TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI

PHÂN HIỆU TẠI TP. HỒ CHÍ MINH

BỘ MÔN CÔNG NGHỆ THÔNG TIN

BÁO CÁO BÀI TẬP LỚN

LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

ĐỀ TÀI: ỨNG DỤNG KỸ THUẬT LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG TRONG LẬP TRÌNH GAME VỚI THƯ VIỆN SDL2

Giảng viên hướng dẫn: ThS. TRẦN THỊ DUNG

Nhóm sinh viên thực hiện: NGUYỄN THÀNH TRÍ - 6451070179

TRẦN NGỌC BIÊN - 6451071004

TRẦN VĂN MINH TÚ - 6451071083

Lớp : CÔNG NGHỆ THÔNG TIN

Khoá : 64

Tp. Hồ Chí Minh, năm 2024

TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI

PHÂN HIỆU TẠI TP. HỒ CHÍ MINH

BỘ MÔN CÔNG NGHỆ THÔNG TIN

BÁO CÁO BÀI TẬP LỚN

LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

ĐỀ TÀI: ỨNG DỤNG KỸ THUẬT LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG TRONG LẬP TRÌNH GAME VỚI THƯ VIỆN SDL2

Giảng viên hướng dẫn: ThS. TRẦN THỊ DUNG

Nhóm sinh viên thực hiện: NGUYỄN THÀNH TRÍ - 6451070179

TRẦN NGỌC BIÊN - 6451071004

TRẦN VĂN MINH TÚ - 6451071083

Lớp : CÔNG NGHỆ THÔNG TIN

Khoá : 64

Tp. Hồ Chí Minh, năm 2024

LỜI CẢM ƠN

Chúng em cảm ơn cô đã tạo cơ hội cho chúng em được học tập môn lập trình hướng đối tượng. Thông qua môn học, chúng em đã biết thêm nhiều kiến thức mới trong ngành lập trình và các ngành khác như thiết kế giao diện. Các kiến thức mà chúng em được học hôm nay sẽ giúp ích rất nhiều cho chúng em trong những giai đoạn về sau. Chúng em rất cảm ơn những lúc em không hiểu, cô luôn sẵn lòng giải đáp các thắc mắc của chúng em. Cảm ơn sự hướng dẫn tận tình của cô.

Bên cạnh đó, chúng em cũng gửi lời cảm ơn tới các bạn. Các bạn đã góp phần giúp tụi em hoàn thành được bài tập lớn này một cách suôn sẻ. Khi chúng em gặp khó khăn đều có các bạn nhiệt tình giúp đỡ, giúp chúng em có thể giải quyết những khó khăn ấy. Khi sản phẩm của chúng em sắp hoàn thành thì nhờ có các bạn với vai trò là tester giúp chúng em tìm ra các lỗi bug để sản phẩm được hoàn thiện nhất, đem lại cho người dùng được những trải nghiệm tốt nhất.

Cuối cùng chúng em xin cảm ơn sự nổ lực, kiên trì của các bạn trong nhóm. Lập trình game là một lĩnh vực rất khó. Mặc dù nó yêu cầu rất nhiều kiên thức về thiết kế giao diện, toán học, vật lý,… nhưng các bạn vẫn nổ lực cố gắng hoàn thành mục tiêu ban đầu đặt ra mà không suy nghĩ đến hướng khác.

Nhờ cô, các bạn và sự nổ lực của nhóm mà bài tập lớn “Lập trình game Mê cung huyền thoại” bằng kỹ thuật lập trình hướng đối tượng với ngôn ngữ C/C++ kết hợp với thư viện SDL2 mới có thể hoàn thành. Nhóm em chân thành cảm ơn sự hướng dẫn, giúp đỡ tận tình của cô và các bạn.

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

STT	Mô tả	Ý nghĩa	Ghi chú
1	VS2022	Visual studio 2022	Công cụ lập trình
2	SDL2	Simple DirectMedia Layer 2	Thư viện hỗ trợ
3	OOP	Lập trình hướng đối tượng	Object-Oriented Programming

BẢNG BIỂU, SƠ ĐỒ, HÌNH VẼ

CHƯƠNG 1: MỞ ĐẦU

I. Tổng quan về bài tập lớn.

Khi học môn lập trình hướng đối tượng nói chung và các môn khác, chúng em đều mong muốn tạo cho mình một dấu ấn riêng, một phong cách riêng và tự đặt ra cho mình một thử thách để thử thách bản thân. Sau khi họp bàn và thống nhất ý kiến, chúng em quyết định chọn đề tài lập trình game - một đề tài ít người chọn vì nó còn mới với tất cả mọi người và cũng là một đề tài khó. Đây cũng là một thử thách mà nhóm em đặt ra và muốn vượt qua để xem khả năng của bản thân. Việc lên ý tưởng cho game cũng gặp nhiều khó khăn vì chưa có nhiều kinh nghiệm. Sau khi tìm hiểu các nguồn thông tin trên mạng và chơi thử một số game thì chúng em chọn thể loại game platform là một thể loại game người chơi sẽ nhảy và tìm cách vượt qua các chướng ngại vật. Đây là một thể loại không quá khó, phù hợp với những người mới bắt đầu lập trình game như chúng em.

II. Mục tiêu nghiên cứu.

Áp dụng các kiến thức lý thuyết về OOP vào để hoàn thành bài tập lớn. Các kiến thức cần thiết như:

III. Phạm vi nghiên cứu.:

Các kiến thức về OOP đã được học ở trường và những kiến thức trong giáo trình.

IV. Cấu trúc bài báo cáo.

- Chương 1: Mở đầu
- Chương 2: Cơ sở lý thuyết
- Chương 3: Xây dựng chương trình
- Chương 4: Tổng kết

V. Giảng viên hướng dẫn:

- Giảng viên: Ths. Trần Thị Dung.

CHƯƠNG 2: CƠ SỞ LÝ THUYẾT

I. OOP là gì?

1. Quá trình phát triển của OOP

Những ngày đầu phát triển, các phần mềm đơn giản với vỏn vẹn vài chục dòng lệnh, chương trình được viết tuần tự với các câu lệnh được thực hiện từ đầu đến cuối. Cách viết chương trình như thế này được gọi là phương pháp lập trình tuyến tính. Sau này, các phần mềm đòi hỏi độ phức tạp cao hơn, lớn hơn rất nhiều, phương pháp lập trình tuyến tính không còn hiệu quả để quản lý những chương trình lớn.

Lúc này phương pháp lập trình cấu trúc ra đời để giải quyết vấn đề trên. Lập trình cấu trúc là áp dụng phương pháp chia, chia nhỏ chương trình thành nhiều chương trình con đơn giản. Các chương trình con này tương đối độc lập với nhau, do đó có thể phân công cho từng nhóm viết các chương trình con khác nhau. Sử dụng cấu trúc dữ liệu và giải thuật để giải quyết các bài toán nên chương trình của phương pháp lập trình cấu trúc có nhược điểm là phụ thuộc vào tổ chức dữ liệu, tức là giải thuật phụ thuộc vào cấu trúc dữ liệu. Nếu cấu trúc dữ liệu thay đổi thì giải thuật sẽ bị sai lệch hoặc không chạy được, phải viết lại chương trình để phù hợp với cấu trúc dữ liệu mới. Một phương pháp lập trình mới ra đời để khắc phục vấn đề này đó là lập trình hướng đối tượng (OOP).

2. Định nghĩa của lập trình hướng đối tượng (OOP)

Lập trình hướng đối tượng (OOP): là phương pháp lập trình thiết kế chương trình xoay quanh dữ liệu hệ thống. Nghĩa là lúc này các thao tác xử lý của hệ thống được gắn liền với dữ liệu và như vậy một sự thay đổi nhỏ của dữ liệu chỉ ảnh hưởng đến một số nhỏ các hàm sử lý liên quan. Sự gắn kết giữa dữ liệu và các hàm xử lý trên chúng tạo ra các đối tượng.

