ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO THỰC HIỆN ĐỒ ÁN PROCESS 2

NHÓM 2 – LỚP 19CLC8

GV HƯỚNG DẪN: THÁI HÙNG VĂN

Thành phố Hồ Chí Minh - 2021

THÔNG TIN VÀ PHÂN CÔNG THÀNH VIÊN NHÓM

MSSV	Họ tên	Email	Phân công	Tiến độ hoàn thành
19127083	Nguyễn Hữu Tuấn	19127083@student.hcmus.edu.vn	UI của máy phụ huynh	100%
19127100	Lê Trần Gia Bảo	19127100@student.hcmus.edu.vn	Xử lí xung đột + xử lí đồng bộ, xử lí đa luồng	100%
19127294	Nguyễn Trần Thiện Toàn	19127294@student.hcmus.edu.vn	Xử lý các tính năng trên máy trẻ em (đọc, cập nhật thông tin, kiểm tra tắt máy, log off màn hình, kiểm tra thời gian sử dụng, tạo các Class	100%
19127363	Lê Văn Đông	19127363@student.hcmus.edu.vn	Thực hiện các chức năng của windows như shutdown, tắt màn hình, chụp màn hình,	100%

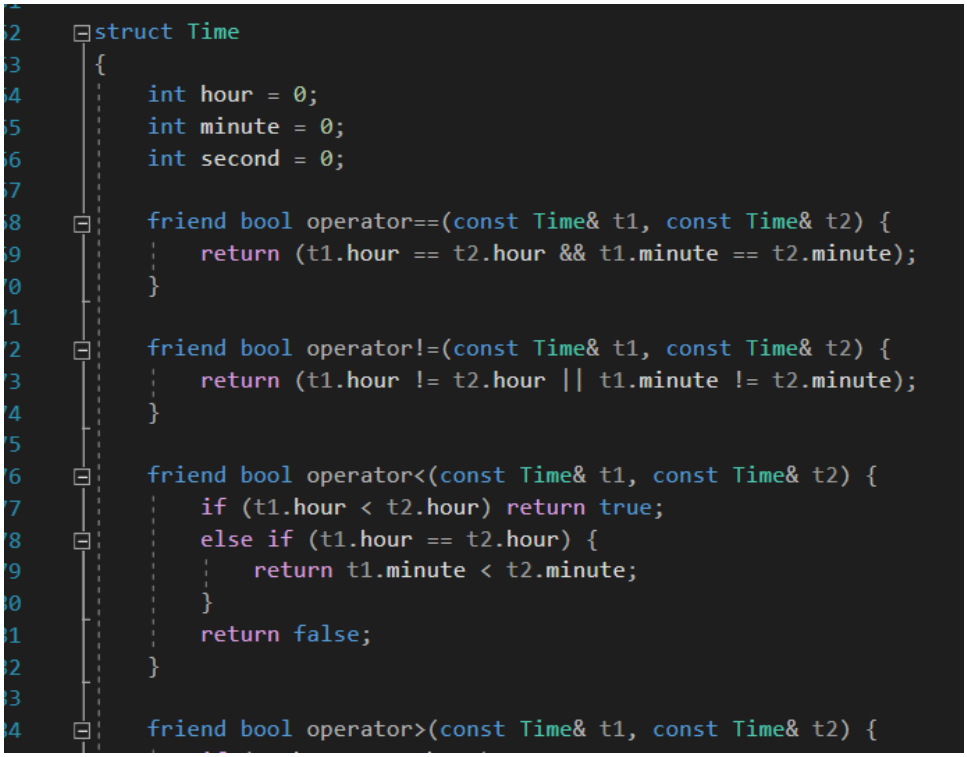
Đánh giá tiến độ hoàn thành: 100%

MỤC LỤC

1. Về xử lí cho máy trẻ em, người bị theo dõi (Chương trình C)

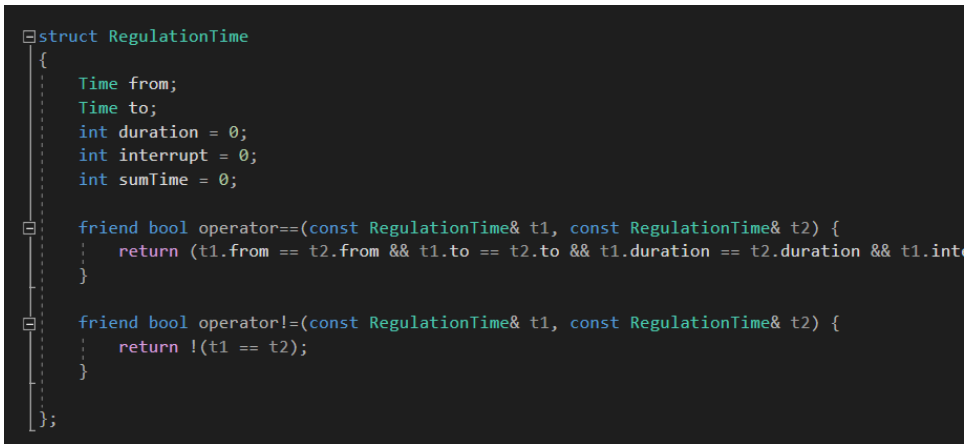
a. Các Lớp đối tượng và cấu trúc được sử dụng chính:

- Struct Time: kiểu dữ liệu lưu trữ thời gian (giờ phút, giây). Cấu trúc này được cài thêm một số operator để so sánh và tính toán.



```
struct Time
{
    int hour = 0;
    int minute = 0;
    int second = 0;

    friend bool operator==(const Time& t1, const Time& t2) {
        return (t1.hour == t2.hour && t1.minute == t2.minute);
    }

    friend bool operator!=(const Time& t1, const Time& t2) {
        return (t1.hour != t2.hour || t1.minute != t2.minute);
    }

    friend bool operator<(const Time& t1, const Time& t2) {
        if (t1.hour < t2.hour) return true;
        else if (t1.hour == t2.hour) {
            return t1.minute < t2.minute;
        }
        return false;
    }

    friend bool operator>(const Time& t1, const Time& t2) {
```

- Struct RegulationTime: Cấu trúc dùng để lưu những thông tin quy định về thời gian sử dụng.

