BỘ GIÁO DỤC VÀ ĐÀO TẠO

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

KHOA CÔNG NGHỆ THÔNG TIN

----- □ & □ -----

A picture containing logo

Description automatically generated

**SEMINAR PROJECT**

**THƯ VIỆN MFC**

PHƯƠNG PHÁP LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

Thành phố Hồ Chí Minh – 2020

BỘ GIÁO DỤC VÀ ĐÀO TẠO

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

KHOA CÔNG NGHỆ THÔNG TIN

----- □ & □ -----

A picture containing logo

Description automatically generated

**SEMINAR PROJECT**

**( THƯ VIỆN MFC)**

**( Giáo viên hướng dẫn )**

TS. Nguyễn Minh Huy

TS. Nguyễn Thị Minh Tuyền

Th.S Trần Duy Quang

Th.S Hồ Tuấn Thanh

Thành phố Hồ Chí Minh – 2020

**MỤC LỤC**

---- □ & □ -----

Mục lục:

[MỤC LỤC ẢNH: 3](#_Toc90029621)

[LỜI CẢM ƠN 4](#_Toc90029622)

[THÔNG TIN CHUNG 5](#_Toc90029623)

[PHÂN CHIA CÔNG VIỆC 6](#_Toc90029624)

[GIỚI THIỆU THƯ VIỆN MFC 10](#_Toc90029625)

[**I.** **Khái niệm:** 10](#_Toc90029626)

[**II.** **Một số lớp cơ bản** 10](#_Toc90029627)

[a. Lớp quản lý tiểu trình **CwinThread:** 10](#_Toc90029628)

[b. Lớp quản lý tiểu trình giao diện chính **CwinApp** 11](#_Toc90029629)

[c. Một số lớp thường dùng: 12](#_Toc90029630)

[**III.** **Ưu điểm và nhược điểm khi lập trình Windows với MFC** 13](#_Toc90029631)

[a. Ưu điểm: 13](#_Toc90029632)

[b. Nhược điểm: 13](#_Toc90029633)

[**IV.** **Đồ án minh họa cho việc sử dụng MFC trong lập trình Windows:** 13](#_Toc90029634)

[XÂY DỰNG ĐỒ ÁN 14](#_Toc90029635)

[**I.** **HỌP ĐẦU ĐỒ ÁN** 14](#_Toc90029636)

[**II.** **GIAI ĐOẠN CODE** 14](#_Toc90029637)

[**1.** **Phần giao diện:** 14](#_Toc90029638)

[**1.1 Cấu trúc chung của một file code giao diện, các hàm mặc định** 15](#_Toc90029639)

[**1.2 Các hàm sử dụng cho code Calculate:** 17](#_Toc90029640)

[**1.3 Thông tin bổ sung:** 20](#_Toc90029641)

[**2.** **Phần chức năng** 21](#_Toc90029642)

[**2.1 Thuật toán:** 21](#_Toc90029643)

[**2.2 Coding:** 22](#_Toc90029644)

[**III.** **GỘP CODE VÀ TEST** 25](#_Toc90029645)

[**IV.** **LÀM REPORT** 38](#_Toc90029646)

[**V.** **HỌP CUỐI ĐỒ ÁN** 39](#_Toc90029647)

[**VI.** **SOURCE CODE** 39](#_Toc90029648)

[NGUỒN THAM KHẢO 40](#_Toc90029649)

# MỤC LỤC ẢNH:

[Hình 1 - Sơ đồ thuật toán hàm addInput 22](file:///C:\Users\dinhv\Downloads\Group13\Report.docx#_Toc90029650)

[Hình 2 - Tính toán với phép cộng - một toán tử 25](#_Toc90029651)

[Hình 3- Tính toán với phép cộng – nhiều toán tử 26](#_Toc90029652)

[Hình 4- Tính toán với phép trừ - một toán tử 27](#_Toc90029653)

[Hình 5- Tính toán với phép trừ - nhiều toán tử 28](#_Toc90029654)

[Hình 6- Tính toán với phép nhân – một toán tử 29](#_Toc90029655)

[Hình 7 - Tính toán với phép chia – một toán tử 31](#_Toc90029656)

[Hình 8- Tính toán với biểu thức 33](#_Toc90029657)

[Hình 9- Tính toán với số thực 34](#_Toc90029658)

[Hình 10- Trước khi xóa 35](#_Toc90029659)

[Hình 11 - Sau khi xóa 36](#_Toc90029660)

[Hình 12 - Lỗi sai khi chia mẫu bằng 0 37](#_Toc90029661)

# LỜI CẢM ƠN

*Kính gửi thầy Huy, cô Tuyền, thầy Quang và thầy Thanh,*

*Cảm ơn thầy Huy vì tâm huyết cũng như thời gian, trí lực của thầy đã bỏ ra để chăm lo cho từng buổi lên lớp, cho từng bài giảng. Cảm ơn thầy vì đã cho chúng em rất nhiều kiến thức bổ ích. Kiến thức thầy truyền đạt cho chúng em đã giúp ích chúng em rất nhiều trong việc hoàn thành đồ án này.*

*Cảm ơn thầy Quang, thầy Thanh và cô Tuyền vì đã bỏ ra tâm huyết và thời gian để giảng dạy chúng em trong các tiết thực hành và giải đáp những thắc mắc của chúng em về môn học, và những kiến thức bên lề môn học. Sự vui vẻ, nhiệt tình và thú vị của thầy cô đã giúp cho các tiết thực hành trở nên nhẹ nhàng hơn và kiến thức truyền đạt cũng dễ dàng hơn.*

*Cảm ơn về đồ án lần này của thầy\cô vì đã cho tụi em cơ hội để tìm hiểu sâu hơn về những gì chúng em đã được học và cụ thể hơn là thư viện MFC. Ngoài ra, đồ án còn cho tụi em hiểu rõ về các bạn cùng nhóm hơn, rèn luyện thêm được các kỹ năng mềm cần thiết như kỹ năng làm việc nhóm, kỹ năng giao tiếp, kỹ năng phân chia công việc, kỹ năng tìm kiếm, kỹ năng biện luận,…*

*Chúc thầy/cô ngày ngày vui vẻ, sức khỏe, và luôn hạnh phúc. Chúc cho con đường đến với “thành công” của thầy/cô luôn thuận lợi.*

*Cảm ơn thầy, cô.*

*Trân trọng,*

*XOOP*

# THÔNG TIN CHUNG

**Môn học:** Phương pháp lập trình hướng đối tượng

**Chủ đề của đồ án:** Topic: Thư viện MFC

**Giảng viên giảng dạy:**

* TS. Nguyễn Minh Huy
* TS. Nguyễn Thị Minh Tuyền
* Th.S Trần Duy Quang
* Th.S Hồ Tuấn Thanh