3. Các đặc tính của OOP

- Tập trung vào dữ liệu thay cho hàm.
- Chương trình được chia thành các đối tượng.
- Các cấu tríc dữ liệu được thiết kế sao cho đặc tả được đối tượng.
- Các hàm thao tác trên các vùng dữ liệu của đối tượng được gắn với cấu trúc dữ liệu đó. Ý nghĩa và ứng dụng của OOP

- Các hàm thao tác trên các vùng dữ liệu của đối tượng được gắn với cấu trúc dữ liệu đó.
- Dữ liệu được đóng gói lại, được che giấu và không cho phép các hàm ngoại lai truy cập tự do
- Các đối tượng tác động và trao đổi thông tin với nhau qua các hàm
- Có thể dễ dàng bổ sung dữ liệu và các hàm mới vào đối tượng nào đó khi cần thiết
- Chương trình được thiết kế theo cách tiếp cận từ dưới lên (bottom-up)

II. Các tính chất của OOP

- **Tính đóng gói (Encapsulation):** Trong lập trình hướng đối tượng, không những các chức năng được đóng gói mà cả dữ liệu cũng như vậy. Với mỗi đối tượng, người ta không thể truy cập trực tiếp vào các thành phần dữ liệu của nó mà phải thông qua các thành phần chức năng (Các phương thức) để làm việc đó.
- **Tính kế thừa (Inheritance):** Sự kế thừa cho phép chúng ta định nghĩa một lớp mới dựa trên cơ sở của các lớp đã tồn tại, tất nhiên có bổ sung những phương thức hay các thành phần dữ liệu mới. Khả năng kế thừa cho phép chúng ta sử dụng lại một cách dễ dàng các module chương trình mà không cần thay đổi các module đó. Đây là một điểm mạnh của lập trình hướng đối tượng so với lập trình hướng cấu trúc.
- **Tính đa hình (Polymorphism):** Tính đa hình giúp cho chương trình dễ quản lý hơn. Ví dụ như hình chữ nhật và hình thoi đề là hình tứ giác. Hai lớp này đề có đầy đủ các thuộc tính và tính chất của lớp tứ giác. Với mỗi hình ta có công thức tính diện tích khác nhau. Vậy thì khi gọi phép tính một tứ giác thì ta áp dụng đúng công thức với mỗi loại tứ giác để tính ra đúng kết quả thì ta gọi đó là tính đa hình.
- **Tính trừu tượng (Abstraction):** Ẩn đi các chi tiết phức tạp bên trong và chỉ cung cấp các tính năng cần thiêt ra bên ngoài, giúp giảm độ phức tạp của chương trình và tập trung vào cách sử dụng thay vì cách hoạt động bên trong.

Các tính chất trên tạo nên sự linh hoạt và hiệu quả trong việc quản lý các chương trình lớn và phức tạp bằng OOP. Nhờ đó lập trình hướng đối tượng có các ưu điểm:

- Thông qua nguyên lý kế thừa chúng ta có thể loại bỏ được những đoạn chương trình lặp lại tỏng quá trình mô tả các lớp và có thể mở rộng khả năng sử dụng các lớp đã xây dựng mà không cần phải viết lại.

- Chương trình được xây dựng từ những đơn thể (đối tượng) trao đổi với nhau nên việc thiết kế và lập trình sẽ được thực hiện theo quy trình nhất định chứ không phải dựa vào kinh nghiệm và kỹ thuật như trước. Điều này đảm bảo rút ngắn được thời gian xây dựng hệ thống và tăng năng suất lao động.
- Nguyên lý đóng gói hay che giấu thông tin giúp người lập trình tạo được những chương trình an toàn không bị thay đổi bởi những đoạn chương trình khác,
- Có thể xây dựng ánh xạ các đối tượng của bài toán vào đối tượng chương trình.
- Cách tiếp cận thiết kế đặt trọng tâm vào dữ liệu, giúp chúg ta xây dựng được mô hình chi tiết và dễ dàng cài đặt hơn.
- Các hệ thống hướng đối tượng dễ mở rộng, nâng cấp thành những hệt thống lớn hơn.
- Kỹ thuật truyền thông báo trong việc trao đổi thông tin giữa các đối tượng làm cho việc mô tả giao diện với các hệ thống bên ngoài trở nên đơn giản hơn.

Những ứng dụng của lập trình hướng đối tượng: Lập trình hướng đối tượng là một trong những thuật ngữ được nhắc đến nhiều nhất hiện nay trong công nghệ phần mềm và nó được ứng dụng trong nhiều lĩnh vực khác nhau. Trong số đó, ứng dụng quan trọng và nổi tiếng nhất hiện nay là thiết kế giao diện với người sử dụng, kiểu như Windows. Các hệ thông tin quản lý trong thực tế thường rất phức tạp, chứa nhiều đối tượng với các thuộc tính và hàm phức tạp. Lập trình hướng đối tượng được áp dụng một cách hiệu quả với các hệ thống như thế. Các lĩnh vực ứng dụng phù hợp với kỹ thuật lập trình hướng đối tượng có thể liệt kê:

- Các hệ thống làm việc theo thời gian thực.
- Các hệ mô hình hóa hoặc mô phỏng các quá trình
- Các hệ cơ sở dữ liệu hướng đối tượng.
- Các hệ siêu văn bản (hypertext), đa phương tiện (multimedia)
- Các hệ thống trí tuệ nhân tạo, và các hệ chuyên gia.
- Các hệ thống song song và mạng nơ rono
- Các hệ tự động hóa văn phòng hoặc trợ giúp quyết định
- Các hệ CAD/CAM.

III. Sử dụng OOP trong lập trình

1. Định nghĩa class

Định nghĩa lớp (class): lớp là một kiểu dữ liệu tự định nghĩa. Trong lập trình hướng đối tượng, chương trình nguồn được phân bố trong khai báo và định nghĩa của các lớp.

```
class NameOfClass
{
private:
    // Khai báo các thành phần riêng tư của đối tượng
public:
    // Khai báo các thành phần công khai của đối tượng
};
// định nghĩa các hàm của đối tượng
```

Hình 2.1 cấu trúc chung của class

- Cú pháp:
  + NameOfClass: tên của class
  + Private, public: phạm vi truy cập

2. Tầm vực và con trỏ

- Trong định nghĩa của lớp, ta có thể xác định khả năng truy cập thành phần của một lớp nào đó từ bên ngoài lớp. Các từ khóa private, public, protected dùng để xác định phạm vi truy xuất của các thuộc tính hoặc phương thức được khai báo. Mọi thành phần được khai báo trong public có thể truy xuất trong bất kỳ hàm nào. Những thành phần được liệt kê trong phần private chỉ được truy xuất bên trong phạm vi lớp hiện tại, bởi chúng thuộc sở hữu của lớp. Còn khi khai báo ở phạm vi protected thì thuộc tính chỉ có thể được truy xuất ở lớp hiện tại và các lớp kế thừa.
- Con trỏ this trong định nghĩa các hàm thành phần của lớp dùng để xác định địa chỉ của đối tượng dùng làm tham số ngầm định cho hàm thành phần. Nói cách khác, con trỏ this tham chiếu đến đối tượng đang gọi hàm thành phần. Như vậy, có thể truy cập đến các thành phần của đối tượng gọi hàm thành phần của đối tượng gọi hàm thành phần gián tiếp thông qua this.

3. Hàm thiết lập và hàm hủy

- Hàm thiết lập: là một hàm thành phần đặc biệt không thể thiếu được trong một lớp. Nó được tự động gọi mỗi khi có đối tượng được khai báo. Chức năng của hàm thiết lập là khởi tạo các giái trị thành phần , xin cấp bộ nhớ động cho các thành phần dữ liệu động. Một số đặc điểm của hàm thiêt lập:
  + Hàm thiết lập phải có cùng tên với lớp và phải là thuộc tính public
  + Hàm thiết lập không có giá trị trả về. Và không cần khai báo void
  + Có thể có nhiều hàm thiết lập trong cùng lớp (nạp chồng hàm thiết lập)
  + Hàm thiết lập có thể được khai báo với các tham số có giá trị ngầm định.
  + Hàm thiết lập ngầm định: là hàm thiết lập do chương trình dịch cung cấp trong khai báo lớp không có định nghĩa hàm thiết lập nào. Đôi khi người ta gọi hàm thiết lập không tham số do người dùng tự định nghĩa là hàm thiết lập ngầm định. Cần phải có hàm thiết lập ngầm định khi khai báo một mảng các đối tượng. Nếu trong lớp chỉ có hàm thiết lập có tham số, khi ta khai báo 1 mảng các đối tượng chương trình sẽ bị lỗi vì không cung cấp đủ tham số cho hàm thiết lập. Giải pháp để giải quyết vấn đề là định nghĩa thêm 1 hàm thiết lập không tham số hoặc là xóa bỏ hàm thiết lập có tham số. Hoặc có thể khai báo giá trị ngầm định cho các tham số của hàm thiết lập có tham số.
  + Con trỏ đối tượng: khi dùng toán tử new để cấp phát động cho một đối tượng động, hàm thiết lập cũng được gọi nên cần cung cấp danh sách các tham số.
- Hàm hủy: ngược lại với hàm thiết lập, hàm hủy được gọi khi đối tượng tương ứng bị xóa khỏi bộ nhớ. Một số quy định đối với hàm hủy
  + Tên của hàm hủy bỏ bắt đầu bằng dấu ~ và theo sau là tên của lớp.
  + Hàm hủy bỏ phải có thuộc tính public
  + Nói chung hàm hủy bỏ không có tham số, mỗi lớp chỉ có một hàm hủy bỏ.
  + Khi không định nghĩa hàm huỷ bỏ, chương trình dịch tự động sản sinh một hàm như vậy (hàm huỷ bỏ ngầm định), hàm này không làm gì ngoài việc “lấp chỗ trống”. Đối với các lớp không có khai báo các thành phần bộ nhớ động, có thể dùng hàm huỷ bỏ ngầm định. Trái lại, phải khai báo hằm huỷ bỏ tường minh để đảm bảo quản lý tốt việc

giải phóng bộ nhớ động do các đối tượng chiếm giữ chiếm giữ khi chúng hết thời gian làm việc.