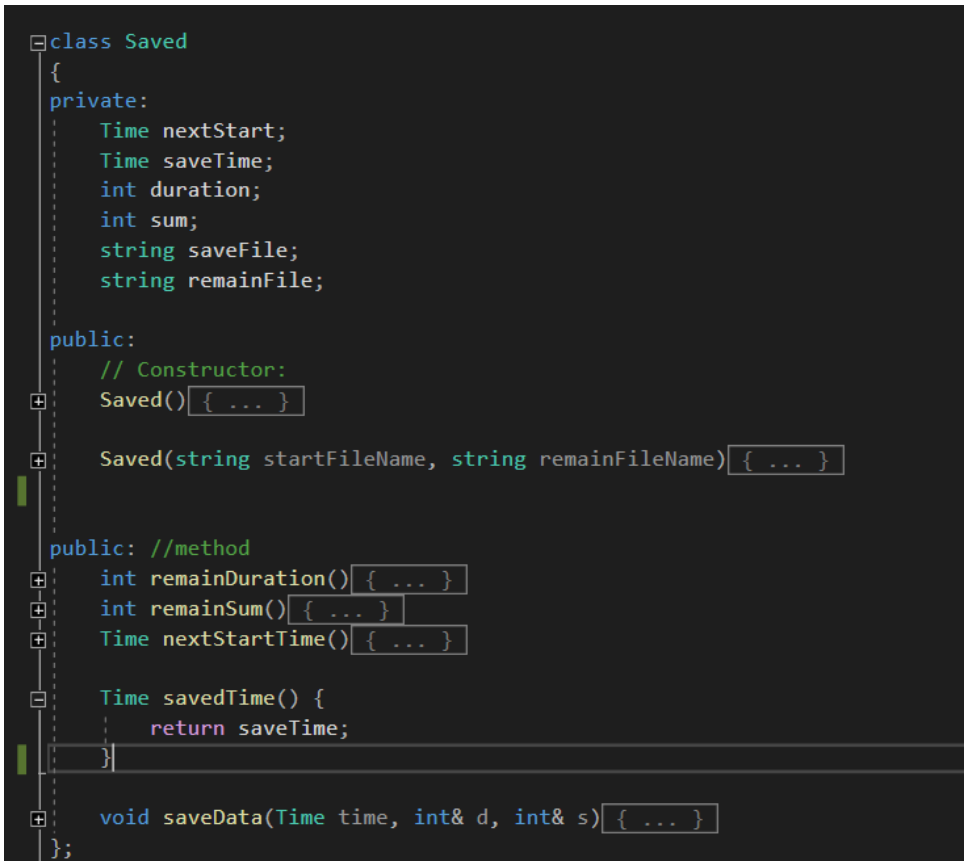


```
struct RegulationTime
{
    Time from;
    Time to;
    int duration = 0;
    int interrupt = 0;
    int sumTime = 0;

    friend bool operator==(const RegulationTime& t1, const RegulationTime& t2) {
        return (t1.from == t2.from && t1.to == t2.to && t1.duration == t2.duration && t1.int
    }

    friend bool operator!=(const RegulationTime& t1, const RegulationTime& t2) {
        return !(t1 == t2);
    }

};
```

- Class Regulation: class này là một tập hợp các thông tin RegulationTime. Các phương thức chủ yếu là đọc data quy định, kiểm tra data có cập nhật không.

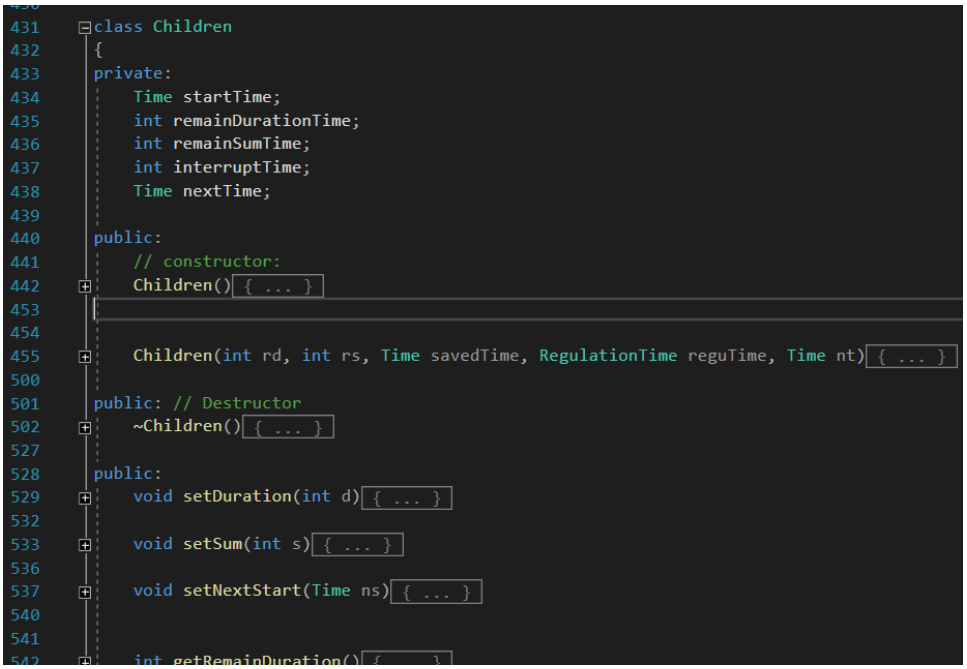
```
class Regulation
{
private:
    vector<RegulationTime> times;
    string fileData;
public:
    // constructor:
    Regulation(string filename){ ... }

public:
    // method:

    bool hasUpdate(){ ... }


    friend bool isInUseTime(Time curTime, Time savedTime, Regulation &regu, Time &nextTime, int &ir

    RegulationTime getFromToTime(int i){ ... }


    void printListReguTime(){ ... }


};
```

- Class Saved: lớp được dùng để lưu các thông tin đã dùng của trẻ như thời gian còn lại được phép sử dụng. Các thông tin về thời gian lưu, thời gian được sử dụng tiếp theo nếu trẻ bị phạt. Thông tin được lấy từ 2 file. Đó là savedStart - thời gian trẻ được phép mở máy (nếu có bị phạt, nếu không có mặc định là thời gian bắt đầu khung giờ hiện tại); file còn lại saveRemain, thông tin chính ở đây là tổng thời gian được sử dụng còn lại trong khung giờ.



```
class Saved
{
private:
    Time nextStart;
    Time saveTime;
    int duration;
    int sum;
    string saveFile;
    string remainFile;

public:
    // Constructor:
    Saved() { ... }

    Saved(string startFileName, string remainFileName) { ... }


public: //method
    int remainDuration() { ... }
    int remainSum() { ... }
    Time nextStartTime() { ... }

    Time savedTime() {
        return saveTime;
    }

    void saveData(Time time, int& d, int& s) { ... }
};
```

- Class Children: Đây là class rất quan trọng. Nó giữ tất cả các thông tin về thời gian, khung giờ. Lớp sẽ được tạo dựa trền thông tin khung giờ đang được sử dụng cùng vời những thông số cũ của khung giờ đó (trong class Saved - nếu có). Các phương thức chủ yếu là update() để cập nhật trạng thái hiện tại, countDown để trừ đi thời gian sử dụng. Destructor được tạo nhằm lưu thông tin của quá trình vừa dụng khi chương trình kết thúc.



```
431  class Children
432  {
433  private:
434      Time startTime;
435      int remainDurationTime;
436      int remainSumTime;
437      int interruptTime;
438      Time nextTime;
439
440  public:
441      // constructor:
442      Children() { ... }
453
454
455      Children(int rd, int rs, Time savedTime, RegulationTime reguTime, Time nt) { ... }
500
501  public: // Destructor
502      ~Children() { ... }
527
528  public:
529      void setDuration(int d) { ... }
532
533      void setSum(int s) { ... }
536
537      void setNextStart(Time ns) { ... }
540
541
542      int getRemainDuration() { ... }
```

b. Các hàm xử lý máy trẻ em

- Hàm getDirCapture() dùng để lấy đường dẫn mặc định, không có tham số. Kết quả trả về kiểu string là đường dẫn: "./history/năm_tháng_ngày/giờ_phút.png", thời gian năm, tháng, ngày, giờ, phút là thời gian hệ tại của hệ thống.
- Hàm captureScreen(string dir). Hàm này dùng để chụp màn hình. Tham số truyền vào là đường dẫn và tên file. Ta có thể dùng hàm getDirCapture() dùng làm tham số.
- Hàm turnoffScreen(int second) hàm này nhằm để tắt màn hình trong second giây. Khi gọi hàm, màn hình sẽ tắt tạm thời, nhằm không cho trẻ em thao tác trên màn hình. Khi hết thời gian tắt màn hình, trẻ có thể tiếp tục sử dụng chương trình.
- Hàm scaleIMG(string des, int height, int width). Chức năng của hàm này nhằm thay đổi kích thước của hình ảnh lại để hiển thị trên GUI của phụ huynh. des là đường dẫn + tên file ảnh cần lưu, height là chiều cao của hình, width là chiều rộng của hình.
- Hàm getTime() sẽ trả về kiểu dữ liệu Time bao gồm giây, phút, giờ hiện tại.

