TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI
PHÂN HIỆU TẠI THÀNH PHỐ HỒ CHÍ MINH

BỘ MÔN CÔNG NGHỆ THÔNG TIN

BÁO CÁO BÀI TẬP LỚN

MÔN: LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

ĐỀ TÀI: QUẢN LÝ TIỆC

Giảng viên hướng dẫn : ThS. Trần Thị Dung
Sinh viên thực hiện : Hoàng Gia Kiệt
Lớp : Công nghệ thông tin
Khoá : 62

TP. Hồ Chí Minh, tháng 11 năm 2022

TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI
PHÂN HIỆU TẠI THÀNH PHỐ HỒ CHÍ MINH

BỘ MÔN CÔNG NGHỆ THÔNG TIN

BÁO CÁO BÀI TẬP LỚN

MÔN: LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

ĐỀ TÀI: QUẢN LÝ TIỆC

Giảng viên hướng dẫn : ThS. Trần Thị Dung
Sinh viên thực hiện : Hoàng Gia Kiệt
Lớp : Công nghệ thông tin
Khoá : 62

TP. Hồ Chí Minh, tháng 11 năm 2022

TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI
PHÂN HIỆU TẠI THÀNH PHỐ HỒ CHÍ MINH

CỘNG HOÀ XÃ HỘI CHỦ NGHĨA VIỆT NAM
Độc lập – Tự do – Hạnh phúc

NHIỆM VỤ THIẾT KẾ BÀI TẬP LỚN

BỘ MÔN: **CÔNG NGHỆ THÔNG TIN**

MÔN: **LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG**

------***------

Mã sinh viên: 6251071049 **Họ tên SV:** Hoàng Gia Kiệt

Khoá: 62 **Lớp:** Công nghệ thông tin

1. Tên đề tài: Quản lý tiệc

2. Mục đích, yêu cầu

a. Mục đích

- Tìm hiểu về các chức năng trong quản lý.

- Tăng cường hiểu biết, ứng dụng linh hoạt ngôn ngữ lập trình C++ vào quản lý cơ bản.

- Hiểu rõ hơn về 4 tính chất quan trọng trong lập trình hướng đối tượng.

- Hiểu rõ hơn về cấu trúc dữ liệu trong C++.

b. Yêu cầu

- Chương trình có giao diện đồ họa, đầy đủ chức năng, sử dụng các cấu trúc dữ liệu và thuật toán đã học để code.

- Dịch sách.

3. Nội dung và phạm vi đề tài

a. Nội dung đề tài

- Ứng dụng hệ thống quản lý tiệc sẽ giúp giảm thiểu việc ghi chú, ghi nhận bằng giấy, biết sắp xếp thời gian các tiệc sẽ diễn ra. Ngoài ra, hỗ trợ quản lý khách hàng, thực đơn, món ăn.

b. Phạm vi đề tài

- Tập trung vào việc quản lý tiệc của gia đình tôi.

- Giải quyết các vấn đề đang tồn đọng.

4. Công nghệ, công cụ và ngôn ngữ lập trình

a. Công cụ

- Visual Studio Code.

- Git.

- Github.

b. Ngôn ngữ lập trình: *C++*.

5. Các kết quả chính dự kiến sẽ đạt được và ứng dụng

- Tạo ra được một công cụ quản lý tiệc hoàn hảo, dễ dùng, dung lượng không quá nặng, hỗ trợ 100% việc lưu trữ xuất - nhập tiệc, giảm sức người và tăng hiệu quả đến mức tối đa.

- Quyển báo cáo bài tập lớn.

- Xây dựng thành công quản lý tiệc đơn giản.

- Ứng dụng vào thực tế giúp các sinh viên cải thiện kỹ thuật lập trình và hiểu rõ về lập trình hướng đối tượng (OOP) và cấu trúc dữ liệu.

6. Giảng viên và cán bộ hướng dẫn:

- Họ tên: ThS. Trần Thị Dung.

- Đơn vị công tác: Bộ môn Công nghệ thông tin – Trường Đại học Giao thông Vận tải Phân hiệu tại thành phố Hồ Chí Minh.

Điện thoại: **0388.389.579** Email: ttdung@st.utc2.edu.vn

Giảng viên hướng dẫn	**Sinh viên thực hiện**
ThS. Trần Thị Dung	**Hoàng Gia Kiệt**

LỜI CẢM ƠN

Lời đầu tiên em xin gửi lời cảm ơn sâu sắc đến cô Trần Thị Dung đã hỗ trợ và giúp đỡ em hoàn thành môn học Lập trình hướng đối tượng (OOP) và hướng dẫn em thực hiện bài tập lớn với đề tài "Quản lý tiệc".

Em xin gửi lời cảm ơn đến quý thầy cô Bộ môn Công nghệ thông tin Trường Đại học Giao thông Vận tải Phân hiệu tại TP. Hồ Chí Minh đã truyền đạt kiến thức và giúp đỡ em trong quá trình học tập để em có thể hoàn thành tốt bài tập lớn của mình.

Ngoài ra, em xin gửi lời cảm ơn đến các anh chị bạn đã đồng hành và giúp đỡ em về tài liệu, xin gửi lời cảm ơn đến bạn Trần Văn Hậu đã cùng thực hiện hệ thống "Quản lý tiệc".

Trong quá trình thực hiện đề tài, bản thân em và các bạn còn gặp nhiều khó khăn, thiếu kiến thức chuyên môn nên báo cáo còn nhiều thiếu sót. Mong cô có thể xem xét, đánh giá cho bài tập lớn của em được cập nhật và chỉnh sửa tốt nhất.

Em xin chân thành cảm ơn./.

TP. Hồ Chí Minh, ngày … tháng 11 năm 2022

Sinh viên thực hiện

Hoàng Gia Kiệt

NHẬN XÉT CỦA GIẢNG VIÊN

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

TP. Hồ Chí Minh, ngày … tháng 11 năm 2022

Giảng viên hướng dẫn

ThS. Trần Thị Dung

MỤC LỤC

DANH MỤC CHỮ VIẾT TẮT

TỪ VIẾT TẮT	Ý NGHĨA	DIỄN GIẢI
VSCode	Visual Studio Code	Trình biên tập văn bản
OOP	Object Oriented Programming	Lập trình hướng đối tượng

DANH MỤC HÌNH ẢNH

CHƯƠNG 1: DỊCH SÁCH

1.1. Nạp chồng toán tử

1.1.1. Giới thiệu

Mọi lĩnh vực kỹ thuật - và hầu hết các lĩnh vực phi kỹ thuật - đều phát triển các ký hiệu viết tắt thông thường để thuận tiện cho việc trình bày và thảo luận liên quan đến các khái niệm được sử dụng thường xuyên. Ví dụ "x+y*z" thay vì "nhân y với z và cộng kết quả cho x".

Giống như hầu hết các ngôn ngữ khác, C++ hỗ trợ một tập hợp các toán tử cho các kiểu tích hợp của nó. Tuy nhiên, hầu hết các khái niệm mà toán tử được sử dụng thông thường không phải là các kiểu tích hợp sẵn trong C++. Nên chúng phải được biểu diễn dưới các kiểu do người dùng tự định nghĩa. Ví dụ: nếu bạn cần số học phức tạp, đại số ma trận, tín hiệu logic hoặc chuỗi kí tự trong C++, bạn sử dụng các lớp để thể hiện những khái niệm này.

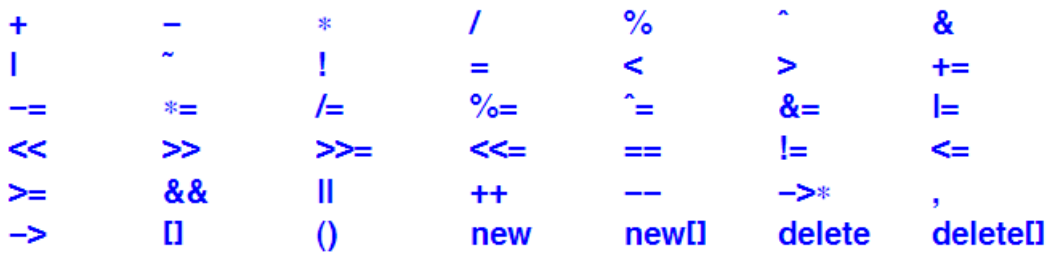
Việc xác định các toán tử cho các lớp như vậy đôi khi cho phép lập trình viên cung cấp một ký hiệu thông thường và thuận tiện hơn để thao tác các đối tượng hơn là có thể đạt được chỉ bằng cách sử dụng chức năng cơ bản.

```
class complex { // số phức đơn giản
double re, im;
public:
complex(double r, double i) :re{r}, im{i} { }
complex operator+(complex);
complex operator∗(complex);
};
```

Đây là một cách triển khai đơn giản của khái niệm số phức.

1.1.2. Hàm toán tử

Các hàm xác định ý nghĩa cho các toán tử có thể được khai báo như sau:



Hình 1.1: Các toán tử trong C++

Tên của một hàm toán tử là keyword "operator" theo sau là chính toán tử đó, ví dụ: "operator <<". Một hàm toán tử được khai báo và có thể được gọi như bất kí hàm nào khác. Việc sử dụng toán tử chỉ là một cách gọi tắt cho một lệnh gọi rõ ràng của một hàm toán tử. Ví dụ:

```
void f(complex a, complex b)
{
   complex c = a + b;          // rút gọn
   complex d = a.operator+(b); // rõ ràng
}
```

Với định nghĩa trên đây là của số phức, hai bộ khởi tạo đồng nghĩa.

1.1.2.1. Toán tử 2 ngôi và 1 ngôi

Toán tử 1 ngôi có thể được định nghĩa bởi một hàm thành viên không tĩnh mang 1 đối số hoặc một hàm không là thành viên mang 2 đối số.

```
class X
{
public:
   void operator+(int);
   X(int);
};
void operator+(X, X);
void operator+(X, double);
void f(X a)
{
   a + 1;   // a.operator+(1)
   1 + a;   // ::operator+(X(1), 1)
   a + 1.0; // ::operator+(a, 1.0)
}
```

Toán tử một ngôi, dù là tiền tố hay hậu tố, nó có thể được xác định bởi một hàm thành viên không tĩnh không có đối số hoặc một hàm không phải là một đối số.

```
class X {
public: // hàm thành viên
X* operator&(); // tiền tố 1 ngôi (&)
X operator&(X); // 2 ngôi & (And)
X operator++(int); // tăng hậu tố
X operator&(X,X); // Lỗi: có 3 thứ
X operator/(); // Lỗi: có 1 ngôi
};
// hàm không là thành viên
X operator-(X); // tiền tố 1 ngôi (-)
X operator-(X,X); // 2 ngôi (-)
X operator--(X&,int); // giảm hậu tố
X operator-(); // Lỗi: không có toán hạng
X operator-(X,X,X); // Lỗi: có 3 thứ
X operator%(X); // Lỗi: có 1 ngôi (%)
```

1.1.2.2. Ý nghĩa được xác định trước cho các toán tử

Ý nghĩa của một số toán tử dựng sẵn được định nghĩa tương đương với một số kết hợp của các toán tử khác trên cùng các đối số.

```
class X {
public:
void operator=(const X&) = delete;
void operator&() = delete;
void operator,(const X&) = delete;
};
void f(X a, X b){
a = b; // Lỗi : không có toán tử =()
&a; // Lỗi : không có toán tử &()
a,b; // Lỗi : không có toán tử,()
}
```

Ngoài ra, chúng có thể được đưa ra các nghĩa mới bằng các định nghĩa phù hợp.

