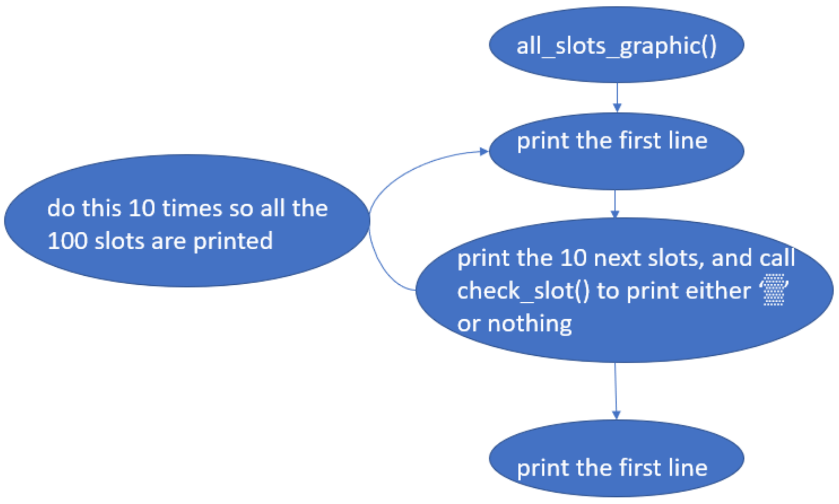
- **all_slots()**


all_slots()
Open the file
Get a line
Do it again until the end of the file
Search for the information (if it’s a 0 or a 1)
Print the slot if it s empty
Close the file

For this function we estimated 30minutes of coding.

This is how the function that shows all the slots works. First as always it opens the file, goes to the first line, searches for the 0 or the 1 and prints the number of the slot if it is a 0 (which means the place is empty). It repeats these operations until the end of the file .

- **all_slots_graphic()**


all_slots_graphic()
print the first line
do this 10 times so all the 100 slots are printed
print the 10 next slots, and call check_slot() to print either '▒' or nothing
print the first line

For this function we estimated 1h30 of coding.
The goal of this function is to print a graphic version of the all_slots() function.
-For this, we begin by printing the first line of the table, like so :

```
┌───┬───┬───┬───┬───┬───┬───┬───┬───┬───┐
```

-Then we print a line of 10 numbers separated by borders :

```
│  1│  2│  3│  4│  5│  6│  7│  8│  9│ 10│
```

(each cell is 3 characters long)
-We print the availability or not of each corresponding slot :

```
│▒▒▒│   │▒▒▒│   │   │▒▒▒│   │▒▒▒│▒▒▒│▒▒▒│
```

-We then close the line :

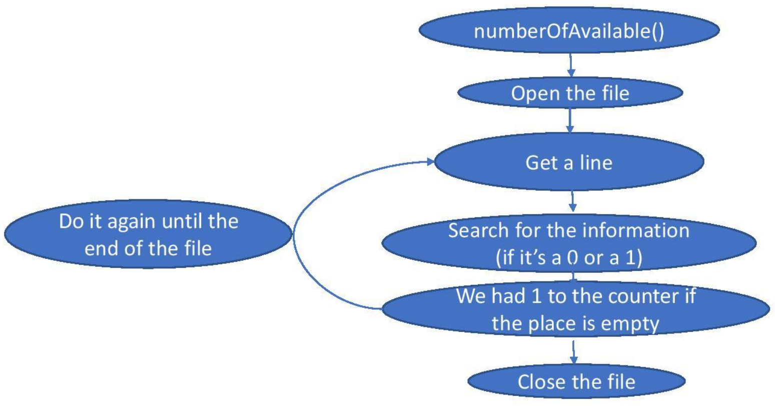
```
├───┼───┼───┼───┼───┼───┼───┼───┼───┼───┤
```

-After repeating this 10 time to cover all the 100 slots, we can draw the last line :

```
└───┴───┴───┴───┴───┴───┴───┴───┴───┴───┘
```

Each of the special characters (┌,─, └, ├,│,▒, ┤) are part of the extended ascii table, which are perfectly supported by the c language.