- Hàm MySystemShutdown(): hàm này tận dụng Window API của hệ điều hành để tắt máy.
- isInUseTime(): hàm kiểm tra xem liệu có đang trong thời được phép xử dụng hay không. Thời gian để so sánh được lấy từ cloud và những thông tin thời gian đã sử dụng trước đó (sum, có bị phạt 10 phút không,…). Đồng thời hàm sẽ giúp biết được khung giờ kế tiếp là mấy giờ.
- InputPassword(): hàm sẽ gọi hàm kiểm tra isInUseTime() và kiểm tra đây là ba mẹ hay trẻ em.
- CheckUpdate(): hàm sẽ gọi hàm update của class Regulation. Nếu có thay đổi sẽ sẽ gọi tiếp các hàm update của các class khác đồng thời kiểm tra luôn liệu còn được sử dụng không.

c. Xử lý đa luồng

- Hai thread được sử dụng để thực hiện song song việc đếm 15s và nhập mật khẩu là:
  - Thread t1(inputThread,BAME): thread với đối số truyền vào là hàm inputThread(). Hàm này sẽ cho người dùng nhập pass và chỉ dừng khi người dùng nhập đúng pass phụ huynh. Sau đó nó sẽ đặt biến toàn cục isLogin bằng true.
  - Threadt2(countDown,ref(countDownTime)): thread với đối số truyền vào là hàm contDown(int& time). Hàm này sẽ liên tục đếm ngược cho đến khi biến time bằng 0, sau đó nó sẽ kiểm tra biến isLogin nếu bằng false se thông báo người dùng bật lại máy sau 10 phút và shutdown hệ điều hành.
  - Lưu ý, ở hàm main() sau khi tạo hai thread trên ta join thread t1 trước để chương trình đợi thread t1 kết thúc. Nếu t1 nhập đúng pass phụ huynh trước 15s thì ta sẽ join thread t2 để ngắt việc shutdown. Nếu t1 nhập sai pass hoặc hết thời gian timeout thì t2 sẽ shutdown hệ điều hành.
- Ba thread được sử dụng để thực hiện song song việc chụp màn hình, kiểm tra thay đổi của file môi phút và thông báo khi thời gian sử dụng còn 1 phút là:

- Thread capture(captureScreen): thread với đối số truyền vào là hàm captureScreen(). Hàm này sẽ chụp lại màn hình và lưu vào folder, sau đó lặp lại mỗi phút và chỉ dừng khi tổng thời gian sử dụng còn lại bằng 0.
- Thread check(checkUpdateData): thread với đối số truyền vào là hàm checkUpdateData(). Hàm này sẽ kiểm tra mỗi phút trạng thái update bằng hàm checkUpdate(), hàm này sẽ kiểm tra và return giá trí vào biến updateStatus. Nếu status = 0 – không có update, status = 1 - có update và thông báo cho người dùng thời gian còn lại, status = -1 - có update và đã hết thời gian sử dụng sau đó shutdown hệ điều hành. Hàm checkUpdateData() chỉ dừng khi tổng thời gian sử dụng còn lại bằng 0.
- Thread notify(notificate()): thread với đối số truyền vào là hàm notificate(). Hàm này sẽ lặp lại mỗi phút và trừ thời gian còn lại và tổng thời gian còn lại đi 1 phút. Hàm sẽ kiểm nếu thời gian còn lại đã hết sẽ tắt màn hình và nghỉ trong thời gian interupt cho trước, nếu thời tổng thời gian còn 1 phút thì sẽ thông báo cho người dùng.
- Ở main, ta sẽ join cả 3 thread trên, main se đợi cả 3 thread thực thi xong. Khi đó, main sẽ shutdown hệ điều hành vì cả 3 thread chỉ dừng khi tổng thời gian còn lại bằng 0.

d. Cơ chế hàm main()

- Đầu tiên một đối tượng Regulation được khởi tạo dựa trên thông tin quy định. Sau đó đến Saved, Children(tre). Các biến toàn cục như gNextTime cũng được sử để tiện cho việc theo dõi. Một vòng lặp do while được thực hiện, nó chỉ lặp lại nếu ứng với người dùng là ba mẹ. Đầu tiên nó sẽ thông báo nhập password. Biến status lưu trạng thái trả về của hàm nhập pass.
- Ứng với status = 1: cha mẹ đang dùng máy của trẻ. Việc kiểm tra chỉ thực hiện sau N*phút (trong code là 2 phút).
- Case = 0: không trong khuôn giờ cho phép. Lúc này một hộp thoại sẽ thông báo đến mấy giờ mới được dùng lại. Đồng thời nó sẽ tạo ra 2 tiến trình song song. Một

tiến trình kiểm tra coundown trong 15s để tắt máy, một cái để kiểm tra nếu nhập đúng pass ba me thì tiếp tục bình thường.

- Case = -1: Nhập đúng pass trẻ em và trong khung giờ sử dụng. Thông báo còn bao nhiêu phút và tạo ra 3 tiến trình. Các tiến trình kiểm tra chụp hình sau mỗi phút, kiểm tra cập nhật, kiểm tra hết giờ.
    - Case = -2: nhập sai 3 lần và tắt máy.

2. Về xử lí cho máy người lớn, người theo dõi (Chương trình P)

a. Về giao diện người dùng

- Giao diện người dùng được cài đặt bằng cách sử dụng thư viện SDL để thiết kế đồ họa
- Về phần code, giao diện người dùng được chia ra làm 3 phần lớn: Các hàm phụ trợ, nhóm hàm Menu và nhóm hàm xử lí Input/Output.
- Ngoài ra, ta cần một lớp dành cho việc hiển thị Text lên màn hình, gọi là Class::TextObject

i. Class TextObject

- Class TextObject được viết theo hướng OOP, được kế thừa từ Class BaseObject với 2 biến cơ bản là rect_ - lưu trữ tọa độ của Object và p_object_ lưu trữ dữ liệu có dạng SDL_Surface – dùng để hiển thị nội dung lên màn hình với các chức năng cơ bản như get, set. Ngoài ra, còn có hàm LoadImg để tải dữ liệu (hình ảnh, text, …) lên biến để hiển thị, ở đây là biến p_object_. Hàm show để hiển thị hình ảnh lên màn hình console.

```cpp
class BaseObject
{
protected:
    SDL_Rect rect_;
    SDL_Surface* p_object_;
public:
    BaseObject();
    ~BaseObject();

    void show(SDL_Surface* des);
    bool LoadImg(std::string filename);

    void setRect(const int& x, const int& y) { rect_.x = x, rect_.y = y; }
    SDL_Rect getRect() const { return rect_; }
    SDL_Surface* getObject() { return p_object_; }

};
```

- Class TextObject với hàm nổi bật là hàm createText, với chức năng để hiển thị chữ lên màn hình

```cpp
class TextObject : public BaseObject
{
private:
    std::string str;
    SDL_Color text_color;

public:
    enum TextColor
    {
        RED_TEXT = 0,
        WHITE_TEXT = 1,
        BLACK_TEXT = 2,
    };

    void setText(const std::string& text) {
        str = text;
    }

    void setColor(const int& type);

    static int InitFont();

    void createText(TTF_Font* font, SDL_Surface* des);

};
```