1.1.2.3. Toán tử và các loại do người dùng xác định

Một hàm toán tử phải là một thành viên hoặc có ít nhất một đối số của kiểu do người dùng xác định (các hàm xác định lại toán tử new và delete không cần). Một hàm toán tử nhằm chấp nhận một kiểu dựng sẵn vì toán hạng đầu tiên của nó không thể là một hàm thành viên.

Liệt kê là kiểu do người dùng xác định để chúng ta có thể xác định các toán tử cho chúng.

```
enum Day { sun, mon, tue, wed, thu, fri, sat };
Day& operator++(Day& d){
return d = (sat==d) ? sun : static_cast<Day>(d+1);
}
```

1.1.3. Một số loại số phức

```
void f(complex x, complex y, complex z){
complex r1 {x+y+z}; // r1 = operator+(operator+(x,y),z)
complex r2 {x}; // r2 = x
r2 += y; // r2.operator+=(y)
r2 += z; // r2.operator+=(z)
}
```

Ngoại trừ sự khác biệt về tính hiệu quả có thể xảy ra, các tính toán của r1 và r2 là tương đương.

2.1. Lớp dẫn xuất

2.1.1. Giới thiệu

Một khái niệm không tồn tại một cách cô lập. Nó tồn tại cùng với các khái niệm liên quan và phần lớn sức mạnh của nó có được từ các mối quan hệ với các khái niệm khác nhau. Ví dụ, cố gắng giải thích những gì một chiếc xe hơi. Bạn sẽ sớm giới thiệu các khái niệm về bánh xe, động cơ, người lái xe, người đi bộ, xe tải, xe cứu thương, đường sá, dầu nhớt, vé chạy quá tốc độ, nhà nghỉ, v.v. Vì chúng tôi sử dụng các lớp để giải thích các khái niệm, vấn đề trở thành cách thể hiện mối quan hệ giữa các khái niệm. Tuy nhiên, chúng tôi không thể diễn đạt mối quan hệ tùy ý trực tiếp bằng ngôn ngữ lập trình. Ngay cả khi chúng tôi có thể, chúng tôi sẽ không muốn. Để 19 | P a g e trở nên hữu ích, các lớp của chúng ta

nên được định nghĩa hẹp hơn so với các khái niệm hàng ngày của chúng và chính xác hơn.

Khái niệm về một lớp dẫn xuất và các cơ chế ngôn ngữ liên quan của nó được cung cấp để thể hiện các mối quan hệ thứ bậc, nghĩa là, để thể hiện tính chung giữa các lớp. Ví dụ, các khái niệm về hình tròn và hình tam giác có liên quan với nhau ở chỗ chúng đều là hình dạng; nghĩa là họ có chung khái niệm về một hình dạng. Do đó, chúng ta xác định rõ ràng lớp Circle và lớp Triangle để có chung lớp Shape. Trong trường hợp đó, lớp chung, ở đây Shape, được gọi là lớp cơ sở hoặc siêu lớp và các lớp dẫn xuất từ lớp đó, ở đây Circle và Triangle, được gọi là lớp dẫn xuất hoặc các lớp con. Biểu diễn hình tròn và hình tam giác trong một chương trình mà không liên quan đến khái niệm về hình dạng sẽ là thiếu một thứ thiết yếu. Chương này là sự khám phá ý nghĩa của ý tưởng đơn giản này, nó là cơ sở cho cái thường được gọi là lập trình hướng đối tượng. Các tính năng ngôn ngữ hỗ trợ xây dựng các lớp mới từ những lớp hiện có:

- Triển khai tính kế thừa : để tiết kiệm việc triển khai bằng cách chia sẻ các phương thức, thuộc tính được cung cấp bởi một lớp cơ sở.

- Kế thừa giao diện: cho phép các lớp dẫn xuất khác nhau được sử dụng thay thế cho nhau thông qua giao diện được cung cấp bởi một lớp cơ sở chung.

- Kế thừa giao diện thường được gọi là đa hình thời gian chạy (hoặc đa hình động). Ngược lại, việc sử dụng thống nhất các lớp không liên quan đến tính kế thừa được cung cấp bởi các mẫu, thường được gọi là đa hình thời gian biên dịch (hoặc đa hình tĩnh).

2.1.2. Lớp dẫn xuất

Xây dựng một chương trình giao dịch với những người được một công ty tuyển dụng. Một chương trình có thể có cấu trúc dữ liệu như thế này.

```
struct Employee
{
   string name;
   char middle_initial;
   Date hiring_date;
};
```

Tiếp theo, chúng ta định nghĩa cấu trúc của người quản lý.

```
struct Manager
{
   Employee emp;
   list<Employee *> group;
};
```

Một nhà quản lý cũng là một nhân viên; dữ liệu Employee được lưu trữ trong thành viên "emp" của đối tượng Manager. Điều này có thể hiển nhiên đối với người đọc là con người - đặc biệt là người đọc cẩn thận - nhưng không có gì nói với trình biên dịch và các công cụ khác rằng Manager cũng là Employee. Manager ∗ không phải là Employee ∗, vì vậy người ta không thể đơn giản sử dụng cái này khi người kia được yêu cầu. Đặc biệt, người ta không thể đặt một Manager vào danh sách Employee mà không cần viết mã đặc biệt. Chúng ta có thể sử dụng chuyển đổi loại rõ ràng trên Manager ∗ hoặc đưa địa chỉ của thành viên "emp" vào danh sách Employee. Tuy nhiên, cả hai giải pháp đều không phù hợp và có thể khá mù mờ. Cách tiếp cận đúng là tuyên bố rõ ràng rằng Manager là Employee, với một vài thông tin được thêm vào:

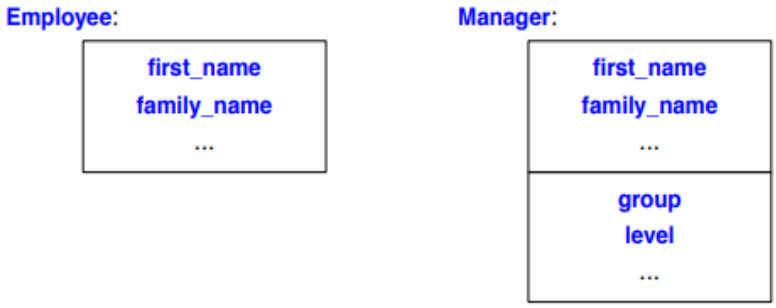
```
struct Manager : public Employee
{
   list<Employee *> group;
}
```

Manager có nguồn gốc từ Employee, và ngược lại, Employee là một lớp cơ sở cho Manager. Lớp Manager có các thuộc tính của lớp Employee (tên người, phòng ban, v.v.) ngoài các thành viên của chính nó (nhóm, cấp, v.v.).

Khởi tạo thường được biểu diễn bằng đồ thị bởi một con trỏ từ lớp dẫn xuất đến lớp cơ sở của nó cho biết rằng lớp dẫn xuất tham chiếu đến cơ sở của nó (thay vì ngược lại):

Một lớp dẫn xuất thường được cho là kế thừa các thuộc tính từ cơ sở của nó, vì vậy mối quan hệ còn được gọi là kế thừa. Một lớp cơ sở đôi khi được gọi là lớp cha và lớp dẫn xuất là lớp con. Tuy nhiên, thuật ngữ này gây nhầm lẫn cho những người quan sát rằng dữ liệu trong một đối tượng lớp dẫn xuất là một tập siêu dữ liệu của một đối tượng thuộc lớp cơ sở của nó. Một lớp dẫn xuất thường lớn hơn (và không bao giờ nhỏ hơn) so với lớp cơ sở của nó theo nghĩa là nó chứa nhiều dữ liệu hơn và cung cấp nhiều chức năng hơn.

Một cách triển khai phổ biến và hiệu quả của khái niệm lớp dẫn xuất có một đối tượng của lớp dẫn xuất được biểu diễn như một đối tượng của lớp cơ sở, với thông tin thuộc về lớp dẫn xuất cụ thể được thêm vào cuối. Ví dụ:



Hình 1.2: Ví dụ về lớp dẫn xuất

CHƯƠNG 2: CƠ SỞ LÝ THUYẾT

2.1. Tổng quan về ngôn ngữ lập trình C++

2.1.1. C++ là gì?

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. [1]

2.1.2. Các đặc điểm nổi bật

C++ là một ngôn ngữ tầm trung, bạn hoàn toàn có thể sử dụng nó để phát triển các chương trình bậc thấp hay những chương trình bậc cao, mà vẫn hoạt động tốt trong phần cứng. [2]

C++ là ngôn ngữ lập trình hướng đối tượng, sử dụng các Class và Object cùng các khái niệm như tính kế thừa, tính đa hình, tính đóng gói... để tạo ra các chương trình.

C++ được tạo ra dựa trên nền tảng ngôn ngữ C, nên nó có hầu hết mọi tính năng của C và được bổ sung thêm khái niệm functions trong quá trình thiết kế chương trình.

Các chương trình được tạo ra bởi C++ đều có thể chạy được trên các hệ điều hành như Mac OS, Windows, hay một số biến thể của Unix.

2.2. Tổng quan về Visual Studio Code

2.2.1. Visual Studio Code là gì?

VSCode chính là ứng dụng cho phép biên tập, soạn thảo các đoạn code để hỗ trợ trong quá trình thực hiện xây dựng, thiết kế website một cách nhanh chóng. VSCode vận hành trên các nền tảng như Windows, macOS, Linux. VSCode còn cho khả năng tương thích với những thiết bị máy tính có cấu hình tầm trung. [3]

VSCode hỗ trợ đa dạng các chức năng Debug, đi kèm với Git, có Syntax Highlighting. Đặc biệt là tự hoàn thành mã thông minh, Snippets, và khả năng cải tiến mã nguồn.

2.2.2. Lí do chọn Visual Studio Code?

Giao diện thân thiện với người dùng.

Dễ dàng sử dụng.

Phù hợp với những đang học lập trình để làm bài tập thuận tiện hơn, ít phát sinh nhiều file không cần thiết.

Hỗ trợ đa ngôn ngữ: C/C++, HTML, CSS, JS…

Số lượng người sử dụng rộng rãi nên sẽ được hỗ trợ dễ dàng khi có vấn đề nào đó trong việc lập trình.