**Danh sách sinh viên thực hiện đồ án:**

**STT** **Tên sinh viên**  **Mã số sinh viên**

**1** Nguyễn Đình Văn 20127662

**2** Phạm Thảo Vy 20127417

**3** Lê Hải Đăng 20127129

**4** Lưu Minh Phát 20127061

# PHÂN CHIA CÔNG VIỆC

**XÂY DỰNG Ý TƯỞNG VÀ CODE MINH HỌA MÁY TÍNH BỎ TÚI**

**BẰNG THƯ VIỆN MFC**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Họ tên & MSSV** | **Nội dung** | **Mức độ hoàn thành công việc** |
| **GIAO DIỆN** | **NGUYỄN ĐÌNH VĂN-20127662** | Thiết kế giao diện:  -Thiết kế, edit bảng history, hiển thị history  -Thiết kế, edit bảng result, hiển thị result  -Thiết kế, chỉnh màu background  Code:  void OnEnChangeEditHistory();  void OnEnChangeEditResult();  HBRUSHCCalculatorDlg::OnCtlColor(CDC\* pDC, CWnd\* pWnd, UINT nCtlColor) (Thiết kế màu sắc cho mỗi đối tượng)  void resetOutput();  void reset();  void doOperation (Calculator::ActionType operation, bool handleNumber = true);  void createHistoryText();  void delText();  Phần báo cáo:  - Phân chia công việc  - Các chức năng của phần đồ họa- Mục lục  - Báo cáo các tính năng của máy tính.  Tổng hợp:  - Tổng hợp code giao diện  - Tổng hợp code cả bài  - Tổng hợp các tính năng của máy tính  - Thuyết trình | 100% |
| **PHẠM THẢO VY-20127417** | Thiết kế giao diện:  -Thiết kế, edit các nút số từ 0->9  -Thiết kế, edit nút DEL, AC  -Thiết kế, edit các nút toán tử +, -, x, /, =  -Thiết kế, edit nút dos  Code:  void OnBnClickedButtonEqual();  void OnBnClickedButtonDel();  void OnBnClickedButtonAc();  void OnBnClickedButtonPlus();  void OnBnClickedButtonSub();  void OnBnClickedButtonMul();  void OnBnClickedButtonDiv();  void OnBnClickedButtonDos();  void OnBnClickedButton0();  void OnBnClickedButton1();  void OnBnClickedButton2();  void OnBnClickedButton3();  void OnBnClickedButton4();  void OnBnClickedButton5();  void OnBnClickedButton6();  void OnBnClickedButton7();  void OnBnClickedButton8();  void OnBnClickedButton9();  void addSymbol(char digit);  Phần báo cáo:  - Giới thiệu MFC  - Qui trình CV  - Tổng hợp nộp dung báo cáo  - Thuyết trình | 100% |
| **CHỨC NĂNG** | **LÊ HẢI ĐĂNG-20127129** | -Thiết lập thuật toán và lập trình dấu chấm thập phân (dos)  -Thiết lập thuật toán và lập trình DEL  -Thiết lập thuật toán và lập trình xóa một phần tử ([C])  Code:  void Calculator::LeftExpression::reset();  void Calculator::LeftExpression::set(double value);  void Calculator::LeftExpression::add(double value);  void Calculator::LeftTerm::reset();  void Calculator::LeftTerm::set(double value);  void Calculator::LeftTerm::multiplyBy(double value);  Calculator::Action Calculator::getLastInput() const;  void Calculator::reset();  bool Calculator::isOperation(ActionType action) const;  bool Calculator::isTerm(ActionType action) const;  bool Calculator::isExpression(ActionType action) const;  Calculator::ActionType Calculator::getLastOperation();  double Calculator::getCurrentResult() const;  double Calculator::getCurrentResult() const;  PPT:   * Slide mở đầu * Slide kết thúc   Tổng hợp:  - Tổng hợp code chức năng  - Thuyết trình | 100% |
| **LƯU MINH PHÁT-20127061** | -Thiết lập thuật toán và lập trình chức năng +, -, x, /  -Thiết lập thuật toán và lập trình chức năng =  Code:  bool Calculator::addInput(const Action& input);  PPT:   * Slide giới thiệu thành viên * Slide các thư viện MFC * Slide ưu điểm nhược điểm * Slide các tính năng của máy tính * Slide Code giao diện và code thuật toán   Tổng hợp:   * Tổng hợp slide * Code addInput * Thiết lập chức năng +, -, x, / cho thuật toán | 100% |

# GIỚI THIỆU THƯ VIỆN MFC

**Thư viện Microsoft Foundation Class (MFC)**

1. **Khái niệm:**

* MFC là thư viện của Microsoft, bao gồm các lớp cơ bản, và được cài đặt bằng ngôn ngữ C++, hỗ trợ cho việc lập trình trong môi trường hệ điều hành Windows. Từ cac lớp này, MFC xác lập nền tảng hình thành ứng dụng của Windows, bao gồm việc định nghĩa bộ khung ứng dụng, các công cụ chuẩn và phổ biến để bổ sung vào bộ khung nói trên nhằm tạo ra ứng dụng hoàn chỉnh. Với thư viện MFC, công việc của người lập trình nhẹ nhàng hơn khi chỉ còn là lựa chọn các thành phần cần thiết, điều chỉnh và phối hợp chúng sao cho hợp lý để có được kết quả như mong muốn.

1. **Một số lớp cơ bản**
   1. Lớp quản lý tiểu trình **CwinThread:**

CwinThread là một lớp của MFC, lớp đối tượng quản lý tiểu trình được tạo ra trong tiến trình của một ứng dụng đang thực hiện. Các dịch vụ cơ bản phục vụ cho quản lý tiểu trình do CwinThread cung cấp thông qua các thuộc tính và hành vi của nó.