+ Giống như hàm thiết lập, hàm huỷ bỏ không có giá trị trả về.

- Trên thực tế, với các lớp không có các thành phần dữ liệu động chỉ cần sử dụng hàm thiết lập và huỷ bỏ ngâm định là đủ. Hàm thiết lập và huỷ bỏ do người lập trình tạo ra rất cần thiết khi các lớp chứa các thành phần dữ liệu động. Khi tạo đối tượng hàm thiết lập đã xin cấp phát một khối bộ nhớ động, do đó hàm huỷ bỏ phải giải phóng vùng nhớ đã được cấp phát trước đó.

```
class example
{
private:
    int n;
    float f;
public:
    example();                  // hàm thiết lập không có tham số
    example(int n_) {n = n_;};  // hàm thiết lập có tham số
    ~example();                 // Hàm hủy
};
```

Hình 2.2 ví dụ về hàm thiết đặt và hàm hủy

4. Các thành phần tĩnh (static)

Có thể cho phép nhiều đối tượng cùng chia sẻ dữ liệu bằng cách đặt từ khoá static trước khai báo thành phần dữ liệu tương ứng

```
class example
{
private:
    static int n;
    float f;
public:
    // Khai báo các thành phần công khai của đối tượng
};
```

- Khi khai báo

```
example a, b;
```

- Chương trình sẽ tạo ra hai đối tượng có chung thành phần n (thuộc tính n có chung địa chỉ ô nhớ)
- Như vậy, việc chỉ định static đối với một thành phần dữ liệu có ý nghĩa là int trong toàn bộ lớp, chỉ có một thể hiện duy nhất của thành phần đó. Thành phần static được dùng

chung cho tất cả các đối tượng của lớp đó và do đó vẫn chiếm giữ vùng nhớ ngay cả khi không khai báo bất kỳ đối tượng nào.

- Khởi tạo các thành phần static: Các thành phần dữ liệu static chỉ có một phiên bản trong tất cả các đối tượng. Như vậy không thể khởi tạo chúng bằng các hàm thiết lập của một lớp. Cũng không thể khởi tạo lúc khai báo các thành phần dữ liệu static.
- Một thành phần dữ liệu static phải được khởi tạo một cách tường minh bên ngoài khai báo lớp bằng một chỉ thị như sau:

```
int example::n = 5;
```

Hình 2.3 Khởi tạo giá trị cho biến static

- Trong C++ việc khởi tạo giá trị như thế này không vi phạm tính riêng tư của các đối tượng. Chú ý rằng cần phải có tên lớp và toán tử phạm vi để chỉ định các thành phần của lớp được khởi tạo.
- Biến static thường dùng để đếm số lượng đối tượng hiện đang sử dụng.

5. Hàm bạn và lớp bạn

Hàm bạn trong c++ là hàm tự do, không thuộc lớp. Tuy nhiên hàm bạn trong c++ có quyền truy cập các thành viên private của lớp.

- Khai báo hàm bạn:

```
class person{
public:
    friend void display(person abc);
};
void display(person abc) // hàm bạn
{
    // code
}
```

Hình 2.4 Khai báo hàm bạn

Tương tự như hàm bạn, lớp bạn (friend class) trong C++ cũng cho phép lớp bạn của lớp kia truy cập các thành viên private

```
class beta; //forward declaration
class alpha{
private:
    int a_data;
public:
    alpha(){a_data = 10;}
    void display(beta);
};
```

Hình 2.5 Khai báo hàm lớp bạn

6. Nạp chồng toán tử

Trong C++, có thể định nghĩa chồng đối với hầu hết các phép toán (một ngôi hoặc hai ngôi) trên các lớp, nghĩa là một trong số các toán hạng tham gia phép toán là các đối tượng. Đây là một khả năng mạnh vì nó cho phép xây dựng trên các lớp các toán tử cần thiết, làm cho chương trình được viết ngắn gọn dễ đọc hơn và có ý nghĩa hơn. Chẳng hạn, khi định nghĩa một lớp complex để biểu diễn các số phức, có thể viết trong C++: a+b, a-b, a*b, a/b với a, b là các đối tượng complex.

Để có được điều này, ta định nghĩa chồng các phép toán +, -, * và / bằng cách định nghĩa hoạt động của từng phép toán giống như định nghĩa một hàm, chỉ khác là đây là hàm toán tử (operator function). Hàm toán tử có tên được ghép bởi từ khoá operator và ký hiệu của phép toán tương ứng. Hàm toán tử có thể dùng như là một hàm thành phần của một lớp hoặc là hàm tự do; khi đó hàm toán tử phải được khai báo là bạn của các lớp có các đối tượng mà hàm thao tác.

Có 2 loại hàm toán tử:

- Toán tử độc lập.
- Toán tử thuộc lớp.

```
class SoPhuc{
private:
    int a;
    int b;
public:
    SoPhuc();
    SoPhuc(int a, int b);
    SoPhuc(const SoPhuc &sp);
    void setSoPhuc(int a, int b);
    int getReal();
    int getImag();
    float module();
    SoPhuc operator+(SoPhuc x);
    SoPhuc operator-(SoPhuc x);
    SoPhuc operator*(SoPhuc x);
    SoPhuc operator/(SoPhuc x);
    int operator==(SoPhuc x);
    int operator < (SoPhuc x);
};
```

Hình 2.6 Khai báo hàm nạp chồng toán tử

Ràng buộc:

- Ngôi của toán tử giữ nguyên.
- Độ ưu tiên của toán tử không đổi.
- Không thể tạo toán tử mới.
- Không thể định nghĩa lại toán tử cho kiểu cơ bản.

Định nghĩa các hàm nạp chồng toán tử

```
SoPhuc SoPhuc:: operator +(SoPhuc x){
    SoPhuc n;
    n.a = a + x.a;
    n.b = b + x.b;
    return n;
}
SoPhuc SoPhuc::operator-(SoPhuc x){
    SoPhuc n;
    n.a = a - x.a;
    n.b = b - x.b;
    return n;
}
SoPhuc SoPhuc::operator*(SoPhuc x){
    SoPhuc n;
    n.a = a*x.a - b*x.b;
    n.b = a*x.b + b*x.a;
    return n;
}
SoPhuc SoPhuc::operator/(SoPhuc x){
    SoPhuc n;
    n.a = (a*x.a + b*x.b)/(a*a + b*b);
    n.b = (a*x.b + b*x.a)/(a*a + b*b);
    return n;
}
 int SoPhuc::operator==(SoPhuc x){
    int tmp = this->module() - x.module();
    if (tmp == 0) return 0;
    if (tmp < 0) return -1;
    if (tmp > 0) return 1;
}
int SoPhuc::operator<(SoPhuc x){
    if (this->module() < x.module())
        return 1;
    else if (this->module() > x.module())
        return -1;
    else
        return 0;
}
```

Hình 2.7 Định nghĩa hàm nạp chồng toán tử

7. Kỹ thuật thừa kế

Thừa kế là một trong bốn nguyên tắc cơ sở của phương pháp lập trình hướng đối tượng. Đặc biệt đây là cơ sở cho việc nâng cao khả năng sử dụng lại các bộ phận của chương trình. Thừa kế cho phép ta định nghĩa một lớp mới, gọi là lớp dẫn xuất, từ một lớp đã có, gọi là lớp cơ sở. Lớp dẫn xuất sẽ thừa kế các thành phần (dữ liệu, hàm) của lớp cơ sở, đồng thời thêm vào các thành phần mới, bao hàm cả việc làm "tốt hơn" hoặc làm lại những công việc mà trong lớp cơ sở chưa làm tốt hoặc không còn phù hợp với lớp dẫn xuất. Chẳng hạn có thể định nghĩa lớp "mặt hàng nhập khẩu" dựa trên lớp "mặt hàng", bằng cách bổ sung thêm thuộc tính "thuế". Khi đó cách tính chênh lệch giá bán, mua cũ trong lớp "mặt hàng" sẽ không phù hợp nữa nên cần phải sửa lại cho phù hợp. Lớp điểm có màu được định nghĩa dựa trên lớp điểm không màu bằng cách bổ sung thêm thuộc tính màu, hàm display() lúc này ngoài việc hiển thị hai thành phần toạ độ còn phải cho biết màu của đối tượng điểm. Trong cả hai ví dụ đưa ra, trong lớp dẫn xuất đều có sự bổ sung và thay đổi thích hợp với tình hình mới.