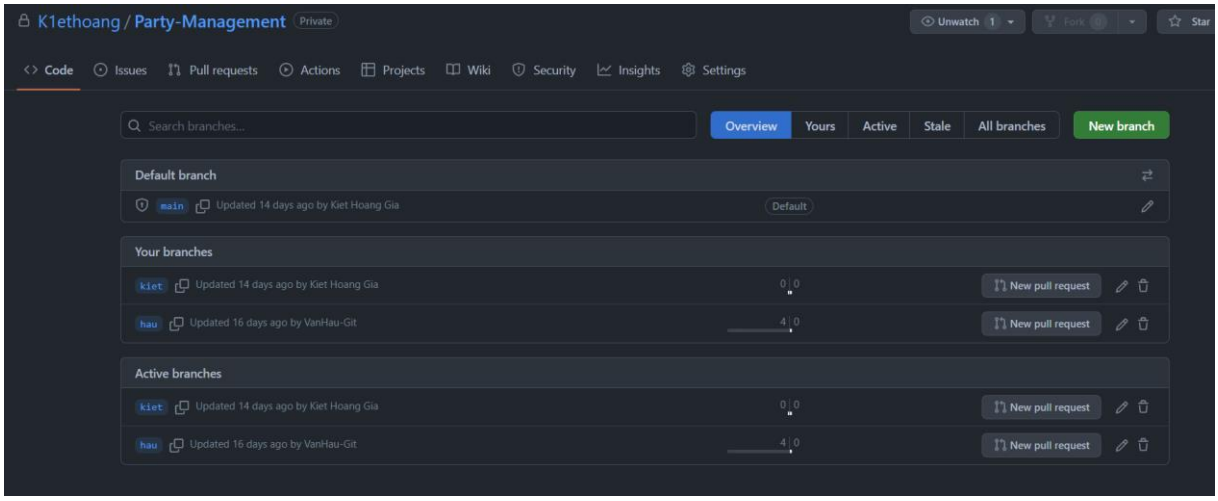
2.3. Tổng quan về Github

2.3.1. Github là gì?

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. [4]

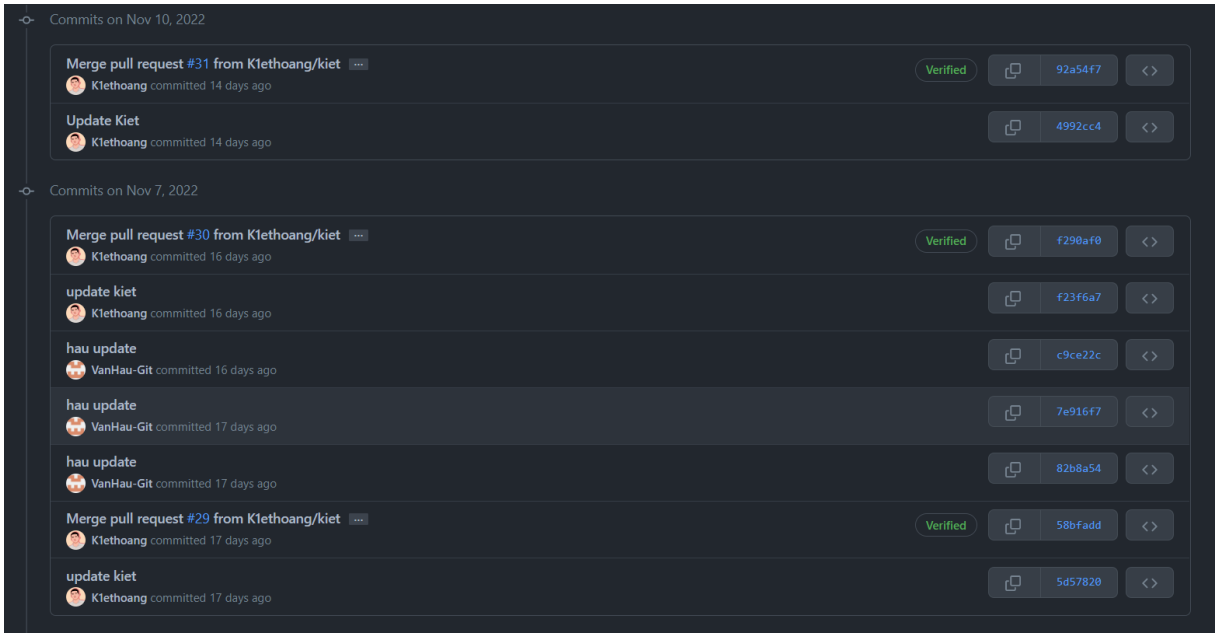
2.3.2. Github với dự án

Trong dự án "Quản lý tiệc" này thì việc chia nhánh (branch) theo người đóng góp dự án.



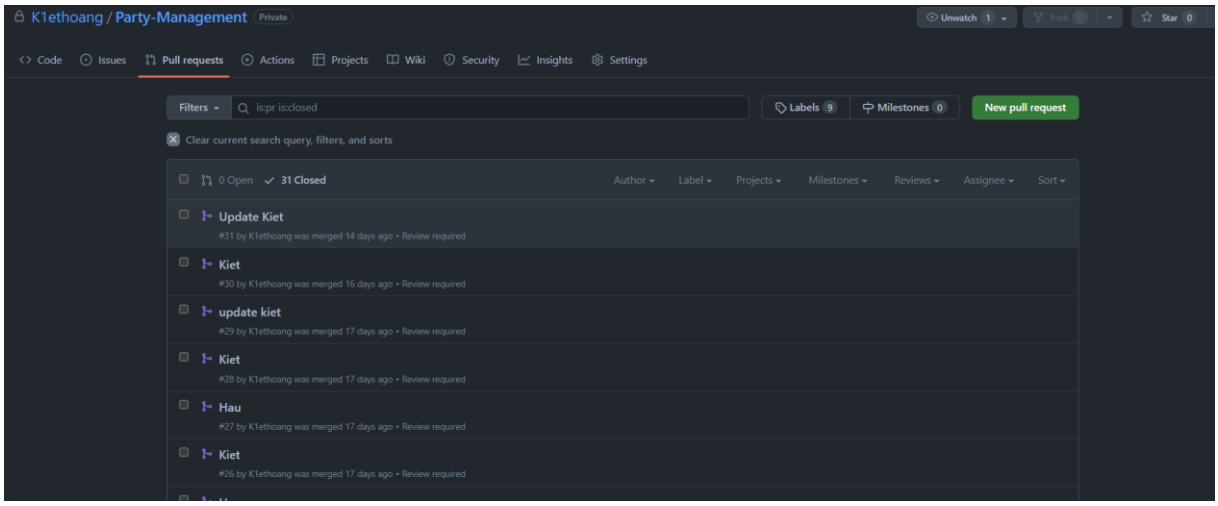
Hình 2.1: Các nhánh trong dự án

Lệnh **commit** để lưu các thông tin của tệp khi thực hiện các thao tác thêm sửa xoá trong tệp.



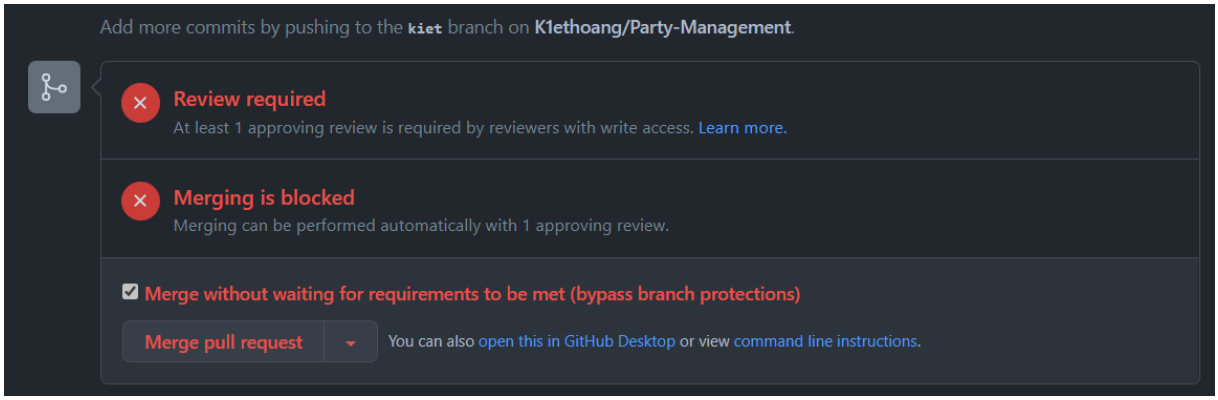
Hình 2.2: Commit trong dự án

Lệnh **pull** cho biết sự thay đổi trong tệp và yêu cầu hợp nhất với nhánh chính (pull request).



Hình 2.3: pull request trong dự án

Lệnh **merge** dùng để hợp nhất các nhánh develop vào nhánh chính.



Hình 2.4: Merge trong dự án

Ngoài ra còn có nhiều lệnh và tính năng, tuy nhiên trong dự án nhỏ này chưa được sử dụng tới.

2.4. Tổng quan về OOP

2.4.1. OOP là gì?

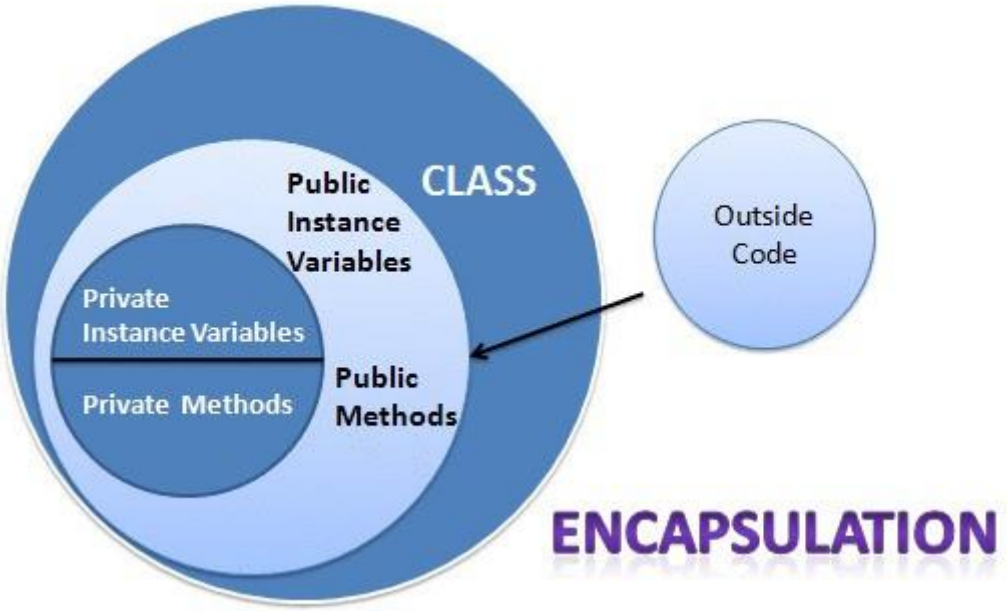
OOP là một phương pháp lập trình dựa trên khái niệm về lớp (class) và đối tượng (object). OOP tập trung vào các đối tượng thao tác hơn là logic để thao tác chúng, giúp code dễ quản lý, tái sử dụng được và dễ bảo trì. [5]

2.4.2. Các tính chất của OOP

2.4.2.1. Tính đóng gói (Encapsulation)

Tính đóng gói cho phép che giấu thông tin và những tính chất xử lý bên trong của đối tượng. Các đối tượng khác không thể tác động trực tiếp đến dữ liệu bên trong và làm thay đổi trạng thái của đối tượng mà bắt buộc phải thông qua các phương thức công khai do đối tượng đó cung cấp.