* DWORD *m\_nThread ID* : Thuộc tính lưu số hiệu của tiểu trình
* CWND *m\_pMain Wnd* : Lưu con trỏ đối tượng cửa sổ giao diện chính của tiểu trình. Khi cửa sổ giao diện chấm dứt hoạt động, tiểu trình liên quan sẽ kết thúc. Nếu tiểu trình thuộc loại tiểu trình xử lí nội thì gía trị này kế thừ từ tiểu trình giao diện cấp cao hơn.
* *CwinThread*() : Tạo lập đối tượng tiểu trình
* Vitual BOOL *initinstance* (): Khởi tạo thông số cho đối tượng tiểu trình và đảm nhận các xử lý bổ sung khác của ứng dụng. Đối với tiểu trình giao diện, hành vi này dùng để khởi tạo đối tượng cửa sổ giao diện và gán địa chỉ của đối tượng của sổ cho *m\_pMain Wnd*
* Virtual int *Exitinstance* (): Kết thúc của đối tượng tiểu trình. Thông qua hoạt động này, đối tượng quản lí tiểu trình thực hiện hoàn trả các tài nguyên của hệ thống mà nó đã đăng kí sử dụng.
* Virtual *Run* (): Hoạt động dành riêng cho tiểu trình giao diện, nó thực hiện vòng lặp nhận message, chuyển message cho hoạt động PreTranslateMessage của lớp. Nếu hoạt động này trả về giá trị 0 thì message sẽ tiếp tục được chuyển đến các hàm diễn dịch phiếm translateMessage và hàm điều phối message Dispatch Message .
* Virtual BOOL *preTranslateMessage* : hoạt động cho phép can thiệp trước các message nhận được từ message queue của ứng dụng. Thông qua đó, tiểu trình giao diện có thể thực hiện các tiền xử lý message đặc trưng nhằm đáp ứng yêu cầu đặt ra cho ứng dụng.
  1. Lớp quản lý tiểu trình giao diện chính **CwinApp**

CwinApp kế thừa từ CwinThread, là lớp đối tượng chuyên dùng để quản lý tiểu trình giao diện chính của ứng dụng. Ứng dụng Windows chỉ được sử dụng một đối tượng thuộc lớp này. Ngoài các thuộc tính, hoạt động kế thừa public từ CwinThread, CwinApp có thêm các thuộc tính và hoạt động bổ sung sau:

* Const char\* *m\_pszAppName* : Lưu chuỗi tên ứng dụng
* LPTSTR *m\_lpcmdLine* : Lưu nội dung chuỗi tham số dòng lệnh. Tham số dòng lệnh là toàn bộ phần nội dung mà người dùng gõ vào ngay sau chuỗi đường dẫn và tên chương trình ứng dụng khi họ thực hiện ứng dụng. Đối với ứng dụng có nhiều chế độ hoạt động khác nhau thì việc sử dụng tham số dòng lệnh là một cơ chế xác lập các giao ước giữa ứng dụng và người dùng để chọn chế độ sử dụng ứng dụng thích hợp.
* Int *m\_nCmdShow* : Lưu giá trị thông số đã được dùng để kích hoạt cửa sổ giao diện chính của ứng dụng
* *CwinApp*() : Tạo lập đối tượng tiểu trình
* HCURSOR LoadCursor (int nID): Nạp cursor từ resource của ứng dụng vào bộ nhớ (nếu cursor chưa được nạp). nID là số hiệu của cursor. Hàm trả về giá trị handle quản lý cursor. Giá trị này kiểu HCURSOR (handle of cursor) mà windows dùng để quán lí cursor.
* HICON LoadIcon (int nID): nạp icon từ recource của ứng dụng vào bộ nhớ. nID là số hiệu của icon. Hàm này trả về giá trị handle quản lí icon. Giá trị này kiểu HICON (handle of icon).
* Hoạt động hiển thị hộp thông báo và chờ nhận ý kiến người dùng:

Virtual int *DomessageBox*

{

LPCT STR lpsz prompt // nội dung thông báo

UINT nType // dạng hộp thông báo

UINT hlpindex=0 // số hiệu mục giúp đỡ (Win help)

}

* Hoạt động này được kích hoạt với tham số tương ứng mỗi khi trong chương trình sử dụng hàm *AfxmessageBox* để hiển thị hộp thông báo.
* *nType* // ấn định hộp thông báo : Giá trị này là sự kết hợp giữa thông số quy định biểu tượng hiển thị và các nút chọn bố trí trong hộp.
* Hoạt động  *DoMessageBox* : chờ người sử dụng trả lời bằng cách chọn một nút chọn xác định trong hộp thông báo. Hoạt động kết thúc với giá trị trả về là số hiệu cả một nút được chọn. Kế thừa hoạt động này để chặn và thực hiện xử lý đặc trưng cho tất cả các lời gọi *AfxMessageBox* trong ứng dụng.
* Hàm *AfxMessageBox* sử dụng *DoMessageBox* để hiển thị hộp thông báo và lấy giá trị của hành vi này làm kết quả trả về của nó.
  1. Một số lớp thường dùng:
* Lớp *CWnd*: Đây là lớp đối tượng quản lý cửa sổ của Windows. Thông qua các thuộc tính và hành vi của lớp CWnd, MFC cung cấp các dịch vụ cần thiết cho phép tạo lập và khai thác các tính năng của cửa sổ Windows một cách dễ dàng.
* Lớp *Cmenu*: Lớp quản lý Menu thông qua các thuộc tính và hành vi đặc trưng
* Các lớp đối tượng nhập liệu: Lớp đối tượng nhập liệu, kế thừa từ lớp CWnd, cho phép quản lý các mục nhập liệu trên cửa sổ giao diện
* *Cstatic*: Lớp đối tượng quản lý mục thông báo hoặc ảnh trên cửa sổ giao diện
* *Cedit*: Lớp đối tượng quản lý hộp nhập trên cửa sổ giao diện. Ngoài việc hỗ trợ xử lý các hoạt động nhập liệu, Cedit còn có khả năng thông tin cho cửa sổ cha của hộp nhập về tình hình nhập liệu đang diễn ra trong hộp
* *Cbutton*: Lớp đối tượng quản lý nút chọn trên của sổ giao diện. Trong Windows, các nút chọn có thể hoạt động độc lập hay theo nhóm.
* Lớp đối tượng quản lý hộp nhập chứa danh sách các mục chọn trên của sổ giao diện. Tùy theo loại *ListBox* mà khi sử dụng, người dùn được phép đánh dấu chọn một hoặc nhiều mục trong *ListBox*.
* Lớp quản lý hộp hội thoại *Cdialog*: Là lớp đối tượng kế thừa từ lớp *Cwnd*, được sử dụng để quản lí dialog.

