**TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI**

**BÁO CÁO TỔNG KẾT**

**ĐỀ TÀI NGHIÊN CỨU KHOA HỌC CỦA SINH VIÊN**

**NĂM 2016**

**Nguyên cứu xử lý đồ họa xây dựng game Tetris**

Sinh viên thực hiện: Nguyễn Đức Hiếu (MSV : 1402959) Nam

Lớp: Công nghệ thông tin 1.

Khoa: Công nghệ thông tin

Người hướng dẫn: ThS Phạm Xuân Tích

**TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI**

**BÁO CÁO TỔNG KẾT**

**ĐỀ TÀI NGHIÊN CỨU KHOA HỌC CỦA SINH VIÊN**

**NĂM 2016**

**Nguyên cứu xử lý đồ họa xây dựng game Tetris**

Thuộc lĩnh vực khoa học và công nghệ:

Sinh viên thực hiện: Nguyễn Đức Hiếu Nam

Dân tộc:Kinh

Lớp, khoa: Công nghệ thông tin 1 Năm thứ: 2 /Số năm đào tạo: 4

Ngành học: Công nghệ thông tin

Người hướng dẫn: ThS Phạm Xuân Tích

***(Ghi rõ họ và tên sinh viên chịu trách nhiệm chính thực hiện đề tài)***

Người hướng dẫn: <Chức danh khoa học, học vị, họ tên của người hướng dẫn>

**MỤC LỤC**

**CHƯƠNG I : TỔNG QUAN.**

1.Tổng quan về C/C++…………………………………………………………………………………………5

2.Tổng quan về cấu trúc dữ liệu(sử dụng cho game). …………………………………………8

3.Tổng quan về game. ………………………………………………………………………………………..14

**CHƯƠNG II :PHÂN TÍCH THIẾT KẾ GAME**

**1.PHÂN TÍCH VÀ THIẾT KẾ THUẬT TOÁN,CẤU TRÚC DỮ LIỆU**

A.Tính điểm và cấp độ ……………………….……………………….……………………….…………….15

B.Bảng Game……………………….……………………….……………………….……………………………16

C.Khối gạch………………………………….……………………….……………………….……………………21

D.Xử lí Game……………………….……………………….……………………….…………………………….26

E.Giao diện……………………….……………………….……………………….……………………………….32

**CHƯƠNG III : XÂY DỰNG GAME**

**1.XÂY DỰNG HÀM MAIN**……………………….……………………….………………………………….37

**2.KẾT THÚC**……………………….……………………….……………………….……………………………..40

**TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI**

**THÔNG TIN KẾT QUẢ NGHIÊN CỨU CỦA ĐỀ TÀI**

**1. Thông tin chung:**

**-** Tên đề tài: Nguyên cứu xử lý đồ họa xây dựng game Tetris.

- Sinh viên thực hiện: Nguyễn Đức Hiếu.

- Lớp: Công nghệ thông tin 1.

Khoa: Công nghệ thông tin Năm thứ: 2 Số năm đào tạo: 4.

- Người hướng dẫn: ThS Phạm Xuân Tích.

**2. Mục tiêu đề tài:**

**-** Trong đề tài này chúng em sẽ nghiên cứu nguyên lí , nguyên tắc ,cấu trúc và lập trình để làm lại 1 trò chơi có tựa game là Xếp hình (Tetris) . Đây là tựa game huyền thoại hay nhất của mọi thời đại ! Được phát hành vào năm 1984, nó trở nên cực kỳ nổi tiếng nhờ sự đơn giản và cuốn hút của Game Play.

**-** Nguyên cứu thuật toán game Tetris.

- Lập trình game bằng phương pháp hướng đối tượng với ngôn ngữ C/C++

- Nâng cao kỹ thuật lập trình.

**3. Tính mới và sáng tạo:**

**-** Ngoài các tính năng ,nguyên tắc cơ bản của trò chơi,chúng em có tạo thêm một số tính năng mới :

+ Tạo chức năng thay đổi giao diện (theme) cho game

+ Tạo phím chức năng (bật/tắt) trong game cho phép người chơi điều khiển 1 lúc 2 khối gạch rơi xuống với nguyên tắc hoạt động tương tự 1 khối gạch rơi .

**4. Kết quả nghiên cứu:**

Xây dựng được game Tetris trên nền DOS (màn hình đen Console) cho phép người chơi ghép những miếng gạch rơi với hình dạng, màu sắc khác nhau, với độ khó tăng dần.

**5. Đóng góp về mặt kinh tế - xã hội,** **giáo dục và đào tạo, an ninh, quốc phòng và khả năng áp dụng của đề tài:**

- Giúp người chơi rèn luyện tư duy , khả năng kiên nhẫn .

**6.** **Công bố khoa học của sinh viên từ kết quả nghiên cứu của đề tài**

Ngày tháng năm

|  |  |
| --- | --- |
|  | **Sinh viên chịu trách nhiệm chính**  **thực hiện đề tài**  *(ký, họ và tên)* |

**Nhận xét của người hướng dẫn về những đóng góp khoa học của sinh viên thực hiện đề tài** *(phần này do người hướng dẫn ghi):*

Ngày tháng năm

|  |  |
| --- | --- |
|  | **Người hướng dẫn**  *(ký, họ và tên)* |

**\*LỜI NÓI ĐẦU**

Trò chơi Xếp gạch (tên gọi quốc tế: Tetris) là một trò chơi thuộc thể loại trí tuệ. Hấp dẫn với mọi lứa tuổi , giới tính .Trò chơi ra đời vào thập kỉ 80 của thế kỷ trước, nhưng cho đến ngày nay vẫn làm nhiều người mất ăn mất ngủ. Trò chơi đã được nhiều công ty phần mềm, nhiều cá nhân đã phát triển ngày càng đa dạng về thể loại cũng như tùy biến về cách chơi.

Tuy đơn giản vậy, nhưng bất kì ai đã từng thử qua trò xếp hình này cũng đều phải công nhận **Tetris** không hề dễ chơi, không gây nhàm chán trong thời gian dài và rất khó đạt tới trình độ ''thượng thừa''. Có thể nói,Tetris hội tụ đầy đủ các yếu tố cần thiết của một tựa game kinh điển.

Trên tinh thần đó, nhóm chúng em cũng xin viết một phiên bản của trò chơi này, đồng thời cũng nhằm nâng cao kĩ năng lập trình .

Trong phần đề tài này chúng em sẽ trình bày tổng quan về chương trình , ưu, nhược điểm, vài nét về giao diện ,cũng như chức năng chình của trò chơi.

Thay mặt nhóm

Sinh viên: Nguyễn Đức Hiếu

**CHƯƠNG I : TỔNG QUAN**

**1.Tổng quan về C/C++.**

-*Ngôn ngữ C:*

**Ngôn ngữ lập trình C** là một ngôn ngữ mệnh lệnh được phát triển từ đầu thập niên 1970 bởi [Dennis Ritchie](https://vi.wikipedia.org/wiki/Dennis_Ritchie) để dùng trong hệ điều hành UNIX. Từ đó, ngôn ngữ này đã lan rộng ra nhiều hệ điều hành khác và trở thành một những ngôn ngữ phổ dụng nhất. C là ngôn ngữ rất có hiệu quả và được ưa chuộng nhất để viết các phần mềm hệ thống, mặc dù nó cũng được dùng cho việc viết các ứng dụng. Ngoài ra, C cũng thường được dùng làm phương tiện giảng dạy trong khoa học máy tính mặc dù ngôn ngữ này không được thiết kế dành cho người nhập môn.

C là một [ngôn ngữ lập trình](https://vi.wikipedia.org/wiki/Ng%C3%B4n_ng%E1%BB%AF_l%E1%BA%ADp_tr%C3%ACnh) tương đối nhỏ gọn vận hành gần với phần cứng và nó giống với ngôn ngữ [Assembler](https://vi.wikipedia.org/w/index.php?title=Assembler&action=edit&redlink=1) hơn hầu hết các [ngôn ngữ bậc cao](https://vi.wikipedia.org/w/index.php?title=Ng%C3%B4n_ng%E1%BB%AF_b%E1%BA%ADc_cao&action=edit&redlink=1). Hơn thế, C đôi khi được đánh giá như là "có khả năng di động", cho thấy sự khác nhau quan trọng giữa nó với [ngôn ngữ bậc thấp](https://vi.wikipedia.org/wiki/Ng%C3%B4n_ng%E1%BB%AF_l%E1%BA%ADp_tr%C3%ACnh_b%E1%BA%ADc_th%E1%BA%A5p) như là Assembler, đó là việc mã C có thể được dịch và thi hành trong hầu hết các máy tính, hơn hẳn các ngôn ngữ hiện tại trong khi đó thì Assembler chỉ có thể chạy trong một số máy tính đặc biệt. Vì lý do này C được xem là [ngôn ngữ bậc trung](https://vi.wikipedia.org/w/index.php?title=Ng%C3%B4n_ng%E1%BB%AF_b%E1%BA%ADc_trung&action=edit&redlink=1).

