**BỘ GIÁO DỤC VÀ ĐÀO TẠO  
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ SÀI GÒN  
KHOA CÔNG NGHỆ THÔNG TIN  
🙡⯎🙣**



**BÁO CÁO ĐỒ ÁN TIN HỌC**

**ĐỀ TÀI XÂY DỰNG CHƯƠNG TRÌNH QUẢN LÝ PHÒNG GAME**

**GVHD:** **HOÀNG KHUÊ**

**SVTH:** **TRẦN VĂN QUỐC THẮNG – DH52007101**

**VÕ XUÂN LAM – DH52006929**

**ĐẶNG XUÂN SANG – DH52006904**

**TRẦN NGUYỄN THANH SANG – DH52007102**

**ĐỖ TRẦN HẢI SƠN – DH52007024**

**Tp. Hồ Chí Minh, 12/2022**

**BỘ GIÁO DỤC VÀ ĐÀO TẠO  
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ SÀI GÒN  
KHOA CÔNG NGHỆ THÔNG TIN  
🙡⯎🙣**



**BÁO CÁO ĐỒ ÁN TIN HỌC**

**ĐỀ TÀI XÂY DỰNG CHƯƠNG TRÌNH QUẢN LÝ PHÒNG GAME**

**GVHD:** **HOÀNG KHUÊ**

**SVTH:** **TRẦN VĂN QUỐC THẮNG – DH52007101**

**VÕ XUÂN LAM – DH52006929**

**ĐẶNG XUÂN SANG – DH52006904**

**TRẦN NGUYỄN THANH SANG – DH52007102**

**ĐỖ TRẦN HẢI SƠN – DH52007024**

**Tp. Hồ Chí Minh, 1/2023**

**NHẬN XÉT CỦA GIẢNG VIÊN HƯỚNG DẪN**

**Tp. Hồ Chí Minh, ngày … tháng … năm …  
Giảng viên hướng dẫn**

**Hoàng Khuê**

**LỜI CẢM ƠN**

Lời đầu tiên chúng em xin gửi lời cảm ơn chân thành đến thầy HOÀNG KHUÊ thuộc khoa Công nghệ thông tin – Trường Đại học Công Nghệ Sài Gòn, người đã hướng dẫn tận tình và giúp đỡ, bỏ thời gian quý báu cho chúng em để hoàn thành đồ án tin học này.

Trong quá trình học tập và tìm hiểu chúng em đã nỗ lực rất nhiều với mong muốn hoàn thành đồ án tin học một cách tốt nhất, nhưng những thiếu sót sẽ không thể tránh khỏi. Chúng em mong thầy có thể thông cảm và cho chúng em những ý kiến, đóng góp để chỉ dạy chúng em có thể hoàn thành đồ án tin học của mình một cách trọn vẹn nhất.

Sau cùng, chúng em xin kính chúc Thầy lời chúc sức khoẻ, luôn hạnh phúc và thành công hơn nữa trong công việc cũng như trong cuộc sống.

Chúng em xin chân thành cảm ơn!

**MỤC LỤC**

[CHƯƠNG 1: MỞ ĐẦU 1](#_Toc123765173)

[1.1. Tổng quan bài toán cần giải quyết 1](#_Toc123765174)

[1.2. Khảo sát nghiệp vụ 1](#_Toc123765175)

[1.3. Phạm vi đề tài 2](#_Toc123765176)

[CHƯƠNG 2: THUẬT TOÁN VÀ CÔNG NGHỆ ÁP DỤNG ĐỂ GIẢI QUYẾT BÀI TOÁN 4](#_Toc123765177)

[2.1. Danh sách liên kết 4](#_Toc123765178)

[2.1.1. Định nghĩa, tổng quan về danh sách liên kết đơn 4](#_Toc123765179)

[2.1.2. Phân loại danh sách liên kết 4](#_Toc123765180)

[2.1.3. Danh sách liên kết đơn 4](#_Toc123765181)

[2.2. Cây nhị phân 10](#_Toc123765182)

[2.2.1. Biểu diễn cây nhị phân 10](#_Toc123765183)

[2.2.2. Các thao tác trên cây nhị phân 11](#_Toc123765184)

[2.2.3. Ưu/Nhược điểm của cây nhị phân 15](#_Toc123765185)

[2.3. Thuật toán sắp xếp Quick Sort 15](#_Toc123765186)

[2.4.1. Giới thiệu thuật toán Quick Sort 15](#_Toc123765187)

[2.4.2. Ý tưởng của thuật toán Quick Sort 15](#_Toc123765188)

[2.4.3. Cài đặt thuật toán Quick Sort 16](#_Toc123765189)

[2.4.4. Ưu/Nhược điểm của thuật toán Quick Sort 18](#_Toc123765190)

[2.4. Thuật toán sắp xếp Merge Sort 18](#_Toc123765191)

[2.4.1. Giới thiệu thuật toán Merge Sort 18](#_Toc123765192)

[2.4.2. Ý tưởng của thuật toán Merge Sort 18](#_Toc123765193)

[2.4.3. Cài đặt thuật toán Merge Sort 20](#_Toc123765194)

[2.4.4. Ưu/Nhược điểm của thuật toán Merge Sort 21](#_Toc123765195)

[2.5. Ngôn ngữ lập trình C++ 22](#_Toc123765196)

[2.5.1. Giới thiệu 22](#_Toc123765197)

[2.5.2. Các đặc điểm của ngôn ngữ lập trình C++ 22](#_Toc123765198)

[2.6. Github 23](#_Toc123765199)

[2.6.1. Tổng quan về Github 23](#_Toc123765200)

[2.6.2. Khái niệm 23](#_Toc123765201)

[2.6.3. Tính năng 23](#_Toc123765202)

[2.6.4. Lợi ích Github mang lại cho lập trình viên 24](#_Toc123765203)

[2.6.5. Các thuật ngữ cơ bản của Github 24](#_Toc123765204)

[CHƯƠNG 3: TRÌNH BÀY HỆ THỐNG 26](#_Toc123765205)

[3.3. Phân tích và thiết kế chương trình 26](#_Toc123765206)

[CHƯƠNG 4: THỬ NGHIỆM ỨNG DỤNG 27](#_Toc123765207)

[4.1 Giao diện đăng nhập 27](#_Toc123765208)

[4.2. Giao diện sau khi đăng nhập 28](#_Toc123765209)

[4.3. Giao diện quản lí máy tính 29](#_Toc123765210)

[4.3.1. Giao diện thêm máy tính 29](#_Toc123765211)

[4.3.2. Giao diện xóa máy tính 30](#_Toc123765212)

[4.4. Giao diện quản lí nhân viên 30](#_Toc123765213)

[4.4.1 Giao diện thêm nhân viên 31](#_Toc123765214)

[4.4.2. Giao diện xóa nhân viên 32](#_Toc123765215)

[4.4.3. Giao diện xem danh sách nhân viên 32](#_Toc123765216)

[4.5. Giao diện xem tình trạng máy 33](#_Toc123765217)

[4.5.1. Giao diện xem máy trống 34](#_Toc123765218)

[4.5.2. Giao diện xem máy đầy 34](#_Toc123765219)

[4.5.3. Giao diện xem tất cả máy 35](#_Toc123765220)

[4.6. Giao diện xem lịch sử thanh toán 35](#_Toc123765221)

[4.7. Giao diện xem lịch sử đăng nhập 36](#_Toc123765222)