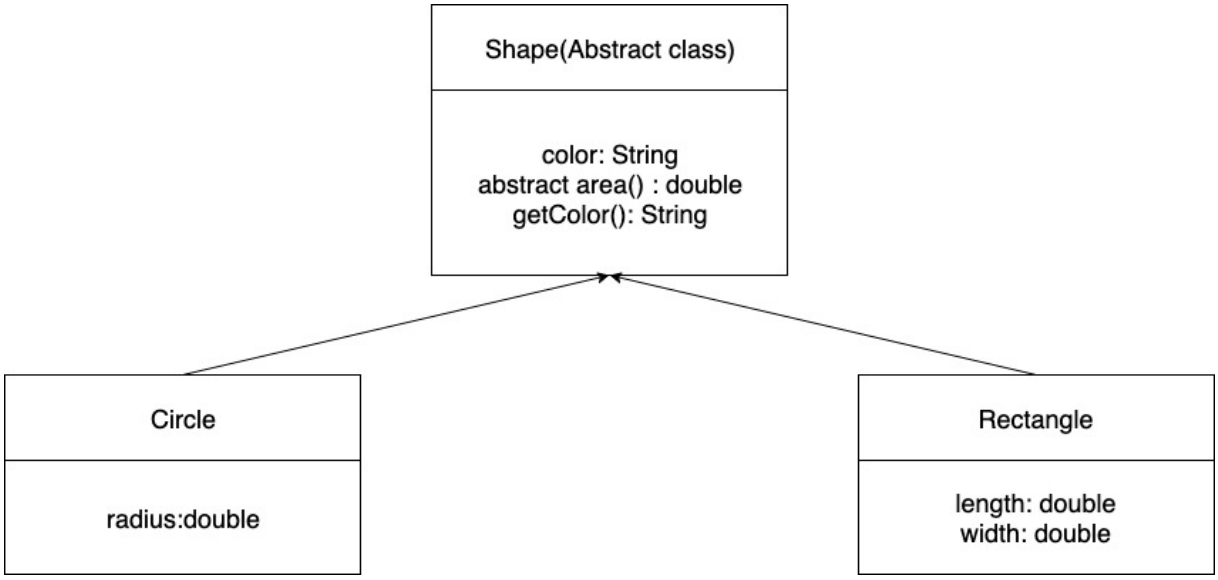
Tính chất này giúp tăng tính bảo mật cho đối tượng và tránh tình trạng dữ liệu bị hư hỏng ngoài ý muốn.



Hình 2.5: Minh hoạ về tính đóng gói trong OOP

2.4.2.2. Tính trừu tượng (Abstraction)

Tính trừu tượng giúp loại bỏ những thứ phức tạp, không cần thiết của đối tượng và chỉ tập trung vào những gì cốt lõi, quan trọng.

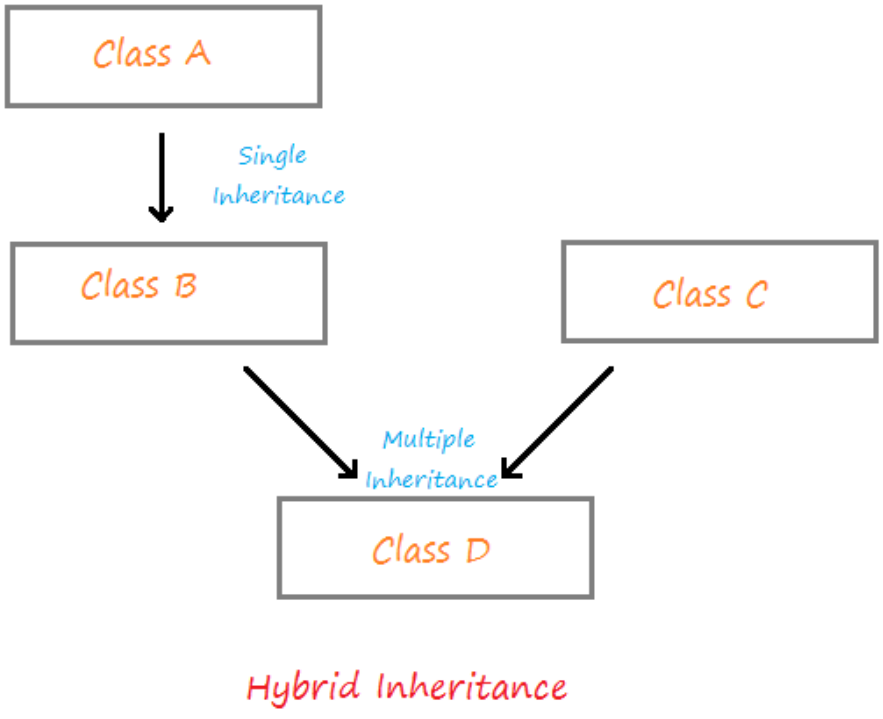


Hình 2.6: Minh hoạ về tính trừu tượng trong OOP

2.4.2.3. Tính kế thừa (Inheritance)

Đây là tính chất được sử dụng khá nhiều. Tính kế thừa cho phép xây dựng một lớp mới (lớp Con), kế thừa và tái sử dụng các thuộc tính, phương thức dựa trên lớp cũ (lớp Cha) đã có trước đó.

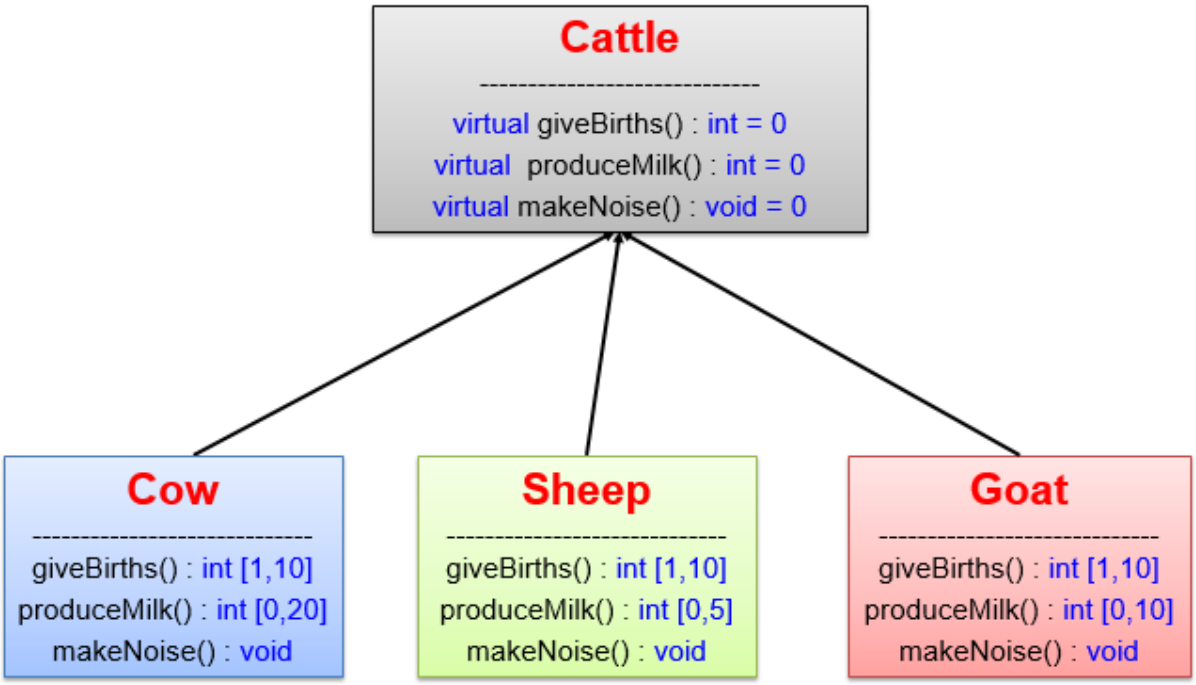
Các lớp Con kế thừa toàn bộ thành phần của lớp Cha và không cần phải định nghĩa lại. Lớp Con có thể mở rộng các thành phần kế thừa hoặc bổ sung những thành phần mới.



Hình 2.7: Minh hoạ tính kế thừa trong OOP

2.4.2.4. Tính đa hình (Polymorphism)

Tính đa hình trong lập trình OOP cho phép các đối tượng khác nhau thực thi chức năng giống nhau theo những cách khác nhau.



Hình 2.8: Minh hoạ tính đa hình trong OOP

2.5. Tổng quan về cấu trúc dữ liệu & giải thuật

2.5.1. Khái niệm

Cấu trúc dữ liệu (Data Structure) là cách lập trình để lưu trữ dữ liệu để dữ liệu có thể được sử dụng một cách hiệu quả.

Thuật toán (Algorithms) là một thủ tục từng bước, xác định một tập hợp các lệnh được thực hiện theo một thứ tự nhất định để có được đầu ra mong muốn.

Cấu trúc dữ liệu và giải thuật là sự kết hợp và áp dụng một hoặc nhiều cấu trúc dữ liệu nào đó vào một hoặc nhiều thuật toán nào đó để có được đầu ra mong muốn một cách tối ưu và tốt nhất khi dữ liệu có số lượng cực lớn. [6]

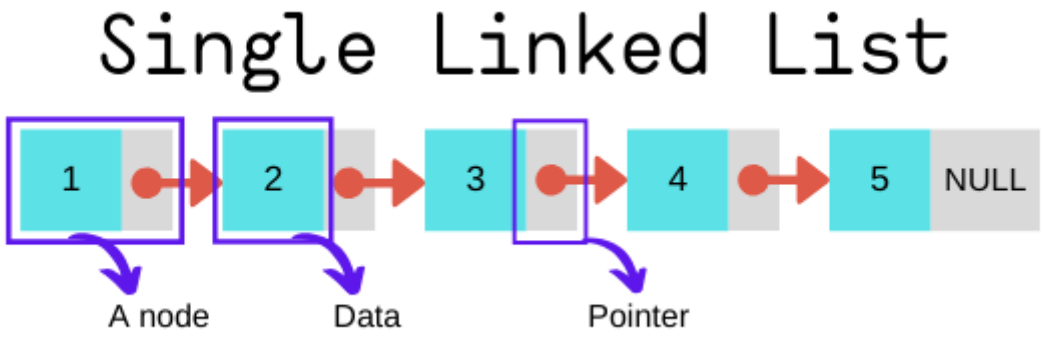
2.5.2. Cấu trúc dữ liệu trong dự án

2.5.2.1. Danh sách liên kết đơn (Singly linked list)

Danh sách liên kết đơn (Single Linked List) là một cấu trúc dữ liệu động, nó là một danh sách mà mỗi phần tử đều liên kết với phần tử đúng sau nó trong

danh sách. Mỗi phần tử (được gọi là một node hay nút) trong danh sách liên kết đơn có 2 thành phần: [7]

- Thành phần dữ liệu: lưu thông tin về bản thân phần tử đó.
- Thành phần liên kết: lưu địa chỉ phần tử đứng sau trong danh sách, nếu phần tử đó là phần tử cuối cùng thì thành phần này bằng NULL.

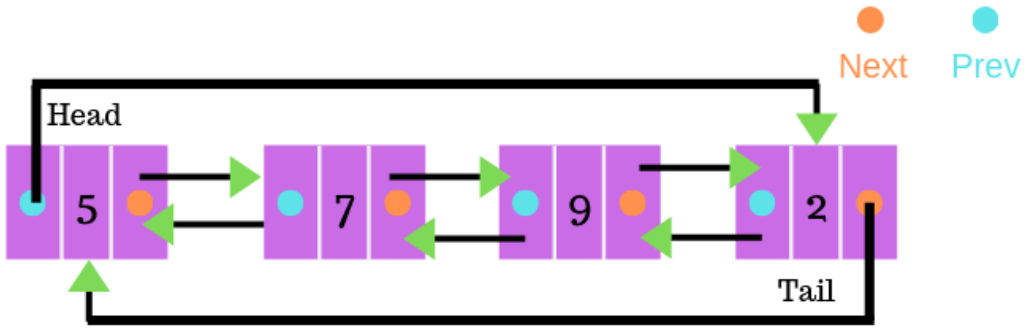


Hình 2.9: Minh hoạ về danh sách liên kết đơn

2.5.2.2. Danh sách liên kết đôi vòng (Circular doubly linked list)

Danh sách liên kết đôi vòng là một biến thể của danh sách liên kết đơn, nhưng phần tử đầu trỏ tới phần tử cuối cùng và phần tử cuối cùng trỏ tới phần tử đầu tiên.

Mỗi phần tử trong danh sách liên kết đơn sẽ có thêm 1 liên kết để lưu địa chỉ của phần tử trước nó.



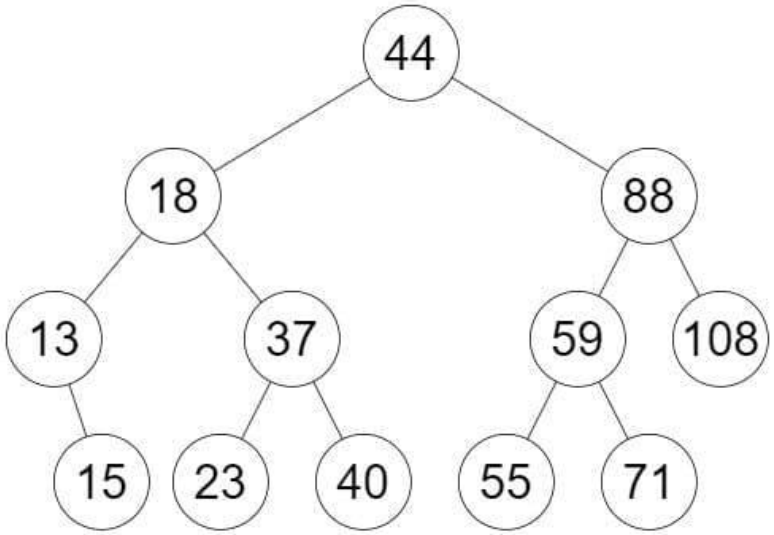
Hình 2.10: Minh hoạ danh sách liên kết đôi vòng

2.5.2.3. Cây nhị phân tìm kiếm (Binary search tree)

Cây nhị phân tìm kiếm là cây nhị phân mà trong đó, các phần tử của cây con bên trái đều nhỏ hơn phần tử hiện hành và các phần tử của cây con bên phải

đều lớn hơn phần tử hiện hành. Do tính chất này, cây nhị phân tìm kiếm không được có phần tử cùng giá trị.

Nhờ vào tính chất đặc biệt này, cây nhị phân tìm kiếm được sử dụng để tìm kiếm phần tử nhanh hơn (tương tự với tìm kiếm nhị phân). Khi duyệt cây nhị phân theo cách duyệt trung tự, bạn sẽ thu được một mảng có thứ tự. Chúng ta sẽ lần lượt tìm hiểu qua chúng.



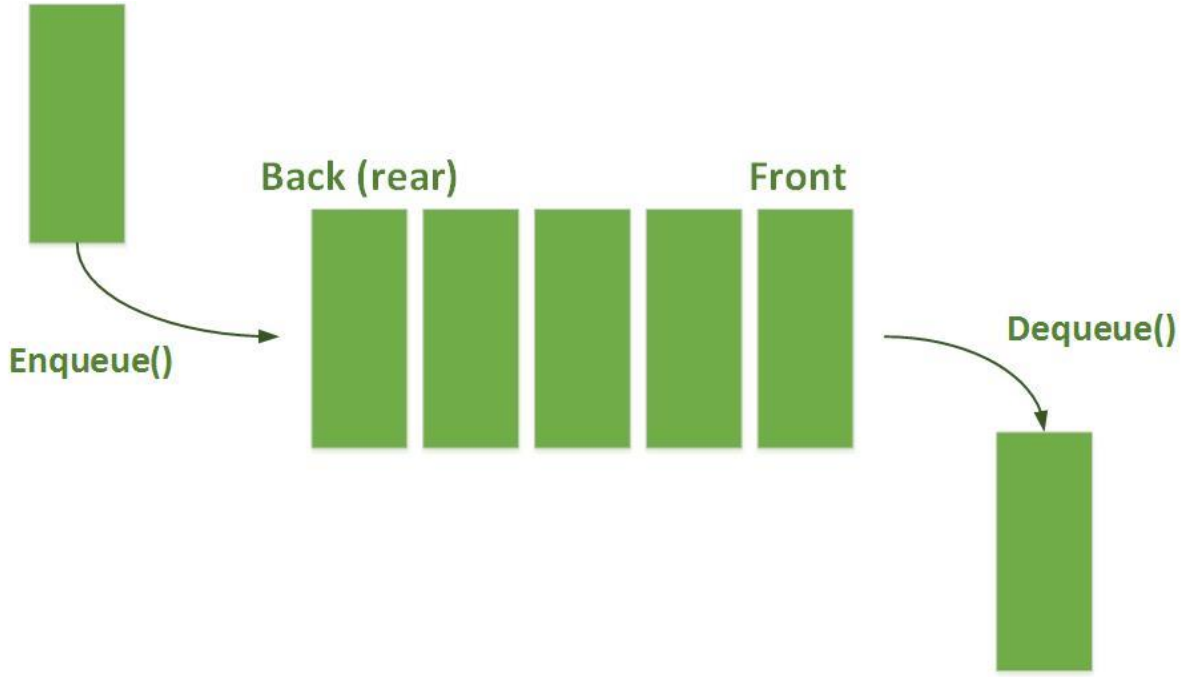
Hình 2.11: Minh hoạ cây nhị phân tìm kiếm

2.5.2.4. Hàng đợi (Queue)

Áp dụng kiến thức về template trong OOP để xây dựng lại hàng đợi. Vậy template trong OOP là gì?

- Template (khuôn mẫu) là một từ khóa trong C++, và là một kiểu dữ liệu trừu tượng tổng quát hóa cho các kiểu dữ liệu int, float, double, bool...
- Template trong C++ có 2 loại đó là function template & class template.
- Template giúp người lập trình định nghĩa tổng quát cho hàm và lớp thay vì phải nạp chồng (overloading) cho từng hàm hay phương thức với những kiểu dữ liệu khác nhau.

Hàng đợi hay queue là một cấu trúc dữ liệu động hoạt động theo cơ chế FIFO (First In, First Out), nghĩa là phần tử được thêm vào đầu tiên sẽ là phần tử được lấy ra đầu tiên.



Hình 2.12: Minh hoạ hàng đợi

2.5.3. Thuật toán trong dự án

2.5.3.1. Sắp xếp chọn (Selection sort)

Thuật toán sắp xếp chọn sẽ sắp xếp một mảng bằng cách đi tìm phần tử có giá trị nhỏ nhất (giả sử với sắp xếp mảng tăng dần) trong đoạn chưa được sắp xếp và đổi cho phần tử nhỏ nhất đó với phần tử ở đầu đoạn chưa được sắp xếp (không phải đầu mảng).

Độ phức tạp thuật toán:

- Trường hợp tốt: O (n^2).
- Trung bình: O (n^2).
- Trường hợp xấu: O (n^2).

Không gian sử dụng bộ nhớ: O (1).

CHƯƠNG 3: PHÂN TÍCH BÀI TOÁN

3.1. Mô tả bài toán

3.1.1. Phân tích nghiệp vụ

Bước 1: Khách hàng đến địa chỉ nhà hàng

Bước 2: Nhà hàng hỏi các thông tin của khách hàng và buổi tiệc

Bước 3: Lưu vào sổ và thanh toán tiền cọc

Bước 4: Chờ tới ngày tiệc và chuẩn bị cho buổi tiệc

Bước 5: Thanh toán tiền còn lại

3.1.2. Mô tả bài toán

Nhà hàng truy cập vào hệ thống quản lý để lưu thông tin của khách hàng gồm: họ tên, số điện thoại, số căn cước công dân. Quản lý tiệc bằng cách: thêm, sửa, xoá, xem các tiệc, chia các loại tiệc, chọn món ăn cho buổi tiệc. Quản lý thực đơn của nhà hàng: xem thực đơn để chọn món ăn. Quản lý hoá đơn: xem và in hoá đơn.

3.2. Yêu cầu hệ thống

3.2.1. Yêu cầu chức năng

Yêu cầu về lưu trữ:

- Thông tin khách hàng:

  - Họ tên
  - Số điện thoại
  - Số căn cước công dân

- Thông tin buổi tiệc:

  - Loại tiệc
  - Số bàn tiệc
  - Thời gian tổ chức
  - Trạng thái tiệc

- Trạng thái thanh toán
- Trạng thái đặt cọc
- Tổng tiền của tiệc
- Thực đơn của tiệc

- Thông tin món:

- Tên món ăn
- Giá cả món ăn

Yêu cầu về tính năng:

- Thêm, sửa, xoá, xem thông tin tiệc

- Thêm, sửa, xem thông tin khách hàng

- Xem thực đơn

- Tự động tạo hoá đơn khi lưu thông tin tiệc, xem và xuất hoá đơn

- Thanh toán tiệc

3.2.2. Yêu cầu phi chức năng

Yêu cầu về chất lượng:

- Dễ dàng nâng cấp hoặc thêm chức năng khác.

- Có đặc tả và hướng dẫn rõ ràng.

- Hoạt động ổn định, đáng tin cậy.

Yêu cầu về giao diện:

- Giao diện thân thiện, dễ sử dụng.

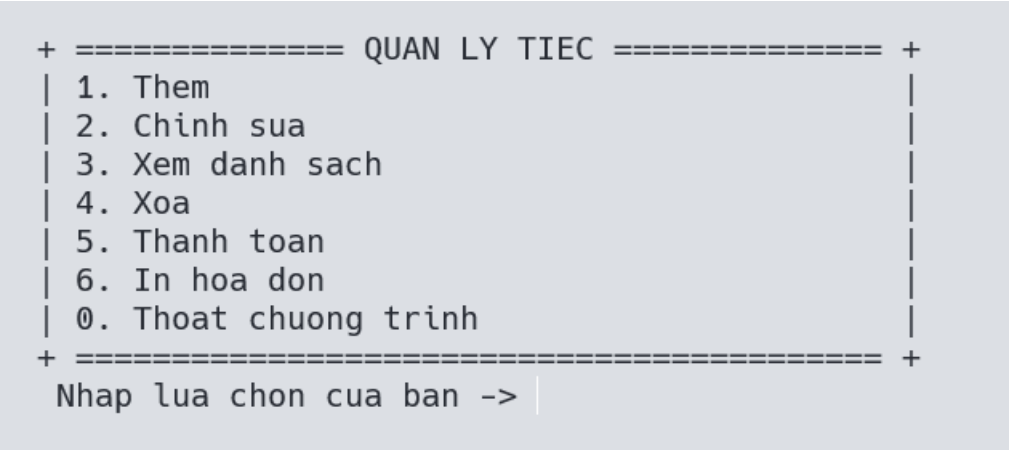
- Hiển thị đầy đủ và chi tiết các thông tin.

- Có tính nhất quán trong tất cả các chức năng.

CHƯƠNG 4: XÂY DỰNG CHƯƠNG TRÌNH

4.1. Các chức năng chính

Khi vào hệ thống thì mình có thể lựa chọn các chức năng.



Hình 4.1: Giao diện chính

4.2. Chức năng thêm tiệc

- Nhập thông tin khách hàng:

  - Sau khi nhập tên sẽ được tự động format lại cho chuẩn (VD: hoàng GIA kIệt -> Hoàng Gia Kiệt).

```
istream &operator>>(istream &is, Customer &c)
{
    cout << "\n\t\t\t\t\t\tNhap ho va ten: ";
    fflush(stdin);
    getline(is, c.fullName);
    formatName(c.fullName);
    // Nhập sđt và kiểm tra
    do
    {
        cout << "\t\t\t\t\t\tNhap so dien thoai: ";
        getline(is, c.phoneNumber);
        if (!isValidPhoneNumber(c.phoneNumber))
            cout << "\n\t\t\t\t\t\t(!) So dien thoai khong hop le! (!)\n";
    } while (!isValidPhoneNumber(c.phoneNumber));
    // Nhập số identityCard và kiểm tra
    do
    {
        cout << "\t\t\t\t\t\tNhap so CCCD: ";
        getline(is, c.identityCard);
        if (!isValidIdentityCard(c.identityCard))
            cout << "\n\t\t\t\t\t\t(!) So CCCD khong hop le! (!)\n";
    } while (!isValidIdentityCard(c.identityCard));
    return is;
}
```

- Kiểm tra số điện thoại, căn cước công dân có hợp lệ hay không, nếu tất cả hợp lệ, xuất ra thông tin khách hàng vừa nhập.

- Nhập thông tin tiệc:

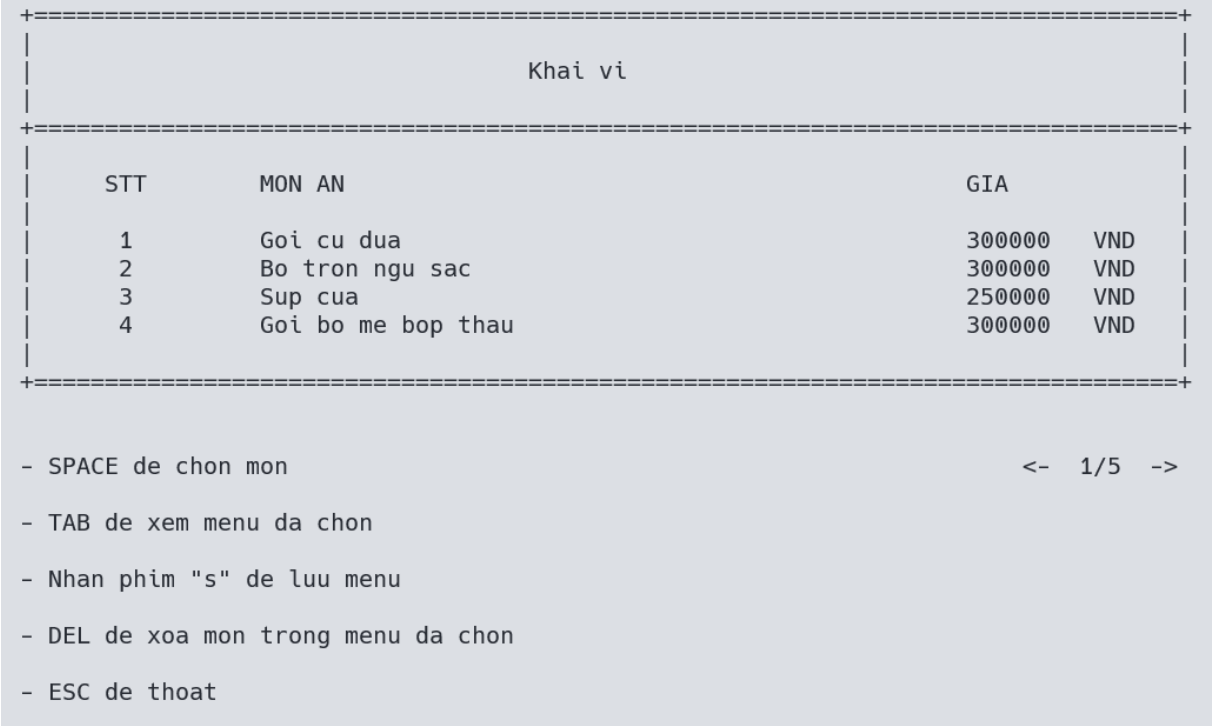
- Khi nhập thông tin của buổi tiệc, kiểm tra xem tiệc này có trùng với các ID đã có không, nếu có thì sẽ nhập lại ID mới.
- Chọn loại tiệc khả dụng của nhà hàng (Tiệc cưới, tiệc trà, tiệc sinh nhật, tiệc set menu).
- Nhập số bàn của buổi tiệc, nếu số bàn bé hơn hoặc bằng 0 thì sẽ nhập lại.
- Nhập thời gian tổ chức tiệc, nếu không hợp lệ sẽ bắt nhập lại. Hợp lệ thì sẽ tự động cập nhật trạng thái cho buổi tiệc.

```
istream &operator>>(istream &is, Party &p)
{
    p.inputID();
    string res = chooseTypeParty();
    p.setTypeParty(res);
    cout << "\n\t\t\t\t\t\tNhap so ban cua buoi tiec: ";
    is >> p.tableNumber;
    cout << "\n\t\t\t\t\t\tNhap thoi gian to chuc tiec:" << endl;
    p.date.inputDate("(!) Ngay thang khong hop le (!)");
    p.setPartyStatus();
    system("pause");
    return is;
}
```

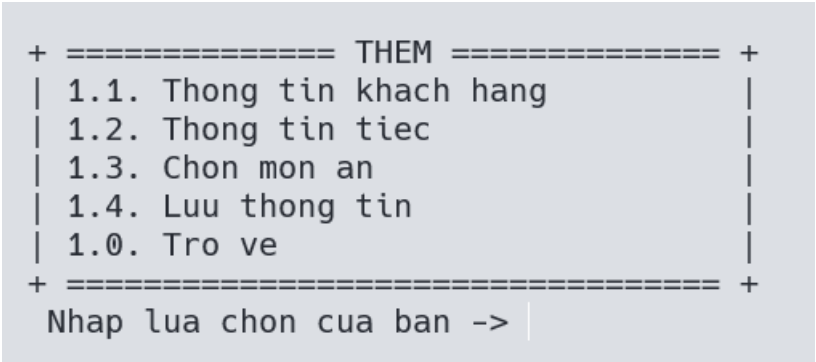
- Thêm thông tin của buổi tiệc vừa nhập vào cây nhị phân tìm kiếm đã tạo.

```
void PartiesBST::add(const ItemP &value)
{
    addPrivate(root, value);
    size++;
}
```

- Chọn món ăn cho buổi tiệc:



Hình 4.2: Giao diện chọn món ăn



Hình 4.3: Giao diện thêm tiệc

4.3. Chức năng chỉnh sửa thông tin

- Nhập ID để tìm bữa tiệc cần chỉnh sửa và kiểm tra xem có tồn tại bữa tiệc không.

- Xây dựng hàm để lựa chọn tiêu chí cần chỉnh sửa.

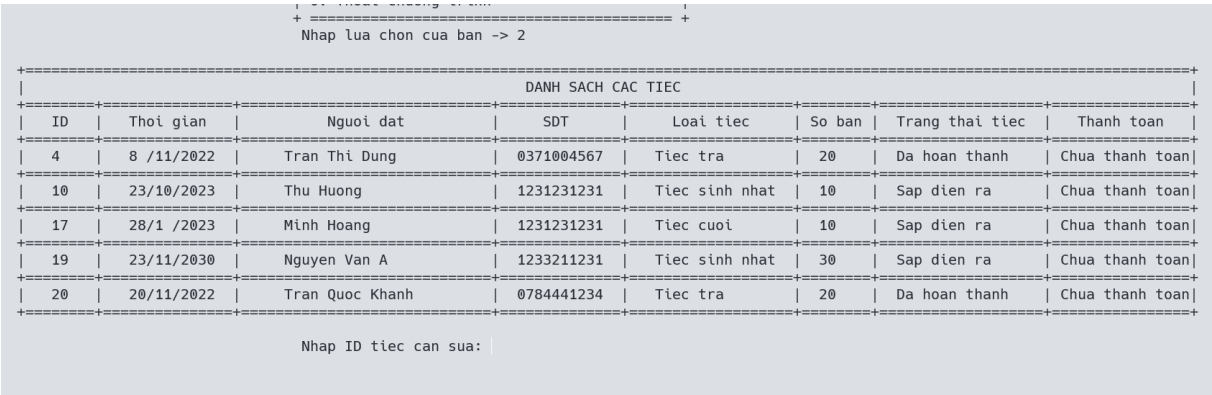
- Gọi hàm lựa chọn tiêu chí chỉnh sửa vừa xây dựng để người đặt chọn tiêu chí chỉnh sửa.

- Xuất ra thông tin mới sau khi đã chỉnh sửa.