C đã được tạo ra với một mục tiêu là làm cho nó thuận tiện để viết các chương trình lớn với số lỗi ít hơn trong [mẫu hình](https://vi.wikipedia.org/wiki/M%E1%BA%ABu_h%C3%ACnh) [lập trình thủ tục](https://vi.wikipedia.org/wiki/L%E1%BA%ADp_tr%C3%ACnh_th%E1%BB%A7_t%E1%BB%A5c) mà lại không đặt gánh nặng lên vai người viết ra [trình dịch](https://vi.wikipedia.org/wiki/Tr%C3%ACnh_bi%C3%AAn_d%E1%BB%8Bch) C, là những người bề bộn với các đặc tả phức tạp của ngôn ngữ. Cuối cùng C có thêm những chức năng sau:

* Một [ngôn ngữ cốt lõi](https://vi.wikipedia.org/w/index.php?title=Ng%C3%B4n_ng%E1%BB%AF_c%E1%BB%91t_l%C3%B5i&action=edit&redlink=1) đơn giản, với các chức năng quan trọng chẳng hạn như là những hàm hay việc xử lý tập tin sẽ được cung cấp bởi các bộ [thư viện các thủ tục](https://vi.wikipedia.org/w/index.php?title=Th%C6%B0_vi%E1%BB%87n_(khoa_h%E1%BB%8Dc_m%C3%A1y_t%C3%ADnh)&action=edit&redlink=1).
* Tập trung trên mẫu hình [lập trình thủ tục](https://vi.wikipedia.org/wiki/L%E1%BA%ADp_tr%C3%ACnh_th%E1%BB%A7_t%E1%BB%A5c), với các phương tiện lập trình theo kiểu [cấu trúc](https://vi.wikipedia.org/wiki/L%E1%BA%ADp_tr%C3%ACnh_c%E1%BA%A5u_tr%C3%BAc).
* Một hệ thống kiểu đơn giản nhằm loại bỏ nhiều phép toán không có ý nghĩa thực dụng.
* Dùng ngôn ngữ [tiền xử lý](https://vi.wikipedia.org/wiki/Ti%E1%BB%81n_x%E1%BB%AD_l%C3%BD), tức là các câu lệnh [tiền xử lý C](https://vi.wikipedia.org/wiki/Ti%E1%BB%81n_x%E1%BB%AD_l%C3%BD_C), cho các nhiệm vụ như là định nghĩa các [macro](https://vi.wikipedia.org/w/index.php?title=Macro&action=edit&redlink=1) và hàm chứa nhiều tập tin [mã nguồn](https://vi.wikipedia.org/wiki/M%C3%A3_ngu%E1%BB%93n) (bằng cách dùng câu lệnh tiền xử lý dạng #include chẳng hạn).
* Mức thấp của ngôn ngữ cho phép dùng tới [bộ nhớ máy tính](https://vi.wikipedia.org/wiki/B%E1%BB%99_nh%E1%BB%9B) qua việc sử dụng kiểu dữ liệu pointer.
* Số lượng từ khóa rất nhỏ gọn.
* Các [tham số](https://vi.wikipedia.org/wiki/Tham_s%E1%BB%91_(khoa_h%E1%BB%8Dc_m%C3%A1y_t%C3%ADnh)) được đưa vào các hàm bằng giá trị, không bằng địa chỉ.
* Hàm các con trỏ cho phép hình thành một nền tảng ban đầu cho [tính đóng](https://vi.wikipedia.org/w/index.php?title=T%C3%ADnh_%C4%91%C3%B3ng_(ng%C3%B4n_ng%E1%BB%AF_l%E1%BA%ADp_tr%C3%ACnh&action=edit&redlink=1) và [tính đa hình](https://vi.wikipedia.org/w/index.php?title=T%C3%ADnh_%C4%91a_h%C3%ACnh&action=edit&redlink=1).
* Hỗ trợ các [bản ghi](https://vi.wikipedia.org/w/index.php?title=B%E1%BA%A3n_ghi_(ng%C3%B4n_ng%E1%BB%AF_l%E1%BA%ADp_tr%C3%ACnh)&action=edit&redlink=1) hay các kiểu dữ liệu kết hợp do người dùng từ khóa định nghĩa struct cho phép các dữ liệu liên hệ nhau có thể được tập hợp lại và được điều chỉnh như là toàn bộ.

Một số chức năng khác mà C không có (hay còn thiếu) nhưng có thể tìm thấy ở các ngôn ngữ khác bao gồm:

* [An toàn kiểu](https://vi.wikipedia.org/w/index.php?title=An_to%C3%A0n_ki%E1%BB%83u&action=edit&redlink=1),
* Tự động [Thu dọn rác](https://vi.wikipedia.org/w/index.php?title=Thu_d%E1%BB%8Dn_r%C3%A1c_(ng%C3%B4n_ng%E1%BB%AF_l%E1%BA%ADp_tr%C3%ACnh)&action=edit&redlink=1),
* Các [lớp](https://vi.wikipedia.org/wiki/L%E1%BB%9Bp_(khoa_h%E1%BB%8Dc_m%C3%A1y_t%C3%ADnh)) hay các [đối tượng](https://vi.wikipedia.org/w/index.php?title=%C4%90%E1%BB%91i_t%C6%B0%E1%BB%A3ng_(OOP)&action=edit&redlink=1) cùng với các ứng xử của chúng (xem thêm [OOP](https://vi.wikipedia.org/wiki/L%E1%BA%ADp_tr%C3%ACnh_h%C6%B0%E1%BB%9Bng_%C4%91%E1%BB%91i_t%C6%B0%E1%BB%A3ng)),
* Các [hàm lồng nhau](https://vi.wikipedia.org/w/index.php?title=H%C3%A0m_l%E1%BB%93ng_nhau&action=edit&redlink=1),
* [Lập trình tiêu bản](https://vi.wikipedia.org/w/index.php?title=L%E1%BA%ADp_tr%C3%ACnh_ti%C3%AAu_b%E1%BA%A3n&action=edit&redlink=1) hay [Lập trình phổ dụng](https://vi.wikipedia.org/w/index.php?title=L%E1%BA%ADp_tr%C3%ACnh_ti%C3%AAu_b%E1%BA%A3n&action=edit&redlink=1),
* [Quá tải](https://vi.wikipedia.org/w/index.php?title=Qu%C3%A1_t%E1%BA%A3i_(ng%C3%B4n_ng%E1%BB%AF_l%E1%BA%ADp_tr%C3%ACnh)&action=edit&redlink=1) và [Quá tải toán tử](https://vi.wikipedia.org/w/index.php?title=Qu%C3%A1_t%E1%BA%A3i_to%C3%A1n_t%E1%BB%AD&action=edit&redlink=1),
* Các hỗ trợ cho [đa luồng](https://vi.wikipedia.org/w/index.php?title=%C4%90a_lu%E1%BB%93ng&action=edit&redlink=1), [đa nhiệm](https://vi.wikipedia.org/w/index.php?title=%C4%90a_nhi%E1%BB%87m&action=edit&redlink=1) và [mạng](https://vi.wikipedia.org/wiki/M%E1%BA%A1ng_m%C3%A1y_t%C3%ADnh).

-*Ngôn ngữ C++:*

**C++** (đọc là "C cộng cộng" hay "xi-plus-plus", [IPA](https://vi.wikipedia.org/wiki/B%E1%BA%A3ng_m%E1%BA%ABu_t%E1%BB%B1_phi%C3%AAn_%C3%A2m_qu%E1%BB%91c_t%E1%BA%BF): /siː pləs pləs/) là một loại ngôn ngữ lập trình. Đây là một dạngngôn ngữ đa mẫu hình tự do có kiểu tĩnh và hỗ trợ lập trình thủ tục, dữ liệu trừu trượng, lập trình hướng đối tượng, và lập trình đa hình. Từ thập niên 1990, C++ đã trở thành một trong những ngôn ngữ thương mại phổ biến nhất trong khi đó.

So với C, C++ tăng cường thêm nhiều tính năng, bao gồm: khai báo như mệnh đề, chuyển kiểu giống như hàm,new/delete, bool, các kiểu tham chiếu, const, các hàm nội tuyến (inline), các đối số mặc định, quá tải hàm, vùng tên (namespace), các lớp (bao gồm tất cả các chức năng liên quan tới lớp như kế thừa, hàm thành viên (phương pháp), hàm ảo, lớp trừu tượng, và cấu tử), sự quá tải toán tử, tiêu bản, toán tử ::, xử lí ngoại lệ, và sự nhận dạng kiểu trong thời gian thi hành.