- Dưới đây là hàm createText():

```
57  void TextObject::createText(TTF_Font* font, SDL_Surface* des) {
58      p_object_ = TTF_RenderText_Solid(font, str.c_str(), text_color);
59      show(des);
60  }
61
```

ii. Nhóm hàm chức năng

- Nhóm hàm chức năng được đưa vào một namespace tên SDLCommonFunction, để tránh trùng lặp với các hàm được thêm vào sau này.
- Nhóm hàm chức năng gồm các hàm sau đây:

```
42  namespace SDLCommonFunction {
43      void menuInit();
44      SDL_Rect ApplySurface(SDL_Surface* src, SDL_Surface* des, int x, int y);
45      void CleanUp();
46      bool checkfocuswithRect(const int& x, const int& y, const SDL_Rect& _rect);
47      SDL_Surface* LoadImg(std::string file_path);
48      void getFilePath();
49      // File
50      vector<string> getFile(string filename);
51      vector<string> getEditFile(string filename);
52      void updateEditFile(string filename, vector<string> data);
53
```

- Hàm nổi bật của nhóm hàm này là ApplySurface(); checkfocuswithRect() và LoadImg().
    - Với hàm ApplySurface(), hàm này đưa object - ở đây là src lên console - ở đây là biến des ở tọa độ (x,y) và trả về tọa độ của object mới đưa lên (bao gồm width – w và height – h)

```
37  SDL_Rect SDLCommonFunction::ApplySurface(SDL_Surface* src, SDL_Surface* des, int x, int y)
38  {
39      SDL_Rect offset;
40      offset.x = x;
41      offset.y = y;
42      SDL_BlitSurface(src, NULL, des, &offset);
43
44      return offset;
45  }
```

    - Với hàm checkfocuswithRect(), hàm này để kiểm tra xem trỏ chuột hiện tại có nằm trong vùng obbject hay không (dùng để click vào một title nào đó)

```
116 bool SDLCommonFunction::checkfocuswithRect(const int& x, const int& y, const SDL_Rect& _rect) {
117     if ((x >= _rect.x) && (x <= _rect.x + _rect.w) && (y >= _rect.y) && (y <= _rect.y + _rect.h))
118         return true;
119     return false;
120 }
```

- Với hàm LoadImg(), hàm này được áp dụng để tải hình (từ file .png, .jpg, .bmp, …) và lưu vào biến SDL_Surface, sau đó có thể đưa lên màn hình nhờ hàm ApplySurface()

```
16 SDL_Surface* SDLCommonFunction::LoadImg(std::string file_path)
17 {
18     SDL_Surface* load_image = IMG_Load(file_path.c_str());
19     SDL_Surface* optimize_image = NULL;
20
21     if (load_image != NULL) {
22         optimize_image = SDL_DisplayFormat(load_image);
23         SDL_FreeSurface(load_image);
24
25         if (optimize_image != NULL)
26         {
27             UINT32 color_key = SDL_MapRGB(optimize_image->format, 0, 0x00, 0x00);
28             SDL_SetColorKey(optimize_image, SDL_SRCCOLORKEY, color_key);
29         }
30     }
31
32     return optimize_image;
33 }
```

iii. Nhóm hàm thực hiện chức năng Nhập/Xuất

- Nhóm hàm này bao gồm 2 hàm, đó là PrintModifiers() và PrintKeyInfo(), có chức năng nhận biết mỗi khi người dùng thực hiện việc Input từ bàn phím (bởi vì SDL không có chức năng nhập từ bàn phím nên ta cần hàm này để thực hiện việc nhập, xuất từ bàn phím)

```
65     // Input From Keyboard
66     void PrintModifiers(SDLMod mod);
67     string PrintKeyInfo(SDL_KeyboardEvent* key);
```

- Hàm PrintModifiers(), có chức năng in lên console những phím chức năng (có thể kể đến như Shift, Enter, Caps lock, …), từ đó ta có thể xử lí việc nhập xuất

```
604  void SDLCommonFunction::PrintModifiers(SDLMod mod) {
605      printf("Modifers: ");
606
607      if (mod == KMOD_NONE) {
608          printf("None\n");
609          return;
610      }
611
612      if (mod & KMOD_NUM) printf("NUMLOCK ");
613      if (mod & KMOD_CAPS) printf("CAPSLOCK ");
614      if (mod & KMOD_LCTRL) printf("LCTRL ");
615      if (mod & KMOD_RCTRL) printf("RCTRL ");
616      if (mod & KMOD_RSHIFT) printf("RSHIFT ");
617      if (mod & KMOD_LSHIFT) printf("LSHIFT ");
618      if (mod & KMOD_RALT) printf("RALT ");
619      if (mod & KMOD_LALT) printf("LALT ");
620      if (mod & KMOD_CTRL) printf("CTRL ");
621      if (mod & KMOD_SHIFT) printf("SHIFT ");
622      if (mod & KMOD_ALT) printf("ALT ");
623      printf("\n");
624  }
```

- Hàm PrintKeyInfo(), cho ta biết thông tin tất cả các phím nhập từ bàn phím, có thể tham khảo link sau để biết tên của các phím: [SDLKey (libsdl.org)]

```
626  string SDLCommonFunction::PrintKeyInfo(SDL_KeyboardEvent* key) {
627      if (key->type == SDL_KEYUP)
628          printf("Release:- ");
629      else
630          printf("Press:- ");
631
632      printf("Scancode: 0x%02X", key->keysym.scancode);
633
634      string name = SDL_GetKeyName(key->keysym.sym);
635      printf(", Name: %s", SDL_GetKeyName(key->keysym.sym));
636
637      if (key->type == SDL_KEYDOWN) {
638
639          printf(", Unicode: ");
640          if (key->keysym.unicode < 0x80 && key->keysym.unicode > 0) {
641              printf("%c (0x%04X)", (char)key->keysym.unicode,
642                  key->keysym.unicode);
643          }
644          else {
645              printf("? (0x%04X)", key->keysym.unicode);
646          }
647      }
648      printf("\n");
649      PrintModifiers(key->keysym.mod);
650      return name;
651  }
```

iv. Nhóm hàm show Menu

- Nhóm hàm show Menu gồm rất nhiều hàm, tuy nhiên nổi bật nhất là hàm showMenu() và Edit(), còn lại thì tương tự như 2 hàm trên, tùy vào chức năng mà ta sẽ code sao cho phù hợp. Dưới đây là các hàm:

```
54      // Show Menu
55      int showMenu(SDL_Surface* des);
56      int showHistory(SDL_Surface* des);
57      int showImage(SDL_Surface* des, int i);
58      int showEdit(SDL_Surface* des);
59      int Edit(SDL_Surface* des, int i);
60      int update(SDL_Surface* des);
61      int isWaiting(SDL_Surface* des);
```

- Hàm showMenu() có công dụng hiển thị menu; showHistory() có nhiệm vụ hiển thị một list hình ảnh để người dùng có thể click vào để xem; showImage() để hiển thị hình ảnh mà người dùng click vào; showEdit() để hiển thị màn hình gồm các phần mà người dùng muốn chỉnh sửa (hoặc thêm mới); Edit() để chỉnh sửa nội dung (ở đây là dòng I trong file) người dùng vừa bấm vào ở hàm showEdit(); hàm update() để cập nhập nội dung người dùng thêm mới vào file và hàm isWaiting() để hiển thị cho người dùng biết có người đang chỉnh sửa chung với mình hay không.
    - Hàm showMenu():
        - Trước tiên, ta cần khởi tạo biến phông nền và font chữ cho chữ được hiển thị trên Menu

```
143 int  SDLCommonFunction::showMenu(SDL_Surface* des) {
144     menu_b = SDLCommonFunction::LoadImg(image_background);
145     if (menu_b == NULL) {
146         std::cout << "\n==============Khong the load anh Menu. Kiem tra lai ten file va duong dan==============\n";
147         return 0;
148     }
149
150     if (TTF_Init() == -1) {
151         cout << "\n================Khong the Init Font================\n";
152         return false;
153     }
154
155     g_font_text = TTF_OpenFont(font_local.c_str(), 40);
156     if (g_font_text == NULL) {
157         std::cout << "\n================Loi font Menu======== o day=========\n";
158         printf("TTF_OpenFont: %s\n", TTF_GetError());
159         return 0;
160     }
```