1. **Ưu điểm và nhược điểm khi lập trình Windows với MFC**
   1. Ưu điểm:

* Bản chất là lập trình hướng đối tượng với C++
* Người lập trình đỡ phải nhớ nhiều hàm vầ cấu trúc vì chúng đã được gói chung vào các lớp cơ bản
* Thời gian phát triển một ứng dụng nhanh bởi sự hỗ trợ của bộ Wizard với Visual Studio
  1. Nhược điểm:
* Chương trình khi biên dịch ra sẽ khá lớn cỡ độ từ vài MB đến vài chục MB tùy vào chế độ biên dịch static link ( đưa toàn bộ vào một exe hoặc một dll duy nhất) hoặc chế độ Dynamic linked ( bắt đính kèm máy tệp dll của MFC)
* Vẫn phải đính kèm các tệp tin của Visual C++ như msvcrtX.dll và msvcppX.dll ( tùy thuộc vào phiên bản VC++)
* Đối với các ứng dụng MFC lớn, việc làm giảm tốc độ của ứng dụng so với việc xây dựng ứng dụng nhằm APIs là không thể tránh khỏi, bởi vì các lớp bao của MFC quá rườm rà phức tạp

1. **Đồ án minh họa cho việc sử dụng MFC trong lập trình Windows:**

Máy tính bỏ túi (chi tiết bên dưới )

# XÂY DỰNG ĐỒ ÁN

**CÁC BƯỚC XÂY DỰNG ĐỒ ÁN**

1. **HỌP ĐẦU ĐỒ ÁN**
   1. Thống nhất ứng dụng minh họa cho đồ án : Máy tính bỏ túi
   2. Lên ý tưởng cho đồ án
      1. Giao diện
         * Có màn hình lịch sử tính toán
         * Có màn hình hiển thị quá trình nhập và hiển thị đáp án
         * Có các nút số (từ [0] -> [9]), nút thập phân([.]), nút toán tử ([+], [-], [x], [/]), nút xóa một phần tử([DEL]), nút xóa tất cả ([AC]), nút bằng ([=])
      2. Chức năng
         * Thực hiện được các phép toán cộng trừ nhân chia
         * Xóa một phần tử
         * Xóa tất cả
   3. Phân chia công việc
      1. Giao diện: Văn & Vy
      2. Chức năng: Phát & Đăng
   4. Thống nhất thời hạn làm việc
2. **GIAI ĐOẠN CODE**
   1. **Phần giao diện:**
      1. **Tìm hiểu kiến thức:**
         * Lớp giao diện mặc định
         * Hàm khởi tạo mặc định
         * Cài đặt Phương thức hỗ trợ DDX, DDV
         * Cách khởi tạo các nút, thao tác với các nút
         * …
      2. **Thuật toán**

- Giao diện có thể chia các đối tượng thành 4 phần:

* Phần 1: Các nút chứa các kí tự “0,1,2,3,4,5,6,7,8,9, .”
* Phần 2: Các hộp hiện thông tin về lịch sử tính toán và kết quả.
* Phần 3: Các nút chứa các toán tử “+, -, \*, /”.
* Phần 4: Các nút chứa các hàm xử lý “DEL, AC”.

- Hàm addSym sẽ xử lý các tạo thành các chuỗi số khi người dùng nhập vào, Hàm doOperation sẽ xử lý các chuỗi có ở addSym và toán tử do người dùng nhập vào sau đó đưa cho thư viện Calculate tính toán và đưa ra kết quả. Chương trình sẽ xuất kết quả ra màn hình sau khi người dùng nhập toán tử “=”. Trong trường hợp người dùng muốn xóa một kí tự thì chương trình sẽ gọi hàm delText() xử lý việc xóa. Nếu người dùng muốn reset lại máy tình thì chương trình sẽ gọi hàm reSet() máy tính sẽ được đưa về trặng thái ban đầu.

* + 1. **Coding:**
* **Chú thích :** 
  + - * Màu xanh lá cây: comment của dev
      * Màu đỏ: giải thích ý nghĩa hàm

### **1.1 Cấu trúc chung của một file code giao diện, các hàm mặc định**

// CMFCDlg dialog

class CMFCDlg : public CdialogEx

Lớp giao diện mặc định được chương trình cung cấp sẵn

{

// Construction

public:

CMFCDlg(CWnd\* pParent = nullptr); // standard constructor

Hàm khởi tạo mặc định, khởi tạo một lớp giao diện có sẵn

// Dialog Data

#ifdef AFX\_DESIGN\_TIME

enum { IDD = IDD\_MFC\_DIALOG };

#endif

protected:

virtual void DoDataExchange(CDataExchange\* pDX); // DDX/DDV support

Cài đặt Phương thức hỗ trợ DDX, DDV

// Implementation

protected:

HICON m\_hIcon;

// Generated message map functions

virtual BOOL OnInitDialog();

Khỏi tạo một Dialog, khởi tạo và các đặt các thuộc tính cho các đối tượng button, edit box, ….

void OnSysCommand(UINT nID, LPARAM lParam);

void OnPaint();

Đối với các ứng dụng MFC sử dụng mô hình tài liệu / chế độ xem, điều này được thực hiện tự động cho bạn bởi khuôn khổ.

HCURSOR OnQueryDragIcon();

Hệ thống gọi hàm này để lấy con trỏ hiển thị trong khi người dùng kéo cửa sổ thu nhỏ.

DECLARE\_MESSAGE\_MAP()

};