**Tính đóng gói**[[sửa](https://vi.wikipedia.org/w/index.php?title=C%2B%2B&veaction=edit&vesection=5) | [sửa mã nguồn](https://vi.wikipedia.org/w/index.php?title=C%2B%2B&action=edit&section=5)]

C++ xây dựng tính đóng bằng cách cho phép mọi [thành viên](https://vi.wikipedia.org/wiki/L%E1%BA%ADp_tr%C3%ACnh_h%C6%B0%E1%BB%9Bng_%C4%91%E1%BB%91i_t%C6%B0%E1%BB%A3ng#CLASS) của một lớp có thể được khai báo bằng các từ khoá public, private, hay protected. (xem thêm [các khái niệm cơ bản trong ngôn ngữ OOP](https://vi.wikipedia.org/wiki/L%E1%BA%ADp_tr%C3%ACnh_h%C6%B0%E1%BB%9Bng_%C4%91%E1%BB%91i_t%C6%B0%E1%BB%A3ng#M.E1.BB.99t_s.E1.BB.91_kh.C3.A1i_ni.E1.BB.87m_c.E1.BA.A7n_bi.E1.BA.BFt_trong_ng.C3.B4n_ng.E1.BB.AF_OOP_hi.E1.BB.87n_.C4.91.E1.BA.A1i)). Một thành viên private chỉ có thể được truy cập từ các phương pháp (hàm nội tại) là thành viên của chính lớp đó hay được truy cập từ các hàm và các lớp được đặc biệt cho phép sử dụng bằng cách dùng từ khóa friend. Một thành viên protected của một lớp sẽ có thể truy cập được từ các thành viên (nào đó) của các lớp có tính kế thừa của nó hay cũng có thể truy cập được từ các thành viện của chính lớp đó và của mọi thành viên friend.

Nguyên lý của OOP là mọi và chỉ có các hàm là có thể truy cập được đến các giá trị nội tại của cùng lớp thì nên có tính đóng. C++ có hỗ trợ đặc tính này (qua các hàm thành viên và các hàm friend), nhưng C++ lại không là yêu cầu bắt buộc: người lập trình có thể khai báo các phần hay tất cả các giá trị nội tại là công cộng (*public*), và cũng cho phép làm cho toàn bộ lớp trở thành công cộng. Lí do là vì C++ hỗ trợ không chỉ lập trình hướng đối tượng mà còn hỗ trợ các [mẫu hình](https://vi.wikipedia.org/wiki/M%E1%BA%ABu_h%C3%ACnh) yếu hơn như là [lập trình mô-đun](https://vi.wikipedia.org/w/index.php?title=L%E1%BA%ADp_tr%C3%ACnh_m%C3%B4-%C4%91un&action=edit&redlink=1).

Một thói quen tốt cần có trong thực hành là khai báo mọi dữ liệu đều là [riêng tư](https://vi.wikipedia.org/wiki/L%E1%BA%ADp_tr%C3%ACnh_h%C6%B0%E1%BB%9Bng_%C4%91%E1%BB%91i_t%C6%B0%E1%BB%A3ng#Ri.C3.AAng_t.C6.B0_.28private.29) (*private*), hay ít nhất ở dạng [bảo tồn](https://vi.wikipedia.org/wiki/L%E1%BA%ADp_tr%C3%ACnh_h%C6%B0%E1%BB%9Bng_%C4%91%E1%BB%91i_t%C6%B0%E1%BB%A3ng#B.E1.BA.A3o_t.E1.BB.93n_.28protected.29), và sau đó, tạo ra một giao diện nhỏ (thông qua các phương pháp) cho người dùng của lớp này dấu đi các chi tiết thiết lập bên trong.

**Tính đa hình**[[sửa](https://vi.wikipedia.org/w/index.php?title=C%2B%2B&veaction=edit&vesection=6) | [sửa mã nguồn](https://vi.wikipedia.org/w/index.php?title=C%2B%2B&action=edit&section=6)]

Khái niệm [đa hình](https://vi.wikipedia.org/w/index.php?title=%C4%90a_h%C3%ACnh_(ng%C3%B4n_ng%E1%BB%AF_l%E1%BA%ADp_tr%C3%ACnh)&action=edit&redlink=1) được dùng khá rộng rãi và là khái niệm bị lạm dụng cũng như không được định nghĩa rõ ràng.

Nói chung tính đa hình trong lập trình hướng muốn nói đến 1 đoạn code nhưng trong 2 trường hợp khác nhau có thể xuất ra 2 kết quả khác nhau. Vì tính chất ra nhiều kết quả khác nhau này nên nó được gọi là đa hình.

Trong trường hợp của C++, khái niệm này thường được nối kết với các tên của các hàm thành viên. Các hàm thành viên này có cùng tên, sự khác nhau chỉ có thể được dựa vào một hay cả hai yếu tố sau:

1. Số lượng và kiểu của các đối số (tức là nguyên mẫu của hàm) -- Tính chất này gọi là **đa hình tĩnh** (*static polymorphism*)
2. Kiểu lớp mà thực thể thực sự thuộc vào. Tính chất này được dùng khi hàm thành viên được định nghĩa là hàm *ảo* qua từ khóa virtual—tính chất này gọi là **đa hình động** (*dynamic polymorphism*)

Khi được gọi thì chương trình sẽ tùy theo hai yếu tố trên để xác định chính xác hàm nào phải được thực thi trong số các hàm cùng tên.

### Tính kế thừa[sửa | sửa mã nguồn]

Kế thừa từ một lớp cơ sở có thể được khai báo thông qua các đặc tính công cộng, bảo tồn, hay riêng tư. Những đặc tính này cho phép xác định khi nào các lớp dẫn xuất hay không liên quan có thể sử dụng các thành viên công cộng, bảo tồn, hay riêng tư của lớp cơ sở. Tuy nhiên, chỉ có sự kế thừa dạng công cộng là hoàn toàn theo đúng ý nghĩa của việc "kế thừa". Hai dạng khác thì ít được dùng hơn. Nếu các đặc tả này không được khai báo thì việc kế thừa được gán mặc định là dạng riêng tư cho lớp cơ sở và dạng công cộng cho một cấu trúc cơ sở.

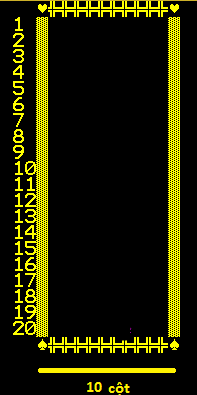
Các lớp cơ sở có thể được khai báo là ảo (thông qua từ khóa virtual). Kế thừa ảo bảo đảm rằng chỉ có một thực thể của lớp cơ sở tồn tại trong đồ thị kế thừa, tránh được một số vấn đề mơ hồ của việc đa kế thừa.

Đa kế thừa cũng là một tính năng có nhiều tranh cãi trong C++. Tính đa kế thừa cho phép một lớp được dẫn xuất từ nhiều hơn một lớp cơ sở; điều này có thể dẫn tới một đồ thị phức tạp của các quan hệ kế thừa. Ví dụ, lớp "Buổi học" có thể kế thừa từ hai lớp "thời gian" và "bộ môn". Một số ngôn ngữ khác như [Java](https://vi.wikipedia.org/wiki/Java_(ng%C3%B4n_ng%E1%BB%AF_l%E1%BA%ADp_tr%C3%ACnh)), tiến hành cách thức tương tự bằng cách cho phép kế thừa của nhiều giao diện trong khi giới hạn số lượng của các lớp cơ sở (kế thừa) chỉ còn là một lớp. (giao diện, không như lớp, không cho phép thiết lập nội dung của các thành viên và do đó không thể có thực thể).

