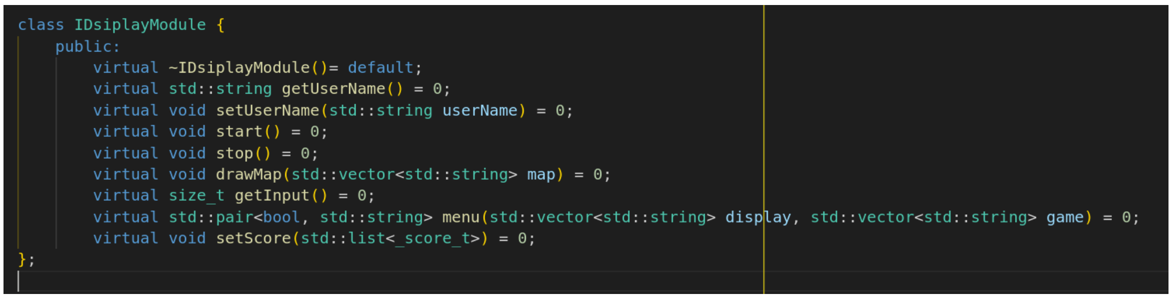
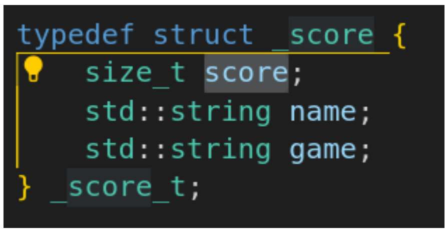
To implement a new graphic library you need to use this class.



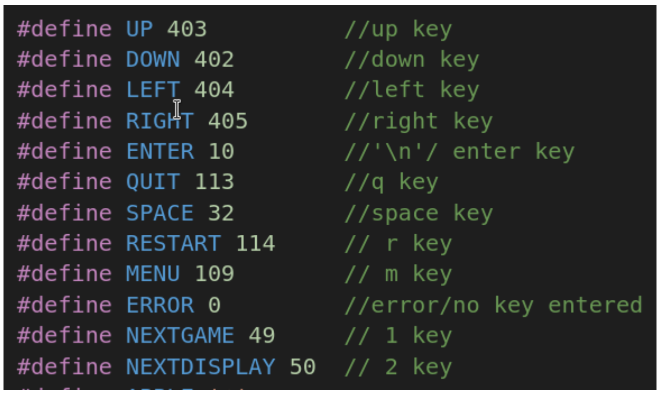
```cpp
class IDsiplayModule {
    public:
        virtual ~IDsiplayModule()= default;
        virtual std::string getUserName() = 0;
        virtual void setUserName(std::string userName) = 0;
        virtual void start() = 0;
        virtual void stop() = 0;
        virtual void drawMap(std::vector<std::string> map) = 0;
        virtual size_t getInput() = 0;
        virtual std::pair<bool, std::string> menu(std::vector<std::string> display, std::vector<std::string> game) = 0;
        virtual void setScore(std::list<_score_t>) = 0;
};
```

Start and **stop** will be like the constructors and destructors of the class, they will be called any time the library will be used or quit.
The **menu** function will be the place that displays the menu with the different graphic libraries and games, it takes two arguments, the names of the graphic libraries, and the names of the games. It returns true if the user decides to change the graphic library or true to launch a game, and the name of the game or the graphic.
setUserName will be called before every call of the menu to set the username if already set.
getUserName will be called after the menu has ended and will take the name if it was not already set.
SetScore will be called before the menu to give the score to the menu to display it. It takes a list of this struct.



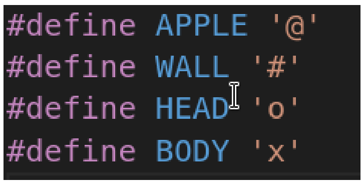
```cpp
typedef struct _score {
    size_t score;
    std::string name;
    std::string game;
} _score_t;
```

After the menu has ended, at every moment **getInput** will be called to take the input, it must be between this event.