### **1.2 Các hàm sử dụng cho code Calculate:**

* void OnEnChangeEditHistory(): Cài đặt phần hiển thị lịch sử tính toán.
* void OnEnChangeEditResult(): Cài đặt phần hiển thị kết quả tính toán.
* void OnBnClickedButtonEqual(): Khi người dùng click vào button “=” thì chương trình sẽ khởi tạo một toán tử “=”.
* void OnBnClickedButtonDel():Khi người dùng click vào button “DEL” thì chương trình sẽ khởi tạo một kí tự “d” sau đó đưa vào hàm addSym để xử lí lệnh xóa.
* void OnBnClickedButtonAc():Khi người dùng click vào button “AC” thì chương trình sẽ khởi tạo hàm reSet() để đưa máy tính về trạng thái mặc định.
* void OnBnClickedButtonPlus(): Khi người dùng click vào button “+” thì chương trình sẽ khởi tạo một toán tử “+”
* void OnBnClickedButtonSub(): Khi người dùng click vào button “-” thì chương trình sẽ khởi tạo một toán tử “-”
* void OnBnClickedButtonMul(): Khi người dùng click vào button “X” thì chương trình sẽ khởi tạo một toán tử “X”
* void OnBnClickedButtonDiv(): Khi người dùng click vào button “/” thì chương trình sẽ khởi tạo một toán tử “/”
* void OnBnClickedButtonDos(): Khi người dùng click vào button “.” thì chương trình sẽ khởi tạo một kí tự “.” Sau đó đưa vào hàm addSym() để xử lý.
* void OnBnClickedButton0():Khi người dùng click vào button “0” thì chương trình sẽ khởi tạo một kí tự “0” Sau đó đưa vào hàm addSym() để xử lý.
* void OnBnClickedButton1(): Khi người dùng click vào button “1” thì chương trình sẽ khởi tạo một kí tự “1” Sau đó đưa vào hàm addSym() để xử lý.
* void OnBnClickedButton2(): Khi người dùng click vào button “2” thì chương trình sẽ khởi tạo một kí tự “2” Sau đó đưa vào hàm addSym() để xử lý.
* void OnBnClickedButton3(): Khi người dùng click vào button “3” thì chương trình sẽ khởi tạo một kí tự “3” Sau đó đưa vào hàm addSym() để xử lý.
* void OnBnClickedButton4(): Khi người dùng click vào button “4” thì chương trình sẽ khởi tạo một kí tự “4” Sau đó đưa vào hàm addSym() để xử lý.
* void OnBnClickedButton5(): Khi người dùng click vào button “5” thì chương trình sẽ khởi tạo một kí tự “5” Sau đó đưa vào hàm addSym() để xử lý.
* void OnBnClickedButton6(): Khi người dùng click vào button “6” thì chương trình sẽ khởi tạo một kí tự “6” Sau đó đưa vào hàm addSym() để xử lý.
* void OnBnClickedButton7(): Khi người dùng click vào button “7” thì chương trình sẽ khởi tạo một kí tự “7” Sau đó đưa vào hàm addSym() để xử lý.
* void OnBnClickedButton8(): Khi người dùng click vào button “8” thì chương trình sẽ khởi tạo một kí tự “8” Sau đó đưa vào hàm addSym() để xử lý.
* void OnBnClickedButton9(): Khi người dùng click vào button “9” thì chương trình sẽ khởi tạo một kí tự “9” Sau đó đưa vào hàm addSym() để xử lý.
* HBRUSH OnCtlColor(CDC\* pDC, CWnd\* pWnd, UINT nCtlColor): Thiết kế cài đặt màu sắc cho các đối tượng button, edit box, ….
* void resetOutput(): Khởi tạo lại những giá trị khi người dùng nhập vào về “0”.
* void reset(): Đưa máy tính trở về trặng thái mặc định ban đầu.
* void addSymbol(char digit): Với mỗi digit thì hàm này sẽ xử lý các trường hợp có thể xảy ra của chuỗi số, khi thêm kí tự “digit” vào chuỗi số này.
* Trường hợp 1: Nếu người dùng nhấn nút “DEL” thì digit thu về là kí tự ‘d’ lúc này hàm sẽ hiểu đây là người dùng muốn xóa đi một giá trị thì hàm sẽ gọi delText() xử lý việc xóa.
* Trường hợp 2: Nếu người dùng đã hoàn thành xong một phép tính và muốn thực hiện phép tính mới thì hàm sẽ tạo ra một chuỗi số mới lưu thông tin về chuỗi mới của người dùng.
* Trường hợp 3: Nếu chuỗi số đang mang giá trị mặc định là “0” mà người dùng tiếp tục nhập kí tự “0” thì hàm sẽ bỏ qua.
* Trường hợp 4: Nếu chuỗi số đang ở giá trị mặc định mà người dùng nhập vào một giá trị khác mặc định thì hàm sẽ cập nhật chuỗi mới.
* Trường hợp 5: Nếu người dùng muốn nhập tiếp giá trị cho chuỗi đang nhập thì hàm sẽ cập nhật chuỗi.
* void delText(): Hàm này xử lý việc xóa đi một kí tự của history và result.
* Trường hợp 1: Nếu người dùng đang xóa đi một số. Nếu chuỗi người dùng muốn xóa đang được cập nhật thì chương trình chỉ xóa kí tự hiện tại. Nếu chuỗi người dùng muốn xóa đã được đưa vào hàm xử lý thì hàm tiến hành lấy chuỗi đó ra khỏi hàm được xử lý trước đó.
* Trường hợp 2: Nếu người dùng muốn xóa đi các toán tử “+, -, \*, /” thì chương trình tiến hành việc xóa history và lấy toán tử ra khỏi hàm xử lí trước đó.
* Trường hợp 3: Khi chuỗi rỗng thì hàm sẽ bỏ qua và không thực hiện việc xóa.
* void createHistoryText(): Cập nhật lại chuỗi lịch sử sau mỗi lần nhận toán tử.
* void doOperation(Calculator::ActionType operation, bool handleNumber = true): Hàm xử lý các toán tử khi người dùng nhập vào.
* Trường hợp 1: Nếu người dùng nhập vào các toán tử “+, -, \*, /” thì hàm sẽ tiến hành đưa chuỗi số mới nhất vào hàm addInput của thư viện Calculate sau đó hàm tiếp tục đưa toán tử vừa nhập vào hàm addInput để cho thư viện Calculate. Sau đó thư viện Calculate sẽ trả về kết quả sau khi tính toán.
* Trường hợp 2: Nếu người dùng nhập vào các toán tử “=” thì hàm sẽ tiến hành đưa chuỗi số mới nhất vào hàm addInput của thư viện Calculate sau đó hàm tiếp tục đưa toán tử vừa nhập vào hàm addInput để cho thư viện Calculate. Sau đó thư viện Calculate sẽ trả về kết quả sau khi tính toán. Hàm tiến hành cập nhật lại kết quả là lịch sử sau đó in ra màn hình.

### **1.3 Thông tin bổ sung:**

* CWinApp: lớp chính của MFC để quản lý ứng dụng. Chứa đựng vòng lặp nhận message và phân phối message đến các cửa sổ của ứng dụng
* CCalculatorApp: lớp kế thừa từ lớp CwinApp
* InitInstance(): hàm khởi tạo ứng dụng, override lên hàm chuẩn của lớp CwinApp
* CWnd: lớp chính của MFC để quản lý các loại cửa sổ giao diện. Có nhiều lớp được kế thừa từ lớp này để quản lý các loại cửa sổ khác nhau (CTreeCtrl, CListBox, Cedit,…)
* CFrameWnd: lớp kế thừa từ lớp CWnd, để quản lý cửa sổ giao diện chính của ứng dụng
* OnPaint: hàm thành phần của lớp CMainWindow, được gọi khi cần cập nhật nội dung cửa sổ. Hàm này được định nghĩa chồng lên hàm chuẩn của lớp CFrameWnd.
* MFC dùng Message Map để liên kết các message với những hàm thành phần của lớp cửa sổ  Mỗi message sẽ được xử lý bởi 1 hàm thành phần tương ứng

BEGIN\_MESSAGE\_MAP(CAboutDlg, CDialogEx)

END\_MESSAGE\_MAP()

* Dialog data exchange (DDX) là một cách dễ dàng để khởi tạo các điều khiển trong hộp thoại của bạn và thu thập dữ liệu đầu vào của người dùng. Dialog data validation (DDV) là một cách dễ dàng để xác thực mục nhập dữ liệu trong hộp thoại.
  1. **Phần chức năng**

**2.1 Thuật toán:**

**Ý tưởng:** Một phép tính là tổng của một nhiều các đơn thức, và mỗi đơn thức là tích của một hoặc nhiều các nhân tử. Vì vậy, để tính kết quả của một phép tính ta cần tính được giá trị của các đơn thức và tính tổng của chúng.