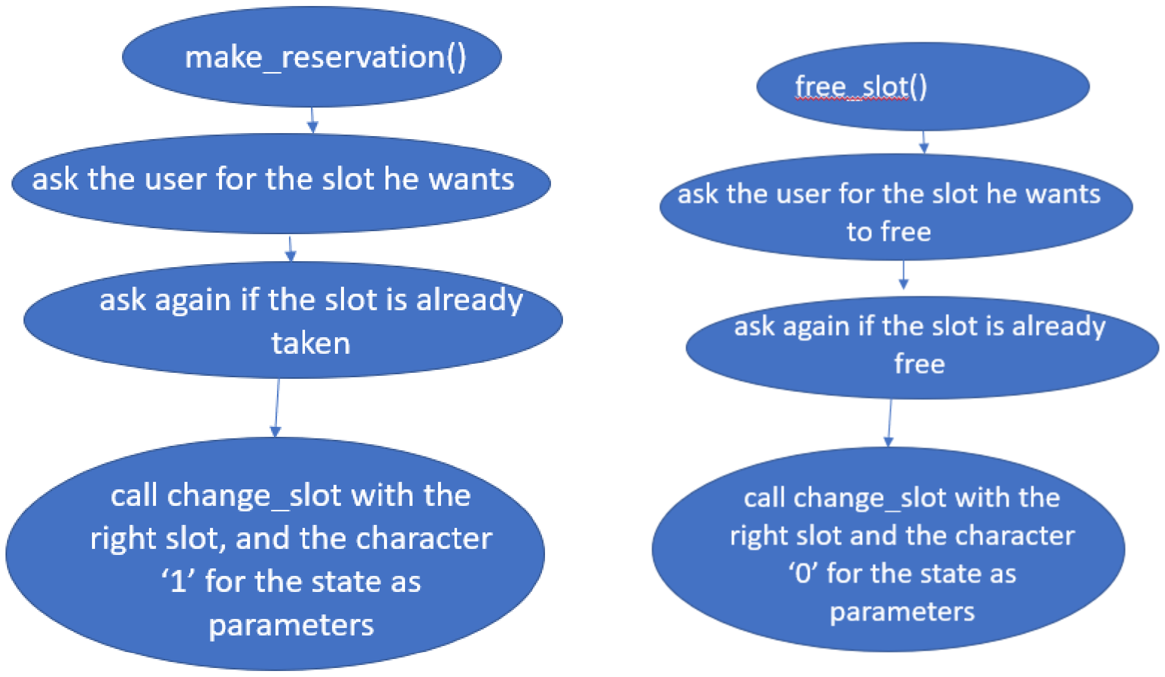
● numberOfAvailable()


numberOfAvailable()
Open the file
Get a line
Search for the information (if it's a 0 or a 1)
We had 1 to the counter if the place is empty
Close the file
Do it again until the end of the file

For this function we estimated 30 minutes of coding.

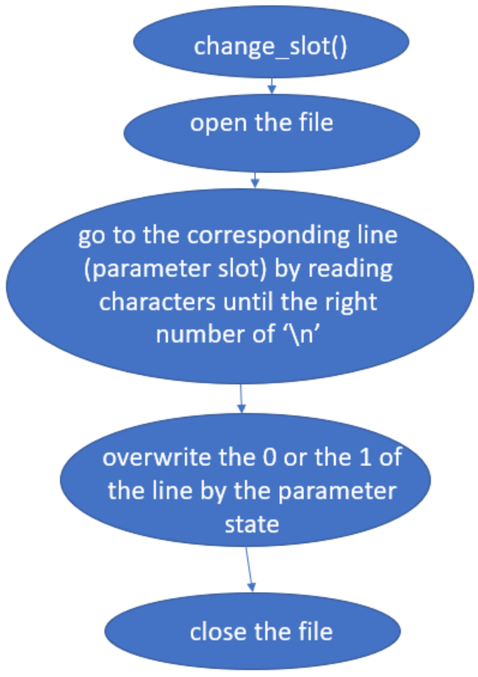
This shows how the function number of slots work. First it opens the files, goes to the first line, searches for the 0 or the 1 and adds 1 to the counter variable if it is a 0 (which means the place is empty). It repeats these operations until the end of the file and then prints the counter, the number of slots available.

- **make_reservation() and free_slot()**


make_reservation()
ask the user for the slot he wants
ask again if the slot is already taken
call change_slot with the right slot, and the character '1' for the state as parameters
free_slot()
ask the user for the slot he wants to free
ask again if the slot is already free
call change_slot with the right slot and the character '0' for the state as parameters

For these functions we estimated 1hour of coding.
These functions basically allow the user to change the slot he wants : he can either reserve it (0 to 1) or free it (1 to 0).
As we can see, we first ask the user for the slot he wants to change, then we check if the slot isn't already in the state the user wants, and then we call the change_slot function that will change the slot in the text file.

