[TYPE THE COMPANY NAME]

Oven Project

OpenCv Team

مروان مجدي عطيه20011861
مهيمن عبدالمجيد صالح20012038
احمد منير علي 20010224
شروق عبدالخالق فتوح 20010726
نوران مصطفي محمد 20012140
احمد عطاالله احمد 20010140
احمد محمد السيد 20010182
عمر ايمن عبدو محمد 20010977
حسام الدين محمد حسن 19015584

Contents

1. Reasons of choice:

Although the project seems easy, it taught us a lot of technical skills and many tools. We chose this as it is more reasonable to code and more clear to understand which helps us to write the code perfectly.

2. Problem analysis:

- ### Required mechanism:

  - Set timer and certain temperature.

  -Increase given temperature by scaling factor until certain limit.

  -When temperature reaches that certain limit oven is off and starts to cool down.

  -when temperature cools down to certain limit oven is on again and so on.

- ### Calculate rate of change and temperature limits:

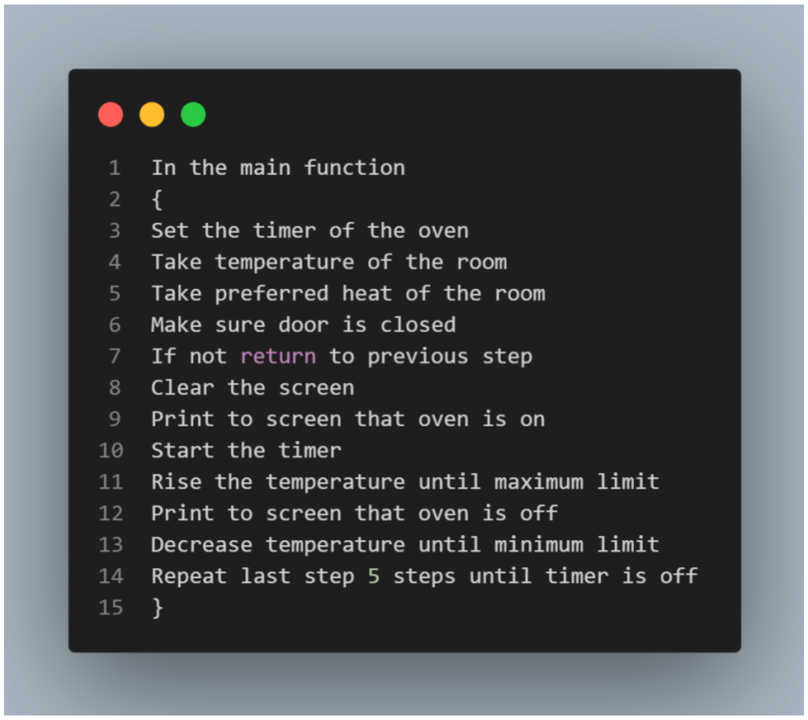
  -Rate of increasing temperature, $Temperature = Ambient_{\mathrm{temp}} + 0.0001 * Required_{temp}$.

  -Maximum limit, $Max_{temp} = Required_{temp} + 0.1Required_{temp}$.

  -Rate of decreasing temperature, $\mathrm{temp} = \mathrm{current}_{\mathrm{temp}} - 0.0001 * Required_{temp}$.

  -Minimum limit, $\mathrm{Minimum}_{\mathrm{temp}} = Required_{temp} - 0.1 * Required_{temp}$

3. Pseudo code:



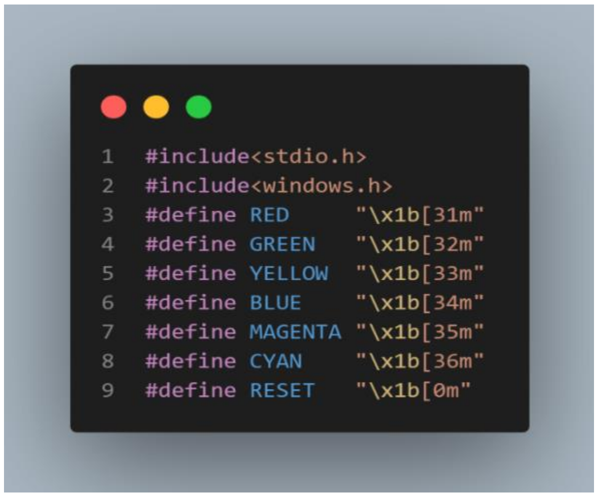
```
In the main function
{
Set the timer of the oven
Take temperature of the room
Take preferred heat of the room
Make sure door is closed
If not return to previous step
Clear the screen
Print to screen that oven is on
Start the timer
Rise the temperature until maximum limit
Print to screen that oven is off
Decrease temperature until minimum limit
Repeat last step 5 steps until timer is off
}
```

4. Code:

a. Includes

We put some pre-processor defines which will obtain no memory, to save more ram storage.

We include window to draw colors and clear screen and easily use sleep function.

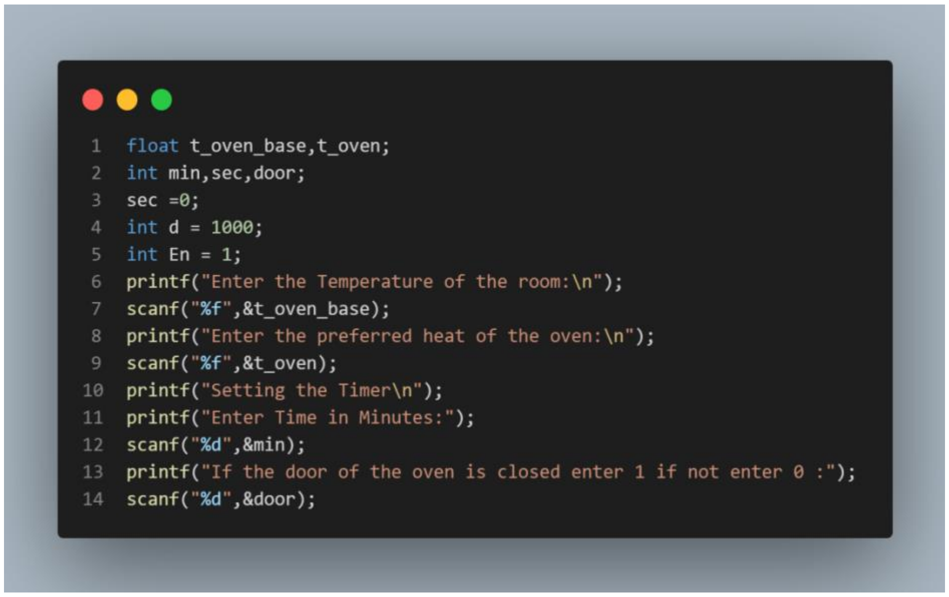


```c
#include<stdio.h>
#include<windows.h>
#define RED      "\x1b[31m"
#define GREEN    "\x1b[32m"
#define YELLOW   "\x1b[33m"
#define BLUE     "\x1b[34m"
#define MAGENTA  "\x1b[35m"
#define CYAN     "\x1b[36m"
#define RESET    "\x1b[0m"
```

Figure 1 pre-processor

b. Variables & inputs

Where'd' is in milliseconds and EN is the active high enable of the oven.

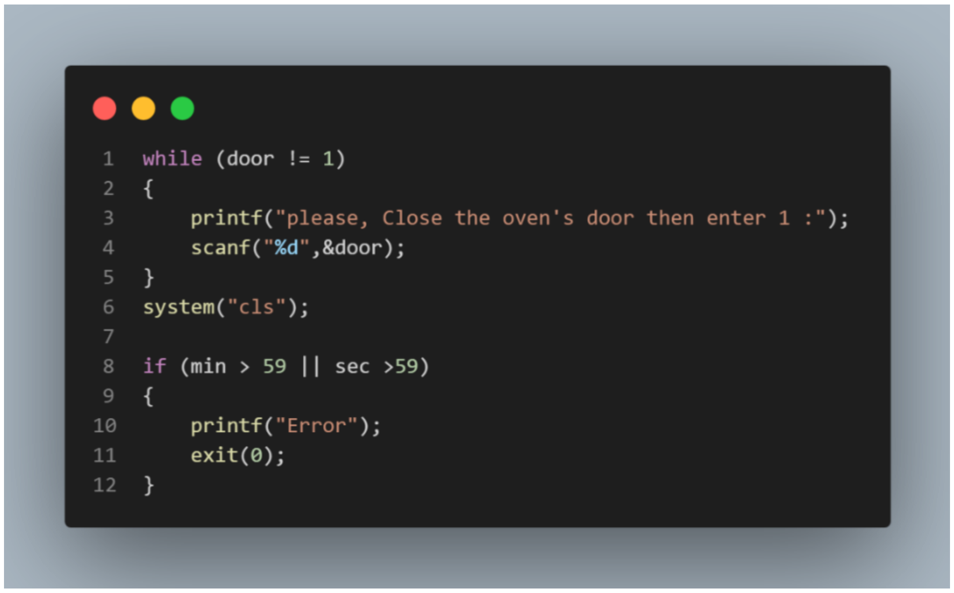


```c
float t_oven_base,t_oven;
int min,sec,door;
sec =0;
int d = 1000;
int En = 1;
printf("Enter the Temperature of the room:\n");
scanf("%f",&t_oven_base);
printf("Enter the preferred heat of the oven:\n");
scanf("%f",&t_oven);
printf("Setting the Timer\n");
printf("Enter Time in Minutes:");
scanf("%d",&min);
printf("If the door of the oven is closed enter 1 if not enter 0 :");
scanf("%d",&door);
```

Figure 2 Variables

c. Checker

It checks if the door is closed, if not, it doesn't function to the next step.



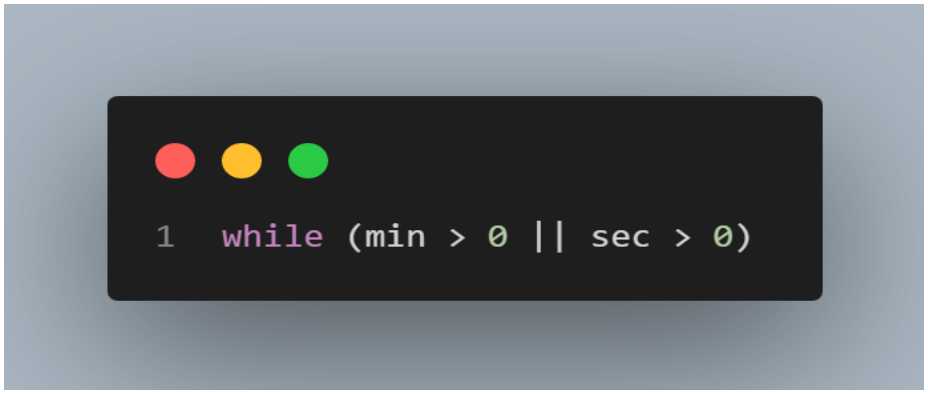
```c
while (door != 1)
{
    printf("please, Close the oven's door then enter 1 :");
    scanf("%d",&door);
}
system("cls");

if (min > 59 || sec >59)
{
    printf("Error");
    exit(0);
}
```

Figure 3 Checker

d. Repeat until end

The will loop will be looping until the Two conditions is False.

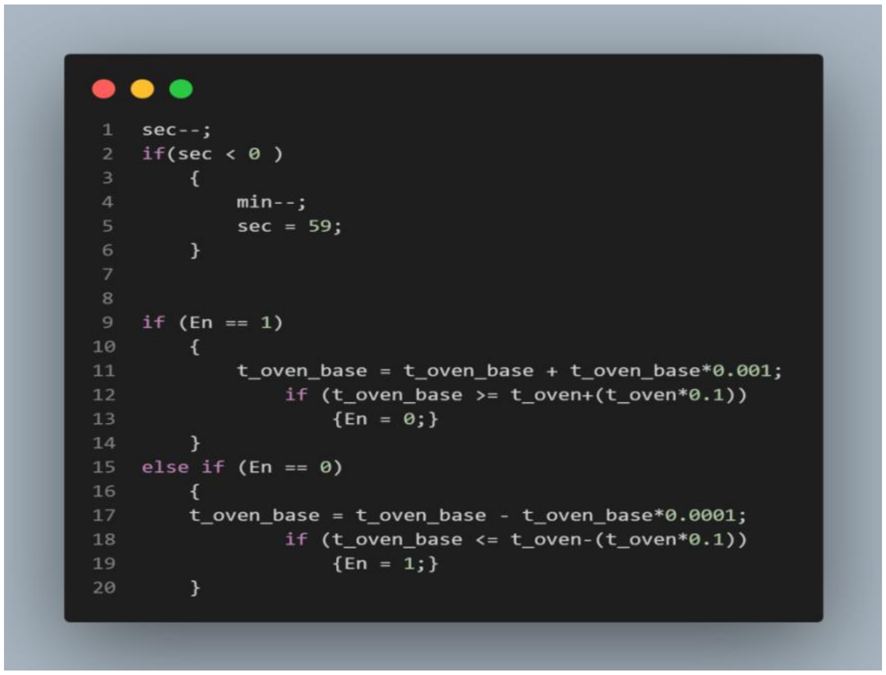


```c
while (min > 0 || sec > 0)
```

Figure 4 Program loop

e. Manipulating the oven

Code that decides whether temperature rises to maximum limit or if it already reached maximum limit and starts to cool down while timer is still on



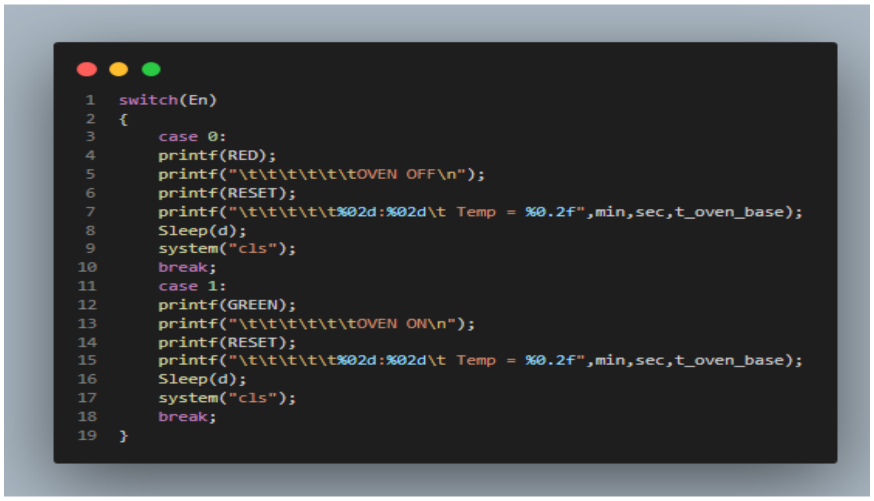
```c
sec--;
if(sec < 0 )
    {
        min--;
        sec = 59;
    }


if (En == 1)
    {
        t_oven_base = t_oven_base + t_oven_base*0.001;
            if (t_oven_base >= t_oven+(t_oven*0.1))
                {En = 0;}
    }
else if (En == 0)
    {
    t_oven_base = t_oven_base - t_oven_base*0.0001;
            if (t_oven_base <= t_oven-(t_oven*0.1))
                {En = 1;}
    }
```

Figure 5 manipulating the oven

f. Test the En variable

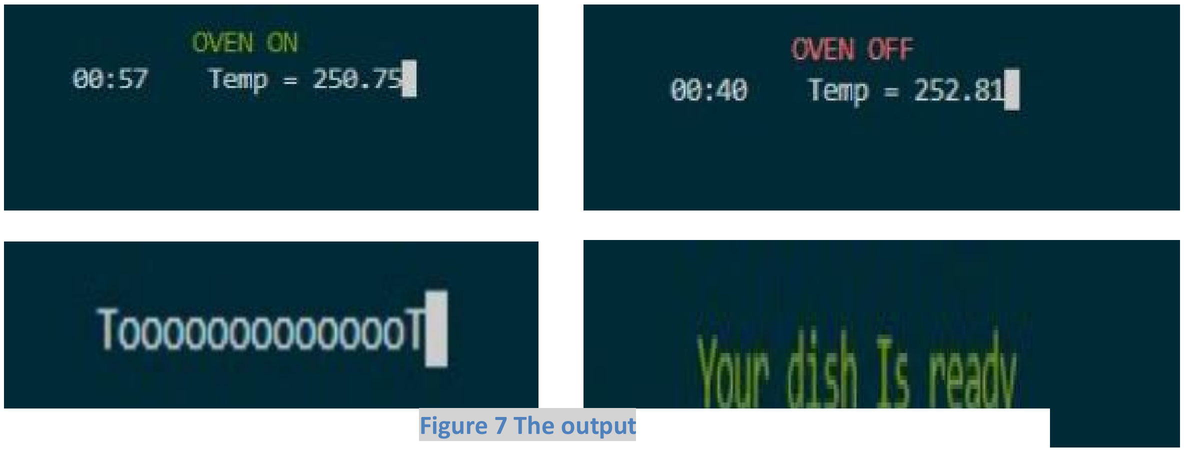
Here we control the oven with the variable En.



```c
switch(En)
{
    case 0:
    printf(RED);
    printf("\t\t\t\t\t\tOVEN OFF\n");
    printf(RESET);
    printf("\t\t\t\t\t%02d:%02d\t Temp = %0.2f",min,sec,t_oven_base);
    Sleep(d);
    system("cls");
    break;
    case 1:
    printf(GREEN);
    printf("\t\t\t\t\t\tOVEN ON\n");
    printf(RESET);
    printf("\t\t\t\t\t%02d:%02d\t Temp = %0.2f",min,sec,t_oven_base);
    Sleep(d);
    system("cls");
    break;
}
```

Figure 6 Testing the En

5. Execution example



```
        OVEN ON
00:57     Temp = 250.75
```

```
        OVEN OFF
00:40     Temp = 252.81
```

```
Toooooooooooooo T
```

```
Your dish Is ready
```

Figure 7 The output

6. Future improvement:

We have decided to make it enable GUI using the SFML library but due to time constrains we do it in console, so in the future it will contain many GUI features like oven simulation with timer.

we could've made better user interface with SFML or grahpics.h instead of just coloring the output.

we could've given the user the option to choose between normal input of time and oven heat or modes of operation as each mode has custom heat and time that has been entered by the programmer.

for Example: fish: 250 degree: 15 minutes

7. Problems we faced while coding:

1- we needed a way to print the time once per second

so, we used sleep(time) and system("cls") to delay the print function and clear what has been printed before and we used the same algorithm with the temperature.

2-we needed a way to show us if the oven is on or off, so we used an active high enable that switches to 0 when the higher temperature limit is reached.

3-we needed a way to decorate the output code, so we used printf("\x1b[31m") to turn the code to red and printf("\x1b[0m") to turn it back to white.