**Mô tả:**

Chương trình cần sử dụng các biến sau:

* m\_leftExpression: dùng để lưu tổng các đơn thức nằm ở bên trái dấu (+, -) đang xử lí của biểu thức.

Ví dụ:

5 -> m\_leftExpression = None

5 + -> m\_leftExpression = 5

5 + 3 -> m\_leftExpression = 5

5 + 3 + = -> m\_leftExpression = 8

5 + 3 + 1 = -> m\_leftExpression = 9

* m\_leftTerm: Dùng để lưu tích các nhân tử nằm bên trái bên trái dấu (\*, /) của đơn thức đang xử lí.

Ví dụ:

5 -> m\_leftTerm = None

5 \* -> m\_ leftTerm = 5

5 \* 3 -> m\_ leftTerm = 15

5 \* 3 \* = -> m\_ leftTerm = 15

5 \* 3 \* 2 = -> m\_ leftTerm = 30

input: lưu hành động người dung nhập vào (số hoặc dấu)

m\_actions: dùng để lưu toàn bộ hành động người dung nhập vào

lastInput: lưu hành động cuối cùng được nhập

lastOperation: lưu dấu cuối cùng được nhập

Diagram

Description automatically generated**Thuật toán:**

Hình - Sơ đồ thuật toán hàm addInput

**2.2 Coding:**

* class CalculatorException : public std::exception : Class loại bỏ ngoại lệ
* class Calculator : Class chứa các biến và các hàm tính toán
* Trong class Calculator:
* enum class ActionType : char {Number, Plus, Minus, Multiply, Divide, Equals, None}: định nghĩa các hành động người dùng nhập vào
* struct Action { ActionType actionType; double value; } : Khi người dùng nhân 1 nút tương ứng với 1 hành động và struct này được dùng để lưu hành động đó. Nếu actionType là kiểu “Number”, value sẽ lưu giá trị số người dùng nhập; nếu actionType không phải “Number”, value = 0.
* void reset(): Khôi phục lại chương trình về trạng thái ban đầu.
* bool addInput(const Action& input): Xử lý các phép tính toán “+, -, \*, /” ứng với các *input* được truyền vào. Nếu action truyền vào là số và action trước đó không phải là dạng số thì chươn trình sẽ thêm số đó vào m\_actions (list quản lý thứ tự action mà người dùng truyền vào). Nếu action truyền vào không là số và action được truyền vào trước đó là số thì sẽ xảy ra các trường hợp sau và thêm action đó vào list m\_actions:
* Trường hợp 1: Nếu toán tử truyền vào trước đó là phép cộng và nếu toán tử đang được truyền vào là toán tử “+” hoặc “-” hoặc “=” thì m\_leftExpression (biến lưu kết quả của phép cộng hoặc trừ) sẽ cập nhật lại giá trị mới và m\_leftTerm (biến lưu kết quả của phép nhân hoặc chia) sẽ trả về 0. Nếu toán tử truyền vào trước đó là phép cộng và nếu nếu toán tử đang được truyền vào khác toán tử “+” hoặc “-” hoặc “=” thì m\_leftTerm sẽ cập nhật lại giá trị bằng giá trị cuối cùng người dùng nhập vào.
* Trường hợp 2: Nếu toán tử truyền vào trước đó là phép trừ và nếu toán tử đang được truyền vào là toán tử “+” hoặc “-” hoặc “=” thì m\_leftExpression (biến lưu kết quả của phép cộng hoặc trừ) sẽ cập nhật lại giá trị mới và m\_leftTerm (biến lưu kết quả của phép nhân hoặc chia) sẽ trả về 0. Nếu toán tử truyền vào trước đó là phép trừ và nếu nếu toán tử đang được truyền vào khác toán tử “+” hoặc “-” hoặc “=” thì m\_leftTerm sẽ cập nhật lại giá trị bằng giá trị cuối cùng người dùng nhập vào.
* Trường hợp 3: Nếu toán tử truyền vào trước đó là phép nhân và nếu toán tử đang được truyền vào là toán tử “+” hoặc “-” hoặc “=” thì m\_leftExpression (biến lưu kết quả của phép cộng hoặc trừ) sẽ cập nhật lại giá trị mới và m\_leftTerm (biến lưu kết quả của phép nhân hoặc chia) sẽ trả về 0. Nếu toán tử truyền vào trước đó là phép nhân và nếu nếu toán tử đang được truyền vào khác toán tử “+” hoặc “-” hoặc “=” thì m\_leftTerm sẽ cập nhật lại giá trị bằng giá trị nhân vào giá trị trước đó.
* Trường hợp 4: Nếu toán tử truyền vào trước đó là phép chia và nếu nếu toán tử đang được truyền vào là toán tử “+” hoặc “-” hoặc “=” và nếu như giá trị cuối cùng của người dùng truyền vào bằng 0 thì trả về lỗi “*Error: Cannot Divide By Zero*”. Nếu toán tử truyền vào trước đó là phép chia và nếu nếu toán tử đang được truyền vào là toán tử “+” hoặc “-” hoặc “=” và nếu như giá trị cuối cùng của người dùng truyền vào khác 0 thì m\_leftExpression (biến lưu kết quả của phép cộng hoặc trừ) sẽ cập nhật lại giá trị mới và m\_leftTerm (biến lưu kết quả của phép nhân hoặc chia) sẽ trả về 0. Ngược lại thì cập nhật lại giá trị của m\_leftTerm bằng cách nhân với 1/ giá trị người dùng nhập cuối cùng.
* Trường hợp 5: Nếu toán tử truyền vào trước đó là toán tử “=” và nếu toán tử đang được truyền vào là toán tử “+” hoặc “-” hoặc “=” thì m\_leftExpression (biến lưu kết quả của phép cộng hoặc trừ) sẽ cập nhật lại giá trị mới và m\_leftTerm (biến lưu kết quả của phép nhân hoặc chia) sẽ trả về 0. Nếu toán tử truyền vào trước đó là toán tử “=” và nếu nếu toán tử đang được truyền vào khác toán tử “+” hoặc “-” hoặc “=” thì m\_leftTerm sẽ cập nhật lại giá trị và m\_leftExpression trả về 0.
* Trường hợp 5: Nếu toán tử truyền vào trước đó là toán tử “None” và nếu toán tử đang được truyền vào là toán tử “+” hoặc “-” thì m\_leftExpression (biến lưu kết quả của phép cộng hoặc trừ) sẽ cập nhật lại giá trị mới và m\_leftTerm (biến lưu kết quả của phép nhân hoặc chia) sẽ trả về 0. Nếu toán tử truyền vào trước đó là toán tử “None” và nếu nếu toán tử đang được truyền vào khác toán tử “+” hoặc “-” hoặc thì m\_leftTerm sẽ cập nhật lại giá trị và m\_leftExpression trả về 0. Nếu toán tử truyền vào trước đó là toán tử “None” và nếu toán tử đang được truyền vào là toán tử “=” thì m\_leftTerm sẽ cập nhật lại giá trị.
* Action getLastInput() const: trả về giá trị cuối cùng của người dùng nhập vào.
* double getCurrentResult() const: Trả về kết quả sau khi tính toán.
* bool hasLeftTermValue() const: trả về kết quả hiện tại sau khi thực hiện phép nhân hoặc chia
* bool hasLeftExpressionValue() const: trả về kết quả hiện tại sau khi thực hiện phép cộng hoặc trừ.
* bool isOperation(ActionType action) const: Kiểm tra xem action truyền vào có phải là một toán tử hay không.
* int getActionsSize(): trả về số lượng action đã được thêm vào list m\_actions.
* const Action& getAction(int i): trả về action tại vị trị thứ i.
* void deleteLastAction(Action): xóa action cuối cùng được thêm vào list m\_actions.
* bool isTerm(ActionType action) const: Kiểm tra xem action có phải toán tử của nhân hoặc chia.
* bool isExpression(ActionType action) const: Kiểm tra xem action có phải toán tử của phép cộng hoặc trừ.
* ActionType getLastOperation(): Trả về toán tử cuối cùng được thêm vào bởi người dùng.
* Class leftExpression:
* void reset(): trả về giá trị ban đầu.
* void set(double value): gán giá trị bằng value.
* void add(double value): cộng giá trị hiện tại với value.
* double getValue() const: trả về giá trị hiện tại của leftExpression.
* bool hasValue() const: kiểm tra xem leftExpression có đang chưa giá trị hay không.
* Class leftTerm:
* void reset(): trả về giá trị ban đầu.
* void set(double value): gán giá trị bằng value.
* void add(double value): cộng giá trị hiện tại với value.
* double getValue() const: trả về giá trị hiện tại của leftTerm.
* bool hasValue() const: kiểm tra xem leftTerm có đang chưa giá trị hay không.