**2.Tổng quan về cấu trúc dữ liệu(sử dụng cho game).**

Đầu tiên, chúng em xin nói về Bảng game (bảng chứa các khối gạch hay bảng chính để hiển thị game ) :

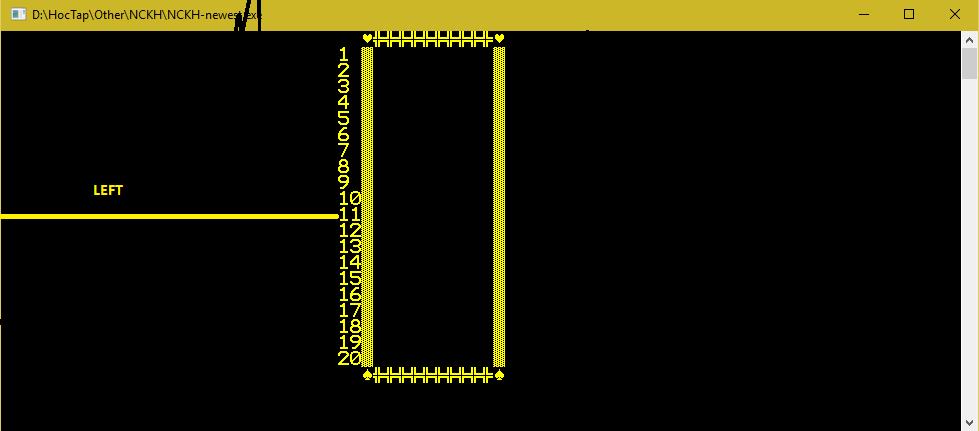
-Bảng được chia thành nhiều ô vuông nhỏ có kích thước là 20 hàng x 10 cột như hình :



Như vậy , Để quản lí được bảng trên ,chúng em sử dụng một mảng nguyên 2 chiều là

int Board[20][10];

Xét đến vị trí đặt bảng Game trên màn hình .gọi LEFT là khoảng cách tọa độ của bảng so với viền trái của màn hình console :



Vì vậy để tiện cho việc xử lí tọa độ và giới hạn bảng game ,em định nghĩa heightBoard = 21

widthBoard = 10 ,và bảng game được khai báo là:

int Board[heightBoard][widthBoard + LEFT + 1]

Thứ hai , đó là về khối gạch . Có 7 khối gạch như sau :

 (0),  (1), (2) ,  (3) , (4) ,  (5) , (6) .

Vậy quản lí những khối gạch này như thế nào ?

- Giải pháp của em đó là sử dụng 1 mảng nguyên để quản lí chúng.cụ thể là :

Dùng 1 mảng nguyên có kích thước đủ để lưu trữ trạng thái của các khối gạch (có thể dùng cấp phát động ); và 1 biến row để lưu số hàng ; 1 biến col lưu số cột của khối cụ thể,ma trận của khối đó là arr[row][col] , ô có khối gạch gán bằng 1 ,còn lại bằng 0.

int arr[4][4];

int row ;

int col ;

Sau đây sẽ áp dụng cho từng khối gạch :

\* khối (0) : row = 4 , col = 1 ,tất cả giá trị của arr[4][1] là 1.

\* khối (1) : row = 2 , col = 2, tất cả giá trị của arr[2][2] là 1.

\* khối (2) : row = 3 ,col = 2 , arr[0][0] = arr[2][0] = 0 ,còn lại bằng 1.

\* khối (3) : row = 3 , col = 2, arr[0][0] = arr[1][0] = 0 , còn lại là 1.

\* khối (4) : row = 3 , col = 2, arr[0][1] = arr[1][1] = 0 ,còn lại là 1.

\* khối (5) : row = 3,col = 2, arr[0][0] = arr[2][1] = 0, còn lại là 1.

\* khối (6) : row = 3 , col = 2, arr[0][1] = arr[2][0] = 0 ,còn lại là 1.

Tiếp tục,dưới đây chúng em xin nói về tổng quan các cấu trúc thành phần của game và đi sâu vào từng phần ở chương II :

-Các thư viện sử dụng :

#include <stdio.h>

#include <conio.h>

#include <Windows.h>

#include "console.h"

#include <time.h>

#include<string.h>

- Trong quá trình lập trình game, nhóm chúng em sử dụng phương pháp hướng đối tượng để xây dựng các lớp quản lí các thành phần như :

+Class MARK : đơn giản chỉ là class quản lí 1 điểm có hoành độ x, tung độ y

class MARK{

public:

int x;

int y;

}

+ Tính điểm và cấp độ :

class Play{

public:

int deley ; //toc do roi cua khoi gach

int score ; //diem

int overScore ;//gioi han diem cua tung level

int at ; //level hien tai

Play(); // ham tao

void restart(); //khoi tao lai level tu dau

}

+ Khối gạch :

class BRICK{

public:

int cl ; // mau sac cua khoi gach.

int arr[4][4] ; // ma tran quan li khoi gach.

int col,row  ; // tuong ung la cot va hang cua ma tran chua khoi gach.

Mark board  ; // toa do cua khoi gach tren BoardGame

void setBrick(); //gan tat ca arr[i][j] = 1;

void CREATE(int id,int vt) ; //dung de khoi tao khoi gach voi id tuong // ung,co hoanh do tren console la vt

void rotate() ;//dung de xoay ma tran 1 goc 90 do

int hasLEFT(); //kiem tra khoi gach co the sang trai hay khong

int hasRIGHT(); //kiem tra khoi gach co the sang phai hay khong

int hasDOWN(); //kiem tra khoi gach co xuong duoc ko , kiem tra gameover

}

+ Bảng chứa các khối gạch

class BoardGame{

public :

void CREATE() ; // khoi tao bang BoardGame.

void EAT(); //xu li khi an diem.

void Update() ; //cap nhat lai trang thai cua Board game.

void show() ; //hien thi BoardGame ra man hinh.

}

+ Khối gạch

class BRICK{

public:

int cl ; // mau sac cua khoi gach.

int arr[4][4] ; // ma tran quan li khoi gach.

int col,row  ; // tuong ung la cot va hang cua ma tran chua khoi gach.

Mark board  ; // toa do cua khoi gach tren console

void setBrick(); //gan tat ca arr[i][j] = 1;

void CREATE(int id,int vt) ; //dung de khoi tao khoi gach voi id tuong // ung,co hoanh do tren console la vt

void rotate() ;//dung de xoay ma tran 1 goc 90 do

int hasLEFT(); //kiem tra khoi gach co the sang trai hay khong

int hasRIGHT(); //kiem tra khoi gach co the sang phai hay khong

int hasDOWN(); //kiem tra khoi gach co xuong duoc ko , kiem tra gameover

}

+ Xử lí game (xử lí ăn điểm , điều khiển khối gạch ,…)

class Game{

public:

int kbhit() ;//xu li tuong tac voi nguoi choi

int Control() ;// xu li toc do roi cua khoi gach

int Process() ;//Xu li cac tinh huong trong game

}

+ Giao diện hiển thị game ( vẽ khung ,TIP , menu game , ....)

class InterFace{

public:

void clear(int x,int y,int k , int p); //dung de xoa man hinh bat dau tu //toa do (x,y) sang phai k , xuong duoi p don vi

void show(); //hien thi toan bo khung tro choi , kit u, ghi chu,..

void GameOver() ;//hien thi menu khi gameover

void drawNext(int id,int x ,int m) ; // dung de ve khung Next va khoi gach // sap roi xuong

void showLevel(); //dung de hien thi thong tin level hien tai

char showMenu() ; //ham hien thi menu khi moi vao game

}

- Cùng với một số hàm bổ trợ như :

+ gotoXY(x,y): dùng để trỏ con trỏ đến vị trí x,y trên màn hình Console

Code:

void gotoXY (int column, int line)

{

COORD coord;

coord.X = column;

coord.Y = line;

SetConsoleCursorPosition(GetStdHandle(STD\_OUTPUT\_HANDLE),coord);

}

+ clrscr() : xóa màn hình

Code:

void clrscr()

{

CONSOLE\_SCREEN\_BUFFER\_INFO csbiInfo;

HANDLE hConsoleOut;

COORD Home = {0,0};

DWORD dummy;

hConsoleOut = GetStdHandle(STD\_OUTPUT\_HANDLE);

GetConsoleScreenBufferInfo(hConsoleOut,&csbiInfo);

FillConsoleOutputCharacter(hConsoleOut,' ',csbiInfo.dwSize.X \* csbiInfo.dwSize.Y,Home,&dummy);

csbiInfo.dwCursorPosition.X = 0;

csbiInfo.dwCursorPosition.Y = 0;

SetConsoleCursorPosition(hConsoleOut,csbiInfo.dwCursorPosition);

}

+TextColor(int color) : thay đổi màu chữ

Code:

void TextColor (int color)

{

SetConsoleTextAttribute(GetStdHandle(STD\_OUTPUT\_HANDLE) , color);

}

Ghi chú: 3 hàm bổ trợ trên chúng em sưu tầm được trên mạng và được tích hợp sẵn trong thư viện console.h và file console.cpp được đính kèm file nộp.