- Sau đó, ta khởi tạo các biến cần thiết, mà ở đây là tọa độ, nội dung. Biến selected để kiểm tra con trỏ chuột có nằm trong vùng chữ hoặc việc bấm chuột có bấm vào trong vùng có chữ hay không. Biến m_event cho ta lấy được các sự kiện diễn ra trong khi chương trình đang chạy (có thể kể đến việc nhập/xuất, việc di chuyển chuột, việc bấm chuột, …)

```
163     SDL_Rect pos_Arr[kMenu];
164     pos_Arr[0].x = x_menu_pixel;
165     pos_Arr[0].y = y_menu_pixel;
166
167     pos_Arr[1].x = x_menu_pixel;
168     pos_Arr[1].y = y_menu_pixel + 40 + 10;
169
170     pos_Arr[2].x = x_menu_pixel;
171     pos_Arr[2].y = pos_Arr[1].y + 40 + 10;
172
173     TextObject text_menu[kMenu];
174
175     text_menu[0].setColor(TextObject::BLACK_TEXT);
176     text_menu[0].setText("XEM LICH SU");
177     text_menu[0].setRect(pos_Arr[0].x, pos_Arr[0].y);
178
179     text_menu[1].setColor(TextObject::BLACK_TEXT);
180     text_menu[1].setText("CHINH SUA");
181     text_menu[1].setRect(pos_Arr[1].x, pos_Arr[1].y);
182
183     text_menu[2].setColor(TextObject::BLACK_TEXT);
184     text_menu[2].setText("THOAT");
185     text_menu[2].setRect(pos_Arr[2].x, pos_Arr[2].y);
186
187     bool selected[kMenu] = { 0, 0, 0 };
188     int xm = 0;
189     int ym = 0;
190     SDL_Event m_event;
```

- Tiếp theo, trong vòng while, ta sẽ cho hiển thị những thứ vừa setup lên màn hình

```
191     while (true) {
192         SDLCommonFunction::ApplySurface(menu_b, des, 0, 0);
193
194         for (int i = 0; i < kMenu; i++) {
195             text_menu[i].createText(g_font_text, des);
196         }
197
```

- Sau đó, ta bắt đầu lấy các sự kiện (event) mà người dùng thao tác. Trường hợp SDL_MOUSEMOTION (di chuyển chuột), ta sẽ thay đổi màu của chữ mỗi khi người dùng di chuột đến vùng có chữ. Dưới đây là code và ví dụ của việc di chuột


```
199         while (SDL_PollEvent(&m_event)) {
200             switch (m_event.type)
201             {
202             case SDL_QUIT: {
203                 return kMenu - 1;
204             }
205             case SDL_MOUSEMOTION:
206             {
207                 xm = m_event.motion.x;
208                 ym = m_event.motion.y;
209 
210                 for (int i = 0; i < kMenu; i++) {
211                     SDL_Rect rectmenu = text_menu[i].getRect();
212                     if (SDLCommonFunction::checkfocuswithRect(xm, ym, rectmenu)) {
213                         if (selected[i] == false) {
214                             selected[i] = 1;
215                             text_menu[i].setColor(TextObject::RED_TEXT);
216                         }
217                     }
218                     else
219                     {
220                         if (selected[i] == true) {
221                             selected[i] = 0;
222                             text_menu[i].setColor(TextObject::BLACK_TEXT);
223                         }
224                     }
225                 }
226                 break;
227             }
```

- Tương tự với việc di chuột, ta cũng có việc nhấn chuột cũng là một event. Ta sẽ trả về giá trị mỗi khi các dòng chữ trên màn hình được click vào; lúc đó ta có thể di chuyển đến một

cửa sổ mới. Sau khi kết thúc các trường hợp của m_event, ta sẽ update lại mành hình bằng hàm SDL_Flip và kết thúc hàm.

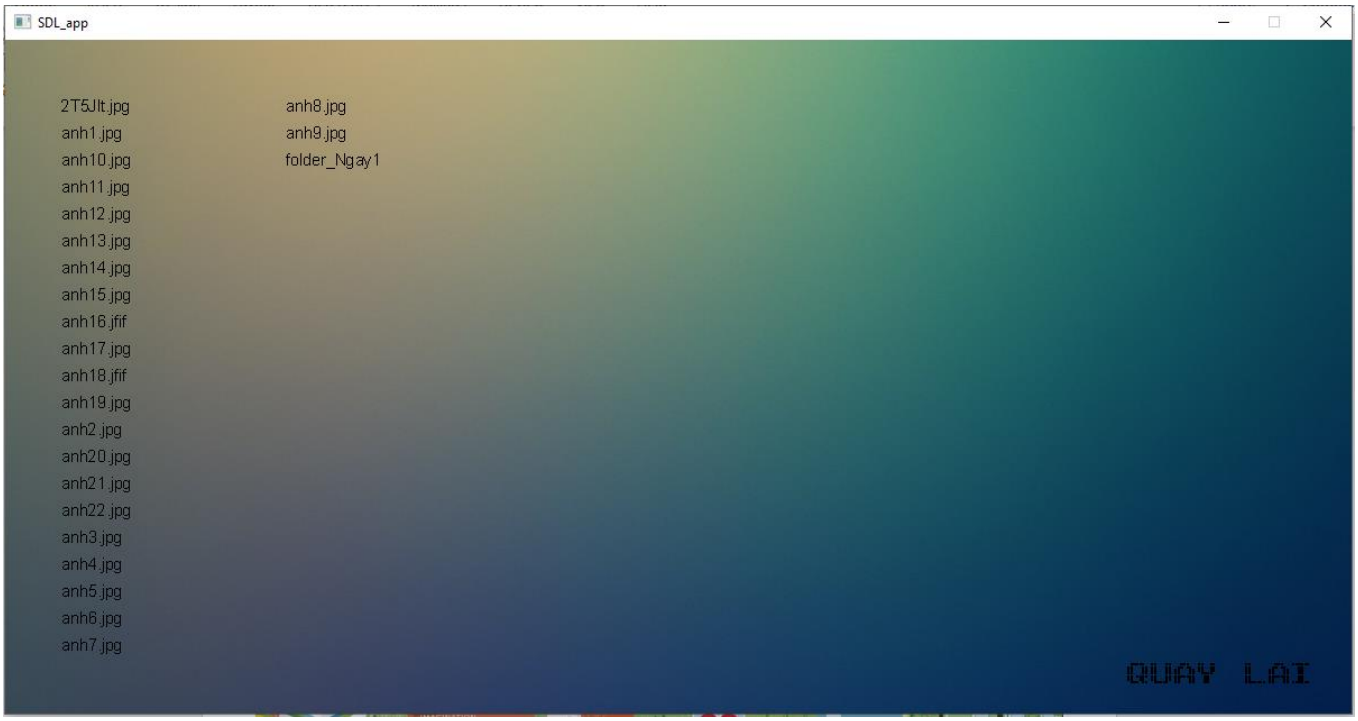
```
228             case SDL_MOUSEBUTTONDOWN:
229             {
230                 xm = m_event.button.x;
231                 ym = m_event.button.y;
232                 int i;
233                 for (i = 0; i < kMenu; i++) {
234 
235                     if (SDLCommonFunction::checkfocuswithRect(xm, ym, text_menu[i].getRect()) == true) {
236                         return i;
237 
238 
239                     }
240                 }
241 
242                 break;
243 
244             }
245             case SDL_KEYDOWN: {
246                 if (m_event.key.keysym.sym == SDLK_ESCAPE)
247                     return kMenu - 1;
248                 break;
249             }
250             default:
251                 break;
252             }
253             SDL_Flip(des);
254         }
255 
256     }
257     return 0;
258 }
```