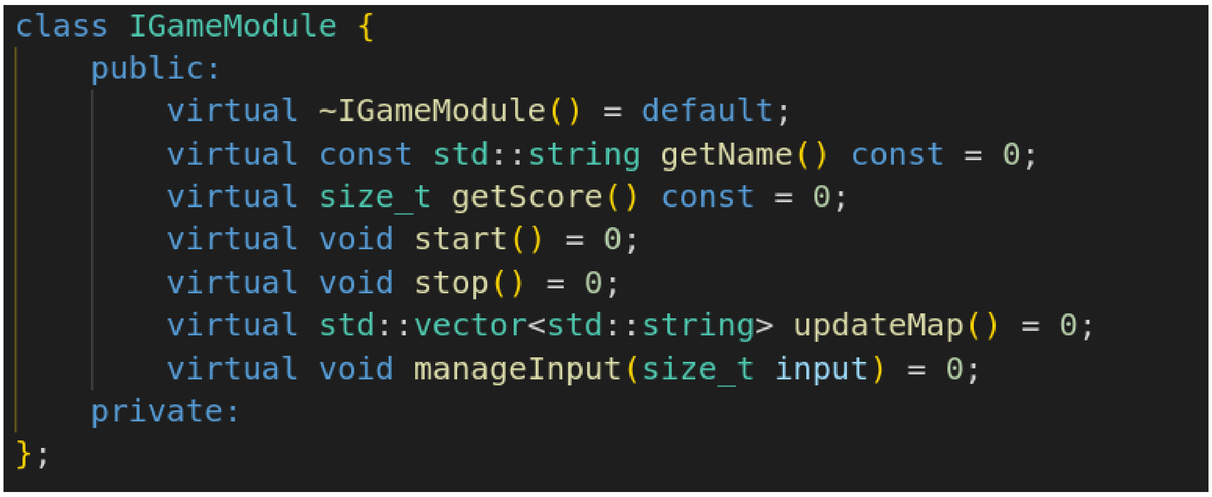
```cpp
#define UP 403           //up key
#define DOWN 402         //down key
#define LEFT 404         //left key
#define RIGHT 405        //right key
#define ENTER 10         //'\n'/ enter key
#define QUIT 113         //q key
#define SPACE 32         //space key
#define RESTART 114      // r key
#define MENU 109         // m key
#define ERROR 0          //error/no key entered
#define NEXTGAME 49      // 1 key
#define NEXTDISPLAY 50   // 2 key
```

and at every tick of the clock in the Core **drawMap** will be called, it takes a vector of string that contains the elements of the games, the elements are this ones.



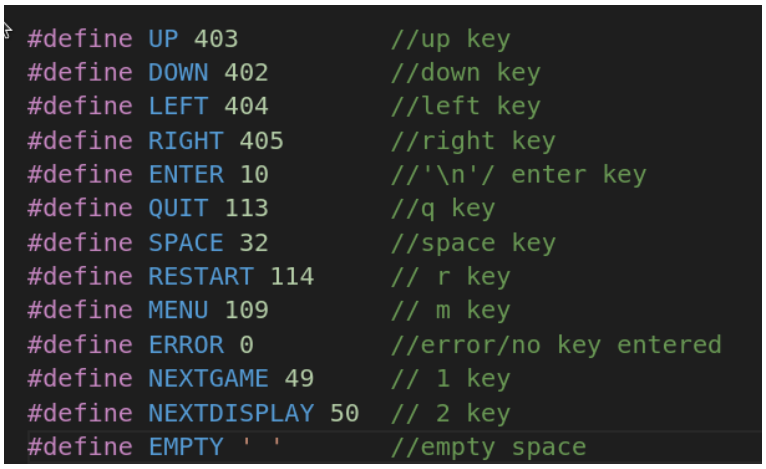
```cpp
#define APPLE '@'
#define WALL '#'
#define HEAD 'o'
#define BODY 'x'
```

To implement a new game library, you need to use this class.
To return the instance of the class the entrypoint must be called `entryDisplay`.



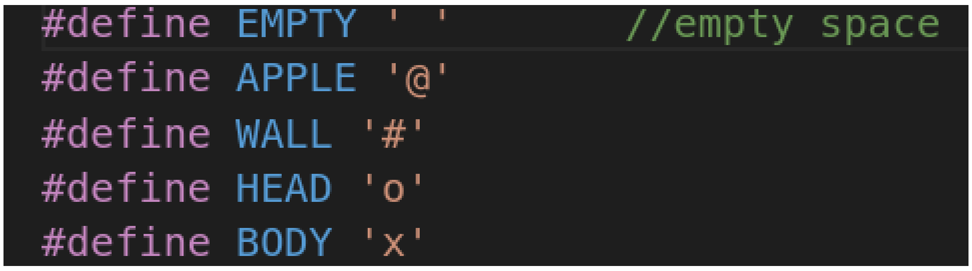
```cpp
class IGameModule {
    public:
        virtual ~IGameModule() = default;
        virtual const std::string getName() const = 0;
        virtual size_t getScore() const = 0;
        virtual void start() = 0;
        virtual void stop() = 0;
        virtual std::vector<std::string> updateMap() = 0;
        virtual void manageInput(size_t input) = 0;
    private:
};
```

Start and **stop** will be like the constructors and destructors of the class, they will be called at any time the library will be used or quit.
in the clock of the core, the **manageInput** function will be the first call to get the input.
they will look like that:



```cpp
#define UP 403              //up key
#define DOWN 402            //down key
#define LEFT 404            //left key
#define RIGHT 405           //right key
#define ENTER 10            //'\n'/ enter key
#define QUIT 113            //q key
#define SPACE 32            //space key
#define RESTART 114         // r key
#define MENU 109            // m key
#define ERROR 0             //error/no key entered
#define NEXTGAME 49         // 1 key
#define NEXTDISPLAY 50      // 2 key
#define EMPTY ' '           //empty space
```

After that the **updateMap** will be called and it will return the actual state of the game, the characters must look like this.



```cpp
#define EMPTY ' '           //empty space
#define APPLE '@'
#define WALL '#'
#define HEAD 'o'
#define BODY 'x'
```

The **getScore** will be called to take the actual score.
the **getName** will be called to take the name
To return the instance of the class the entrypoint must be called `entryGame`.