**3.Tổng quan về Game :**

Là 1 game cổ điển huyền thoại như đã được giới thiệu

Sau đây chúng em xin trình bày về điều khiển , nguyên lí ăn điểm ,quy tắc của game giúp người chơi sau khi đọc có thể áp dụng vào thử nghiệm trò chơi :

+ Điều khiển và các phím chức năng :

-Sử dụng phím mũi tên lên , xuống , trái , phải để tương tác với khối gạch :

\* Phím lên dùng để xoay khối gạch một góc 90 độ , tùy vào người chơi chọn được khối hình phù hợp với nhu cầu của mình

\* Phím xuống dùng để di chuyển nhanh khối gạch xuống vị trí xác định không thể xuống được nữa

\* Phím trái , phải tương ứng dùng để di chuyển sang trái , sang phải khối gạch

- Một số phím chức năng :

\* ESC : Quay lại Menu game

\* Space : Tạm dừng / tiếp tục game

\* O : thay đổi màu giao diện

\* P : Bật / tắt chế độ 2 khối gạch rơi xuống (điều khiển khối gạch thứ 2 tương tự phím mũi tên lên xuống ,trái ,phải bằng các phím W,S,A,D .

+ Hệ thống tính điểm :

- Người chơi sử dụng phím xuống để cho khối gạch rơi nhanh với số điểm bằng khoảng cách nó rơi được

- Khi xếp được n hàng kín thì sẽ được cộng 100\*n điểm . n = (1,2,3,4,..)

- Sau khi qua 1 level ,tốc độ rơi sẽ giảm đi 50 ms.( ban đầu là 500 ms)

+ Xét Game Over:

- Khi có 1 khối gạch rơi hết mà chưa được hiển thị hết ra màn hình sẽ bị tính thua game.

**CHƯƠNG II : PHÂN TÍCH VÀ THIẾT KẾ GAME**

**1.PHÂN TÍCH VÀ THIẾT KẾ THUẬT TOÁN :**

**A.Tính điểm và Cấp độ :**

class Play{

public:

long deley;

long score;

long overScore;

int at;

Play(){

deley = 500;

score = 0;

overScore = 1000;

at = 1;

}

void restart(){

deley = 500;

score = 0;

overScore = 1000;

at = 1;

numBrick = 1;

flag = 0;

}

};

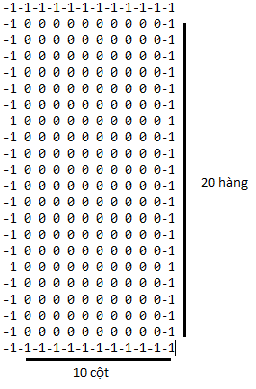
Ta khai báo luôn biến cục bộ Play Level sử dụng cho toàn bài

**B.Bảng Game ( class BoardGame)**

Bây giờ , em xin nói về cách quản lí các khối gạch trên Board , làm cách nào để quản lí các màu của từng ô , xác định trạng thái vẫn đang rơi hay đã được xếp của các khối gạch :

-void CREATE(int x = 0)

Đầu tiên , đó là hàm khởi tạo Board ,gán các giá trị chính của bảng là 0 ( tức là không có ô gạch nào ) , giới hạn là -1.Board sẽ có dạng :

****

Code:

x != 0 sẽ không khởi tạo lại khung giới hạn -1 nữa dành cho Play again .

void CREATE(int x = 0){

for (int i = 1; i < heightBoard ; i++){

for (int j = LEFT + 1; j <= LEFT + widthBoard; j++)

Board[i][j] = 0;

if(x ==0) {

Board[i][LEFT] = -1;

Board[i][LEFT + widthBoard + 1] = -1;

}

}

if(x==0){

for (int j = LEFT ; j <= LEFT + widthBoard; j++){

Board[0][j] = -1;

Board[heightBoard][j] = -1;

}

}

}

-void Update() : cập nhật trạng thái của Board

Giả sử có khối gạch BRICK brick ( biến toàn cục )

Gọi x,y là tọa độ của từng ô của ma trận của brick.arr

Ban đầu brick.board.x và brick.board.y là tọa độ của brick.arr[0][0] trên console .

Vậy cách xác định x ,y của từng brick.arr[i][j] trên Board sẽ là :

int x = brick.board.y + i;

int y = brick.board.x + j;

Lưu ý : khi x > 0 thì mới xét vào Board .

Cách quản lí là :

Ô gạch vẫn đang rơi sẽ lưu mã màu của nó=> có Board[x][y] < 16 ( chỉ có mã màu từ 1->15)

Ô gạch đã xếp xong sẽ cộng thêm 16 .tức là Board[x][y] > 16 .

Mô tả :



Với A < 16 và B-16,C-16,D-16 (B,C,D > 16 ) tương ứng là mã màu của từng khối gạch (khối gạch đã xếp có mã màu là Board[x][y] tại đó -16 ).

Code:

void Update(){

for (int i = 0; i < brick.row; i++){

for (int j = 0; j < brick.col; j++){

int x = brick.board.y + i;

int y = brick.board.x + j;

if (x > 0 && x < heightBoard){

if (brick.arr[i][j] == 1){

if (time2 - time1 <= level.deley) Board[x][y] = brick.cl;

else Board[x][y] = brick.cl + 16;

}

else{

if (Board[x][y] < 16 ) Board[x][y] = 0;

}

}

if(numBrick == 2 ) {

for (int i = 0; i < brick2.row; i++){

for (int j = 0; j < brick2.col; j++){

int x = brick2.board.y + i;

int y = brick2.board.x + j;

if (x > 0 && x < heightBoard){

if (brick2.arr[i][j] == 1){

if (time2\_2 - time1\_2 <= level.deley) Board[x][y] = brick2.cl;

else Board[x][y] = brick2.cl + 16;

}

else{

if (Board[x][y] < 16 ) Board[x][y] = 0;

}

}

}

}

}

}

Time2 và time1 là 2 biến toàn cục dùng để xử lí khoảng thời gian khi gạch chạm vào Board[x][y] > 16 còn có thể di chuyển .sẽ nói ở phần class Game.

-numBrick là biến khai báo toàn cục để xét chế độ 2 khối gạch rơi ( = 2 là 2 khối gạch)

Đoạn code sẽ tương tự như khối gạch 1.

-void show() : hàm hiển thị BoardGame ra màn hình .

Đơn giản là xét đến từng ô của Board.thay màu chữ là màu của từng ô mà in ra kí tự thể hiện từng ô của khối gạch. Tại Board[x][y] = 0 thì sẽ có màu đen trùng với màu nền.Sau đó lại xóa đi các khối đang rơi để update cho lượt tiếp theo.

Gọi f1 là kí tự in ra

ở đây em dùng #define f1 15

Code:

void show(){

for (int i = 1; i < heightBoard ; i++){

for (int j = LEFT + 1; j <= LEFT + widthBoard; j++){

gotoXY(j, i);

int cl = Board[i][j];

if(Board[i][j] > 16 ) cl-=16;

TextColor(cl);

printf("%c",f1);

if (Board[i][j] < 16 ){

Board[i][j] = 0;

}

}

}

}

-void EAT() : xử lí ăn điểm .

void EAT(){

TextColor(162);

//to mau nhung hang an

for (int i = 0; i < num; i++){

int k = \_eat[i];

level.score += 100;

for (int j = LEFT + 1; j <= LEFT + widthBoard; j++){

gotoXY(j, k);

printf("%c", f1);

}

}

num = 0;

//xu li khi an

for (int i = \_eat[0]; i > 1; i--){

for (int j = LEFT + 1; j <= LEFT + widthBoard; j++){

Board[i][j] = Board[i - 1][j];

}

}

//xu li sau khi an

for (int i = \_eat[0]; i > 1; i--){

for (int j = LEFT + 1 ; j <= LEFT + widthBoard ; j++){

if(Board[i][j] > 16) {

if(Board[i][j-1] == 0 && Board[i][j+1] == 0 && Board[i+1][j] == 0 ){

int x = i;

while(Board[++x][j] == 0){

Board[x][j] = Board[x-1][j] ;

Board[x-1][j]= 0;

};

}

}

}

}

TextColor(color);

}

-Ở đây sẽ nói đến phần xử lí ăn điểm.đơn giản ta chỉ cần xóa đi hàng ăn khỏi Board.

Tiếp đến là sau khi ăn,sẽ có trường hợp viên gạch bị lơ lửng trên không.vì vậy cần cho nó rơi xuống đến khi không thể rơi nữa.

-Ta khai báo biến cục bộ BoardGame bg sử dụng cho toàn bài

**C.Khối gạch : (class BRICK)**

- void setBrick() : Gán giá trị 1 cho tất cả các ô trong arr.