- **change_slot()**


change_slot()
open the file
go to the corresponding line (parameter slot) by reading characters until the right number of '\n'
overwrite the 0 or the 1 of the line by the parameter state
close the file

for this function we estimated 45 minutes of coding.
The goal of this function is to change the character in the file that corresponds to a specific slot.
First, it needs to go to the right line : the parameter slot. To do this, it will go through every character of the file until it reaches the right amount of '\n' characters. Then, it will move forward the right amount of character (1 if we are below ten, 2 if we are between 10 and 99 and 3 if slot is equal to 100).
Finally, it can write the new availability (contained in the parameter 'state')

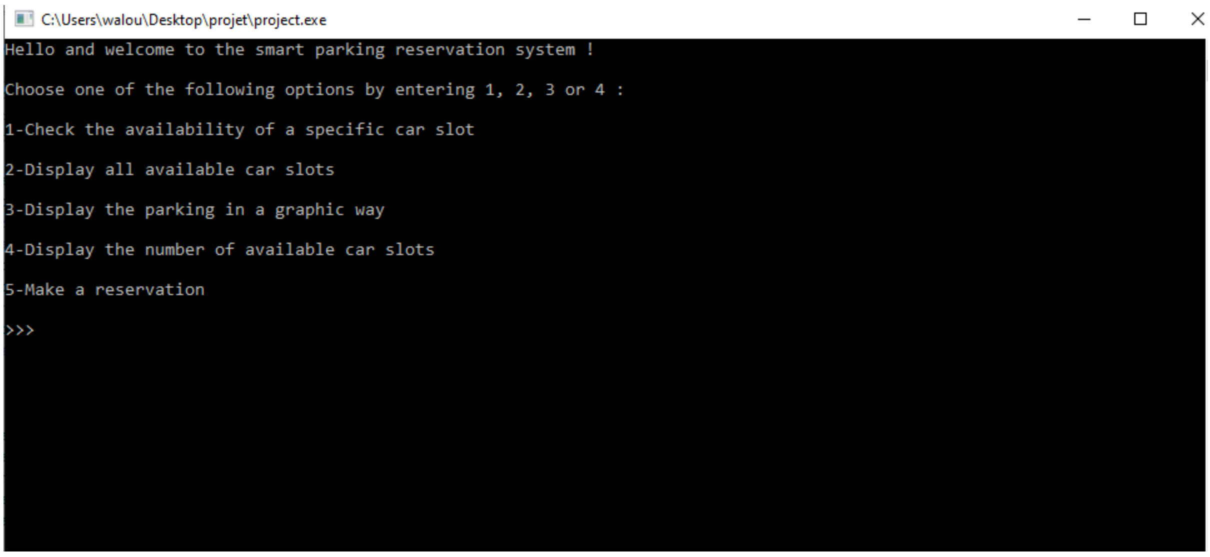
Encountered Problems

One of the problems we encountered was to write in the file after reading something. We didn't know where this problem came from but after replacing the cursor in the file we were able to fix it : a call of the fseek() function did it well.

Another problem we encountered was about a the number of character we were collecting with the "fscanf" function, as the last line was the only one to contain 6 character ("100:0\n") we didn't thought about it and made a variable to contain 5 character,which result with all the functions that seemed to work unless the print all_slots. This function was working fine until the line 99 and then printed an error message continuously.

Instruction Manual for the program

First, run the projet.exe file, it will will launch a window that looks like this :



```
C:\Users\walou\Desktop\projet\project.exe
Hello and welcome to the smart parking reservation system !

Choose one of the following options by entering 1, 2, 3 or 4 :

1-Check the availability of a specific car slot

2-Display all available car slots

3-Display the parking in a graphic way

4-Display the number of available car slots

5-Make a reservation

>>>
```

When you are here you need to choose which features of the program you want to use and type the corresponding number, when it's done type enter.