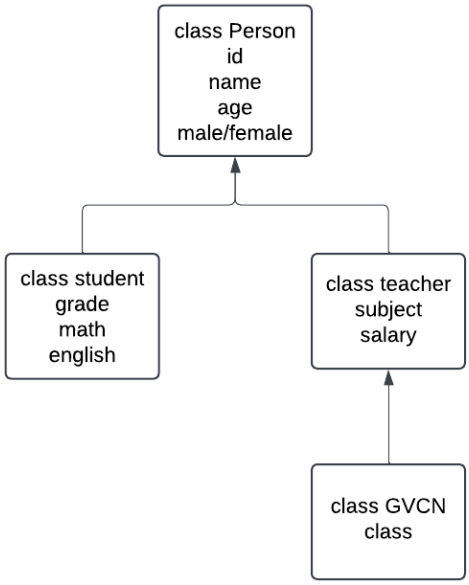
Thừa kế cho phép không cần phải biên dịch lại các thành phần chương trình vốn đã có trong các lớp cơ sở và hơn thế nữa không cần phải có chương trình nguồn tương ứng. Kỹ thuật này cho phép chúng ta phát triển các công cụ mới dựa trên những gì đã có được. Người sử dụng Borland C hay Turbo Pascal 6.0/7.0 rất thích sử dụng Turbo Vision - một thư viện cung cấp các lớp, đối tượng là cơ sở để xây dựng các giao diện ứng dụng hết sức thân thiện đối với người sử dụng. Tất cả các lớp này đều được cung cấp dưới dạng các tập tin *.obj, *.lib nghĩa là người sử dụng hoàn toàn không thể và không cần phải biết rõ phần chương trình nguồn tương ứng. Tuy nhiên điều đó không quan trọng khi người lập trình được phép thừa kế các lớp định nghĩa trước đó.

Thừa kế cũng cho phép nhiều lớp có thể dẫn xuất từ cùng một lớp cơ sở, nhưng không chỉ giới hạn ở một mức: một lớp dẫn xuất có thể là lớp cơ sở cho các lớp dẫn xuất khác. Ở đây ta thấy rằng khái niệm thừa kế giống như công cụ cho phép mô tả cụ thể hoá các khái niệm theo nghĩa: lớp dẫn xuất là một cụ thể hoá hơn nữa của lớp cơ sở và nếu bỏ đi các dị biệt trong các lớp dẫn xuất sẽ chỉ còn các đặc điểm chung nằm trong lớp cơ sở.

Tính đa hình cũng là một trong các điểm lý thú trong lập trình hướng đối tượng, được thiết lập trên cơ sở thừa kế trong đó đối tượng có thể có biểu hiện khác nhau tuỳ thuộc vào tình huống cụ thể. Tính đa hình ấy có thể xảy ra ở một hành vi của đối tượng hay trong toàn bộ đối tượng. Ví dụ trực quan thể hiện tính đa hình là một ti vi có thể vừa là đối tượng của mặt hàng vừa là đối tượng của lớp mặt hàng điện tử dân dụng. Các đối tượng hình học như

hình vuông, hình tròn, hình chữ nhật đều có cùng cách vẽ như nhau: xác định hai điểm đầu và cuối, nối hai điểm này. Do vậy thuật toán tuy giống nhau đối với tất cả các đối tượng hình, nhưng cách vẽ thì phụ thuộc vào từng lớp đối tượng cụ thể. Ta nói phương thức nối điểm của các đối tượng hình học có tính đa hình. Tính đa hình còn được thể hiện trong cách thức hiển thị thông tin trong các đối tượng điểm màu/không màu.

Các ngôn ngữ lập trình hướng đối tượng đều cho phép đa thừa kế, theo đó một lớp có thể là dẫn xuất của nhiều lớp khác. Do vậy dẫn tới khả năng một lớp cơ sở có thể được thừa kế nhiều lần trong một lớp dẫn xuất khác, ta gọi đó là sự xung đột thừa kế. Điều này hoàn toàn không hay, cần phải tránh. Từng ngôn ngữ sẽ có những giải pháp của riêng mình, C++ đưa ra khái niệm thừa kế ảo.



Hình 2.8 Sơ đồ kế thừa

- Tầm vực trong kế thừa
  + Dẫn xuất public: các thành phần, các hàm bạn và các đối tượng của lớp dẫn xuất không thể truy nhập đến các thành phần private của lớp cơ sở. Các thành phần protected trong lớp cơ sở trở thành các thành phần private trong lớp dẫn xuất. các thành phần public của lớp cơ sở vẫn là public trong lớp dẫn xuất.
  + Dẫn xuất private: các thành phần public trong lớp cơ sở không thể truy nhập được từ các đối tượng lớp dẫn xuất, nghĩa là chúng trở thành các thành phần private trong lớp dẫn xuất. các thành phần protected trong lớp cơ sở có thể truy nhập được từ các hàm thành phần và các hàm bạn của lớp dẫn xuất. Dẫn xuất private được sử dụng trong các

tình huống đặc biệt khi lớp dẫn xuất không khai báo thêm các thành phần hàm mới mà chỉ định nghĩa lại các phương thức đã có trong lớp cơ sở.

+ Dẫn xuất protected: trong dẫn xuất loại này, các thành phần public, protected trong lớp cơ sở trở thành các thành phần protected trong lớp dẫn xuất.

❖ Quan hệ IS-A và HAS-A:

- IS-A: A là trường hợp đặc biệt của B => A kế thừa B.
- HAS-A: A bao hàm B => B là thuộc tính của A.

8. Tính đa hình

Thực tế có nhiều vấn đề khi xác định phương thức hay hành động cụ thể tuỳ thuộc vào thời điểm tham chiếu đối tượng, chẳng hạn khi vẽ các đối tượng hình chữ nhật, hình vuông, hình tròn, tất cả đều gọi tới phương thức vẽ được thừa kế từ lớp tổng quát hơn (lớp hình vẽ). Trong định nghĩa của phương thức đó có lời gọi đến một phương thức khác là nối điểm chỉ được xác định một cách tường minh trong từng đối tượng cụ thể là hình chữ nhật, hình vuông hay hình tròn.

Để gọi được phương thức tương ứng với đối tượng được trỏ đến, cần phải xác định được kiểu của đối tượng được xem xét tại thời điểm thực hiện chương trình bởi lễ kiểu của đối tượng được chỉ định bởi cùng một con trỏ có thể thay đổi trong quá trình thực hiện của chương trình. Ta gọi đó là “gán kiểu động - dynamic typing" hay “gán kiểu muộn - late binding", nghĩa là xác định hàm thành phần nào tương ứng với một lời gọi hàm thành phần từ con trỏ đối tượng phụ thuộc cụ thể vào đối tượng mà con trỏ đang chứa địa chỉ. Khái niệm hàm ảo được C++ đưa ra nhằm đáp ứng nhu cầu này.

- Không nhất thiết phải định nghĩa lại hàm virtual: trong trường hợp tổng quát, ta luôn phải định nghĩa lại ở trong các lớp dẫn xuất các phương thức đã được khai báo là virtual trong lớp cơ sở. trong một số trường hợp, không nhất thiết phải làm như vậy.
- Định nghĩa chồng hàm ảo: có thể định nghĩa chồng hàm ảo và các hàm định nghĩa chồng như thế có thể không còn là hàm ảo nữa. Hơn nữa, nếu ta định nghĩa một hàm ảo trong một lớp và lại định nghĩa chồng nó trong một lớp dẫn xuất với các tham số khác thì có thể xem hàm định nghĩa chồng đó là một hàm hoàn toàn khác không liên quan gì đến hàm ảo hiện tại, nghĩa là nếu nó không được khai báo virtual thì nó có tính chất “gán kiểu tĩnh –

static typing". Nói chung, nếu định nghĩa chồng hàm ảo thì tất cả các hàm định nghĩa chồng của nó nên được khai báo là virtual để việc xác định lời gọi hàm đơn giản hơn.