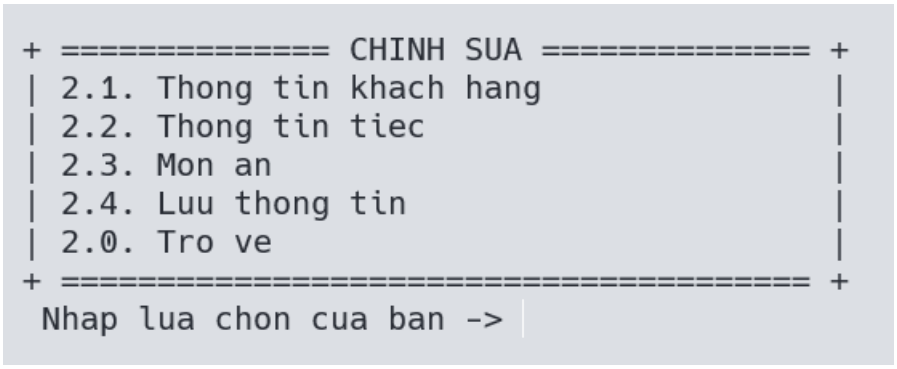
- Chỉnh sửa gồm:

- Thông tin tiệc.
- Thông tin khách hàng.
- Món ăn của tiệc.

```
parties.display();
cout << endl;
long _ID;
_ID = returnChoose("Nhap ID tiec can sua: ");
if (parties.isExistID(_ID))
{
    editDisplay(parties, foods, _ID);
    parties.exportPartiesData(PARTY_DATA_PATH);
}
else
{
    cout << "\n\t\t\t\t\t\t(!) Khong co ton tai tiec nay (!)";
    pressAnyKey();
}
```



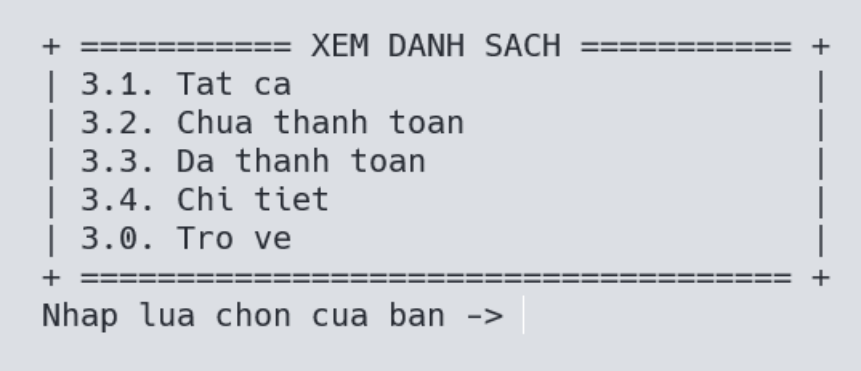
Hình 4.4: Giao diện chọn tiệc để chỉnh sửa



Hình 4.5: Giao diện các thông tin cần sửa

4.4. Chức năng xem danh sách các tiệc

- Xây dựng danh sách xuất thông tin tiệc theo chiều ngang

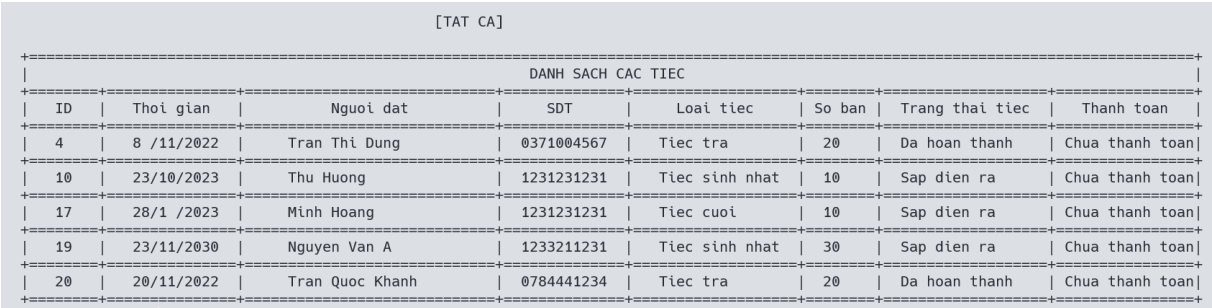


Hình 4.6: Giao diện xem danh sách các tiệc

- 4 lựa chọn xem danh sách:

- Xem tất cả: Duyệt theo Left-Node-Right trong cây nhị phân tìm kiếm, và xuất danh sách thông tin các tiệc theo chiều ngang.

```
void PartiesBST::printInOrderPrivate(NodeP *root)
{
    if (root != NULL)
    {
        printInOrderPrivate(root->left);
        cout << root->data;
        printInOrderPrivate(root->right);
    }
}
```

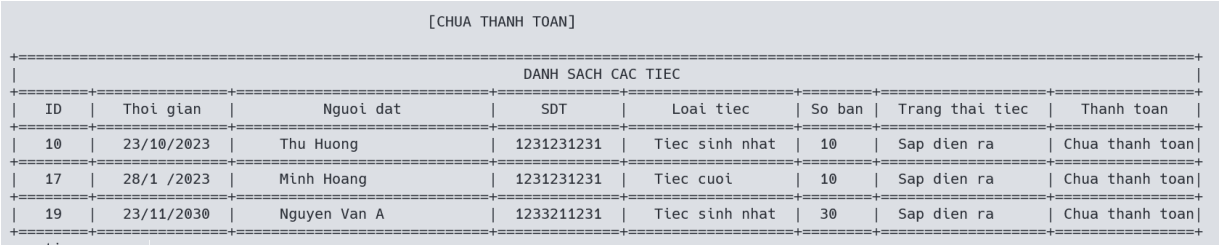


Hình 4.7: Giao diện xem tất cả tiệc

- Xem các tiệc chưa thanh toán: Duyệt theo Node-Left-Right trong cây nhị phân tìm kiếm, và xuất danh sách thông tin các tiệc theo chiều ngang.

- Xem các tiệc đã thanh toán: Duyệt theo Node-Left-Right trong cây nhị phân tìm kiếm, và xuất danh sách thông tin các tiệc theo chiều ngang.

```
void PartiesBST::printPreOrderPrivate(NodeP *root, const bool &_isPaid)
{
    if (root != NULL)
    {
        printPreOrderPrivate(root->left, _isPaid);
        if (root->data.getIsPaymentStatus() == _isPaid)
            cout << root->data;
        printPreOrderPrivate(root->right, _isPaid);
    }
}
```

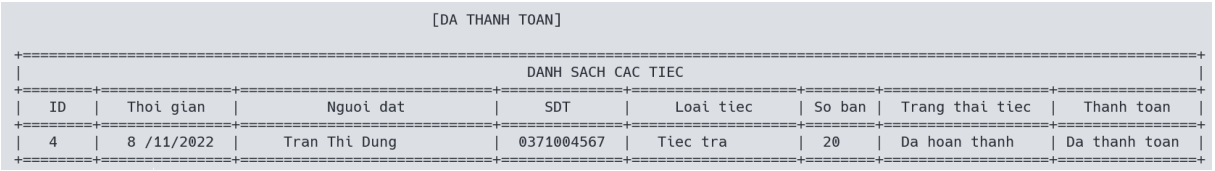


[CHUA THANH TOAN]

DANH SACH CAC TIEC

ID	Thoi gian	Nguoi dat	SDT	Loai tiec	So ban	Trang thai tiec	Thanh toan
10	23/10/2023	Thu Huong	1231231231	Tiec sinh nhat	10	Sap dien ra	Chua thanh toan
17	28/1 /2023	Minh Hoang	1231231231	Tiec cuoi	10	Sap dien ra	Chua thanh toan
19	23/11/2030	Nguyen Van A	1233211231	Tiec sinh nhat	30	Sap dien ra	Chua thanh toan

Hình 4.8: Giao diện tiệc chưa thanh toán



[DA THANH TOAN]

DANH SACH CAC TIEC

ID	Thoi gian	Nguoi dat	SDT	Loai tiec	So ban	Trang thai tiec	Thanh toan
4	8 /11/2022	Tran Thi Dung	0371004567	Tiec tra	20	Da hoan thanh	Da thanh toan

Hình 4.9: Giao diện tiệc đã thanh toán

- Xem chi tiết thông tin các tiệc:

>>>HOA DON THANH TOAN<<<

> Ten: Tran Thi Dung
> SDT: 0371004567
> So CCCD: 090823455678

> ID: 4
> Loai tiec: Tiec tra
> So ban: 20
> Thoi gian to chuc: 8 /11/2022

STT	Mon an	Gia ca
1	Goi cu dua	300000
2	Muc xao sa te	350000
3	Lau thai	350000
4	Pepsi	70000
5	Nho my	200000

TONG TIEN: 25400000 VND

>>>HOA DON THANH TOAN<<<

Hình 4.10: Giao diện xem chi tiết

4.5. Chức năng xoá tiệc

- Nhập ID để tìm tiệc cần xóa.

- Xác nhận xem có thật sự muốn xóa tiệc có ID vừa nhập hay không.

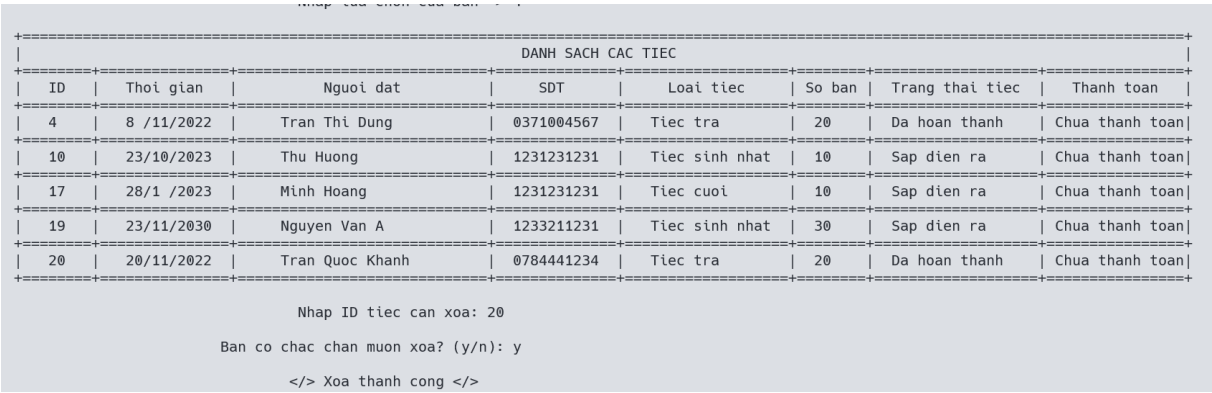
```
parties.display();
cout << endl;
long ID;
ID = returnChoose("Nhap ID tiec can xoa: ");
if (parties.isExistID(ID))
{
    char answer;
    do
    {
        cout << "\n\t\t\t\t\tBan co chac chan muon xoa? (y/n): ";
        fflush(stdin);
        cin >> answer;
        if (answer == 'y')
        {
            parties.remove(ID);
            parties.exportPartiesData(PARTY_DATA_PATH);
            cout << "\n\t\t\t\t\t\t</> Xoa thanh cong </>\n";
        }
        else if (answer == 'n')
            break;
    } while (answer != 'y' && answer != 'n');
}
else
    cout << "\n\t\t\t\t\t\t>>> Khong co ton tai tiec nay <<<";
pressAnyKey();
```