Depending on the feature you have chosen the next step will be different:

-if you typed “1” : the program will ask you which slot you want to check , just type the number of the slot you want to check and press enter, It’ll display “Available” or “Not Available”.
-if you typed “2” : the program will print you all the slot numbers available in the parking, one by line .
-if you typed “3” : the program will print you all the slots in the parking in a graphic way, you will have a blank case with the slot number if it’s empty, and a stripped box if it’s already occupied.
-if you typed “4” : the program will print you the number of available slots.
-if you typed “5” : the program will ask you which slot do you want to book, just type the number of the slot you want and press enter, It’ll display “sorry, the slot is already taken” if the place is not available and else it’ll book the place.

After finishing the instruction, the program will ask if you want to do something else, just press “2” and enter if you want to stop or “1” and then the number of the feature you want to do next.

Conclusion

To conclude, we think that our program responds well to the expectations of the subject. We even added a feature to show the parking in a graphic way. To optimize it we could have added some timer to make a slot available again after some time and make the smart parking work alone. It also could have been optimized for the user, by having a graphic application to navigate instinctively and use the different options of the program. Concerning the time spent on this project we actually didn't respect our estimated coding time, some took longer some were quicker but looking through the code to find errors at the end of the project and a lot of small details took us a really long time that we completely underestimated before starting coding. We finally arrived at 5h30-6h of coding on this project but we were not detailing the time spent into each function.

Appendix

References :

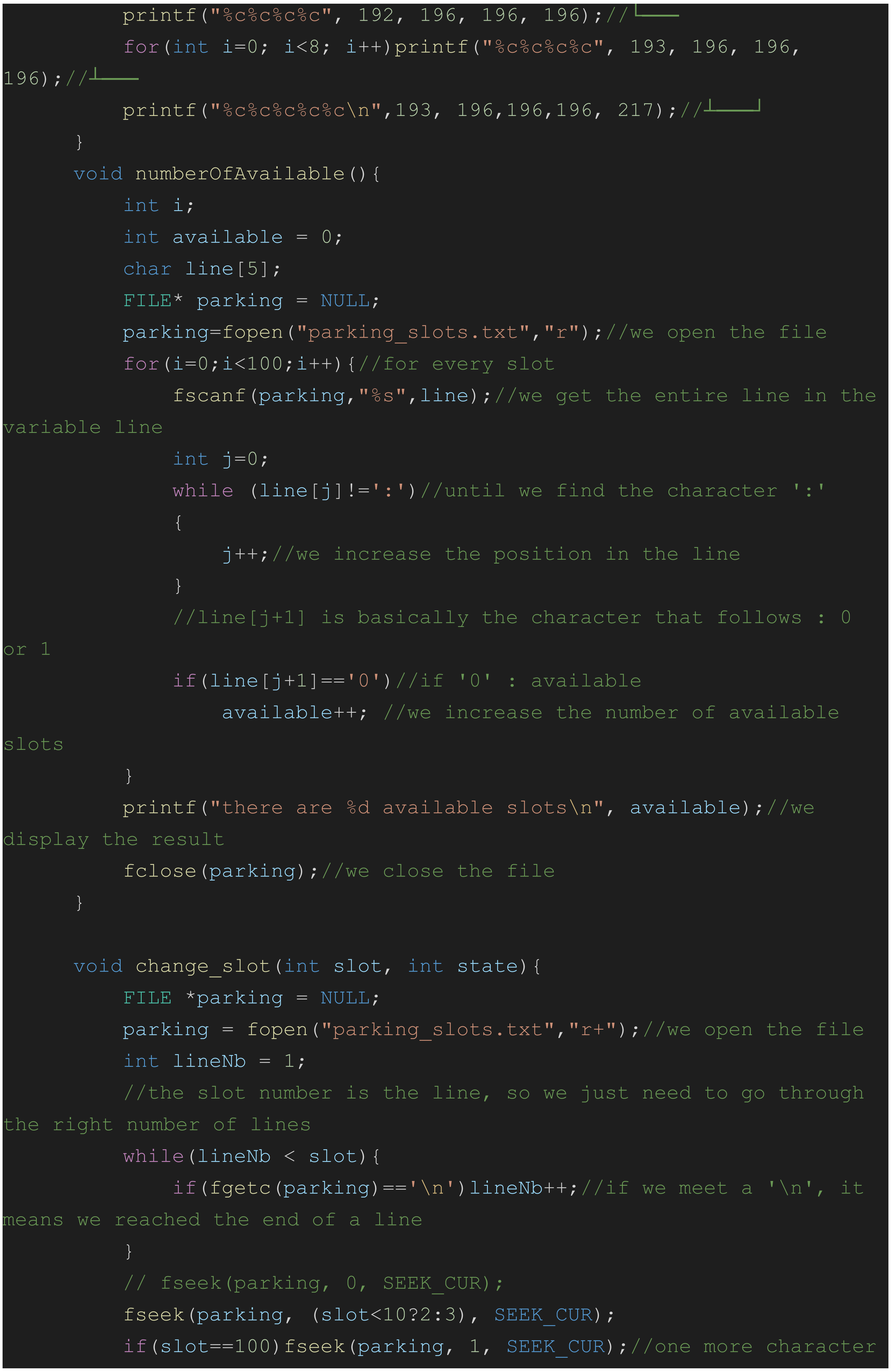
- NF05A Courses
- Open classrooms : https://openclassrooms.com/
- Extended Ascii Table : https://theasciicode.com.ar/
- Doxygen documentation : https://www.doxygen.nl/manual/index.html