- Hàm hủy bỏ ảo: Hàm thiết lập không thể là hàm ảo, trong khi đó hàm huỷ bỏ lại có thể. Ta quan sát sự cố xảy ra khi sử dụng tính đa hình để xử lý các đối tượng của các lớp trong sơ đồ thừa kế được cấp phát động. Nếu mỗi đối tượng được dọn dẹp tường minh nhờ sử dụng toán tử delete cho con trỏ lớp cơ sở chỉ đến đối tượng thì hàm huỷ bỏ của lớp cơ sở sẽ được gọi mà không cần biết kiểu của đối tượng đang được xử lý, cũng như tên hàm huỷ bỏ của lớp tương ứng với đối tượng (tuy có thể khác với hàm huỷ bỏ của lớp cơ sở).
- Một giải pháp đơn giản cho vấn đề này là khai báo hàm huỷ bỏ của lớp cơ sở là hàm ảo, làm cho các hàm huỷ bỏ của các lớp dẫn xuất là ảo mà không yêu cầu chúng phải có cùng tên.

CHƯƠNG 3: THƯ VIỆN SDL2

I. Tổng quan về thư viện SDL2

SDL2 (Simple DirectMedia Layer) là một thư viện đa nền tảng mã nguồn mở, được sử dụng rộng rãi trong lập trình game, các ứng dụng đa phương tiện và các phần mềm tương tác đồ họa khác. SDL2 giúp đơn giản hóa việc truy cập các chức năng cơ bản của hệ điều hành, hỗ trợ cho việc xử lý đồ họa, âm thanh, sự kiện bàn phím, chuột, và các thiết bị ngoại vi khác.

Ưu điểm của SDL2:

- Đa nền tảng: SDL2 hỗ trợ nhiều hệ điều hành như Windows, macOS, Linux, iOS, và Android, giúp lập trình viên dễ dàng chuyển đổi phần mềm giữa các nền tảng.
- Hiệu suất cao: SDL2 tối ưu hóa hiệu suất với sự hỗ trợ phần cứng, đặc biệt là khi kết hợp với các API đồ họa như OpenGL và Vulkan.
- Dễ sử dụng: SDL2 có cú pháp đơn giản, thân thiện và dễ học, phù hợp cho cả người mới bắt đầu và lập trình viên chuyên nghiệp.

Ứng dụng của SDL2: SDL2 chủ yếu được dùng trong:

- Phát triển game 2D: SDL2 thường được sử dụng để tạo các game 2D trên nhiều nền tảng khác nhau.
- Ứng dụng đa phương tiện: Nhờ khả năng xử lý âm thanh, video, và các loại hình ảnh, SDL2 phù hợp cho các ứng dụng liên quan đến xử lý đa phương tiện.
- Phát triển công cụ mô phỏng: SDL2 có thể được sử dụng trong các ứng dụng mô phỏng và hình ảnh động nhờ vào khả năng tương tác và render đồ họa.

Một trong những hàm thường dùng trong SDL2:

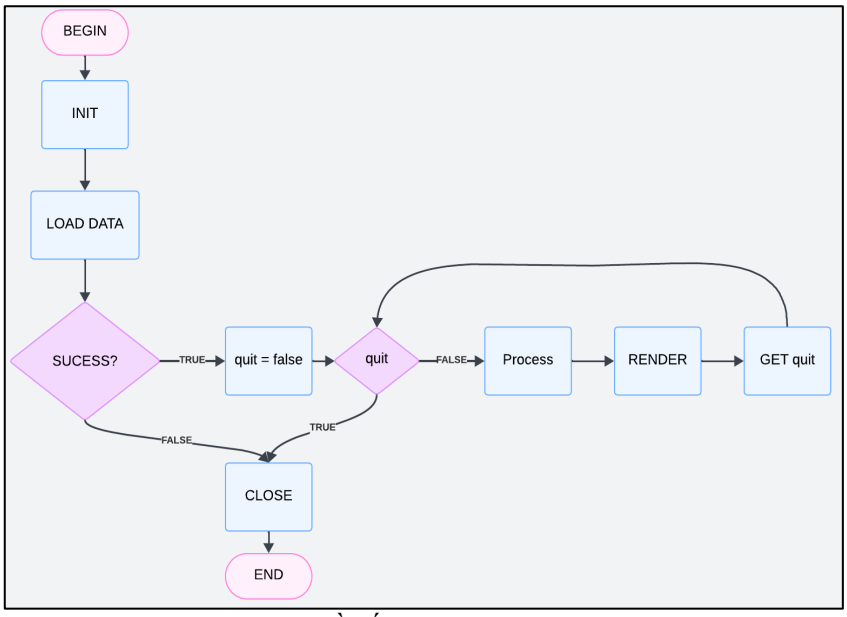
- SDL_Init(): hàm khởi tạo các hệ thống con của SDL (video, audio,…)
- SDL_Quit(): giải phóng tất cả tài nguyên
- SDL_CreateRenderer(): trả về con trỏ kiểu SDL_Renderer*
- SDL_RenderClear(): xóa màn hình với màu đã thiết lập
- SDL_RenderCopy(): vẽ các texture lên màn hình
- SDL_RenderPresent(): hiển thị nội dung đã vẽ lên màn hình

II. Cấu trúc cơ bản của việc lập trình game với SDL2

Cách sử dụng cơ bản: việc sử dụng SDL2 yêu cầu cài đặt và liên kết thư viện với dự án, sau đó khởi tạo các thành phần như cửa sổ, renderer, và vòng lặp game. Một chương trình SDL2 cơ bản thường bao gồm các bước:

- Khởi tạo SDL2 bằng cách gọi SDL_Init() để bắt đầu các thành phần cần thiết.
- Tạo cửa sổ (SDL_CreateWindow()) và renderer (SDL_CreateRenderer()).
- Vòng lặp chính để kiểm tra các sự kiện (như bàn phím và chuột) và thực hiện render.
- Giải phóng tài nguyên bằng cách gọi SDL_DestroyRenderer(), SDL_DestroyWindow() và SDL_Quit().

Cấu trúc chung của chương trình sử dụng thư viện SDL2:



Hình 3.1 Sơ đồ cấu trúc chương trình SDL2

Bên cạnh đó, còn có các thư viện hỗ trợ mở rộng

- SDL_image: thư viện quản lý việc đọc và render các file hình ảnh có định dạng phổ biến như .png, .jepg,…
- SDL_mixer: Thư viện quản lý phát âm thanh đa kênh.
- SDL_ttf: Thư viện hỗ trợ các font chữ.

CHƯƠNG 4: CÔNG CỤ QUẢN LÝ

I. Jira với quy trình Scrum

1. Quy trình Scrum là gì?

Scrum là một phương pháp quản lý dự án trong lĩnh vực phát triển phần mềm, nằm trong khung làm việc Agile. Scrum được thiết kế để giúp các đội ngũ phát triển sản phẩm có thể linh hoạt, dễ dàng điều chỉnh và cải tiến dự án theo từng giai đoạn ngắn, liên tục, gọi là Sprint. Mỗi Sprint thường kéo dài từ 1 đến 4 tuần và tạo ra một sản phẩm hoàn chỉnh, có thể sử dụng được, giúp đội ngũ có thể điều chỉnh theo phản hồi của người dùng và yêu cầu kinh doanh một cách nhanh chóng.

Scrum có ba thành phần chính: **Artifacts** (các tài liệu hoặc sản phẩm), **Events** (các sự kiện), và **Roles** (các vai trò). Các thành phần này phối hợp với nhau để đảm bảo tiến trình và chất lượng của dự án.

- **Product Backlog:** Đây là danh sách các yêu cầu, tính năng và công việc cần làm để hoàn thành sản phẩm. Product Backlog được sắp xếp theo thứ tự ưu tiên, các công việc quan trọng hơn sẽ được làm trước. Product Backlog được Product Owner cập nhật và quản lý liên tục.
- **Sprint Backlog:** Đây là danh sách công việc được chọn từ Product Backlog để thực hiện trong mỗi Sprint. Development Team sẽ chịu trách nhiệm hoàn thành tất cả các công việc trong Sprint Backlog.
- **Increment:** Đây là phiên bản cập nhật của sản phẩm sau mỗi Sprint, có thể sử dụng được và đã hoàn thiện các tính năng mới. Increment giúp Product Owner và các bên liên quan thấy được sự tiến bộ của dự án

Quy trình Scrum bao gồm các sự kiện chính diễn ra theo một chu kỳ lặp lại (Sprint):

- **Sprint Planning (Lập kế hoạch Sprint):** Đội Scrum, bao gồm Product Owner, Scrum Master và Development Team, thảo luận và chọn các mục tiêu từ Product Backlog để đưa vào Sprint Backlog. Mục tiêu của Sprint Planning là xác định những công việc nào cần làm và cách hoàn thành chúng.