- Hàm Edit(): tương tự với hàm showMenu(), tuy nhiên ở hàm Edit(), ngoài các sự kiện với chuột, ta còn có các sự kiện với phím. Dưới đây là xử lí sơ bộ cho việc Nhập/Xuất:

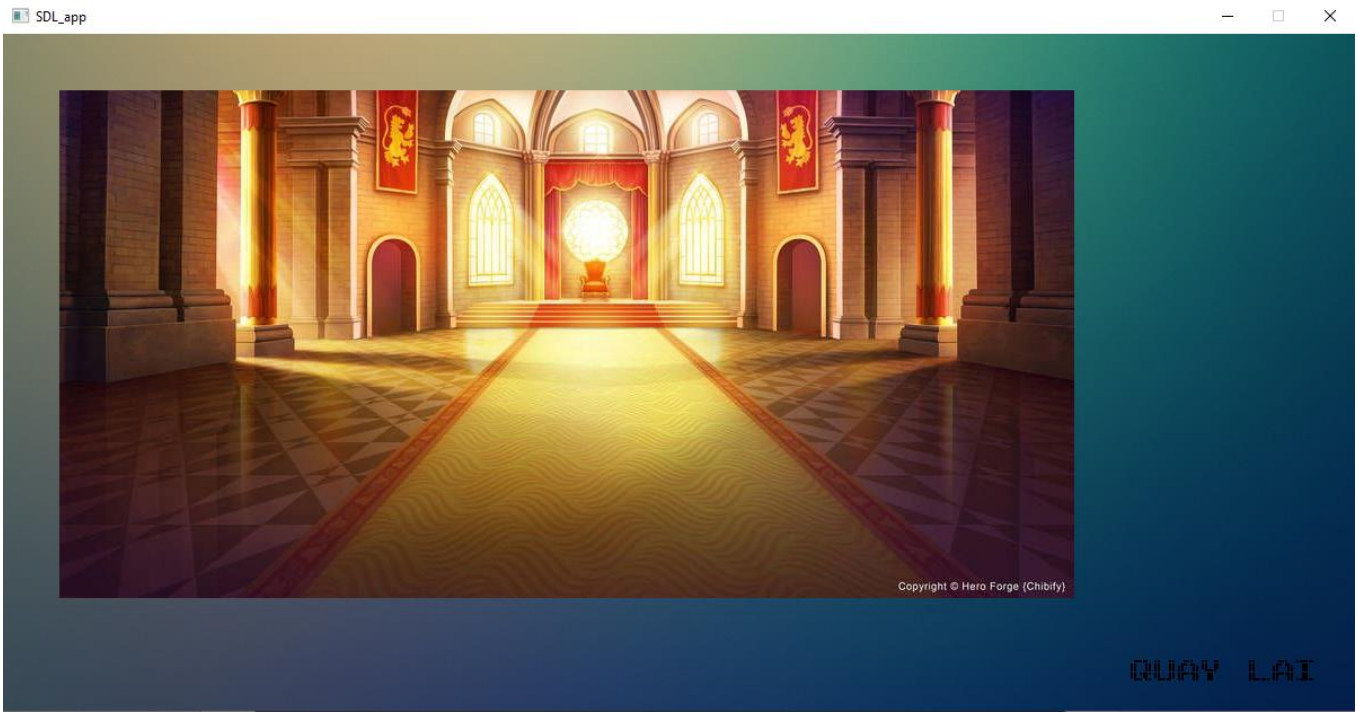
```
739             case SDL_KEYUP: {
740                 temp = SDLCommonFunction::PrintKeyInfo(&m_event.key);
741
742                 if (temp == "backspace") {
743                     if (s.size() != 0)
744                         s.pop_back();
745                 }
746                 else if (temp == "space") {
747                     s += " ";
748                 }
749                 else if (temp == "return") {
750                     file[i - 2] = s;
751                     SDLCommonFunction::updateEditFile(editing_path, file);
752                     return 0;
753                 }
754                 else if (temp == "semicolon") {
755                     s += ';';
756                 }
757                 else s += toupper(temp[0]);
758                 SDLCommonFunction::ApplySurface(g_image_history, des, 0, 0);
759                 back.createText(g_font_text, des);
760                 text.createText(g_font_text, des);
761                 str.setText(s);
762                 str.createText(g_font_text, des);
763                 break;
764             }
```

- Dưới đây là màn hình của toàn bộ chương trình:
    - Menu chính

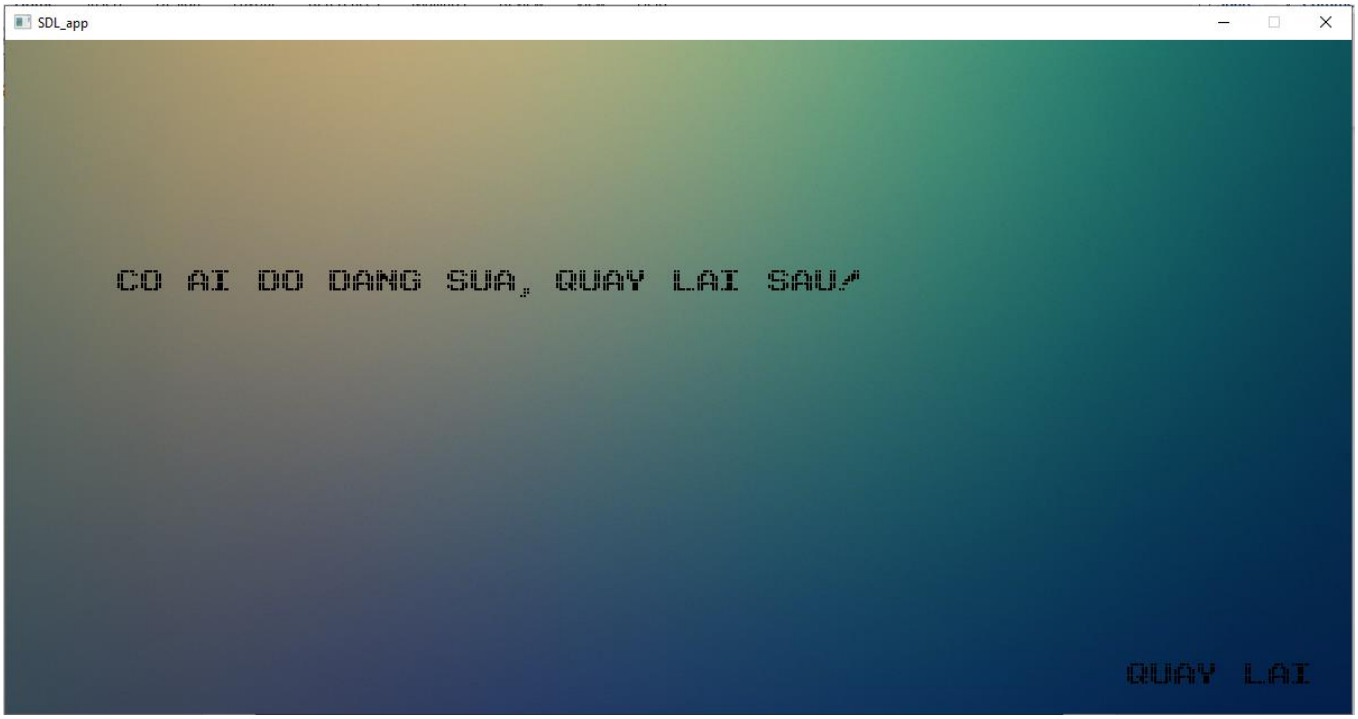
    - Menu xem ảnh



- Click vào ảnh thì hiển thị ảnh

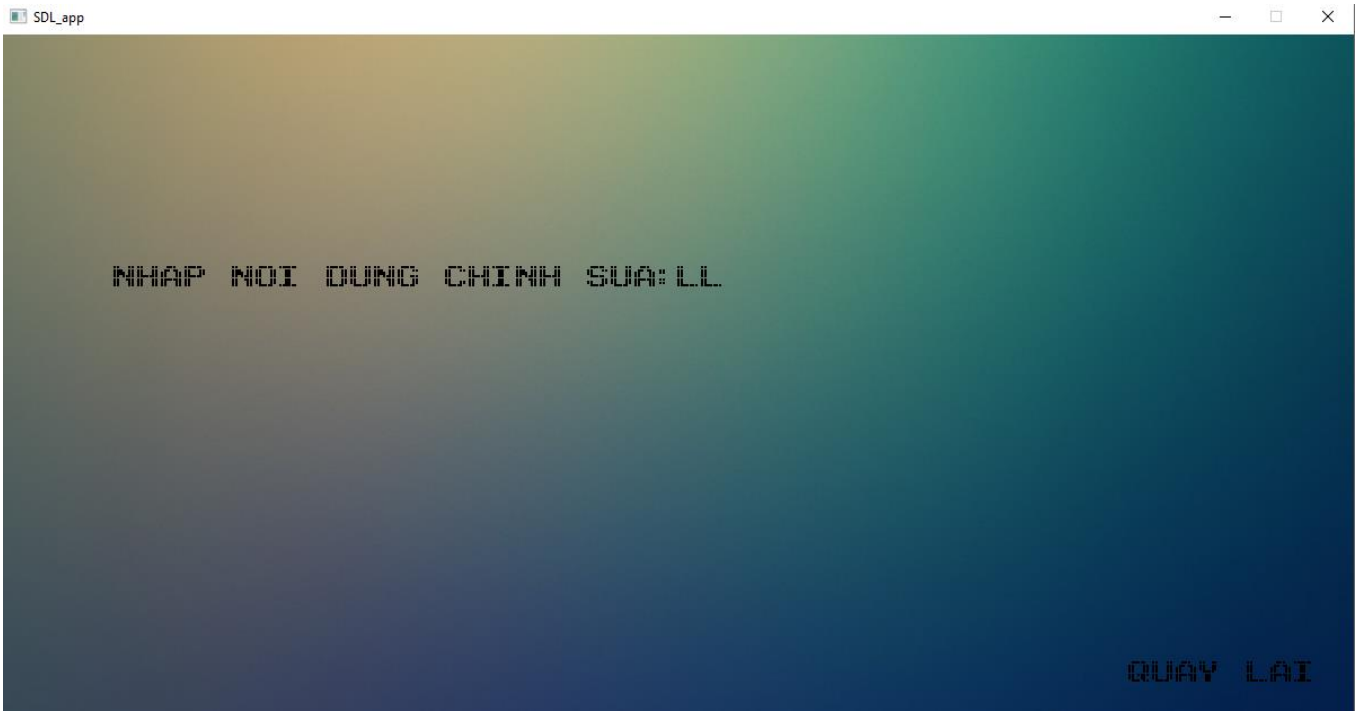


- Menu chỉnh sửa khi có người đang chỉnh

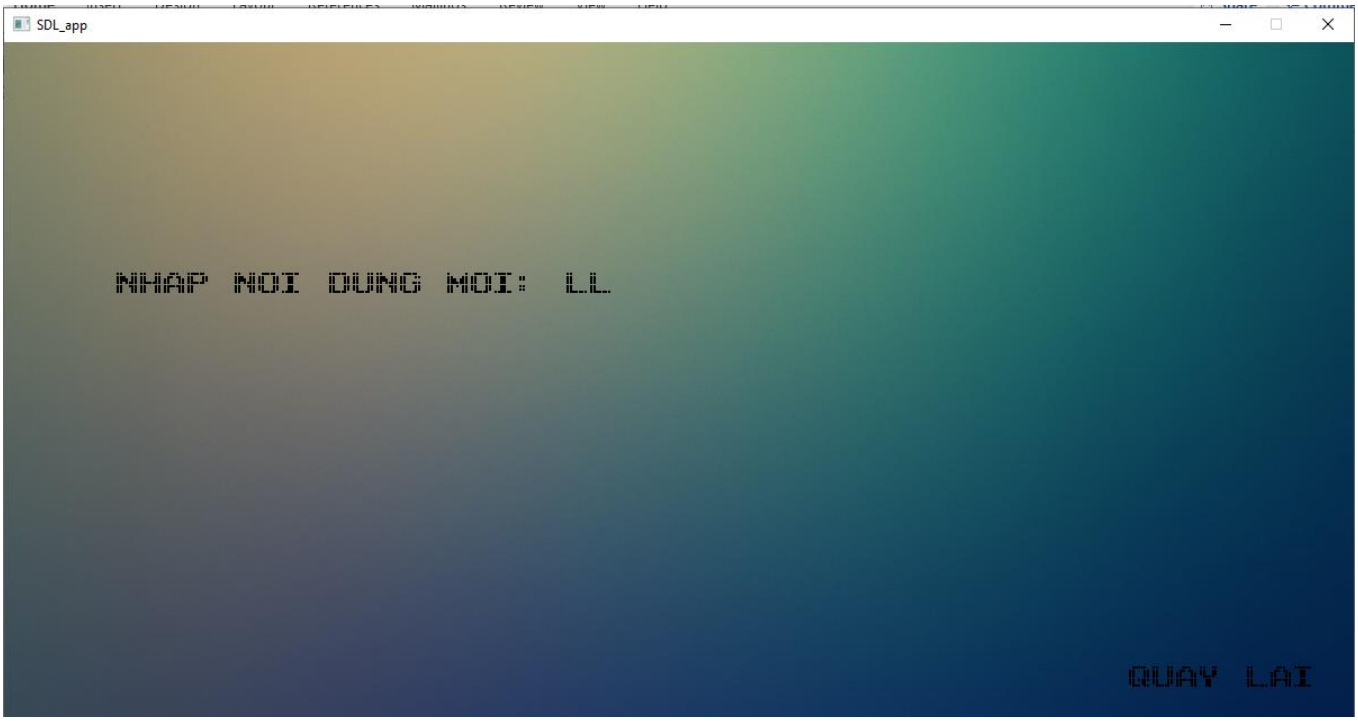


- Nếu không có ai chỉnh sửa

- Khi click vào một nội dung



- Khi click vào “THEM MOI”

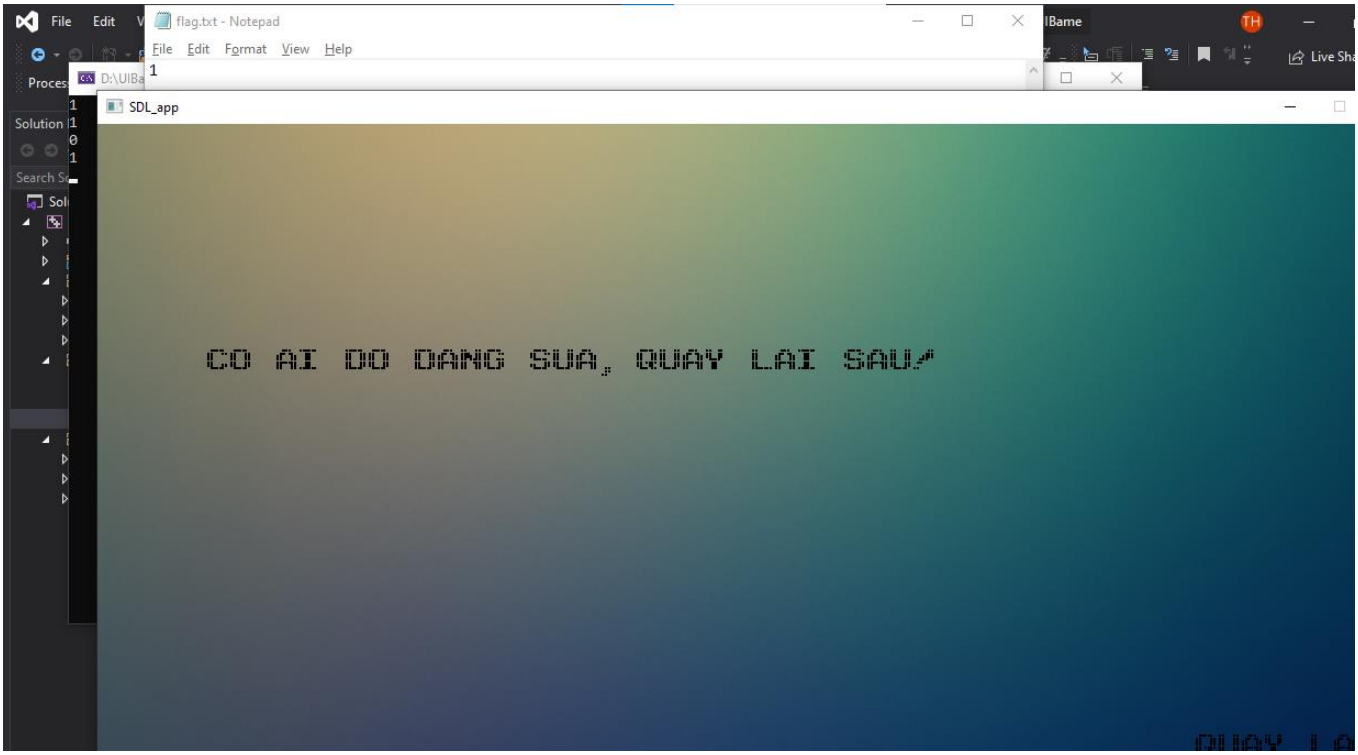


b. Về xử lí critical section

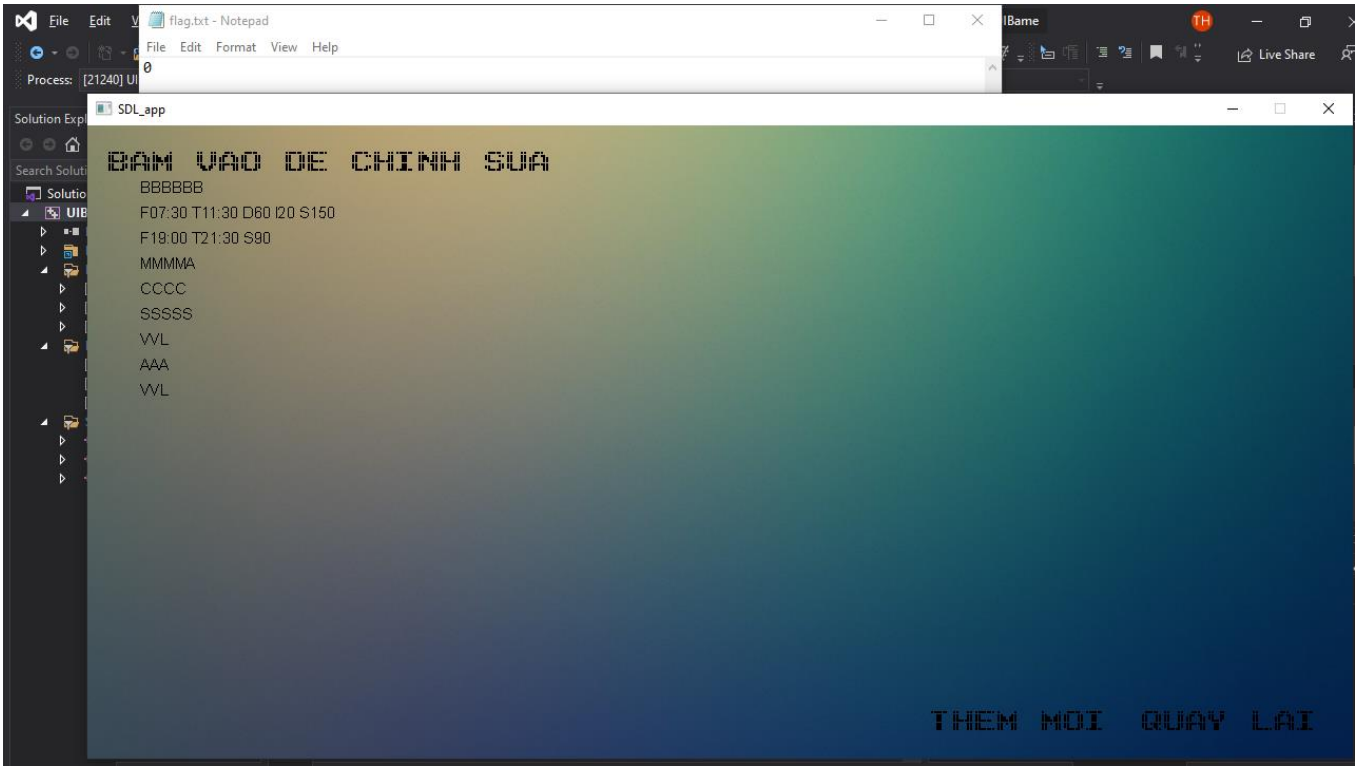
- Ở đây, ta sẽ sử dụng cờ hiệu (flag) để xử lí tranh chấp giữa 2 người dùng. Nếu có người đang chỉnh sửa, flag sẽ lưu thành 1 và với người dùng đến sau, sẽ không có quyền chỉnh sửa. Ngược lại, nếu flag = 0 thì người dùng đến sau sẽ có quyền chỉnh sửa.
- Ta có hàm checkFlag() và setFlag() để lấy và chỉnh sửa lại cờ hiệu trong file:

```
123  void SDLCommonFunction::checkFlag(int& flag) {
124      fstream flag_file;
125      flag_file.open("flag.txt", ios::in);
126      if (flag_file.is_open()) {
127          string temp;
128          getline(flag_file, temp);
129          flag = stoi(temp);
130          flag_file.close();
131      }
132  }
133
134  void SDLCommonFunction::setFlag(int value) {
135      fstream flag_file;
136      flag_file.open("flag.txt", ios::out | ios::trunc);
137      flag_file << value;
138      flag_file.close();
139  }
```

- Đây là kết quả với việc sử dụng cờ hiệu:
    - Nếu flag = 1:


flag.txt - Notepad
File Edit Format View Help
1
SDL_app
CO AI DO DANG SUA, QUAY LAI SAU!

- Nếu flag = 0:


flag.txt - Notepad
File Edit Format View Help
0
SDL_app
BAM VAO DE CHINH SUA
BBBBBB
F07:30 T11:30 D60 I20 S150
F19:00 T21:30 S90
MMMMA
CCCC
SSSSS
VVL
AAA
VVL
THEM MOI QUAY LAI

THAM KHẢO

Introduce to SDL. (n.d.). Retrieved from https://thenumbat.github.io/cpp-course/sdl2/01/01.html

Lập trình game với SDL. (n.d.). Retrieved from https://phattrienphanmem123az.com/lap-trinh-game-cpp/bai-1-gioi-thieu-cai-dat.html?fbclid=IwAR38kocA-SFmz3SEi70BSDm_qIzEElEKdpBrS5N7oVKk5jq_90bhiCAr7xk

SDL Input Handling. (n.d.). Retrieved from https://www.libsdl.org/release/SDL-1.2.15/docs/html/guideinputkeyboard.html