Hàm này chỉ dùng bổ trợ khi khởi tạo khối gạch:

void setBrick(){

for (int i = 0; i < row; i++)

for (int j = 0; j < col; j++)

arr[i][j] = 1;

}

- void CREATE(int id,int vt) : khởi tạo khối gạch

Id sẽ tương ứng với chỉ số của từng hình nói ở chương trước ( id = 0,1,2,3,4,5,6)

Khi khởi tạo ,ta tạo vị trí tọa độ MARK board của khối gạch (tọa đọ này trỏ đến arr[0][0] của khối gạch ) trên console ban đầu là (vt,-1)

+ Biến cl sẽ khởi tạo là mã màu ngẫu nhiên từ 1 -> 15 bằng cách sử dụng hàm rand() trong thư viện <time.h>

Code:

void CREATE(int id,int vt){

switch (id){

case 0:

row = 4;

col = 1;

setBrick();

break;

case 1:

row = col = 2;

setBrick();

break;

case 2:

row = 2;

col = 3;

setBrick();

arr[0][0] = arr[0][2] = 0;

break;

case 3:

row = 3;

col = 2;

setBrick();

arr[0][0] = arr[1][0] = 0;

break;

case 4:

row = 3;

col = 2;

setBrick();

arr[0][1] = arr[1][1] = 0;

break;

case 5:

row = 3;

col = 2;

setBrick();

arr[2][1] = arr[0][0] = 0;

break;

case 6:

row = 3;

col = 2;

setBrick();

arr[0][1] = arr[2][0] = 0;

break;

}

this->board.x = vt;

this->board.y = -1;

cl = 1 + rand() % 15;

}

- void rotate() : hàm xoay khối gạch

+khai báo Brick b để lưu ma trận xoay lại

+ Xoay khối gạch thực chất ta chỉ cần xoay ma trận trạng thái của khối gạch đó 1 góc 90o theo chiều kim đồng hồ ,vì ma trân ở đây không phải là ma trận vuông nên khi xoay thì kích thước của ma trận sẽ bị thay đổi b.row=this->col, b.col=this ->row.Ta chỉ cần viết 1 hàm xoay duy nhất thì có thể xoay được tất cả các loại khối gạch.

+Sau đó xét xem vị trí của ma trận sau khi xoay tương ứng trên ma trân Board có hợp lệ hay không (không có đè lên ô Board[i][j] != 0 nào và không vượt ra khỏi giới hạn của Board).Nếu hợp lệ thì mới chấp nhận ,còn ngược lại thì vẫn giữ nguyên ma trận trạng thái như ban đầu.

+Tọa độ của khối gạch trên Board không bị thay đổi.

Code :

void rotate(){

BRICK b;

b.row = this->col;

b.col = this->row;

//xoay ma tran gan gia tri sang Brick b

for (int i = this->row – 1; i >= 0; i--){

for (int j = this->col – 1; j >= 0; j--)

b.arr[j][this->row – I – 1] = this->arr[i][j];

}

//doan code kiem tra xem khoi gach b co vi tri co the ton tai hay k

for (int i = 0; j < b.row; i++){

int x = this->board.y + i; //tinh toa do x cua tung o tren Board

for (int j = 0; j < b.col; j++){

int y = this->board.x + j; //tinh tung do y cua tung o //tren Board

if (Board[x][y] != 0 && x >= 0 && b.arr[i][j] == 1) return;

}

}

//neu phu hop thi thay doi khoi gach cua minh

this->row = b.row;

this->col = b.col;

for (int i = 0; i < this->row; i++)

for (int j = 0; j < this->col; j++)

this->arr[i][j] = b.arr[i][j];

}

- int hasLEFT() : kiểm tra xem khối gạch có đi tiếp được hay không

Nếu có trả về 1, ngược lại trả về 0.

Ý tưởng :

Để xem 1 khối gạch có di chuyển sang trái được hay không ta đi xét từng ô gạch bên trái nhất của khối gạch và xét từng ô gạch đó.nếu 1 ô bất kì không di chuyển được thì cả khối gạch không di chuyển được.

=> Giả sử tọa độ của ô gạch đó trên Board là x,y : vậy ta cần xét các ô là Board[x][y-1] .

+ Nếu x > 0 Board[x][y-1] != 0 thì là không đi được

+ Board[x][y-1] != 0 && Board[x+1][y-1] == -1 && Board[x+2][y-1] == -1 để xét trường hợp khi khối gạch chưa rơi hết ta vẫn có thể di chuyển.

int hasLEFT(){

int d;

for (int i = 0; i < this->row; i++){

int x = this->board.y + i;

for (int j = 0; j < this->col; j++)

if (this->arr[i][j] == 1) {

d = j;

break;

}

if (Board[x][this->board.x + d - 1] != 0 && this->board.y >= 0) return 0;

if(Board[x+2][this->board.x + d - 1] == -1 ) return 0;

}

return 1;

}

-int hasRIGHT() : kiểm tra xem khối gạch có đi sang phải được hay không

Nếu có trả về 1, ngược lại trả về 0.

Ý tưởng :

Tương tự như hasLEFT() , các ô cần xét là Board[x][y+1].

int hasRIGHT(){

for (int i = 0; i < this->row; i++){

int x = this->board.y + i;

for (int j = 0; j < this->col; j++){

int y = this->board.x + j;

if (this->arr[i][j] == 1){

if ( (Board[x][y + 1] < 0 || Board[x][y+1] > 16) && this->board.y >= 0) return 0;

if ( Board[x][y+1] != 0 && Board[x+1][y + 1] == -1 ) return 0;

}

}

}

return 1;

}

Int hasDOWN() : kiểm tra xem khối gạch có đi xuống được hay không

Nếu có trả về 1, ngược lại trả về 0.đồng thời kiểm tra gameover ( trả về 2)

Ý tưởng :

Tương tự như 2 phương thức trên. Các ô cần xét là [x+1][y] :

+ nếu x+1 == HeightBoard hoặc Board[x+1][y] đã có gạch xếp rồi ( > 16 ) thì không di chuyển xuống được .

Trong trường hợp x<=0 mà gạch không rơi xuống được nữa thì trả về 2

int hasDOWN(){

for (int i = 0; i < this->row; i++){

int x = this->board.y + i;

for (int j = 0; j < this->col; j++){

int y = this->board.x + j;

if (this->arr[i][j] == 1){

if (Board[x + 1][y] > 16 && x <= 0 && (x+1) > 0) return 2;

if ((x+1 == heightBoard || Board[x + 1][y] > 16) && x >= 0) return 0;

}

}

}

return 1;

}

-Ta khai báo biến cục bộ BRICK brick,brick2 sử dụng cho toàn bài

**D.Xử lí game (class Game)**

Giả sử có biến toàn cục giao diện InterFace ITF .

-int kbhit() : xử lí tương tác người chơi

TextColor(color);

if (\_kbhit()){

char key = \_getch();

if (key == 75){

if (brick.hasLEFT()){

brick.board.x--;

}

}

else if (key == 77){

if (brick.hasRIGHT()){

brick.board.x++;

}

}

else if (key == 72)

brick.rotate();

else if (key == 80){

if (brick.hasDOWN()){

while (brick.hasDOWN()) {

level.score++;

brick.board.y++;

}

}

}

else if(key == 'p' || key == 'P'){

if(flag %2 ==0){

numBrick = 2;

id\_2 = rand()%7;

brick2.CREATE(id\_2,LEFT + 2);

id\_2= rand()%7;

gotoXY(LEFT -25 ,17);

printf("%cUsing A,S,D,W to ",16);

gotoXY(LEFT -25 ,18);

printf("move the left-brick");

}

else {

numBrick = 1;

ITF.clear(LEFT - 14 ,2,8,6);

ITF.clear(LEFT-25,17,19,2);

}

flag++;

}

else if(key == 'o' || key == 'O'){

color = 1 + rand()%15;

changeColor = 1;

}

else if(key == 27){

clrscr();

pause = 1;

ITF.showMenu();

return 1;

}

else if(key == 32) {

gotoXY(LEFT+3,5);

printf("PAUSE");

while(1){

char k = \_getch();

if(k == 32) {

ITF.clear(LEFT+3,5,5,0);

break;

}

};

}

if(numBrick ==2 ) {

if (key == 'a' || key=='A'){

if (brick2.hasLEFT()){

brick2.board.x--;

}

}

else if (key == 'd' || key == 'D' ){

if (brick2.hasRIGHT()){

brick2.board.x++;

}

}

else if (key == 'w' || key == 'W')

brick2.rotate();

else if (key == 's' || key == 'S'){

if (brick2.hasDOWN()){

while (brick2.hasDOWN()) {

level.score++; brick2.board.y++;

}

}

}

}

}

return 0;

}

- Trong hàm này :

75,77,72,80 tương ứng là mã của phím mũi tên trái ,phải,lên ,xuống với cách xử lí tương ứng chức năng của phím.

Xử lí chung ở đây là : kiểm tra xem có thể đi được hay không rồi thay đổi tọa độ cho hợp lí .