- Xoá tiệc là xoá 1 node trong cây nhị phân tìm kiếm.

```
void PartiesBST::removeNodePrivate(NodeP *&root, const long &_ID)
{
    if (root == NULL)
        return;
    else
    {
        if (_ID < root->data.getID())
            removeNodePrivate(root->left, _ID);
        else if (_ID > root->data.getID())
            removeNodePrivate(root->right, _ID);
        else // đã tìm ra node cần xoá - root->data.getID() == _ID
        {
            // node x là node thế mạng - tí nữa xoá nó
            // xoá được cho cả node lá
            NodeP *x = root;
            if (root->left == NULL) // nếu nhánh trái NULL <=> đây chính là con
phải
            {
                root = root->right; // duyệt sang phải của cái node cần xoá
                // để update mlk giữa node cha của node cần xoá với node con của
node cần xoá
            }
            else if (root->right == NULL) // nếu nhánh phải NULL <=> đây là node
có 1 con chính là con trái
            {
                root = root->left; // duyệt sang trái của cái node cần xoá
                // để update mlk giữa node cha của node cần xoá với node con của
node cần xoá
            }
            else // node cần xoá là node có 2 con
            {
                // cách 1: Tìm node trái nhất của cây con phải (cây con phải của
node cần xoá)
                NodeP *y = root->right; // node y là node thế mạng cho node cần
xoá - node này sẽ đảm bảo nhận nhiệm vụ là tìm ra node trái nhất

                // cách 2: ngược lại với cách 1
            }
            delete x;
        }
    }
    size--;
}

void PartiesBST::remove(const long &ID)
{
    removeNodePrivate(root, ID);
}
```



```
+=============================================================================================================================================+
|                                                        DANH SACH CAC TIEC                                                                   |
+========+==============+============================+=============+==================+========+==================+================+
|   ID   |   Thoi gian  |          Nguoi dat         |     SDT     |     Loai tiec    | So ban |  Trang thai tiec |   Thanh toan   |
+========+==============+============================+=============+==================+========+==================+================+
|   4    |   8 /11/2022 |     Tran Thi Dung          |  0371004567 |   Tiec tra       |  20    |  Da hoan thanh   | Chua thanh toan|
+========+==============+============================+=============+==================+========+==================+================+
|   10   |   23/10/2023 |     Thu Huong              |  1231231231 |   Tiec sinh nhat |  10    |  Sap dien ra     | Chua thanh toan|
+========+==============+============================+=============+==================+========+==================+================+
|   17   |   28/1 /2023 |     Minh Hoang             |  1231231231 |   Tiec cuoi      |  10    |  Sap dien ra     | Chua thanh toan|
+========+==============+============================+=============+==================+========+==================+================+
|   19   |   23/11/2030 |     Nguyen Van A           |  1233211231 |   Tiec sinh nhat |  30    |  Sap dien ra     | Chua thanh toan|
+========+==============+============================+=============+==================+========+==================+================+
|   20   |   20/11/2022 |     Tran Quoc Khanh        |  0784441234 |   Tiec tra       |  20    |  Da hoan thanh   | Chua thanh toan|
+========+==============+============================+=============+==================+========+==================+================+

                                Nhap ID tiec can xoa: 20

                        Ban co chac chan muon xoa? (y/n): y

                                </> Xoa thanh cong </>
```

Hình 4.11: Giao diện xoá tiệc

4.6. Chức năng thanh toán tiệc

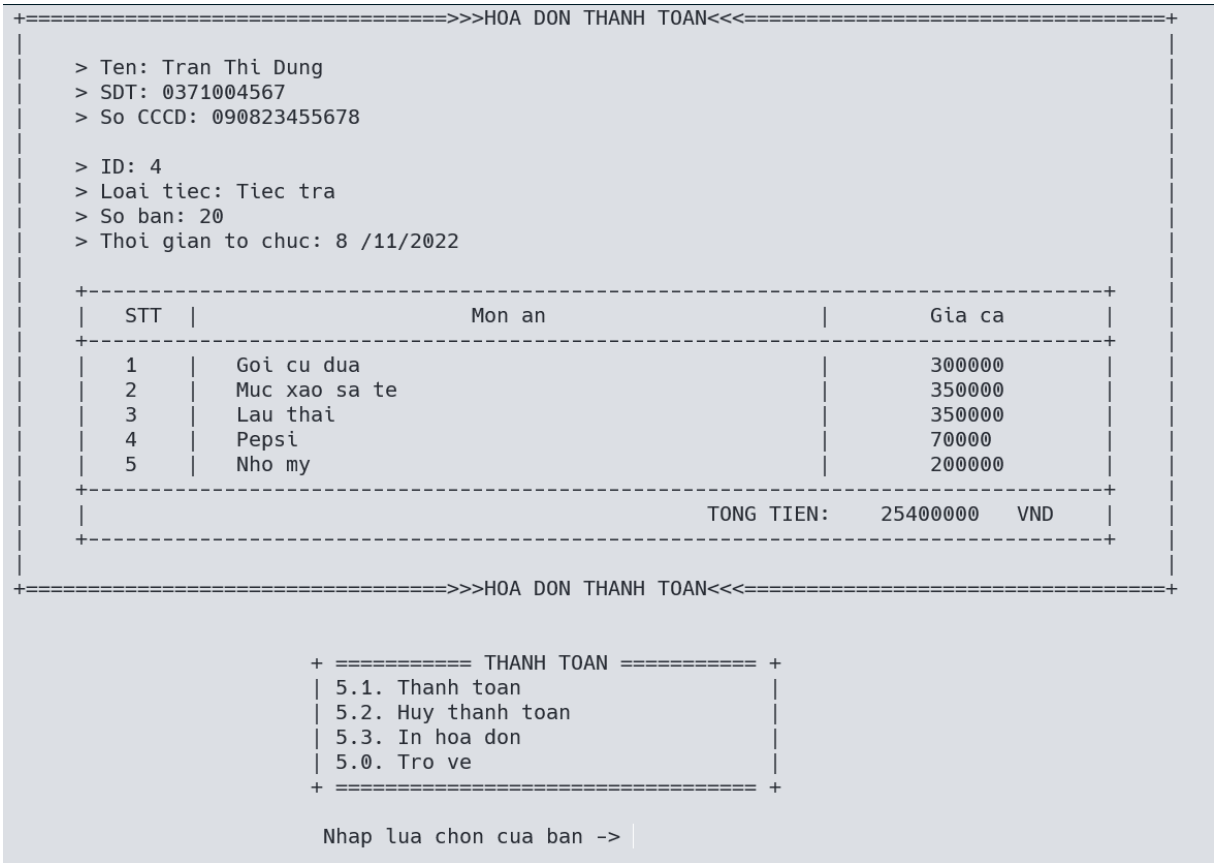
- Hiển thị danh sách các tiệc chưa thanh toán

- Nhập ID của tiệc cần thanh toán, kiểm tra xem có tồn tại loại tiệc chứa ID này không

- Hiển thị các lựa chọn của quá trình thanh toán

- Sau khi thanh toán thành công, thông tin của buổi tiệc đó tự động xóa khỏi cây nhị phân.

```
parties.displayByPaymentStatus(0);
cout << endl;
long _ID;
_ID = returnChoose("Nhap ID tiec can thanh toan: ");
cout << endl;
ItemP _party = parties.search(_ID);
if (parties.isPaid(_party.getID()) && _party.getID() != -1)
    cout << "\n\t\t\t\t\t\t(!) Tiec nay da thanh toan (!)";
else if (_party.getID() != -1)
    paymentDisplay(parties, _party);
else
    cout << "\n\t\t\t\t\t\t(!) Khong co ton tai tiec nay (!)";
pressAnyKey();
```



Hình 4.12: Giao diện thanh toán tiệc

4.7. Chức năng in hoá đơn

- Hiển thị danh sách thông tin các tiệc thuê.

- Tìm kiếm theo ID để nhận biết tiệc cần in hóa đơn.

- Hiển thị ra hóa đơn thanh toán.

```
parties.display();
cout << endl;
long _ID;
_ID = returnChoose("Nhap ID tiec can in bill: ");
ItemP _party = parties.search(_ID);
system("cls");
if (_party.getID() != -1)
    exportBillParty(_party);
else
    cout << "\n\t\t\t\t\t\t(!) Khong co ton tai tiec nay (!)";
pressAnyKey();
```

```
+==================================>>>HOA DON THANH TOAN<<<==================================+
|                                                                                             |
|                                                    24/11/2022   13:21                       |
|                                                                                             |
|    > Ten: Tran Thi Dung                                                                     |
|    > SDT: 0371004567                                                                        |
|    > So CCCD: 090823455678                                                                  |
|                                                                                             |
|    > ID: 4                                                                                  |
|    > Loai tiec: Tiec tra                                                                    |
|    > So ban: 20                                                                             |
|    > Thoi gian to chuc: 8 /11/2022                                                          |
|                                                                                             |
|    +-------------------------------------------------------------------------------+        |
|    |   STT  |                     Mon an                      |        Gia ca        |        |
|    +-------------------------------------------------------------------------------+        |
|    |   1    |   Goi cu dua                                    |        300000        |        |
|    |   2    |   Muc xao sa te                                 |        350000        |        |
|    |   3    |   Lau thai                                      |        350000        |        |
|    |   4    |   Pepsi                                         |        70000         |        |
|    |   5    |   Nho my                                        |        200000        |        |
|    +-------------------------------------------------------------------------------+        |
|    |                                              TONG TIEN:    25400000   VND    |        |
|    +-------------------------------------------------------------------------------+        |
|                                                                                             |
+==================================>>>HOA DON THANH TOAN<<<==================================+


        Hoa don cua ban duoc luu theo duong dan sau:

code -> data -> Hoa-Don -> 4_Tran Thi Dung.txt
```

Hình 4.13: Giao diện in hoá đơn

CHƯƠNG 5: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

5.1. Kết luận

5.1.1. Kết quả đạt được

Chương trình đã được phát triển thành công. Đảm bảo đầy đủ tính các tính năng cơ bản để quản lý tiệc một cách hiệu quả, dễ thao tác. Đặc biệt là đã áp đụng được đầy đủ 4 tính chất của OOP và ứng dụng được cấu trúc dữ liệu (danh sách liên kết đơn, liên kết đôi vòng, cây nhị phân tìm kiếm, hàng đợi) vào trong ứng dụng quản lý.

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

5.1.2. Kiến nghị

Do thời gian tim hiểu, phân tích thiết kế còn hạn hẹp nên chương trình vẫn còn một số hạn chế: đồ họa của chương trình vẫn là đang sử dụng đồ hoạ trên màn hình console.

5.2. Hướng phát triển

Xây dựng hoàn thiện chương trình, tìm hiểu thêm nhiều hơn về các chức năng trong quá trình quản lý để bổ sung vào trong chương trình. Khắc phục giao diện của chương trình.

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

TÀI LIỆU THAM KHẢO

[1] Glints, "Glints," [Online]. Available: https://glints.com/vn/blog/lap-trinh-cpp-la-gi/. [Accessed 15 11 2022].

[2] DAISY, "tuhoclaptrinh," [Online]. Available: https://tuhoclaptrinh.edu.vn/bai-viet/tim-hieu-ve-ngon-ngu-lap-trinh-c-422.html. [Accessed 20 11 2022].

[3] L. N. H. Quân, "FPTShop," [Online]. Available: https://fptshop.com.vn/tin-tuc/danh-gia/visual-studio-code-la-gi-cac-tinh-nang-noi-bat-cua-visual-studio-code-146213. [Accessed 13 11 2022].

[4] TopDev, "TopDev," [Online]. Available: https://topdev.vn/blog/github-la-gi/. [Accessed 21 11 2022].

[5] itviecBlog, "itviecBlog," [Online]. Available: https://itviec.com/blog/oop-la-gi/. [Accessed 22 11 2022].

[6] D. Xuân, "CafeDev," [Online]. Available: https://cafedev.vn/cau-truc-du-lieu-va-giai-thuat-la-gi-tai-sao-no-lai-quan-trong-voi-dan-lap-trinh/. [Accessed 23 11 2022].

[7] K. Lê, "Khiem Le," [Online]. Available: https://www.khiemle.dev/blog/danh-sach-lien-ket-don. [Accessed 23 11 2022].