1. **GỘP CODE VÀ TEST**
   1. Tổng hợp code chức năng vào code giao diện

* Tính toán với phép cộng:
* Thuật toán: Chương trình sẽ lần lượt nhận toán tử “+” và chuỗi số do người dùng nhập, khi người dùng nhập đến đâu thì chương trình sẽ cộng dồn và lưu lại kết quả. Ví dụ ban đầu người dùng nhập chuỗi số “abc” thì chương trình sẽ chờ người dùng nhập tiếp toán tử, khi người dùng nhâp toán tử “+” thì chương trình lưu lại chuỗi số vào biến *result* và chờ người dùng nhập chuỗi số tiếp theo. Khi người dùng nhập chuỗi số tiếp theo là “def” thì chương sẽ chờ người dùng nhập tiếp toán tử. Khi người dùng nhập toán tử bất kì thì chương trình sẽ lưu lại kết quả phép tính của toán tử trước đó *result* = *result* + “def”. Và cứ lặp lại cho đến khi người dùng nhập toán tử “=” thì chương trình sẽ xuất kết quả ra màn hình.

Graphical user interface, application

Description automatically generated

Hình - Tính toán với phép cộng - một toán tử

Graphical user interface

Description automatically generated

Hình - Tính toán với phép cộng – nhiều toán tử

* Tính toán với phép trừ:
* Thuật toán: Chương trình sẽ lần lượt nhận toán tử “-” và chuỗi số do người dùng nhập, khi người dùng nhập đến đâu thì chương trình sẽ trừ và lưu lại kết quả. Ví dụ ban đầu người dùng nhập chuỗi số “abc” thì chương trình sẽ chờ người dùng nhập tiếp toán tử, khi người dùng nhâp toán tử “-” thì chương trình lưu lại chuỗi số vào biến *result* và chờ người dùng nhập chuỗi số tiếp theo. Khi người dùng nhập chuỗi số tiếp theo là “def” thì chương sẽ chờ người dùng nhập tiếp toán tử. Khi người dùng nhập toán tử bất kì thì chương trình sẽ lưu lại kết quả phép tính của toán tử trước đó *result* = *result* - “def”. Và cứ lặp lại cho đến khi người dùng nhập toán tử “=” thì chương trình sẽ xuất kết quả ra màn hình.

Graphical user interface, application

Description automatically generated

Hình - Tính toán với phép trừ - một toán tử

Graphical user interface

Description automatically generated

Hình - Tính toán với phép trừ - nhiều toán tử

* Tính toán với phép nhân:
* Thuật toán: Chương trình sẽ lần lượt nhận toán tử “x” và chuỗi số do người dùng nhập, khi người dùng nhập đến đâu thì chương trình sẽ nhân và lưu lại kết quả. Ví dụ ban đầu người dùng nhập chuỗi số “abc” thì chương trình sẽ chờ người dùng nhập tiếp toán tử, khi người dùng nhâp toán tử “x” thì chương trình lưu lại chuỗi số vào biến *result* và chờ người dùng nhập chuỗi số tiếp theo. Khi người dùng nhập chuỗi số tiếp theo là “def” thì chương sẽ chờ người dùng nhập tiếp toán tử. Khi người dùng nhập toán tử bất kì thì chương trình sẽ lưu lại kết quả phép tính của toán tử trước đó *result* = *result* \* “def”. Và cứ lặp lại cho đến khi người dùng nhập toán tử “=” thì chương trình sẽ xuất kết quả ra màn hình.