-‘P’ ở đây là tùy chọn chế độ 2 khối gạch rơi (bật/tắt):

Biến flag dùng để đánh dấu in ra dòng chữ hướng dẫn điều khiển khối gạch 2(lẻ thì in ra,chẵn sẽ xóa đi ).sau mỗi lần ấn P thì flag ++.

Ta tạo sẵn BRICK brick2,id\_2 là biến toàn cục

Tạo id\_2 ngẫu nhiên và khởi tạo khối gạch mới brick2.CREATE(id\_2,LEFT +2 ) là xong.(LEFT +2 là vị trí rơi của gạch 2.)

-‘O’ là thay đổi thêm khung :

-Khai báo biến color,changeColor là biến toàn cục .

Đơn giản chỉ là thay đổi màu chữ với biến color (màu chữ) ,gán changeColor = 1 .(trạng thái có đổi màu hay không ,1 là đổi , 0 là không).

- 27 , 32 là mã của phím ESC và Space với chức năng lần lượt là quay ra menu (return 1) và tạm dừng màn hình chơi game.

-Trong khung numBrick = 2 tương tự các phím di chuyển là A,D,W,S .

-void Control() : đưa hàm kbhit() vào xử lí và căn thời gian rơi của gạch

Biến toàn cục : int t1,t2,check .

Code :

int Control(){

int x = this->kbhit();

if(x == 1) return 1;

//xu li thoi gian roi

if (check == 0){

t1 = clock();

t2 = clock();

check = 1;

}

else{

t2 = clock();

if (t2 - t1 >level.deley){

check = 0;

if (brick.hasDOWN()) {

brick.board.y++;

}

if(numBrick == 2 && brick2.hasDOWN()){

brick2.board.y++;

}

}

}

return 0;

}

+Nếu kbhit() = 1 tức là đã nhấn ESC thì sẽ không xử lí rơi nữa.

-int Process() : Xử lí gameOver,ăn điểm,kiểm tra xác định khối gạch đã xếp và tạo khối gạch mới .

int time1,time2,check2 = 0,time1\_1,tim2\_2,check2\_2 = 0 là các biến toàn cục được sử dụng.

ở phần kiểm tra ăn hay không chỉ cần xét từng hàng nếu có đầy gạch thì thêm hàng đó vào mảng \_eat là mảng int được khai báo toàn cục,num là số phần tử của mảng cũng được khai báo toàn cục.

=>nếu num > 0 thì gọi đến hàm xử lí ăn của BoardGame. Với bg là biến Boardgame toàn cục.

int Process(){

//kiem tra gameover

if (brick.hasDOWN() == 2 || (brick2.hasDOWN() == 2 && numBrick == 2) ) return 2;

//kiem tra an hay ko

for (int i = 20; i > 0; i--){

int eat = 0;

for (int j = LEFT + 1; j <= LEFT + widthBoard; j++){

if (Board[i][j] < 16 ) {

eat = 1;

break;

}

}

if (eat == 0) \_eat[num++] = i;

}

if (num > 0) bg.EAT();

if(numBrick == 2){

if (!brick2.hasDOWN()){

if (check2\_2 == 0){

time1\_2 = clock();

time2\_2 = clock();

check2\_2 = 1;

}

else{

time2\_2 = clock();

bg.Update();

if (time2\_2 - time1\_2 > level.deley){

time1\_2 = time2\_2;

brick2.CREATE(id\_2,LEFT + 2);

id\_2 = rand()%7;

check2\_2 = 0;

return 3;

}

}

}

}

if (!brick.hasDOWN()){

if (check2 == 0){

time1 = clock();

time2 = clock();

check2 = 1;

}

else{

time2 = clock();

bg.Update();

if (time2 - time1 > level.deley){

time1 = time2;

brick.CREATE(id,LEFT + 6);

id = rand()%7;

check2 = 0;

}

}

return 3;

}

return 0;

}

- Khi có khối gạch đã rơi hết sẽ tạo brick mới , sau đó, tạo id mới ,trả về 3 để phục vụ cho việc vẽ khối tiếp the

**E.Giao diện (class InterFace)**

-Mỗi người sẽ có 1 cách thiết kế giao diện riêng

Ở phần này ,em sẽ chỉ nói đến phần hiển thị khối gạch tiếp theo :

-void DrawNext(int d,int x,int m) : m là giá trị của Process

Nếu m =3 sẽ xóa khối gạch hiển thị ở ô next trước đó đi.

Id là mã của khối gạch đã được tạo ở hàm Process

void drawNext(int id,int x,int m){

//ve khung

TextColor(color);

gotoXY(x-2,2);

printf("\_Next\_");

for(int i = 0 ; i < 4 ; i++){

gotoXY(x-3,3 + i);

putchar(16);

gotoXY(x+4,3 + i);

putchar(17);

}

for(int i = 0 ; i < 8 ; i++){

gotoXY(x-3+i,7);

putchar(196);

}

gotoXY(x-3,7);

putchar(192);

gotoXY(x+4,7);

putchar(217);

//hien thi ra khoi gach tiep theo

BRICK b;

b.CREATE(id,LEFT + 7);

if(m == 3){

for(int i = 0 ; i < 4; i++){

for(int j = 0 ; j < 4 ; j ++){

gotoXY(x + j,3+ i);

putchar(' ');

}

}

}

for(int i = 0 ; i < b.row ; i++){

for(int j = 0 ; j < b.col ; j ++){

gotoXY(x+ j,3 + i);

if(b.arr[i][j]==1){

putchar(f1);

}

}

}

}

Sau đây là các đoạn code vẽ lên giao diện của nhóm em:

-void clear(int x,int y,int k,int p ) :xóa 1 phần màn hình console bắt đầu từ(x,y) sang phải k đơn vị,xuống dưới p đơn vị :

void clear(int x , int y, int k,int p){

//xoa bat dau tu toa do (x,y) sang phai k , xuong duoi p don vi

for(int i = 0 ; i < k ; i++){

for(int j = 0 ; j < p ; j++){

gotoXY(x+i,y+ j);

putchar(' ');

}

}

}

-void show() : vẽ khung và các chú thích

void show(){

TextColor(color);

for (int i = 1; i <= 20; i++){

gotoXY(LEFT - 2, i);

printf("%d ", i);

gotoXY(LEFT, i);

printf("%c", 178);

gotoXY(LEFT + widthBoard + 1, i);

printf("%c", 178);

}

for (int i = LEFT + 1; i <= widthBoard + LEFT; i++){

gotoXY(i, 21);

putchar(206);

gotoXY(i, 0);

putchar(206);

}

gotoXY(LEFT, 0);

putchar(3);

gotoXY(LEFT, 21);

putchar(6);

gotoXY(LEFT + widthBoard + 1, 0);

putchar(3);

gotoXY(LEFT + widthBoard + 1, 21);

putchar(6);

//TIP

gotoXY(LEFT -25 ,10);

printf("%c<P>: [ON]/[OFF] ",16);

gotoXY(LEFT - 19,11);

printf("2 BRICK model");

gotoXY(LEFT -25 ,13);

printf("%c<Space> : Pause/Play",16);

gotoXY(LEFT -25 ,14);

printf("%c<ESC> : menu ");

gotoXY(LEFT - 25,15);

printf("%c<O> : Change Theme");

if(changeColor == 1 && numBrick ==2){

gotoXY(LEFT -25 ,17);

printf("%cUsing A,S,D,W to ",16);

gotoXY(LEFT -25 ,18);

printf("move the left-brick");

}

gotoXY(LEFT + widthBoard + 8 ,17);

printf("%cUsing %c,%c,%c,%c to ",16,24,25,26,27);

gotoXY(LEFT + widthBoard + 8 ,18);

printf("move the right-brick");

}

-void GameOver() : hiển thị khung game Over

void GameOver(){

clear(LEFT - 2,3,6 + widthBoard,6);

for(int i = 0 ; i < widthBoard + 10 ;i++){

gotoXY(LEFT - 4 + i ,3);

putchar(196);

gotoXY(LEFT - 4 + i ,9);

putchar(196);

}

TextColor(14);

gotoXY(LEFT +1,4);

printf("GAME OVER");

gotoXY(LEFT -1 , 5);

printf("Your Score : %d",level.score);

gotoXY(LEFT + 1 , 6);

printf("<1> Play Again");

gotoXY(LEFT + 1 , 7);

printf("<2> Goto Menu ");

gotoXY(LEFT + 1 , 8);

printf("<3> Exit");

}

-void showLevel() : hiển thị thông tin cấp độ,điểm,tốc độ:

void showLevel(){

TextColor(color);

if(level.score >= level.overScore){

level.at++;

if(level.deley > 50 ) level.deley -= 50;

level.overScore +=1000;