- **Daily Scrum:** Là cuộc họp hằng ngày của Development Team, thường kéo dài khoảng 15 phút, nhằm theo dõi tiến độ, chia sẻ thông tin và xác định các trở ngại. Các thành viên thường trả lời ba câu hỏi:
  + Hôm qua đã làm gì?
  + Hôm nay sẽ làm gì?
  + Có gặp khó khăn gì không?
- **Sprint Review (Đánh giá Sprint):** Cuối mỗi Sprint, đội Scrum tổ chức cuộc họp với các bên liên quan để trình bày sản phẩm đã hoàn thành và nhận phản hồi. Sprint Review giúp Product Owner hiểu rõ hơn nhu cầu của người dùng và điều chỉnh Product Backlog.
- **Sprint Retrospective (Cuộc họp cải tiến)**: Đây là cuộc họp nội bộ của đội Scrum để đánh giá và cải tiến quá trình làm việc, nhằm tăng hiệu quả trong các Sprint tiếp theo. Đội Scrum thảo luận về những gì làm tốt, những gì chưa tốt và cách cải thiện trong tương lai.
- **Sprint:** Một chu kỳ phát triển, thường kéo dài từ 1 đến 4 tuần. Trong khoảng thời gian này, Development Team hoàn thành các công việc trong Sprint Backlog và tạo ra một phiên bản sản phẩm hoàn chỉnh.

2. Jira

Jira là một công cụ quản lý dự án và theo dõi công việc được phát triển bởi Atlassian. Nó thường được sử dụng trong các dự án phát triển phần mềm để theo dõi tiến độ công việc, quản lý các lỗi và hỗ trợ trong phương pháp Agile.

Jira giúp theo dõi tiến độ của từng hạng mục công việc trong dự án, từ các giai đoạn lập kế hoạch đến triển khai và hoàn thiện. Đối với dự án lập trình hướng đối tượng: “Lập trình game Mê cung huyền thoại”, Jira có thể được sử dụng để phân chia và theo dõi các nhiệm vụ liên quan đến giao diện, thiết kế, và kiểm thử.

II. Quản lý mã nguồn với Git và Github

1. Git

Git là một hệ thống quản lý phiên bản phân tán (distributed version control system). Nó giúp theo dõi và quản lý các thay đổi trong mã nguồn của dự án. Git rất phổ biến trong việc phát triển phần mềm vì nó giúp các lập trình viên:

- Theo dõi lịch sử thay đổi của mã nguồn.
- Phối hợp làm việc nhóm trên cùng một dự án mà không gây xung đột.
- Dễ dàng quay lại phiên bản trước đó nếu có lỗi xảy ra.
- Tạo nhánh (branch) để thử nghiệm các tính năng mới mà không ảnh hưởng đến mã nguồn chính.

2. Github

GitHub là một nền tảng lưu trữ dự án và cộng tác trực tuyến, giúp bạn quản lý mã nguồn và hợp tác với những người khác trên các dự án. GitHub sử dụng Git làm hệ thống quản lý phiên bản.

Các tính năng chính của GitHub bao gồm:

- Lưu trữ kho mã nguồn trên đám mây.
- Chia sẻ và hợp tác với lập trình viên khác.
- Pull Request: Cho phép bạn yêu cầu xem xét và hợp nhất thay đổi của mình vào nhánh chính.
- Issues: Theo dõi các vấn đề, lỗi hoặc yêu cầu tính năng.
- Actions: Tự động hóa các quy trình như kiểm thử, triển khai.

3. Ứng dụng Github vào quản lý mã nguồn của dự án

GitHub được sử dụng để lưu trữ mã nguồn và theo dõi quá trình phát triển của dự án. Trong dự án này, GitHub có thể hỗ trợ quản lý các phiên bản và lưu trữ mã nguồn để dễ dàng chia sẻ và làm việc nhóm.

CHƯƠNG 5: XÂY DỰNG CHƯƠNG TRÌNH GAME

I. Quy trình thực hiện:

1. Phân chia công việc:

Dự án của nhóm chúng em được quản lý bằng Jira theo phương pháp Scrum. Scrum hoạt động theo chu kỳ ngắn gọi là **Sprint**, thường kéo dài từ 1-2 tuần. Trong từng Sprint, nhóm sẽ hoàn thành các nhiệm vụ cụ thể để tiến dần đến việc hoàn thành dự án tổng thể.

- **Nguyễn Thành Trí:** Leader/Developer
  + **Vai trò:** Là Scrum Master kiêm Developer chính, Trí chịu trách nhiệm lãnh đạo, phân chia công việc, đảm bảo tiến độ dự án và quản lý các vấn đề phát sinh.
  + **Nhiệm vụ cụ thể:**
    - **Quản lý dự án:** Tổ chức các cuộc họp Sprint Planning và Daily Stand-up thông qua Google Meet, đặt ra các mục tiêu Sprint và giám sát tiến độ hoàn thành.
    - **Chịu trách nhiệm Git:** Quản lý việc merge các commit, xử lý xung đột mã (conflicts), đảm bảo rằng mọi thành viên đều làm việc trên mã nguồn sạch.
    - **Cài đặt thư viện SDL2 và thiết kế khung chương trình:** Liên kết thư viện SDL2 với VS22. Tạo các class cơ bản và cấu trúc khung chương trình làm việc theo luồng đã lên ý tưởng trước đó.
    - **Xây dựng các class nhân vật, quái vật:** Thiết kế ra các class nhân vật và quái vật dựa vào hình ảnh đã được vẽ trước đó.
    - **Xây dựng các logic của game:** Kiểm tra va chạm giữa nhân vật và các vật thể ở trong map.
- **Trần Ngọc Biên:** Designer/Developer
  + **Vai trò:** Thiết kế các hình ảnh của nhân vật, quái vật, background từ các tileset có sẵn và xây dựng lớp Map.
  + **Nhiệm vụ cụ thể:**
    - **Thiết kế assets game:** Dùng các tileset có sẵn để tạo nên các background, các hình ảnh nhân vật và quái, các giao diện menu
    - **Xây dựng lớp Map** để quản lý các thuộc tính của Map như các con quái, các cạm bẫy, các vật phẩm,…

- **Trần Văn Minh Tú:** Tester/Developer
  + **Vai trò:** xây dựng giao diện menu và kiểm thử phần mềm
  + **Nhiệm vụ cụ thể:**
    - **Xây dựng giao diện menu:** Xây dựng giao diện menu dựa theo format và hình ảnh đã được thiết kế bao gồm trang menu chính, trang tutorial, trang submenu
    - **Kiểm thử phần mềm:** Kiểm tra các lỗi và tính ổn định của các tính năng, chức năng trong trò chơi.

2. Triển khai dự án

a. Lên ý tưởng và tìm kiếm tileset phù hợp

Game mê cung huyền thoại – The Legendary Maze là một game thuộc thể loại platform. Người chơi trong vai một pháp sư bảo vệ khu rừng mê cung và hành trình đi tìm lại những viên ngọc năng lượng của khu rừng đang bị những thế lực hắc ám nắm giữ. Thế lực hắc ám này được dẫn đầu bởi vua rồng hắc ám đã chiếm giữ những viên ngọc năng lượng và trú ngụ ở trong khu rừng mê cung. Nhiệm vụ của người chơi là phải đi tìm lại những viên ngọc năng lượng ở trên bản đồ. Trong quá trình thu thập, người chơi phải vượt qua những cạm bẫy ở khắp nơi. Bên cạnh những viên ngọc năng lượng, trên bản đồ cũng sẽ tồn tại vật phẩm hỗ trợ, đó là các bình năng lượng, giúp người chơi có thể thi triển kỹ năng và tiêu diệt các thế lực hắc ám.

b. Cài đặt thư viện SDL2

Quá trình cài đặt thư viện SDL2 trên môi trường Window và sử dụng công cụ lập trình VS22 diễn ra theo các bước:

- **Bước 1: Tải xuống SDL2**
  + Truy cập trang chính thức của SDL: https://libsdl.org.
  + Tải SDL2 Development Library cho Windows (SDL2-devel-2.x.x-VC.zip).
- **Bước 2: Giải nén**
  + Giải nén file .zip vào một thư mục, ví dụ: C:\SDL2.
- **Bước 3: Cấu hình Visual Studio**
  + **Tạo một dự án mới:** Mở Visual Studio và tạo một dự án C++ mới (Console App hoặc Empty Project).