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
void create_parking ()
{
    FILE *parking = NULL;
    parking=fopen("parking_slots.txt","w");
    for (int i=1;i<=100;i++)
    {
        fprintf(parking,"%d:0\n",i);
    }
    fclose(parking);
}
int check_slot(int slot_number)
{
    int i,availability ;
    //this variable is going to stock the line
    char line[5];
    //we open the file
    FILE *parking = NULL;
    parking=fopen("parking_slots.txt","r");
```

```c
          //we are getting to the slot asked by the user reading each
line but storing only the last one
          for(i=0;i<slot_number;i++)
          {
              fscanf(parking,"%s",line);
          }
          i=0;
          //until we find the character ':'
          while(line[i]!=':')
          {
              i++;//we increase the position in the line
          }
          //line[j+1] is basically the character that follows : 0 or 1
          if(line[i+1]=='0')//if '0' : available
          {
             availability=0;
              return availability;
          }
          else//if '1' : Not available
          {
              availability=1;
              return availability;
          }

          //we close the file
          fclose(parking);

      }
      void all_slots()
      {
          int i, j;
          char line[6];
          FILE* parking = NULL;
          parking = fopen("parking_slots.txt","r");//we open the file
          //we are checking each slot to see if it's available
          for(i=1;i<=100;i++)
          {
              //we get the line
              fscanf(parking,"%s",line);
              j=0;
              while (line[j]!=':')//until we find the character ':'
              {
                  j++;//we increase the position in the line
              }
              //line[j+1] is basically the character that follows : 0
  or 1
              if(line[j+1]=='0')//if '0' : available
                  printf("%d\n",i);//we print the number of the parking
  slot
          }
          //we close the file
          fclose(parking);
      }
      void all_slots_graphic(){
          //first line
          printf("%c%c%c%c", 218, 196, 196, 196);//┌───
          for(int i=0; i<8; i++)printf("%c%c%c%c", 194, 196, 196,
  196);//┬───
          printf("%c%c%c%c%c\n",194, 196,196,196, 191);//┬───┐
          for(int j=0; j<100; j+=10){
              for(int i=1; i<=10; i++){
                  printf("%c", 179);//│
                  int slotNb = i+j;
                  if(slotNb<10)
                      printf("  %d", slotNb);
                  else if(slotNb==100)
                      printf("%d", slotNb);
                  else printf(" %d", slotNb);
              }
              printf("%c\n", 179);//│
              for(int i=1; i<=10; i++){
                  printf("%c", 179);//│
                  if(check_slot(i+j)==1)
                      printf("%c%c%c", 177, 177, 177);//▒
                  else
                      printf("   ");
              }
              printf("%c\n", 179);//│
              if(j==90)break;//in order to close the table
              printf("%c%c%c%c", 195, 196, 196, 196);//├───
              for(int i=0; i<8; i++)printf("%c%c%c%c", 197, 196, 196,
  196);//┼───
              printf("%c%c%c%c%c\n",197, 196,196,196, 180);//┼───┤
              //printf("%c\n", 179);//│
          }
```