}

gotoXY(LEFT + widthBoard + 8 ,10);

printf("%cLevel : %d",16,level.at);

gotoXY(LEFT + widthBoard + 8 ,11);

printf("%cScore : %ld / %ld ",16,level.score,level.overScore);

gotoXY(LEFT + widthBoard + 8 ,12);

printf("%cSpeed: %ld (ms) ",16,level.deley);

}

-void showMenu() : menu lúc mới mở game

char showMenu(){

int lf = 0;

int f = 0;

while(1){

int mau = rand()%16;

TextColor(mau);

Sleep(150);

gotoXY(lf,3);

printf(" \_|\_|\_|\_|\_| \_|\_|\_|\_| \_|\_|\_|\_|\_| \_|\_|\_| \_|\_|\_| \_|\_|\_| ");

gotoXY(lf,4);

printf(" \_| \_| \_| \_| \_| \_| \_| ");

gotoXY(lf,5);

printf(" \_| \_|\_|\_| \_| \_|\_|\_| \_| \_|\_| ");

gotoXY(lf,6);

printf(" \_| \_| \_| \_| \_| \_| \_| ");

gotoXY(lf,7);

printf(" \_| \_|\_|\_|\_| \_| \_| \_| \_|\_|\_| \_|\_|\_|");

char s[] = "Thuc hien de tai NCKH - CNTT1 - UTC - K55";

if(f == 0 ){

int sp;

if(pause == 0) sp = 50;

else sp = 0;

TextColor(6);

for(int i = 0 ; i < strlen(s) ; i++ ){

Sleep(sp);

gotoXY(lf +10 + i,10);

putchar(s[i]);

}

f = 1;

}

TextColor(10);

if(pause == 1) {

gotoXY(30,13);

printf("0.Continue");

}

gotoXY(30,15);

printf("1.New Game");

gotoXY(30,17);

printf("2.Exit");

if(kbhit()){

char key = \_getch();

if(key == '1' || key == '2' ) {

clrscr();

return key;

}

}

if(pause == 1) return 'c';

}

}

-Ta khai báo biến cục bộ InterFace ITF sử dụng cho toàn bài

**CHƯƠNG III : XÂY DỰNG GAME**

**1. Xây dựng hàm main (vòng lặp cho game) :**

Cấu trúc chương trình sẽ là :

int main(){

-khởi tạo bg

-khởi tạo color ban đầu

-Khai báo biến class Game

-khởi tạo bộ đếm ngẫu nhiên

-khởi tạo id

-khởi tạo khối gạch đầu tiên

Menu:

-xử lí chọn menu

-vòng lặp cho game

}

Code:

int main()

{

bg.CREATE();

color = 7;

Game game;

srand(time(NULL));

id = rand() % 7;

brick.CREATE(id,LEFT + 6);

id = rand() % 7;

menu:

char k = ITF.showMenu();

if(k == '2') return 0;

ITF.showTIP();

while(\_getch() != 13);

start:

clrscr();

ITF.show();

int m = 0;

//vong lap cho game

while (1){

if(changeColor == 1){

//xu li doi mau nen

TextColor(color);

ITF.show();

changeColor = 0;

}

bg.Update();

bg.show();

game.Control();

if(pause == 1){

//xu li quay lai menu

while(1){

char key = \_getch();

if(key == '0') {

clrscr();

ITF.show();

pause = 0;

break;

}

else if(key == '1'){

pause =0;

bg.CREATE(1);

level.restart();

id = rand()%7;

brick.CREATE(id,LEFT + 7);

goto start;

}

else if(key == '2') return 0;

};

}

m = game.Process();

if (m == 2 ){

//xu li gameOver

ITF.GameOver();

while(1){

char key=getch();

if(key == '1') {

bg.CREATE(1);

level.restart();

goto start;

}

if(key == '2') {

clrscr();

bg.CREATE(1);

level.restart();

goto menu;

}

if(key == '3') return 0;

}

}

//ve khung next

ITF.drawNext(id,LEFT + widthBoard + 11,m);

if(numBrick == 2) ITF.drawNext(id\_2,LEFT - 11,m);

ITF.showLevel();

};

}

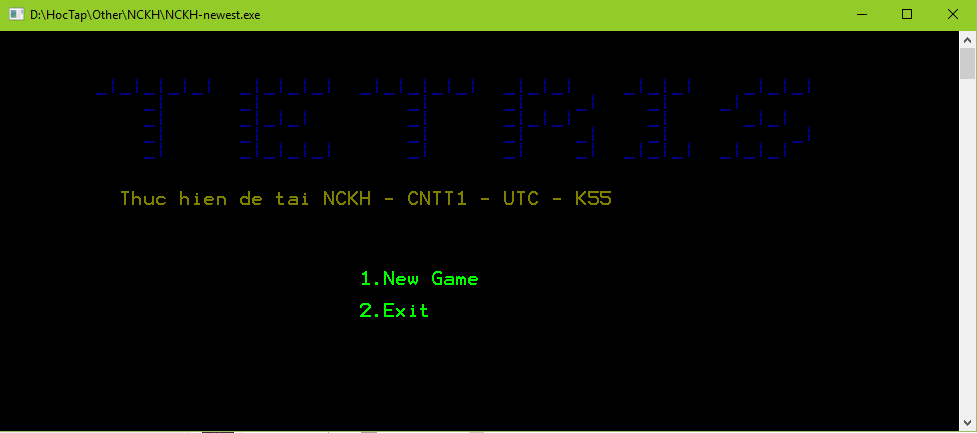
**KẾT THÚC.**

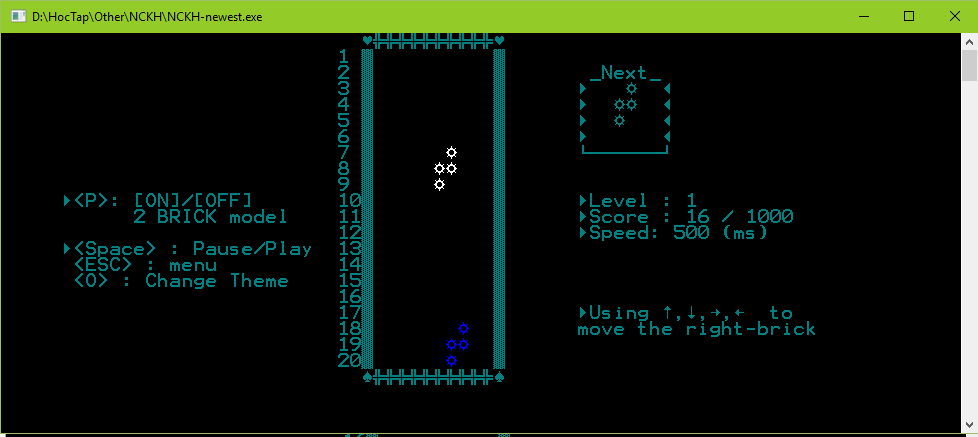
-Bản báo cáo đề tài của chúng em đến đây là kết thúc.Do kiến thức còn hạn hẹp nên chương trình chắc sẽ còn nhiều thiếu sót. Chúng em mong các thầy cô giáo trong khoa đánh giá ,chỉ bảo thêm để chương trình được hoàn thiện một cách tốt nhất !

Nhóm chúng em xin chân thành cảm ơn !

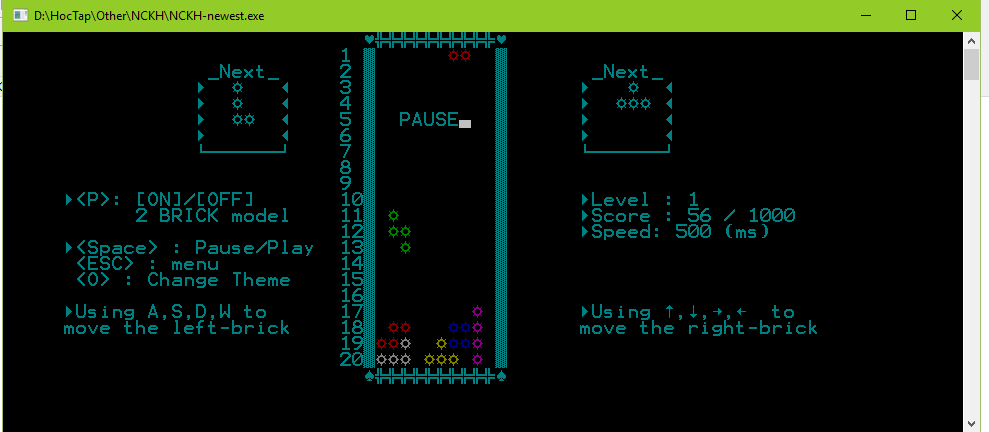
Một số hình ảnh trong game :

Menu chính :

New game :



Chế độ 2 viên gạch rơi:



Game Over :

