GROUP / INT-1

Alexandre DIDIER
Samuel WEISTROFFER

C PROJECT

efrei
PARIS PANTHÉON-ASSAS UNIVERSITÉ

ABOUT THE PROJECT

The goal of the whole project is to code a drawing software. But the purpose of the first part we began to code was just the preparation for this software, without any interactive display (just values and a basic menu).

We both think that dividing the project into two parts, respectively the structure and the display, was very useful to get organized, so that every new function could easily be implemented.

In this part, the code is just really about defining structures and using pointers effectively in order to pass shapes in parameters. The “user interface” in this part is the input of coordinates and the resulting output .

SUMMARY

1. THE MAIN FILE

For the first part, the main file is just a test in order to see if everything works fine.

We first define a linked list `l` as `NULL` that will store each shape in the order of their creation.

Then we use an infinite loop to display the following menu and ask the user for his choice:

Please select an action:

A- Add a shape

B- Display the list of shapes

Then we use an `if` condition to analyze his choice and display the correct output.

2. Presentation of the FUNCTIONS

MANDATORY FUNCTIONS
- `Point* create_point(int px, int py)`
- `void delete_point(Point* point)`
- `void print_points(Point *p)`
- `Line* create_line(Point *p1, Point *p2)`
- `void delete_line(Line *line)`
- `void print_line(Line *line)`
- `print_line(Line *line)`
- `Square* create_square(Point *point, int length)`
- `void delete_square(Square *square)`
- `void print_square(Square *square)`
- `Rectangle* create_rectangle(Point* point, int width, int height)`
- `void delete_rectangle(Rectangle* rectangle)`
- `print_rectangle(Rectangle* rectangle)`
- `Circle* create_circle(Point *center, int radius)`
- `void delete_circle(Circle* circle)`
- `void print_circle(Circle* circle)`
- `Polygon* create_polygon(int n)`
- `void delete_polygon(Polygon* polygon)`

- void print_polygon(Polygon* polygon)
- Shape *create_empty_shape(SHAPE_TYPE shape_type)
- Shape *create_point_shape(LIST*)
- Shape *create_line_shape(LIST*)
- Shape *create_square_shape(LIST*)
- Shape *create_rectangle_shape(LIST*)
- Shape *create_circle_shape(LIST*)
- Shape *create_polygon_shape(LIST*)
- void delete_shape(Shape * shape)
- void print_shape(Shape * shape)
- unsigned int get_next_id()
NEW FUNCTIONS
- Point *get_point()
- Line *get_line()
- Square *get_square()
- Rectangle *get_rectangle()
- Circle *get_circle()
- Polygon *get_polygon()
- NODE * create_node(Shape)
- bool empty_list(LIST)
- LIST add_tail_list(LIST , Shape)
- void print_list(LIST)
- shape *get_shape_from_node(NODE *)
- void menu(char *)
- void menu2(int, LIST)

Summary of functions

The functions below are explained and ordered based on the functionality of the code.

`[Desired_shape_type] *get_[desired_shape]()` : Get the coordinates of the desired shape from the user with this format : **"[desired_shape] : [input]"**. We secured the memory allocation with the statement **if (p == NULL)**. This function uses the function **create_[desired_shape]([coordonates_of_desired_shape])** with the coordinates that the user gave.

```
NODE * create_node(Shape)
bool empty_list(LIST)
LIST add_tail_list(LIST , Shape)
void print_list(LIST)
```

Those are the basic functions in order to create, modify and display a simple linked list

```
void menu(char*)
void menu2(int, LIST)
```

Those functions are used to display the menu

3. Difficulties Encountered

We didn't run into major difficulties in the first part of the project, we only faced one small problem, that is getting the input from the user. We found it difficult to use fgets() for the code of the menu, so we used scanf(), but the thing is that we couldn't secure the input. In fact, if the user enters a wrong input during the input of coordinates, then those coordinates will be set to 0.

4. Test and User INTERFACE

We ran different types of tests to see if our program worked. We started the project, set A as the input, chose 1 to add a point with coordinates 10 10, then chose B to display the list with only the point we had created. We then chose A to add a line with coordinates 10 10 12 12 and displayed the list by choosing B. The list we displayed contained the point and the line.

```
Please select an action:
A- Add a shape
B- Display the list of shapes
A

Please select an action:
1- Add a point
2- Add a line
3- Add square
4- Add a rectangle
5- Add a circle
6- Add a polygon
7- Return to previous menu
1
Enter the coordinates x y:10 10
POINT 10 10
ID: 1

Please select an action:
A- Add a shape
B- Display the list of shapes
B

The list of shapes:
POINT 10 10
ID: 1

Please select an action:
A- Add a shape
B- Display the list of shapes
```

```
Please select an action:
1- Add a point
2- Add a line
3- Add square
4- Add a rectangle
5- Add a circle
6- Add a polygon
7- Return to previous menu
2
Enter the coordinates of the first point x y:10 10
Enter the coordinates of the second point x y:12 12
LINE 10 10 12 12
ID: 2

Please select an action:
A- Add a shape
B- Display the list of shapes
B

The list of shapes:
POINT 10 10
ID: 1
LINE 10 10 12 12
ID: 2

Please select an action:
A- Add a shape
B- Display the list of shapes
```

5. CONCLUSION

This first part was quite simple but very enriching. We learned a lot about the use of structures and pointers in particular.

We were also able to improve our way of working in groups with github in particular which simplified things compared to the first project where we used discord.

We think that the second part will be more complicated, especially to set up the display algorithm. One improvement for the second part we think will be important to implement is the replacement of scanf() by fgets() without bugging our program.