Graphical user interface

Description automatically generated

Hình - Tính toán với phép nhân – một toán tử

Graphical user interface, application

Description automatically generated

Tính toán với phép nhân - nhiều toán tử

* Tính toán với phép chia:
* Thuật toán: Chương trình sẽ lần lượt nhận toán tử “/” và chuỗi số do người dùng nhập, khi người dùng nhập đến đâu thì chương trình sẽ chia và lưu lại kết quả. Ví dụ ban đầu người dùng nhập chuỗi số “abc” thì chương trình sẽ chờ người dùng nhập tiếp toán tử, khi người dùng nhâp toán tử “x” thì chương trình lưu lại chuỗi số vào biến *result* và chờ người dùng nhập chuỗi số tiếp theo. Khi người dùng nhập chuỗi số tiếp theo là “def” thì chương sẽ chờ người dùng nhập tiếp toán tử. Khi người dùng nhập toán tử bất kì thì chương trình sẽ lưu lại kết quả phép tính của toán tử trước đó *result* = *result* / “def”. Và cứ lặp lại cho đến khi người dùng nhập toán tử “=” thì chương trình sẽ xuất kết quả ra màn hình.

Graphical user interface, application

Description automatically generated

Hình - Tính toán với phép chia – một toán tử

Graphical user interface, application

Description automatically generated

Tính toán với phép chia – nhiều toán tử

* Tính toán với biểu thức:
* Thuật toán: Chương trình sẽ lần lượt nhận toán tử và chuỗi số do người dùng nhập, khi người dùng nhập đến đâu thì chương trình sẽ chia và lưu lại kết quả, chương trình dùng 2 biến để lưu lại kết quả *result\_1* (lưu kết quả của phép cộng và phép trừ) và *result\_2* (lưu kết quả của phép nhân và phép chia). Ví dụ ban đầu người dùng nhập chuỗi số “abc” thì chương trình sẽ chờ người dùng nhập tiếp toán tử, khi người dùng nhập toán tử thì chương trình sẽ phân loại đó là kết quả quả phép tính nào sau đó lưu lại chuỗi số vào biến result\_1/ result\_2 và chờ người dùng nhập chuỗi số tiếp theo. Khi người dùng nhập chuỗi số tiếp theo là “def” thì chương sẽ chờ người dùng nhập tiếp toán tử. Khi người dùng nhập toán tử bất kì thì chương trình sẽ lưu lại kết quả phép tính của toán tử trước đó *result\_1/ result\_2* = *result\_1/ result\_2* <toán tử> “def”. Và cứ lặp lại cho đến khi người dùng nhập toán tử “=” thì chương trình sẽ xuất kết quả ra màn hình.

Graphical user interface

Description automatically generated

Hình - Tính toán với biểu thức

* Tính toán với phép nhân:
* Thuật toán: Xem kí tự “.” Như là một phần của chuỗi số sau đó chuyển đổi chuỗi số thành số và thực hiện các phép tính toán.

Graphical user interface

Description automatically generated

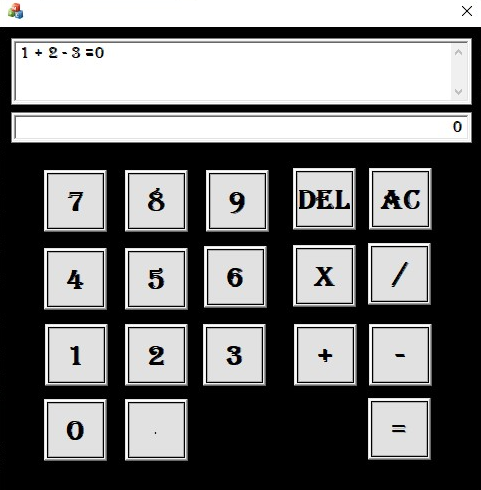
Hình - Tính toán với số thực

* Xóa biểu thức:
* Thuật toán: Khi người dùng nhấn nút del thì chương trình sẽ xóa đi toán tử và xóa đi trả lại kết quả trước khi toán thử đó thực hiện.

Graphical user interface, application

Description automatically generated

Hình - Trước khi xóa



Hình - Sau khi xóa

* Tính năng bắt lỗi:

Graphical user interface

Description automatically generated

Hình - Lỗi sai khi chia mẫu bằng 0

1. **LÀM REPORT**
   1. Bản word:

Dựa theo yêu cầu đã đưa ra:

- Yêu cầu: + Xác định đúng vấn đề cần phải giải quyết.

+ Các luận cứ và luận chứng chính xác và có sức thuyết phục, giải quyết được vấn đề, thể hiện năng lực tư duy lý luận tốt.

+ Có dẫn chứng cụ thể (hình ảnh, tài liệu thầy cho)

+ Trình bày rõ ràng, các ý cụ thể.

+ Báo cáo xác thực.

Người được giao làm báo cáo:

* + - * 20127662 - Nguyễn Đình Văn
      * 20127417 – Phạm Thảo Vy
  1. Bản PowerPoint:
     + - Yêu cầu: + Slide đẹp (số dòng trên 1 trang, hình ảnh phong phú)

  + Có trích dẫn rõ ràng

  + Bố cục hợp lý

Người được gia làm PowerPoint:

* + - * 20127129 – Lê Hải Đăng
      * 20127061 – Lưu Minh Phát

1. **HỌP CUỐI ĐỒ ÁN**
   1. Bàn giao công việc:
      * + Kiểm tra lại các tính năng của máy tính.
        + Chia công việc thuyết trình.
        + Tổng hợp lại source code, word, powerpoint.
   2. Thuyết trình nội bộ
      * + Tập dợt thuyết trình.
        + Kiểm tra những chỗ chưa ổn
2. **SOURCE CODE**

Link:

<https://drive.google.com/file/d/11B9T7ayt5qsMPWYyNB-pd1hTySOTknpn/view?usp=sharing>

# NGUỒN THAM KHẢO

* Tham khảm trên Youtube:

<https://www.youtube.com/watch?v=YXUx97uRNT0&list=PLfszubEEhakf7mGTDjsImyp-YGU69_S5k>

<https://www.youtube.com/watch?v=yXfMlq7SZTM&list=PLbbMKXzfxUBRrAdO-fc3FSdaHihb7XHqI&index=13>

* Tham khảo tài liệu:

<https://123docz.net/document/2639592-tim-hieu-ve-mfc-va-xay-dung-ung-dung-su-dung-mfc.htm?fbclid=IwAR2sMNyBx3pRAg-ejxQkhCKsCdh3IwsxYZ64nqGhE4pReHAFV7ZTLgnf6yg>

<https://123docz.net/document/638274-thu-vien-mfc-cua-microsoft-va-ung-dung-co-ban-trong-windows.htm?fbclid=IwAR3lUuyJCQCI8KKZVNaQ85ntk42XvY7J-GWiGwNgIeANL4oXQk-KYYmKcOY>

https://www.tutorialspoint.com/mfc/mfc\_overview.htm