```
          printf("%c%c%c%c", 192, 196, 196, 196);//└───
          for(int i=0; i<8; i++)printf("%c%c%c%c", 193, 196, 196, 
196);//┴───
          printf("%c%c%c%c%c\n",193, 196,196,196, 217);//┴───┘
      }
      void numberOfAvailable(){
          int i;
          int available = 0;
          char line[5];
          FILE* parking = NULL;
          parking=fopen("parking_slots.txt","r");//we open the file
          for(i=0;i<100;i++){//for every slot
              fscanf(parking,"%s",line);//we get the entire line in the 
variable line
              int j=0;
              while (line[j]!=':')//until we find the character ':'
              {
                  j++;//we increase the position in the line
              }
              //line[j+1] is basically the character that follows : 0 
or 1
              if(line[j+1]=='0')//if '0' : available
                  available++; //we increase the number of available 
slots
          }
          printf("there are %d available slots\n", available);//we 
display the result
          fclose(parking);//we close the file
      }

      void change_slot(int slot, int state){
          FILE *parking = NULL;
          parking = fopen("parking_slots.txt","r+");//we open the file
          int lineNb = 1;
          //the slot number is the line, so we just need to go through 
the right number of lines
          while(lineNb < slot){
              if(fgetc(parking)=='\n')lineNb++;//if we meet a '\n', it 
means we reached the end of a line
          }
          // fseek(parking, 0, SEEK_CUR);
          fseek(parking, (slot<10?2:3), SEEK_CUR);
          if(slot==100)fseek(parking, 1, SEEK_CUR);//one more character
```

```
        fputc(state==1?'1':'0', parking);
        fclose(parking);
    }
    void makeReservation(){
        int slot_number;
        printf("Which slot do you want to reserve?\n");
        scanf("%d",&slot_number);
        while(slot_number<1 || slot_number>100)//while the slot is
invalid
        {
            printf("Error the slot is out of range, please try again
\n");
            scanf("%d",&slot_number);//we ask the user again to enter
the slot
        }
        if(check_slot(slot_number)){//slot not available
            printf("sorry, the slot is already taken\n");
            makeReservation();//we ask again
        }
        change_slot(slot_number, 1);
    }

    void free_slot(){
        int slot_number;
        printf("Which slot do you want to free?\n");
        scanf("%d",&slot_number);
        while(slot_number<1 || slot_number>100)//while the slot is
invalid
        {
            printf("Error the slot is out of range, please try again
\n");
            scanf("%d",&slot_number);//we ask the user again to enter
the slot
        }
        if(!check_slot(slot_number)){//slot not available
            printf("sorry, the slot is already empty\n");
            makeReservation();//we ask again
        }
        change_slot(slot_number, 0);
    }
    int main()
    {
        int features, slot_number;
```

```c
          int continue_loop = 1;
          while(continue_loop==1){
              // we initialize the parking
              if(access("parking_slots.txt", F_OK) != 0){//if file
doesn't exist
                  create_parking();
              }
              // this is the interface for the user
              printf("Hello and welcome to the smart parking
reservation system !\n\nChoose one of the following options by entering
1, 2, 3 or 4 : \n\n1-Check the availability of a specific car
slot\n\n2-Display all available car slots\n\n3-Display the parking in a
graphic way\n\n4-Display the number of available car slots \n\n5-Make a
reservation \n\n6-Free a slot\n\n");
              //the user choose an option
              printf(">>>");
              scanf("%d",&features);
              while(features<1 || features>6)
              {
                  int c;while((c = getchar()) != '\n' && c !=
EOF){}//this line allows to empty the input, that way we can't get
stuck in this while loop

                  printf("Error during the selections of the option
please try again \n");
                  printf(">>>");
                  scanf("%d",&features);
              }
              switch (features)
              {
                  case 1://asking for the user
                      printf("Which slot do you want to check the
availability ?\n");
                      scanf("%d",&slot_number);
                      //checking if his response is acceptable
                      while(slot_number<1 || slot_number>100)
                      {
                          printf("Error the slot is out of range,
please try again \n");
                          scanf("%d",&slot_number);
                      }
                      if (check_slot(slot_number)==0)
                          printf("Available\n");
                      else
                          printf("Not Available\n");
                      break;
                  case 2:all_slots();
                      break;
                  case 3:all_slots_graphic();
                      break;
                  case 4:numberOfAvailable();
                      break;
                  case 5:makeReservation();
                      break;
                  case 6:free_slot();
                      break;
              }
              printf("\nDo you want to do anything else? 1-yes 
2-no\n");
              printf(">>>");
              while(continue_loop<1 || continue_loop>2)
              {
                  printf("Error during the selections of the option 
please try again \n");
                  printf(">>>");
                  scanf("%d",&continue_loop);
              }
              scanf("%d", &continue_loop);
          }
      }
```