+ **Thêm đường dẫn thư viện SDL2 vào dự án:**
  - Nhấp chuột phải vào tên dự án của bạn → Properties.
  - Vào Configuration Properties → VC++ Directories.
  - Include Directories: Thêm đường dẫn C:\SDL2\include.
  - Library Directories: Thêm đường dẫn C:\SDL2\lib\x64 (hoặc x86 nếu bạn sử dụng phiên bản 32-bit).
+ **Link thư viện SDL2:**
  - Vào Configuration Properties → Linker → Input.
  - Thêm SDL2.lib và SDL2main.lib vào Additional Dependencies.
+ **Sao chép file DLL:**
  - Sao chép SDL2.dll từ thư mục C:\SDL2\lib\x64 vào thư mục Debug và Release của dự án (nơi chứa file .exe).

- **Bước 4: Viết chương trình kiểm tra**
  + Viết chương trình:

```
#include <SDL.h>
#include <iostream>

int main(int argc, char* argv[]) {
    if (SDL_Init(SDL_INIT_VIDEO) != 0) {
        std::cout << "SDL Initialization Error: " << SDL_GetError() << std::endl;
        return 1;
    }
    SDL_Window* window = SDL_CreateWindow("Kiem tra", SDL_WINDOWPOS_CENTERED, SDL_WINDOWPOS_CENTERED, 800,
600, 0);
    if (window == nullptr) {
        std::cout << "SDL Window Error: " << SDL_GetError() << std::endl;
        SDL_Quit();
        return 1;
    }
    SDL_Delay(3000);
    SDL_DestroyWindow(window);
    SDL_Quit();
    return 0;
}
```

Hình 3.2 Kiểm tra chương trình SDL2

+ **Build và Run:** Nếu mọi thứ đúng, bạn sẽ thấy một cửa sổ trống hiện ra.

c. Xây dựng khung sườn cho game

Chương trình được tổ chức gồm các file nhỏ (file .h và file .cpp) và được gọi lại bằng #include. Mỗi file sẽ đảm nhận một nhiệm vụ nhất định và có thể tái sử dụng nhiều lần. Chương trình sẽ gồm các loại file: file main, file setup, các file định nghĩa lớp như map.h, mainObject.h,…

Khung chương trình gồm các file đảm bảo cho chương trình có thể hoạt động được:

- **File function:** đây là file chứa các biến và macro cần thiết cho việc triển khai mã
- **File setup:** file setup chứa các hàm cơ bản để chạy được chương trình của thư viện SDL2 như hàm khởi tạo, hàm đóng game, hàm khởi động game, hàm tải dữ liệu
- **File main:** đây là file sẽ được chương trình thực thi.
- **File baseObject:** Là một lớp được định nghĩa các thuộc tính và phương thức với mục đích tải các hình ảnh có sẵn và vẽ lên màn hình.

```
bool BaseObject::loadImg(std::string filepath, SDL_Renderer* screen) {
        SDL_Texture* new_texture = NULL;                //tao moi texture
        SDL_Surface* surface = IMG_Load(filepath.c_str());       // load file hinh len texture
        if (surface != NULL) {
                SDL_SetColorKey(surface, SDL_TRUE, BACKGROUND_COLOR);            // Xoa nen file anh
                new_texture = SDL_CreateTextureFromSurface(screen, surface);             // chuyen doi
surface sang texture
                if (new_texture != NULL) {              // lay thong so cho khung hinh
                        rect_.w = surface->w;
                        rect_.h = surface->h;
                }
                SDL_FreeSurface(surface);               // giai phong vung nho
        }
        SDL_DestroyTexture(object_);     // xoa vung nho cua texture hien tai
        object_ = new_texture;
        return object_ != NULL;
}
```

Hình 3.3: hàm loadImg lưu hình ảnh vào texture

d. Xây dựng giao diện menu

Giao diện menu bao gồm khung menu và các nút bấm để người dùng tương tác với chương trình như tiếp tục, khởi động lại game, thoát game,…

Các bước xây dựng giao diện menu:

- Vẽ các khung menu: sử dụng các hàm loadImg() và render của BaseObject.
- Vẽ các nút bấm bằng các hình chữ nhật.
- Liên kết nút bấm với các flag điều khiển chương trình: khi người dùng nhấn vào nút bấm thì cờ tương ứng sẽ thay đổi giúp điều hướng chương trình theo ý muốn.

e. Xây dựng lớp Map

Lớp Map trong chương trình sẽ bao gồm 6 map nhỏ là các màn chơi mà người chơi phải vượt qua. Mỗi màn chơi sẽ chứa các dữ liệu bao gồm: các loại quái, các loại vật phẩm, các loại tường, đường đi, cạm bẫy,… Các dữ liệu về map sẽ được tải lên nhờ hàm loadMap(). Trong đó:

- Đường đi trong map được tạo nên nhờ kỹ thuật tilemap (chia map làm các ô nhỏ). Các ô tilemap được lưu trong file map.dat
- Các thông tin khác của map như các item, các quái vật, các cạm bẫy được lưu trong file map.json. File json sẽ được nhập vào chương trình và xử lý bằng thư viện json.hpp

f. Xây dựng lớp nhân vật

Lớp nhân vật là lớp hình ảnh có chuyển động (animation) nên hình ảnh sẽ được tải lên dưới dạng spire sheet. Người chơi có thể điều khiển nhân vật di chuyển, nhảy, bắn bằng các phím A,D,Space,J,…chương trình sẽ luôn kiểm tra sự kiện người dùng nhấn phím rồi thực hiện các lệnh theo yêu cầu

```
if (evn.type == SDL_KEYDOWN && died == false) {
        switch (evn.key.keysym.sym) {
        case SDLK_d:
                if (status != RIGHT) {
                        status = RIGHT;
                }
                if (object_ != run)
                        updateImg(run);
                input_type.right = 1;
                input_type.left = 0;
                break;
        case SDLK_a:
                if (status != LEFT) {
                        status = LEFT;
                }
                if (object_ != run)
                        updateImg(run);
                input_type.right = 0;
                input_type.left = 1;
                break;
        case SDLK_SPACE:
                input_type.jump = 1;
                if (object_ != jump) updateImg(jump);
                break;
        }
}
```

Hình 3.4: Xử lý đầu vào từ bàn phím

Bên cạnh việc di chuyển, nhân vật còn có các chức năng khác. Đơn cử như chức năng nhặt vật phẩm. Ở mỗi màn chơi sẽ có các vật phẩm như bình năng lượng mana và viên ngọc năng lượng. Nếu nhân vật chạm trúng vật phẩm thì sẽ nhận được vật phẩm đó. Bình năng lượng sẽ giúp nhân vật nhận được 15 điểm mana và 1 mạng hồi sinh. Mỗi lần bắn 1 viên đạn thì người chơi sẽ tốn 10 điểm mana. Vật phẩm viên ngọc năng lượng là chìa khóa để có thể thắng được trò chơi. Ở mỗi màn chơi sẽ có các cạm bẫy mà người chơi phải vượt qua. Nếu nhân vật chạm trúng các gai nhọn, các dây gai thì sẽ chết. Bên cạnh đó người chơi cũng phải tránh né những viên đạn do các con quái, boss bắn ra. Nếu va chạm với viên đạn hoặc va chạm trực tiếp với quái vật thì sẽ chết.

g. Xây dựng lớp quái vật, Boss

Lớp quái vật giống như lớp nhân vật, cũng đều là hình ảnh chuyển động. Chuyển động của quái vật không theo sự điều khiển của trò chơi mà chuyển động theo quỹ đạo đã được lập trình từ trước. Các quái vật cũng có thể bắn đạn, di chuyển qua lại,… Nhân vật chạm vào quái hoặc bị quái bắn trúng thì sẽ chết. Mỗi màn chơi sẽ có số lượng quái khác nhau, vị trí khác nhau, tùy theo độ khó của màn chơi đó. Càng về sau thì độ khó càng cao.