[4.8. Giao diện kiểm tra doanh thu 36](#_Toc123765223)

[4.8.1. Giao diện kiểm tra doanh thu 7 ngày gần nhất 37](#_Toc123765224)

[4.8.2. Giao diện kiểm tra doanh thu ngày tùy chỉnh 37](#_Toc123765225)

[4.9. Giao diện mở máy trực tiếp 38](#_Toc123765226)

[4.10. Giao diện thanh toán 39](#_Toc123765227)

[4.11. Giao diện thiết lập giá tiền 39](#_Toc123765228)

[4.12. Giao diện thông tin giới thiệu 40](#_Toc123765229)

[CHƯƠNG 5: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN 41](#_Toc123765230)

[5.1. Kết luận 41](#_Toc123765231)

[5.2. Hướng phát triển 41](#_Toc123765232)

[TÀI LIỆU THAM KHẢO 42](#_Toc123765233)

# MỞ ĐẦU

## 1.1. Tổng quan bài toán cần giải quyết

*Câu hỏi được đặt ra là:“****Tại sao phải cần tự động hóa cho công việc quản lý phòng game ?****”*

Yêu cầu của bài toán là tạo ra chương trình có thể thực hiện các thao tác quản lý phòng game một cách dễ dàng, tiện lợi dựa trên sự trợ giúp của máy tính. Mọi công việc phải được thao tác trên một vùng dữ liệu chung để đảm bảo việc đồng bộ với nhau trong khâu quản lý. Chính vì thế chương trình quản lý phòng game là một giải pháp tối ưu nhằm:

* Tiết kiệm chi phí quản lý.
* Tiết kiệm thời gian tối đa trong quy trình quản lý.
* Quản lý phòng game một cách hiệu quả, khoa học.
* Thống kê doanh thu chính xác.
* Tạo ra sự tiện ích cho chủ phòng game trong việc quản lý số lượng, loại máy.

## 1.2. Khảo sát nghiệp vụ

* Quản lý số lượng, loại máy:
* Người quản lý sẽ ghi số lượng máy và loại máy vào sổ để quản lý hoặc tự ghi nhớ.
* Quản lý nhân viên:
* Người quản lý sẽ ghi thông tin nhân viên vào sổ để quản lý hoặc tự ghi nhớ.
* Xem tình trạng máy:
* Người quản lý sẽ quan sát bằng mắt tổng quát phòng game của mình xem có máy tính nào đang trống hay đã được sử dụng hết.
* Mở máy cho người thuê máy:
* Khách hàng vào nhận máy do quản lý hoặc nhân viên chỉ định. Nếu máy trống khách hàng có thể tự chọn.
* Người quản lý hoặc nhân viên ghi thời gian lên phiếu thuê máy đưa cho khách thuê và tiến hành mở máy cho khách.
* Người quản lý ghi ngày giờ khách thuê bắt đầu sử dụng máy vào sổ.
* Tính tiền người thuê máy:
* Khách hàng trả máy mang phiếu thuê máy đưa cho quản lý hoặc nhân viên khi không muốn sử dụng máy nữa.
* Người quản lý hoặc nhân viên sẽ kiểm tra trong sổ và tính tiền.
* Người quản lý hoặc nhân viên sẽ ghi lại số tiền đã thanh toán vào sổ.
* Thống kê doanh thu:
* Người quản lý kiểm tra trong sổ và tự tính tổng doanh thu thủ công.
* Ghi lại thời gian vào ca, ra ca của nhân viên.
* Nhân viên khi vào ca, ra ca phải ghi thời gian vào sổ sau đó ký tên.

## 1.3. Phạm vi đề tài

Dựa trên những khảo sát nghiệp vụ trên, nhóm chúng em sẽ tiến hành xác định tự động hóa những phần như sau:

* Quản lý số lượng, loại máy tính:
* Người quản lý thao tác sử dụng hệ thống để thêm hoặc xóa thông tin máy tính ra khỏi danh sách máy tính một cách nhanh chóng.
* Quản lý số lượng, thông tin nhân viên:
* Người quản lý thao tác sử dụng hệ thống để thêm hoặc xóa thông tin nhân viên ra khỏi danh sách nhân viên.
* Hệ thống hiển thị danh sách nhân viên một cách nhanh chóng.
* Xem tình trạng máy:
* Người quản lý hoặc nhân viên thao tác sử dụng hệ thống để xem tình trạng máy hiện tại của phòng game.
* Hệ thống hiển thị các máy đang trống và đang được sử dụng một cách nhanh chóng.
* Tính tiền người thuê máy:
* Khách hàng chỉ cần đọc số máy muốn thanh toán, người quản lý hoặc nhân viên sẽ thao tác sử dụng hệ thống để tính tiền cho khách hàng.
* Thông tin số máy, loại máy và thời gian sử dụng sẽ được hiển thị trong hóa đơn thanh toán.
* Thông tin thanh toán sẽ tự động lưu trữ trong hệ thống.
* Thống kê doanh thu:
* Người quản lý thao tác sử dụng hệ thống để có thể xem thống kê doanh thu theo ngày hoặc theo tuần một cách nhanh chóng.
* Ghi lại thời gian vào ca, ra ca của nhân viên:
* Người quản lý cấp tài khoản, mật khẩu cho nhân viên bằng cách tạo trên hệ thống.
* Khi nhân viên sử dụng tài khoản, mật khẩu đã được cấp đăng nhập hoặc đăng xuất khỏi ca làm việc thì sẽ tự động lưu trữ thời gian trong hệ thống. Người quản lý có thể xem lịch sử thời gian vào ca, ra ca một cách nhanh chóng.

# THUẬT TOÁN VÀ CÔNG NGHỆ ÁP DỤNG ĐỂ GIẢI QUYẾT BÀI TOÁN

## 2.1. Danh sách liên kết

### 2.1.1. Định nghĩa, tổng quan về danh sách liên kết đơn

Danh sách liên kết là danh sách mà các phần tử (Node) liên kết với nhau nhờ vào vùng liên kết của chúng. Mỗi node bao gồm 2 phần: phần Data dùng để chứa dữ liệu cần xử lý và phần liên kết dùng để liên kết tới các node khác.

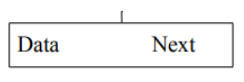
### 2.1.2. Phân loại danh sách liên kết

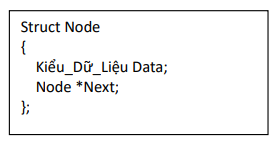
Tùy cách liên kết giữa các phần tử, danh sách liên kết gồm có nhiều loại khác nhau:

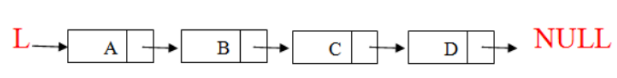
* Danh sách liên kết đơn: mỗi phần tử liên kết với phần tử đứng sau nó trong danh sách.
* Danh sách liên kết đôi/kép: mỗi phần tử liên kết với các phần tử đứng trước và sau nó trong danh sách.
* Danh sách liên kết vòng: phần tử cuối danh sách liên kết với phần tử đầu danh sách.

### 2.1.3. Danh sách liên kết đơn

#### 2.1.3.1. Khái niệm

Danh sách liên kết đơn là một tập hợp các Node được phân bố động, được sắp xếp theo cách sao cho mỗi Node chứa một giá trị (Data) và một con trỏ (Next). Con trỏ sẽ trỏ đến phần tử kế tiếp của danh sách liên kết đó. Nếu con trỏ mà trỏ đến NULL, nghĩa là đó là phần tử cuối cùng của link list. Các phần tử trong danh sách không thể truy cập trực tiếp, muốn truy cập bất kỳ phần tử nào đó trong danh sách phải xuất phát từ phần tử đầu tiên. Mỗi Node trong danh sách có thể được biểu diễn như sau:

Cú pháp:

Có thể hình dung danh sách liên kết qua hình vẽ như sau:

#### 2.1.3.2. Thao tác trên 1 Node

* Khai báo biến con trỏ p là biến con trỏ kiểu Node: Node \*p
* Cấp phát bộ nhớ cho Node mới – được trỏ bởi con trỏ p:
* Node \*p = new Node;
* Truy cập vào trường info: p->info
* Truy cập vào trường next: p->next
* Xóa Node được trỏ bởi p: delete p

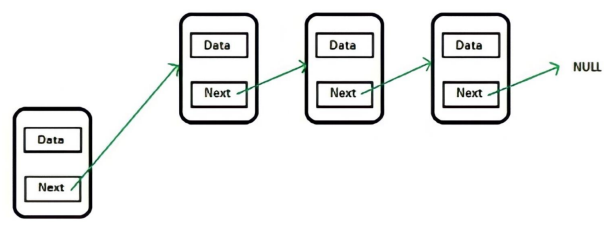
#### 2.1.3.3. Các thao tác trên danh sách liên kết đơn

Thao tác trên danh sách liên kết khác với thao tác trên mảng khi cần thêm Node vào danh sách liên kết thì ta xin cấp phát bộ nhớ cho Node và nối Node đó vào dánh sách liên kết và khi không cần sử dụng thì ta giải phóng nó khỏi danh sách. Truy cập trên danh sách cũng khác so với trên mảng. Nếu ở mảng ta truy cập trực tiếp thông qua chỉ số mảng (thông qua địa chỉ ô nhớ) thì ở danh sách liên kết đơn ta phải truy cập một cách tuần tự. Khi làm việc trên danh sách liên kết đơn cũng tương tự như vậy nên ta cần chú ý những điểm sau:

* Danh sách luôn có con trỏ đầu danh sách.
* Danh sách luôn có giá trị báo kết thúc danh sách.
* Trường Next của mỗi Node chỉ chứa địa chỉ của Node sau đó, trừ Node cuối.
* Không tách danh sách thành 2 danh sách con nếu danh sách phần sau chưa có con trỏ tới.
* Khởi tạo danh sách rỗng.
* **Thêm một phần tử (node) vào danh sách:**
* Trường hợp 1: Thêm phần tử vào đầu danh sách:
* Bước 1: Cho con trỏ của node cần thêm (node p) liên kết với node đầu danh sách pHead.

p->pNext = l.pHead;

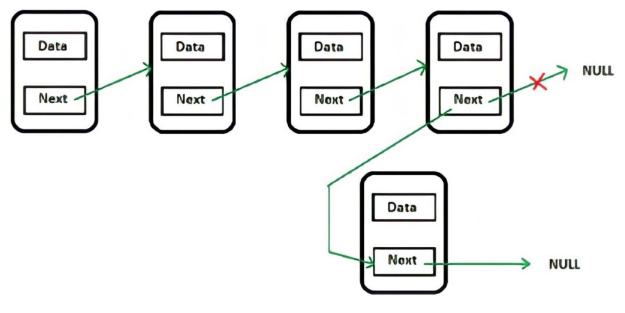
* Bước 2: Cập nhập lại node đầu pHead là node vừa thêm (node p).

l.pHead = p;

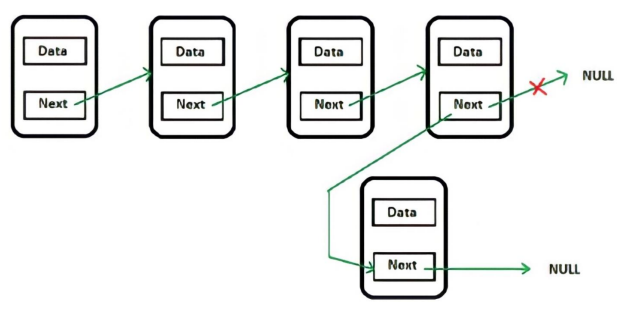
* Trường hợp 2: Thêm phần tử vào cuối danh sách:
* Bước 1: Cho con trỏ của node cuối danh sách pTail liên kết với node cần thêm (node p).

l.pTail->pNext = p;

* Bước 2: Cập nhật lại node cuối pTail chính là node mới thêm (node p).

l.pTail = p;

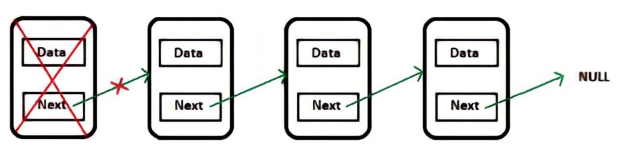
* Trường hợp 3: Thêm phần tử vào vị trí bất kì:



* **Xóa một phần tử ra khỏi danh sách:**
* Trường hợp 1: Xóa phần tử (p) đầu danh sách:
* Bước 1: Cho con trỏ node đầu danh sách (pHead) trỏ đến node sau nó (pNext).

l.pHead = l.pHead->pNext;

* Bước 2: Xóa đi node node (p) nằm ở đầu danh sách.

delete p;

* Trường hợp 2: Xóa phần tử cuối danh sách:
* Bước 1 : Nếu danh sách có một phần tử thực hiện hàm xóa đầu (Trường hợp 1).
* Bước 2 : Nếu danh sách có nhiều phần tử ta thực hiện duyệt danh sách từ đầu đến kế cuối.

for(NODE \*k = l.pHead; k != NULL; k = k->pNext)

* Bước 3 : Phát hiện phần tử cần xóa ở vị trí kế cuối.

if(k->pNext == l.pTail)

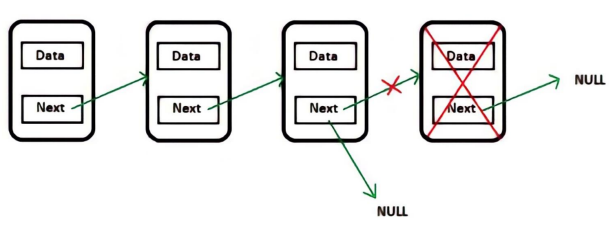
* Bước 4 : Xóa đi phần tử cuối.

delete l.pTail;

* Bước 5 : Cho con trỏ kế cuối trỏ đến NULL.

k->pNext = NULL;

* Bước 6 : Cập nhập lại pTail.

l.pTail = k;

* Trường hợp 3 : Xóa phần tử ở vị trí bất kì trong danh sách :
* Bước 1 : Nếu node cần xóa nằm đầu danh sách thì xóa đầu (Trường hợp 1).
* Bước 2 : Nếu node cần xóa nằm cuối danh sách thì xóa cuối (Trường hợp 2).
* Bước 3 : Nếu node cần xóa không nằm ở đầu hoặc cuối danh sách:
* Duyệt danh sách liên kết đơn.
* Phát hiện node cần xóa ta cập nhập mối liên kết của node trước node cần xóa với node sau node cần xóa.
* Diagram

  Description automatically generatedXóa phần tử cần xóa.
* Tìm kiếm phần tử trong danh sách :
* Bước 1: Cho con trỏ p trỏ vào đầu danh sách liên kết (l) và

khởi tạo 1 biến int found = 0

Node \*p = l.pHead;

int found = 0;

* Bước 2: Thực hiện duyệt các phần tử từ đầu đến cuối danh sách kiểm tra nếu có data = X thì trả về địa chỉ của Node đó, ngược lại thì cho con trỏ p trỏ đến vị trí phần tử kế tiếp

while ((p != NULL)&&(p->data != x)) {  
 p = p->pNext;

}

return p;

#### 2.1.3.4. Ưu/Nhược điểm của danh sách liên kết đơn

* Ưu điểm:
* Danh sách liên kết có kích thước động, có thể mở rộng hay thu hẹp rất dễ ràng.
* Việc chèn hay xóa một phần tử trong danh sách liên kết là rất dễ dàng, ta chỉ cần thay đổi vị trí trỏ của các con trỏ thay vì phải dịch toàn bộ phần dữ liệu còn lại
* Nhược điểm:
* Danh sách liên kết mang yếu điểm của cấp phát động về hiệu năng. Điều này cũng dễ hiểu vì mảng có vị trí bộ đệm tốt hơn (là các khối các ô nhớ liên tiếp) so với danh sách liên kết.
* Ta không thể truy cập ngẫu nhiên một phần tử trong danh sách liên kết ngay lập tức, mà phải thực hiện duyệt từ đầu danh sách cho tới khi gặp phần tử đó. Do vậy ta không thể thực hiện tìm kiếm nhị phân với các danh sách liên kết.
* Cần thêm không gian bộ nhớ cho con trỏ trong mỗi phần tử của danh sách liên kết.

**Kết luận:**

* Với ưu điểm như vậy của danh sách liên kết đơn nên nhóm chúng em chọn để xử lý công việc quản lý máy tính.

## 2.2. Cây nhị phân

### 2.2.1. Biểu diễn cây nhị phân

Cây nhị phân là một cấu trúc dữ liệu bao gồm các nút được kết nối với nhau theo quan hệ cha con với mỗi nút cha có tối đa 2 nút con. Mỗi nút của cây nhị phân sẽ lưu các thông tin sau:

* + Giá trị lưu tại nút đó. Giá trị này có thể là bất kỳ kiểu dữ liệu nào. Ví dụ, cây nhị phân lưu trữ các số nguyên thì kiểu dữ liệu là int.
  + Địa chỉ nút gốc của cây con bên trái.
  + Địa chỉ nút gốc của cây con bên phải.

Chúng ta có thể sử dụng danh sách liên kết để biểu diễn cây nhị phân. Đầu tiên, cần định nghĩa một nút trong cấu trúc dữ liệu dạng cây.

struct tNode{

int data;

tNode \*pLeft, \*pRight;

Diagram

Description automatically generated};

Để lưu trữ cây, chúng ta chỉ cần xác định nút gốc của cây.

tNode \*root;

### 2.2.2. Các thao tác trên cây nhị phân

Có nhiều thao tác trên cây nhị phân như:

* Chèn một nút mới vào cây nhị phân
* Xóa một nút ra khỏi cây nhị phân
* Duyệt cây nhị phân
* Tìm kiếm trên cây nhị phân
* …
* **Chèn một nút mới vào cây nhị phân để làm con của nút p:**

void insertNode(tNode \*p, int value){

tNode \*node = newNode(value);

if (p->pLeft == NULL){

p->pLeft = node;

}else if (p->pRight == NULL) {

p->pRight = node;

}else{

node->pLeft = p->pLeft;

p->pLeft = node;

}

}

* **Chèn nút mới vào vị trí gốc của cây:**

void insertNewRoot(int value){

tNode \*node = newNode(value);

node->pLeft = root;

root = node;

}

* **Duyệt cây nhị phân:**

Có 3 kiểu duyệt chính có thể áp dụng trên cây nhị phân:

* **Duyệt theo thứ tự trước NodeLeftRight (NLR):**

Xử lý nút đang duyệt trước, sau đó mới duyệt đến các cây con bên trái và bên phải.

void NLR(tNode \*root){

if(root!=NULL){

if(root!=NULL){

cout<<root->data<<" ";

}

NLR(root->pLeft);

NLR(root->pRight);

}

}

* **Duyệt theo thứ tự giữa LeftNodeRight (LNR):**

Xử lý cây con bên trái trước, rồi mới xử lý nút đang duyệt, cuối cùng xử lý cây con bên phải.

void LNR(tNode \*root){

if(root!=NULL){

LNR(root->pLeft);

if(root!=NULL){

cout<<root->data<<" ";

}

LNR(root->pRight);

}

}

* **Duyệt theo thứ tự sau LeftRightNode (LRN):**

Xử lý cây con bên trái trước, rồi xử lý cây con bên phải, cuối cùng mới xử lý nút đang duyệt.

void LRN(tNode \*root){

if(root!=NULL){

LRN(root->pLeft);

LRN(root->pRight);

if(root!=NULL){

cout<<root->data<<" ";

}

}

}

### 2.2.3. Ưu/Nhược điểm của cây nhị phân

* Ưu điểm:
* Phân cấp dữ liệu.
* Tìm kiếm dễ dàng hơn.
* Thao tác trên các danh sách dữ liệu đã sắp xếp.
* Thao tác hủy hoặc chèn rất nhanh.
* Nhược điểm:
* Tốn thêm bộ nhớ để lưu trữ địa chỉ nút con trái, con phải.

**Kết luận:**

* Với ưu điểm như vậy của cây nhị phân nên nhóm chúng em chọn để xử lý công việc quản lý nhân viên.

## 2.3. Thuật toán sắp xếp Quick Sort

### 2.4.1. Giới thiệu thuật toán Quick Sort

Thuật toán sắp xếp nhanh hay còn gọi là QuickSort Algorithm là một trong 6 thuật toán sắp xếp thông dụng nhất của khoa học máy tính. Thuật toán sử dụng tư tưởng chia để trị nên còn được ví là part sort và thuộc loại sắp xếp phức tạp. Từ dãy ban đầu, người ta sẽ chia dãy thành hai phần nhỏ theo một quy tắc xác định, từ đó giúp cải thiện tốc độ của việc săp xếp.

### 2.4.2. Ý tưởng của thuật toán Quick Sort

Quicksort là thuật toán ứng dụng ý tưởng chia nhỏ để trị. Đầu tiên nó chia mảng đầu vào thành hai mảng con một nửa nhỏ hơn, một nửa lớn hơn dựa vào một phần tử trung gian. Sau đó, nó sắp xếp đệ quy các mảng con để giải quyết bài toán.

Mấu chốt của thuật toán ở việc chia mảng con hay còn gọi là thuật toán phân đoạn. Cách giải quyết có thể tóm tắt như sau:

* Chọn một phần tử trong mảng làm phần tử trung gian để chia đôi mảng gọi là pivot. Thông thường ta thường chọn phần tử đầu tiên, phần tử ở giữa mảng hoặc phần tử cuối cùng của mảng để làm pivot.
* Phân đoạn: di chuyển phần tử có giá trị nhỏ hơn pivot về một bên, tất cả các phần tử có giá trị lớn hơn pivot xếp về một bên (các giá trị bằng pivot có thể đi theo một trong hai cách).
* Sau bước phân đoạn di chuyển pivot về đúng vị trí giữa của 2 mảng con
* Áp dụng đệ quy các bước phân đoạn trên cho hai mảng con để thực hiện sắp xếp.

Trường hợp dừng của đệ quy là khi các mảng con có kích thước bằng 0 hoặc 1, khi đó nó đã được sắp xếp theo định nghĩa, vì vậy chúng không bao giờ cần phải sắp xếp.

### 2.4.3. Cài đặt thuật toán Quick Sort

Như đã nói ở phần trên, việc phân đoạn chính là việc quan trọng nhất. Bạn có thể xây dựng thuật toán phân đoạn riêng hoặc có thể viết chung luôn với quickSort trong cùng một hàm đều được. Để dễ hiểu mình sẽ viết riêng thành hai hàm riêng biệt.

Có 3 cách chọn pivot (phần tử làm chốt), mục đích là để chọn được phần tử có giá trị trung gian để chia làm hai vế cho danh sách:

* Chọn phần tử đầu mảng
* Chọn phần tử ở giữa mảng
* Chọn phần tử cuối mảng
* Hàm phân đoạn:
* Có 3 tham số truyền vào: mảng a, low, high. low và hight lần lượt là chỉ số đầu và chỉ số cuối
* Chia mảng thành 2 phần: bên trái nhỏ hơn pivot, bên phải lớn hơn pivot
* Chọn pivot là phần tử cuối cùng: pivot = a[high];
* lef chỉ sổ đầu của mảng: left = low;
* right chỉ số cuối của mảng trừ pivot: right = high – 1;

Thực hiện vòng lặp trong khi left < right như sau:

* Trong khi left <= right và a[left] < pivot: left++;

(tìm tới vị trị a[left] > pivot)

* Trong khi right >= left và a[right] > pivot: right –;

(tìm tới vị tri a[right])

* Khi tìm được vị trí của left, right thích hợp, tức là ( a[left] > pivot và a[right] < pivot):

swap(a[left], a[right]);

Cuối cùng là chuyển pivot về giữa mảng: swap(a[left], a[high]);

int partition(int a[], int low, int high) {

int pivot = a[high];

int right = high - 1, left = low;

while (true) {

while (left <= right && a[left] < pivot) left++;

while (right >= left && a[right] > pivot) right--;

if (left >= right)

break;

swap(a[left], a[right]);

left++;

right--;

}

swap(a[left], a[high]);

return left;

}

* Hàm Quick Sort:
* Chia đôi mảng
* Đệ quy bên trái
* Đệ quy bên phải

void quickSort(int a[], int low, int high) {

if (low < high) // Mảng có nhiều hơn 0 phần tử

{

int pivot = partition(a, low, high); // Chia đôi mảng

quickSort(a, low, pivot - 1); // Đệ quy bên trái

quickSort(a, pivot + 1, high); // Đệ quy bên phải

}

}

### 2.4.4. Ưu/Nhược điểm của thuật toán Quick Sort

* Ưu điểm:
* Thuật toán có độ phức tạp nhỏ hơn các thuật toán sắp xếp đơn giản, tốc độ xử lý tương đối nhanh. Trong một số trường hợp, quicksort là thuật toán có tốc độ tốt nhất.
* Thông dụng, được áp dụng nhiều trong lập trình, trong thư viện của các ngôn ngữ lập trình như C++, Java, C# . . .
* Có thể ứng dụng vào xử lý dữ liệu lớn.
* Nhược điểm:
* Thuật toán không có tính ổn định, không có tính thích ứng, dễ ảnh hưởng bởi dữ liệu đầu vào.
* Tốn không gian bộ nhớ hơn so với các thuật toán sắp xếp đơn giản.
* Tư tưởng và giải thuật khá phức tạp.
* Khó khăn trong việc lựa chọn phần tử làm chốt trong phân hoạch. Việc lựa chọn có thể ảnh hưởng rất nhiều đến hiệu suất của thuật toán tuy nhiên ta không thế đưa ra lựa chọn tối ưu nhất.

**Kết luận:**

* Với ưu điểm như vậy của thuật toán Quick Sort nên nhóm chúng em chọn để xử lý công việc sắp xếp danh sách khách thuê máy tính.

## 2.4. Thuật toán sắp xếp Merge Sort

### 2.4.1. Giới thiệu thuật toán Merge Sort

Thuật toán sắp xếp merge sort là một trong những thuật toán có độ phức tạp ở mức trung bình và cùng sử dùng phương pháp chia để trị giống thuật toán sắp xếp nhanh quick sort.

### 2.4.2. Ý tưởng của thuật toán Merge Sort

Giống như Quick sort, Merge sort là một thuật toán chia để trị. Thuật toán này chia mảng cần sắp xếp thành 2 nửa. Tiếp tục lặp lại việc này ở các nửa mảng đã chia. Sau cùng gộp các nửa đó thành mảng đã sắp xếp. Hàm merge() được sử dụng để gộp hai nửa mảng. Hàm merge(arr, l, m, r) là tiến trình quan trọng nhất sẽ gộp hai nửa mảng thành 1 mảng sắp xếp, các nửa mảng là arr[l…m] và arr[m+1…r] sau khi gộp sẽ thành một mảng duy nhất đã sắp xếp.

Hãy xem ý tưởng triển khai code dưới đây để hiểu hơn:

mergeSort(arr[], l, r)

If r > l

1. Tìm chỉ số nằm giữa mảng để chia mảng thành 2 nửa:

middle m = (l+r)/2

2. Gọi đệ quy hàm mergeSort cho nửa đầu tiên:

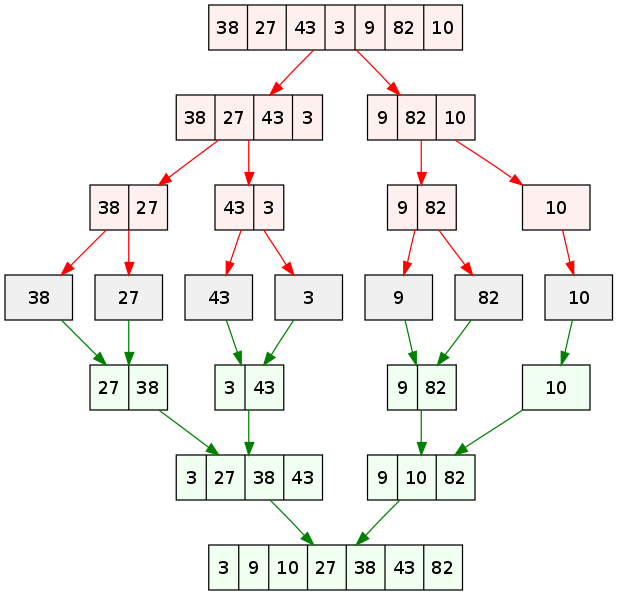
mergeSort(arr, l, m)

3. Gọi đệ quy hàm mergeSort cho nửa thứ hai:

mergeSort(arr, m+1, r)

4. Gộp 2 nửa mảng đã sắp xếp ở (2) và (3):

merge(arr, l, m, r)

Hình ảnh dưới đây từ wikipedia sẽ hiển thị cho bạn toàn bộ sơ đồ tiến trình của thuật toán merge sort cho mảng {38, 27, 43, 3, 9, 82, 10}. Nếu nhìn kỹ hơn vào sơ đồ này, chúng ta có thể thấy mảng ban đầu được lặp lại hành động chia cho tới khi kích thước các mảng sau chia là 1. Khi kích thước các mảng con là 1, tiến trình gộp sẽ bắt đầu thực hiện gộp lại các mảng này cho tới khi hoàn thành và chỉ còn một mảng đã sắp xếp.

### 2.4.3. Cài đặt thuật toán Merge Sort

Với giải thuật nêu trên, ta cần thiết kế hai hàm, một hàm dùng để chia nhỏ danh sách làm thân hàm Merge sort, một hàm dùng để trộn hai danh sách con lại với nhau.

Trộn hai dãy con hay còn gọi là hòa nhập hai dãy con là trái tim quan trọng nhất của giải thuật. Có lẽ đây là lý do mà người ta gọi nó là sắp xếp trộn.

* Hàm trộn:
* Hàm trộn sẽ ghép hai dãy con lại thành một dãy có thứ tự, dãy con nhỏ nhất ta cần ghép sẽ có 1 phần tử. Thực chất đó là ghép hai mảng liền nhau đã sắp xếp thành một mảng có thứ tự.
* Ta sẽ viết hàm trộn trực tiếp trên dãy ban đầu và khai báo thêm 2 mảng trung gian (left, right) để lưu và trộn. Do đó sẽ là trộn hai dãy con liền kề nhau.

void merge(int a[], int start, int mid, int end) {

int n1 = mid - start + 1;

int n2 = end - mid;

int left[n1];

int right[n2];

for (int x = 0; x < n1; x++) left[x] = a[start + x];

for (int y = 0; y < n2; y++) right[y] = a[mid + 1 + y];

int i = 0, j = 0;

int k = start;

while (i < n1 && j < n2) {

if (left[i] >= right[j]) {

a[k] = right[j];

j++;

} else {

a[k] = left[i];

i++;

}

k++;

}

while (j < n2) {

a[k] = right[j];

k++;

j++;

}

while (i < n1) {

a[k] = left[i];

k++;

i++;

}

}

* Hàm Merge Sort:
* Chia nhỏ mảng ban đầu thành 2 mảng con.
* Thực hiện đệ quy mergeSort hai mảng con đó.
* Cuối cùng thì trộn hai mảng con lại bằng hàm merge đã viết ở trên.

void mergeSort(int a[], int first, int end) {

int t;

if (first < end) {

t = (first + end) / 2;

mergeSort(a, first, t);

mergeSort(a, t + 1, end);

merge(a, first, t, end);

} else

return;

}

### 2.4.4. Ưu/Nhược điểm của thuật toán Merge Sort

* Ưu điểm:
* Độ phức tạp trung bình O(n log n), tốc độ giải quyết khá nhanh.
* Có tính ổn định và thích ứng, tốc độ không bị ảnh hưởng nhiều bởi dữ liệu đầu vào.
* Xử lý khá tốt với dữ liệu lớn đặc biệt là dạng list, file.
* Nhược điểm:
* Tốn nhiều bộ nhớ nếu sử dụng đệ quy.
* Code khó cài đặt, tương đối phức tạp.
* Trong hầu hết các trường hợp, thuật toán này không được đánh giá cao hơn Quick Sort.

**Kết luận:**

* Với ưu điểm như vậy của thuật toán Merge Sort nên nhóm chúng em chọn để xử lý công việc sắp xếp danh sách máy tính.

## 2.5. Ngôn ngữ lập trình C++

### 2.5.1. Giới thiệu

C ++ là một ngôn ngữ lập trình được phát triển bởi Bjarne Stroustrup vào năm 1979 tại Bell Labs. C ++ được coi là ngôn ngữ bậc trung (middle-level) như một phần mở rộng của ngôn ngữ lập trình C, hoặc “C với các lớp Class” vì nó bao gồm sự kết hợp của cả các tính năng của ngôn ngữ cấp cao và cấp thấp.

Ngôn ngữ đã được mở rộng đáng kể theo thời gian và C ++ hiện nay có nhiều tính năng như:

* Lập trình tổng quát.
* Lập trình hướng đối tượng.
* Lập trình thủ tục.
* Ngôn ngữ đa mẫu hình tự do có kiểu tĩnh.
* Dữ liệu trừu tượng.
* Lập trình đa hình.

### 2.5.2. Các đặc điểm của ngôn ngữ lập trình C++

**Tính phổ biến**: C++ là một trong những ngôn ngữ lập trình phổ biến nhất trên thế giới.

**Tính thực thi nhanh:** Nếu bạn đã sành sỏi về C++ thì bạn có thể lập trình rất nhanh. Một trong những mục tiêu của C++ chính là khả năng thực thi. Và nếu bạn cần thêm các tính năng cho chương trình, C++ cho phép bạn sử dụng ngôn ngữ Assembly (Hợp ngữ) – Ngôn ngữ lập trình bậc thấp nhất dùng để giao tiếp trực tiếp với phần cứng của máy tính.

**Thư viện đầy đủ:** Có rất nhiều tài nguyên sử dụng cho người lập trình bằng C++, bao gồm cả đồ hoạ API, 2D, 3D, vật lý các thiết bị âm thanh hỗ trợ giúp cho lập trình viên dễ dàng thực thi.

**Đa mô hình:** C++ cũng cho phép bạn lập trình theo cấu trúc tuyến tính, hướng chức năng, hướng đối tượng đa dạng tuỳ theo yêu cầu của người lập trình.

## 2.6. Github

### 2.6.1. Tổng quan về Github

GitHub được viết bằng Ruby on Rails và Erlang do Tom Preston-Werner, Chris Wanstrath, và PJ Hyett phát triển trang web được đưa ra và chạy chính thức vào tháng 4 năm 2008.

Tính đến thời điểm tháng 3 năm 2018 Github đang là dịch vụ máy chủ lưu trữ các mã nguồn lập trình lớn nhất thế giới. Với hơn 25 triệu người dùng và hơn 80 triệu mã nguồn dự án, Github đã trở thành một phần không thể thiêu đối với cộng đồng phát triển mã nguồn mở và cộng đồng lập trình viên trên toàn thế giới.

### 2.6.2. Khái niệm

GitHub là một dịch vụ nổi tiếng cung cấp kho lưu trữ mã nguồn Git cho các dự án phần mềm. Github có đầy đủ những tính năng của Git, ngoài ra nó còn bổ sung những tính năng về social để các developer tương tác với nhau. Git là công cụ giúp quản lý source code tổ chức theo dạng dữ liệu phân tán.

### 2.6.3. Tính năng

Follow User : tính năng này giúp bạn có thể theo dõi những hoạt động của người khác.

Watch Project : đây là tính năng giúp mọi người có thể theo dõi được các hoạt động của nhiều chương trình khác nhau. Nhờ đó họ có thể hiểu xem quá trình để mọi người phát triển các phần mềm như thế nào, hoạt động nhờ project này ra làm sao.Wiki, thống kê, issue… được đặt vào các namespace và coi là các user.

### 2.6.4. Lợi ích Github mang lại cho lập trình viên

**Dễ dàng Markdown:** bạn có thể chỉnh sửa được các cách hiển thị của document, ví dụ như định dạng in nghiêng, in đậm, tạo list, thêm hình…Có thể tracking về sự thay đổi qua nhiều version: nếu bạn có nhiều member cùng thực hiện một dự án, sử dụng Github sẽ giúp bạn lưu lại thay đổi mà bạn đã đẩy lên repository. Có thể nói rằng chức năng này khá giống với Google Drive.

**Quản lý các source code cực kỳ dễ dàng:** nếu bạn muốn tạo ra repo, các nguồn code từ repo đó đều đã được lưu sẵn ở trên Github. Nó giúp bạn có thể coi lại quá trình mà mình đã làm việc thông qua những bình luận sau mỗi lần mà bạn commit. Trên thực tế, nhiều người có thể cùng làm một repo chứ không phải chỉ một người.

**Kết luận:** Với những lợi ích như vậy của Github nên nhóm chúng em chọn để quản lý mã nguồn đồ án và phân công công việc cho các thành viên trong nhóm.

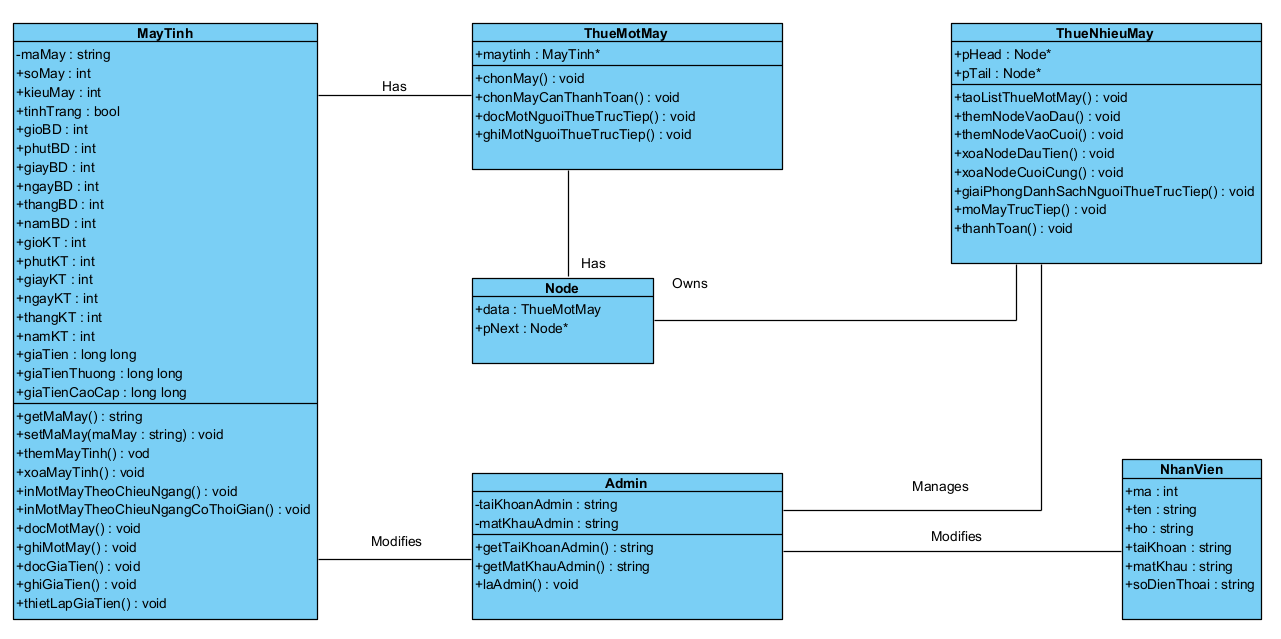
### 2.6.5. Các thuật ngữ cơ bản của Github

* **git init:** tạo một repo trên máy của bạn.
* **git add:** cập nhập lại project.
* **git clone:** sao chép một repo trên server về máy.
* **git commit:** xác nhận và lưu những thay đổi của project.
* **git push:** cập nhập những thay đổi của project và gửi lên server.
* **git checkout:** tạo một nhánh mới và chuyển qua nhánh đó hoặc chuyển nhánh.
* **git remote:** Xem danh sách remote, kết nối repo của bạn đến máy chủ remote, đổi tên remote.
* **git branch:** tạo nhánh mới, xóa nhánh, kiểm tra nhánh hiện tại.
* **git pull:** lấy source mới nhất trên server về và tiến hành trộn.
* **git merge:** để trộn một nhánh vào nhánh đang hoạt động.
* **git log:** hiển thị danh sách cách commit trên một nhánh và các thông tin của nó.
* **git diff:** xem xung đột với tệp cơ sở, với nhánh được trộn trước khi trộn.

# TRÌNH BÀY HỆ THỐNG

## 3.3. Phân tích và thiết kế chương trình

Để chương trình quản lí phòng game được chặt chẽ thì mỗi lớp máy tính, nhân viên, cũng như mỗi máy được thuê có thuộc tính và phương thức như sau:



*Biểu đồ lớp*

# THỬ NGHIỆM ỨNG DỤNG

## 4.1 Giao diện đăng nhập

Cũng như bao ứng dụng, chương trình khác. Đầu tiên chúng ta tiến hành đăng nhập để có quyền truy cập vào hệ thống. Tài khoản và mật khẩu đã được lưu trước đó, chỉ có quản lý và nhân viên mới biết và có quyền truy cập.Graphical user interface, text

Description automatically generated

## 4.2. Giao diện sau khi đăng nhập

Graphical user interface, text

Description automatically generatedĐây là giao diện người quản lý sẽ sử dụng. Có đầy đủ các chức năng hiệu quả để quản lý phòng game.

Text

Description automatically generatedĐây là giao diện nhân viên sẽ sử dụng. Tương tự như của người quản lý nhưng sẽ bị lược bỏ một số chức năng của quản lý.

## 4.3. Giao diện quản lí máy tính

Có 2 tùy chọn là thêm hoặc xóa máy tính.

Text

Description automatically generated

### 4.3.1. Giao diện thêm máy tính

Đây là giao diện thêm máy tính mới. Phải nhập thông tin số máy và kiểu máy cho máy mới.

Treemap chart

Description automatically generated

### 4.3.2. Giao diện xóa máy tính

Tương tự giao diện thêm máy tính.

Treemap chart

Description automatically generated

## 4.4. Giao diện quản lí nhân viên

Có 3 tùy chọn là thêm nhân viên, xóa nhân viên hoặc xem danh sách nhân viên.

Text

Description automatically generated

### 4.4.1 Giao diện thêm nhân viên

Đây là giao diện thêm nhân viên. Phải nhập thông tin họ tên, tài khoản, mật khẩu và số điện thoại cho nhân viên mới.

Graphical user interface, text

Description automatically generated

### 

### 4.4.2. Giao diện xóa nhân viên

Người quản lý bắt buộc nhập mã nhân viên cần xóa để xóa nhân viên đó ra khỏi danh sách nhân viên.

Graphical user interface, text

Description automatically generated

### 4.4.3. Giao diện xem danh sách nhân viên

Chương trình sẽ hiển thị thông tin của từng nhân viên.

## Graphical user interface, text Description automatically generated4.5. Giao diện xem tình trạng máy

Có 3 tùy chọn là tình trạng máy trống, máy đầy hoặc tất cả.

Text

Description automatically generated

### 4.5.1. Giao diện xem máy trống

Chương trình chỉ hiển thị thông tin và trạng thái những máy chưa được sử dụng.

Graphical user interface

Description automatically generated

### 4.5.2. Giao diện xem máy đầy

Chương trình chỉ hiển thị thông tin và trạng thái những máy đang được sử dụng.

Text

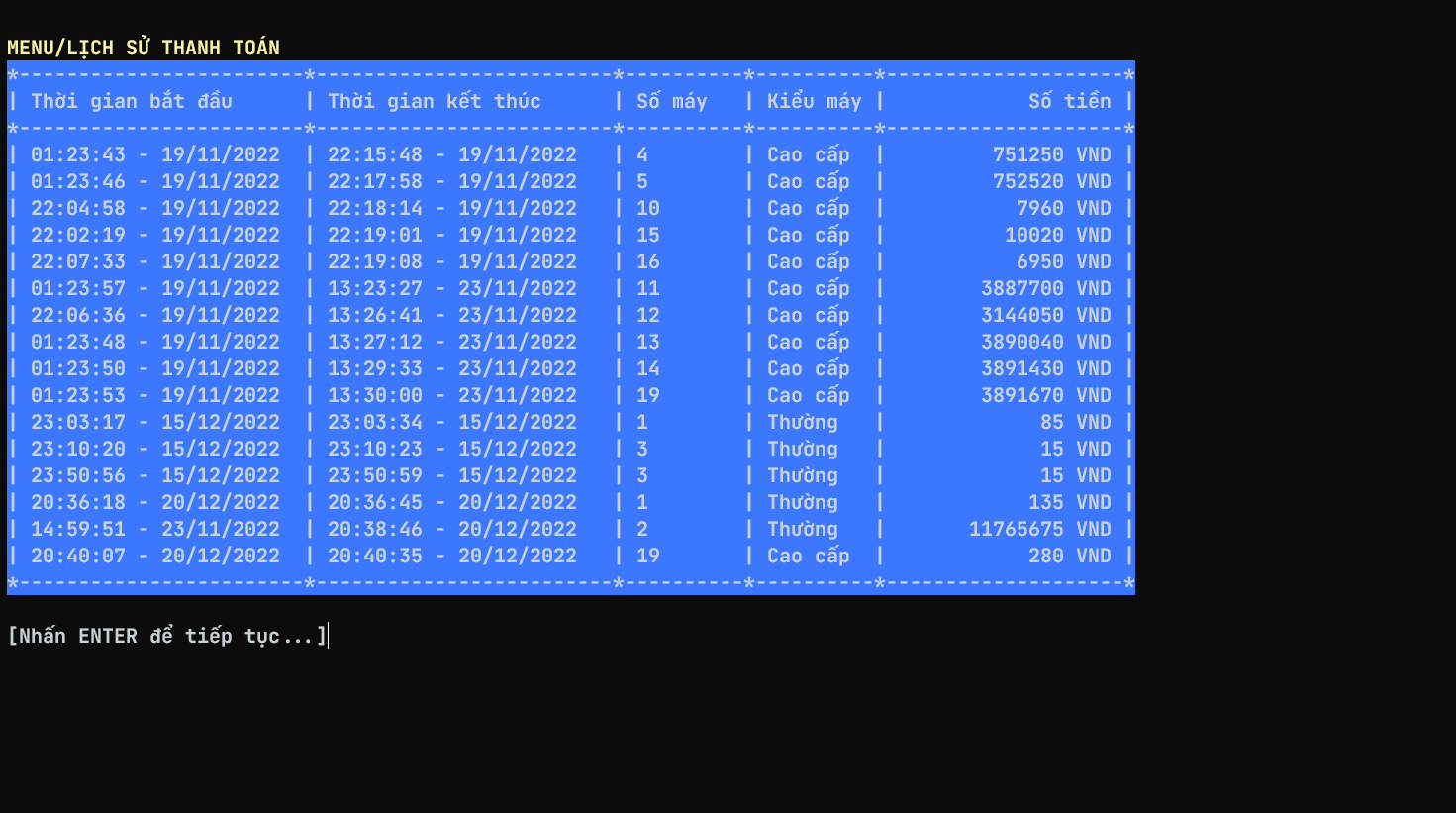
Description automatically generated

### 4.5.3. Giao diện xem tất cả máy

Chương trình sẽ hiển thị thông tin và trạng thái tất cả các máy.

## Treemap chart Description automatically generated with low confidence4.6. Giao diện xem lịch sử thanh toán

Chương trình sẽ hiển thị 2 mốc thời gian bắt đầu và thời gian kết thúc, số máy, kiểu máy và số tiền thanh toán.



## 4.7. Giao diện xem lịch sử đăng nhập

A computer screen capture

Description automatically generated with medium confidenceChương trình sẽ hiển thị mốc thời gian cụ thể, tài khoản thực hiện thao tác và trạng thái đăng nhập hoặc đăng xuất.

## 4.8. Giao diện kiểm tra doanh thu

Có 2 loại là kiểm tra doanh thu 7 ngày gần nhất và doanh thu ngày tùy chỉnh.

Text

Description automatically generated

### 4.8.1. Giao diện kiểm tra doanh thu 7 ngày gần nhất

Chương trình sẽ xuất ra doanh thu từng ngày từ ngày hiện tại lùi lại thêm 6 ngày.

Graphical user interface

Description automatically generated

### 4.8.2. Giao diện kiểm tra doanh thu ngày tùy chỉnh

Người quản lý sẽ nhập ngày/tháng/năm cần xem doanh thu. Sau đó chương trình sẽ xuất ra doanh thu của ngày đó.

Graphical user interface, text, application

Description automatically generated

## 4.9. Giao diện mở máy trực tiếp

Nếu danh sách máy tính trống thì sẽ xuất thông báo danh sách máy tính trống.

Nếu danh sách máy tính đều đang được sử dụng thì sẽ xuất thông báo hết máy.

Người quản lý hoặc nhân viên sẽ nhập số máy cần mở để mở máy.

Graphical user interface

Description automatically generated

## 

## 4.10. Giao diện thanh toán

Người quản lý hoặc nhân viên nhập số máy cần thanh toán, sau đó sẽ chương trình sẽ hiển thị thông tin hóa đơn cần thanh toán.

Text

Description automatically generated

## 4.11. Giao diện thiết lập giá tiền

Graphical user interface, application, Teams

Description automatically generatedNgười quản lý phải bắt buộc nhập giá tiền giá máy mới (thường và cao cấp).

## Graphical user interface, text, application Description automatically generated4.12. Giao diện thông tin giới thiệu

# KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

## 5.1. Kết luận

Chương trình đã được phát triển thành công. Đảm bảo đầy đủ các tính năng cơ bản để quản lí phòng game một cách hiệu quả, thuận lợi.

Chương trình hoạt động tốt trên máy tính, laptop. Hiệu suất nhanh chóng ổn định, giao diện thân thiện, dễ sử dụng, dễ thao tác.

## 5.2. Hướng phát triển

Xây dựng hoàn thiện chương trình, thêm các tính năng như nạp giờ chơi,...

Tìm hiểu kết hợp nhiều ngôn ngữ khác để phát triển chương trình thành phần mềm, ứng dụng.

# TÀI LIỆU THAM KHẢO

[1]. <https://teky.edu.vn/blog/github-la-gi>

[2]. <https://glints.com/vn/blog/lap-trinh-cpp-la-gi/#.Y6INQXZByMo>

[3]. <https://github.com/hugorplobo/colors.hpp>