Ở màn chơi cuối cùng sẽ có sự xuất hiện của Boss – quái vật mạnh và to nhất bản đồ. Lớp boss được kế thừa từ lớp quái. Bên cạnh đó còn thêm vào các thuộc tính như lượng máu, khiên chắn, ngọn lửa xanh,… Làm cho việc tiêu diệt trùm cuối khó khăn hơn.

h. Cơ chế thắng và thua

Ở mỗi màn chơi, nhiệm vụ của người chơi là thu thập các viên ngọc năng lượng. Nếu người chơi thu thập đủ tất cả 5 viên thì người chơi sẽ chiến thắng. Tuy nhiên trong quá trình thu thập các viên ngọc, nếu người chơi để mất hết số mạng hồi sinh thì người chơi sẽ thua và phải chơi lại từ đầu.

```
int i = 0;
while (i < crystal.size()){
        if (crystal[i] == 0) break;
        i++;
}if (i == crystal.size()){
        is_win = true; is_lose = false;
        return;
}if (life <= 0){
        is_lose = true; is_win = false;
        return;
}
```

Hình 3.5: Cơ chế thắng thua

i. Thêm các hiệu ứng âm thanh

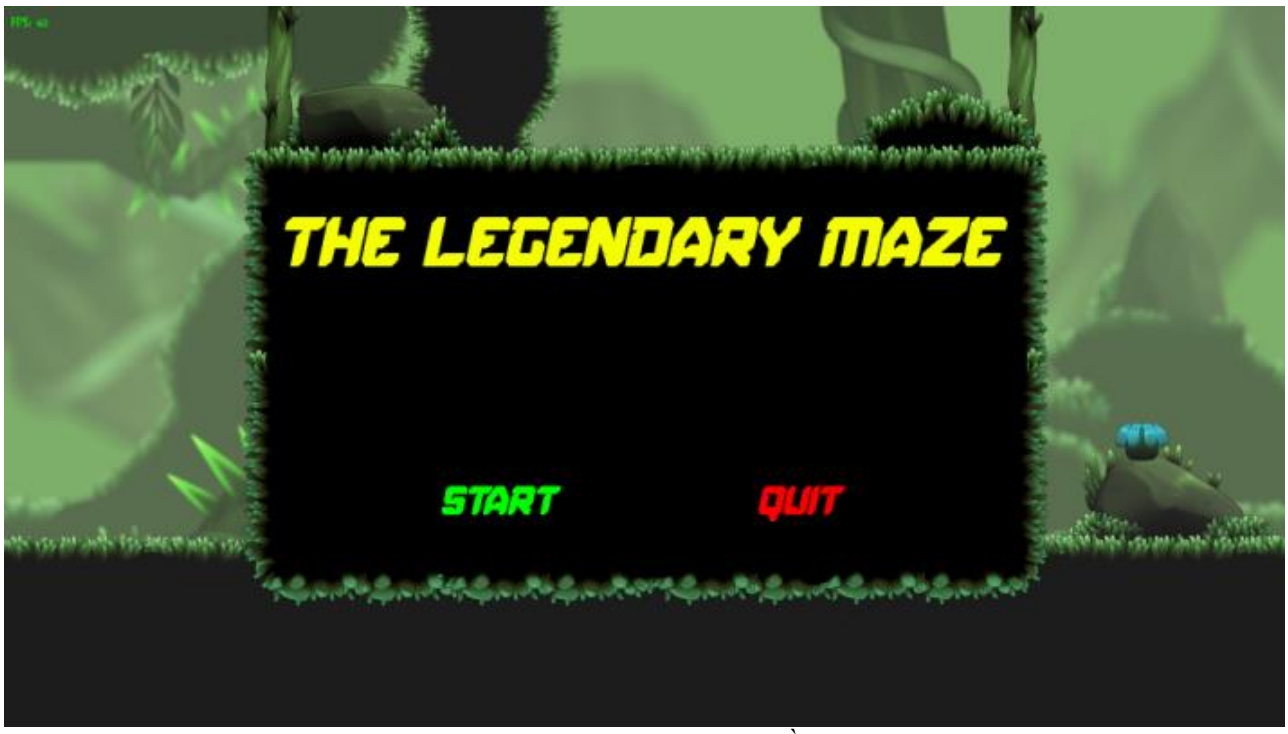
Âm thanh được thêm vào trò chơi thông qua thư viện con SDL_mixer. Thư viện này có các hàm phát, dừng nhạc, và các hiệu ứng âm thanh. Các âm thanh được tải lên từ file .wav hoặc file .mp3. SDL2 hỗ trợ âm thanh đa kênh tức là sẽ phát được nhiều âm thanh cùng 1 lúc sẽ tránh được tình trạng âm thanh bị ngắt để phát âm thanh khác.

```
private:
        Mix_Chunk* climb;
        Mix_Chunk* crystal_pick_up;
        Mix_Chunk* mana_pick_up;
        Mix_Chunk* jump;
        Mix_Chunk* land;
        Mix_Chunk* fire;
        Mix_Chunk* death;
        Mix_Chunk* walk;
        Mix_Music* background;
        Mix_Music* bossFight;
        Mix_Music* victory;
        Mix_Music* defeat;
};
```

Hình 3.6: Các loại hiệu ứng âm thanh

II. Kết quả đạt được

1. Hoàn thiện giao diện



Hình 3.7: Giao diện mở đầu

Hình 3.8: Giao diện hướng dẫn

Hình 3.9: Giao diện trò chơi

2. Những kiến thức đã học hỏi được

Sau bài tập này, chúng em đã có thể sử dụng thành thạo kỹ thuật lập trình hướng đối tượng vào các project để giải quyết các vấn đề phức tạp. Chúng em cũng nắm rõ hơn về các

tính chất của OOP như tính kế thừa, tính đóng gói, …Bên cạnh đó, chúng em cũng biết thêm nhiều về các kiến thức mới như chia tách file, kỹ năng sử dụng VS22, kỹ năng debug và fix lỗi, kỹ năng sử dụng git, github, jira để quản lý dựa án.

III. Kiến nghị và hướng phát triển trong tương lai

1. Cải tiến các tính năng hiện tại

- Cần tối ưu mã C++ của các tính năng hiện tại giúp tối ưu hóa hiệu năng, giảm tải CPU và ổn định chỉ số khung hình trên giây.
- Viết lại các đoạn mã chưa tối ưu giúp thuận lợi hơn cho việc bảo trì mã nguồn sau này.
- Thêm các chuyển động cho hình ảnh thêm mượt

2. Bổ sung các tính năng mới

- Các tính năng cần bổ sung thêm như:
  + Thêm nhiều loại quái khác và các loại đạn khác nhau
  + Save game
  + Đăng nhập tài khoản
  + Bảng xếp hạng
  + Báo cáo lỗi

3. Hỗ trợ đa nền tảng

Trong tương lai, chương trình có thể phát triển thêm, hỗ trợ đa nền tảng như android, MacOS, Linux,… để trò chơi có thể chơi được trên tất cả thiết bị thông minh, tiếp cận với nhiều người chơi hơn. Thông tin người chơi sẽ được lưu trên các máy chủ, các cơ sở dữ liệu nên người dùng có thể dễ dàng chuyển đổi tài khoảng giữa các thiết bị mà không sợ bị mất dữ liệu.

PHỤ LỤC

1. Tải và cài đặt game

- Tải file nén của game theo đường link sau:

+ https://drive.google.com/drive/u/0/folders/1ydAUwADNdvhnWLPj5Y6rSKKMH3PMnMfE

- Có 2 phiên bản bình thường và siêu khó.
- Nhấn vào file MeCungHuyenThoai.exe để chạy game.

2. Cách chơi

- Người chơi sẽ vượt qua các chướng ngại vật là những bẫy chông, những con quái… và sau đó lấy những viên ngọc. Sau khi tìm đủ 5 viên thì sẽ chiến thắng. Đặt biệt ở giai đoạn lấy viên ngọc cuối cùng, người chơi phải chiến đấu với rồng hắc ám – trùm cuối của trò chơi. Hãy nhớ tiết kiệm năng lượng vì nó có giới hạn.
- Người chơi phải điều khiển nhân vật bằng các nút:
  + A, D: di chuyển sang hai bên trái phải.
  + W: leo dây
  + SPACE: Nhảy
  + J: bắn
  + F: mở khóa cửa
  + ESC: mở menu tab
  + F1: mở bản hướng dẫn
  + Nhấn vào nút thoát ở góc phải dưới để thoát
  + Nhấn vào nút reset ở góc phải dưới để reset lại game

3. Quản lý mã nguồn và triển khai dự án

- Github quản lý source code:

+ https://github.com/TriNguyenThanh/MeCungHuyenThoai

- Jira quản lý dự án:

+ https://bienbeo05.atlassian.net/jira/software/projects/BTLOGMCHT/boards/2

TÀI LIỆU THAM KHẢO

[1]. Maaot, assets game, https://maaot.itch.io/mossy-cavern, truy cập lúc 17/9/2024.

[2]. Lê Đăng Hưng, Tạ Tuấn Anh, Nguyễn Hữu Đức, Nguyễn Thanh Thủy, *Lập trình hướng đối tượng với C++,* nhà xuất bản Khoa học và Kỹ thuật Hà Nội năm 2006

[3]. Các hàm của thư viện SDL2, https://wiki.libsdl.org/SDL2/FrontPage, truy cập lúc 17/9/2024.

[4]. Trang web chính thức của thư viện SDL2: