ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

KHOA CÔNG NGHỆ PHẦN MỀM

BÁO CÁO ĐỒ ÁN CUỐI KÌ

ĐỀ TÀI

PHẦN MỀM MÔ PHỎNG LƯU ĐỒ THUẬT TOÁN HỖ TRỢ HỌC LẬP TRÌNH

Môn học: **Lập trình trực quan**

Giảng viên lý thuyết: **Thầy Mai Trọng Khang**

Giảng viên lý thực hành : **Cô Huỳnh Hồ Thị Mộng Trinh**

Nhóm thực hiện: **Team ProVision**

Lê Hoàng Minh Sơn - 18520350

Quản Tiến Nghĩa - 18520111

Võ Thành Trung - 18520180

TP. Hồ Chí Minh, tháng 12 năm 2019

LỜI NÓI ĐẦU

Hiện nay, có rất nhiều công cụ, từ online cho đến offline, phục vụ cho việc thiết kế và lưu trữ các lưu đồ (flowchart), khung sườn thiết kế (wireframe), biểu đồ (diagram), tiến trình dự án (progress),... một cách tiện lợi và hiệu quả.

Trong lĩnh vực công nghệ thông tin, đặc biệt đối với ngành kĩ thuật phần mềm thì việc có cho mình một lưu đồ thuật toán trước khi đi vào quá trình thiết kế sản phẩm cụ thể là vô cùng quan trọng. Đồng thời, việc học cách thiết kế một lưu đồ thuật toán sao cho hợp lí là cần thiết đối với một sinh viên IT.

Nắm bắt được nhu cầu đó, nhóm đã hướng đến đề tài thiết kế phần mềm mô phỏng lưu đồ thuật toán đơn giản với đối tượng sử dụng là những bạn đang trong quá trình học lập trình.

Sản phẩm của đề tài là quá trình nỗ lực hợp tác và làm việc giữa các thành viên, với hy vọng được nâng cao kĩ năng chuyên môn cũng như phong cách làm việc trong môi trường phát triển phần mềm chuyên nghiệp sau này.

LỜI CẢM ƠN

Trân trọng gửi lời cảm ơn thầy Mai Trọng Khang đã tạo điều kiện cho chúng em có cơ hội được thực hiện đồ án.

Với những kiến thức lý thuyết thầy đề cập tới trong các tuần học, nhóm đã vận dụng được rất nhiều trong việc hoàn thành đề tài.

Chỉ trong vòng 15 tuần, nhờ sự chỉ dẫn nhiệt tình của thầy, chúng em đã tiếp thu được những kiến thức quan trọng cùng những góp ý chân thành để có thể làm được một chương trình hoàn chỉnh.

Cũng xin cảm ơn thầy cô và bạn bè trong khoa Công nghệ phần mềm đã nhiệt tình hỗ trợ, tạo điều kiện cho nhóm em làm bài báo cáo này.

NHẬN XÉT CỦA GIẢNG VIÊN

I. TỔNG QUAN

1. Thông tin chung

a. Tên đề tài:

Tiếng Việt: Phần mềm Mô phỏng lưu đồ Thuật toán hỗ trợ học lập trình

Tiếng Anh: FlowArt – Flowchart Simulation for Basic Programming

b. Thời gian thực hiện: 4 tháng (và sẽ tiếp tục phát triển)

c. Thành viên tham gia:

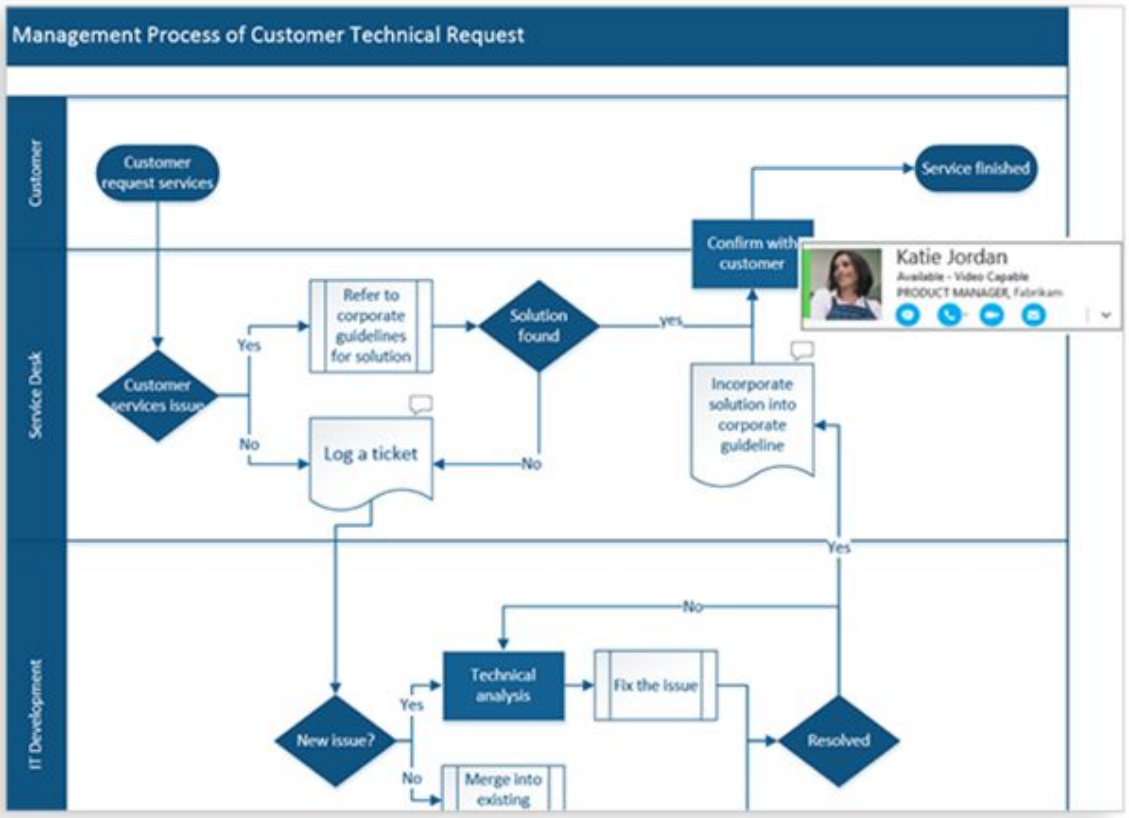
TT	MSSV	Họ tên
1	18520350	Lê Hoàng Minh Sơn
2	18520111	Quản Tiến Nghĩa
3	18520180	Võ Thành Trung

2. Quá trình chọn đề tài

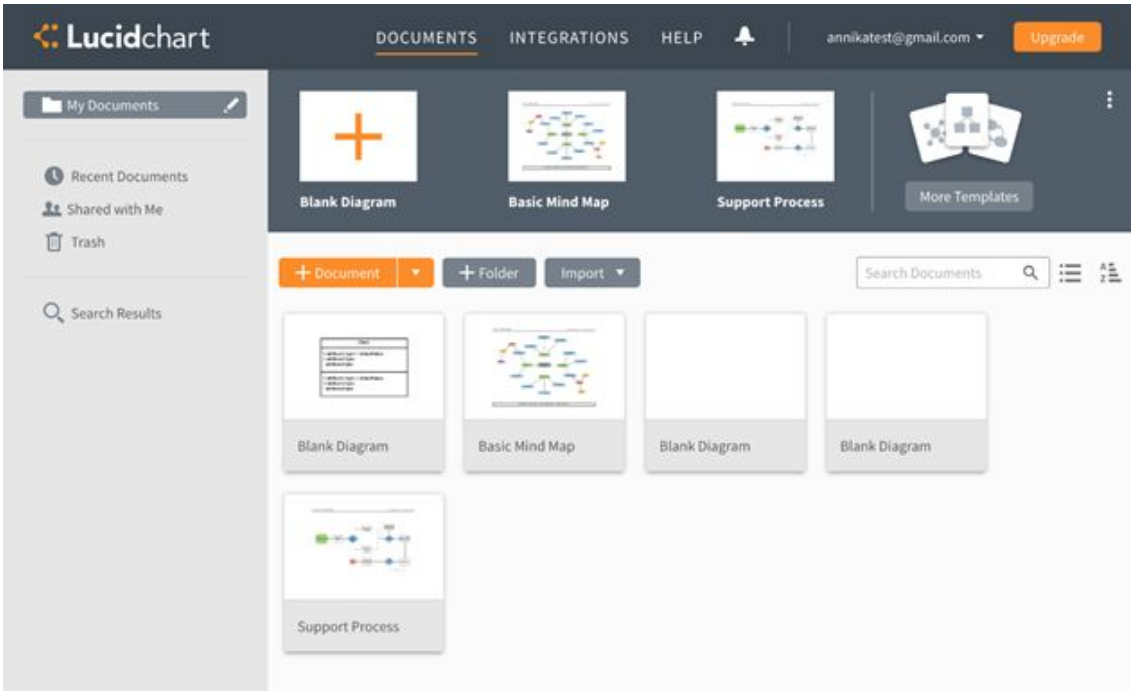
Trước khi tiến hành chọn đề tài, nhóm đã khảo sát một số ứng dụng, trang web thiết kế lưu đồ phổ biến, chúng được phân thành 2 loại:

Loại 1: Phần mềm vẽ sơ đồ khối

- Microsoft Visio.


Management Process of Customer Technical Request
Customer
Service Desk
IT Development
Customer request services
Service finished
Confirm with customer
Katie Jordan
Available - Video Capable
PRODUCT MANAGER, Fabrikam
Refer to corporate guidelines for solution
Solution found
yes
Yes
Customer services issue
No
Log a ticket
No
Incorporate solution into corporate guideline
Yes
No
Technical analysis
Fix the issue
Yes
New issue?
Resolved
No
Merge into existing

- Lucid Chart


Lucidchart
DOCUMENTS
INTEGRATIONS
HELP
anniikatest@gmail.com
Upgrade
My Documents
Recent Documents
Shared with Me
Trash
Search Results
Blank Diagram
Basic Mind Map
Support Process
More Templates
+ Document
+ Folder
Import
Search Documents
Blank Diagram
Basic Mind Map
Blank Diagram
Blank Diagram
Support Process

Ngoài ra còn có nhiều website và ứng dụng hỗ trợ khác như:

- Creately
- Pidoco
- Draw.io
- Google Slides
- OmniGraffle
- Dia Diagram Editor
- ClickCharts Diagram
- …

**** Ưu điểm:*** các ứng dụng trên đều đáp ứng tốt nhu cầu vẽ và lưu trữ lưu đồ, hỗ trợ các chức năng tự hoàn chỉnh lưu đồ, đồng thời có nhiều thư viện lớn cho nhiều loại lưu đồ khác nhau, cũng như chức năng chia sẻ và tích hợp với các mạng xã hội lớn, thuận tiện cho công việc. Trong đó 2 ứng dụng Microsoft Visio và LucidChart rất dễ sử dụng với giao diện đơn giản nhưng chuyên nghiệp

**** Nhược điểm:*** không có chức năng mô phỏng quá trình lưu đồ chạy thực tế, vì thế người sử dụng có thể không biết mình đã vẽ hợp lí chưa; đa số phần mềm trên phải trả phí để sử dụng đầy đủ chức năng

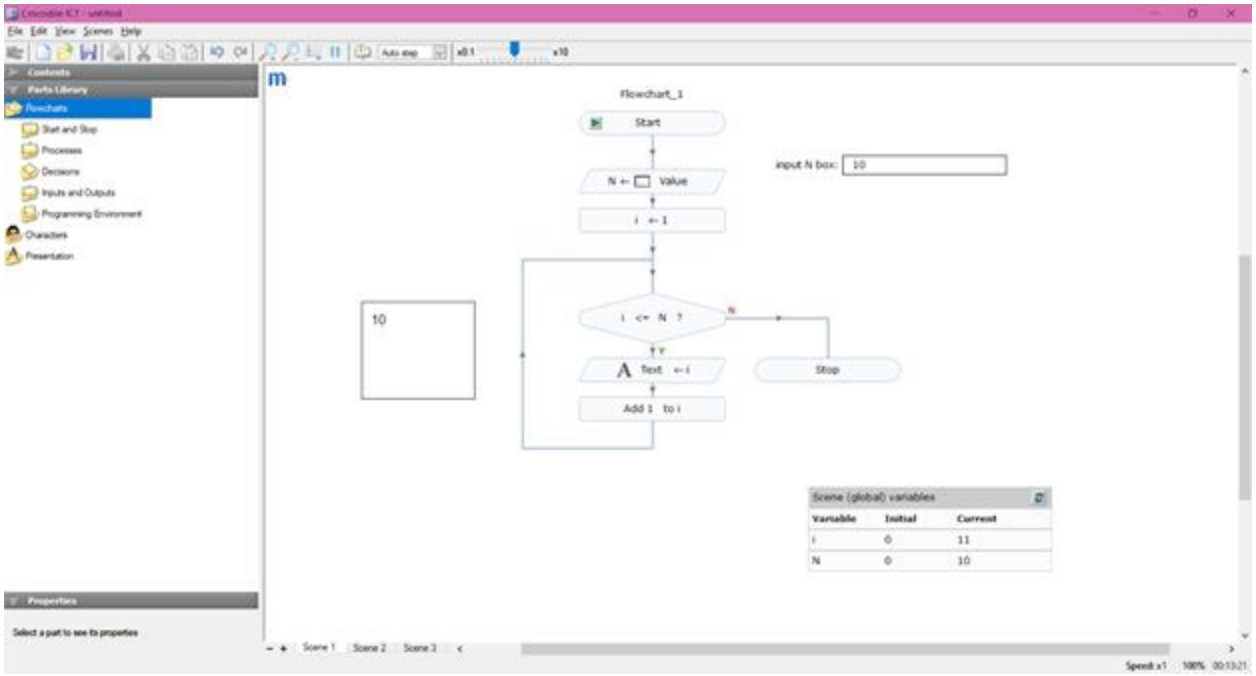
Loại 2: Phần mềm tích hợp vẽ và mô phỏng (động) lưu đồ thuật toán

- Phần mềm Crocodile ICT :

Giao diện chính của phần mềm Crocodile ICT và một lưu đồ đơn giản

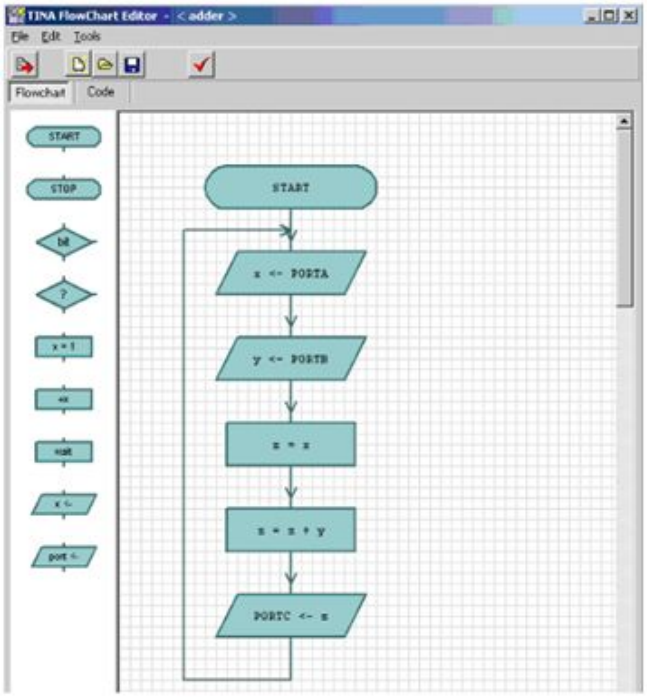
**** Ưu điểm:*** Phần mềm Crocodile ICT hỗ trợ rất đầy đủ các công cụ cần thiết cho việc thiết kế và mô phỏng các lưu đồ đơn giản. Phần mềm còn hỗ trợ một số hệ thống như "Scene variable" được sử dụng như tính năng watch thường thấy ở các IDE, các hình thức input khác nhau,…

**** Nhược điểm:*** giao diện người dùng (UI) của phần mềm còn quá khó hiểu để tiếp xúc, đặc biệt không thân thiện với những người mới bắt đầu luyện tập về lưu đồ thuật toán. Mặc dù cố tỏ ra thân thiện bằng cách hỗ trợ nhiều loại khối khác nhau. song, ở đó, một số khối không hiểu làm gì, một số khối trở nên dư thừa, không cần thiết. Một số thao tác cơ bản nhất lại cần được vẽ quá cầu kỳ (VD: nhập / xuất), đồng thời việc nhập và xuất cũng chưa thực sự thân thiện với người dùng.



Giao diện chính của phần mềm Crocodile ICT và một lưu đồ đơn giản

- TINA: là phần mềm thiết kế mạch điện tử, nhưng trong đó tích hợp tính năng vẽ lưu đồ thuật toán



**** Ưu điểm:*** tiện lợi, phục vụ việc thiết kế và lắp ráp mạch, là phần mềm lâu năm nên độ tin cậy khá cao

**** Nhược điểm:*** không thích hợp cho người mới học lập trình với các câu lệnh phức tạp, không trực quan

→ Tổng hợp những kết quả khảo sát trên, nhóm đã đưa ra quyết định sẽ làm một ứng dụng với chức năng tương tự: có thể mô phỏng quá trình chạy của một lưu đồ thuật toán, với giao diện thân thiện (user-friendly) và nhiều chức năng tích hợp khác phục vụ nhu cầu học lập trình căn bản, cũng như đáp ứng được khái niệm "Lập trình trực quan" của môn học này. Nhóm chọn phần mềm **Crocodile ICT** để dựa vào và phát triển.

3. Mục đích đề tài

Mục đích cơ bản là hiện thực hóa những kiến thức đạt được trong môn học cũng như quá trình triển khai đồ án. Đó là xây dựng được một chương trình hoàn thiện cả về mặt UI và UX, đồng thời rèn luyện tác phong lập trình nhóm có tính kỉ luật.

Mục đích sâu xa là tạo ra một sản phẩm có thể phục vụ cho cộng đồng các bạn trẻ đam mê tin học và muốn trau dồi khả năng lập trình, tư duy thuật toán. FlowArt giúp việc tiếp cận với lập trình và thuật toán thú vị hơn thông qua việc hỗ trợ vẽ và chạy sơ đồ thuật toán (thay vì chạy những dòng code mang tính hàn lâm và khô khan, hoặc vẽ những lưu đồ chỉ "hiểu được" chứ không "thấy được").

4. Phạm vi đề tài

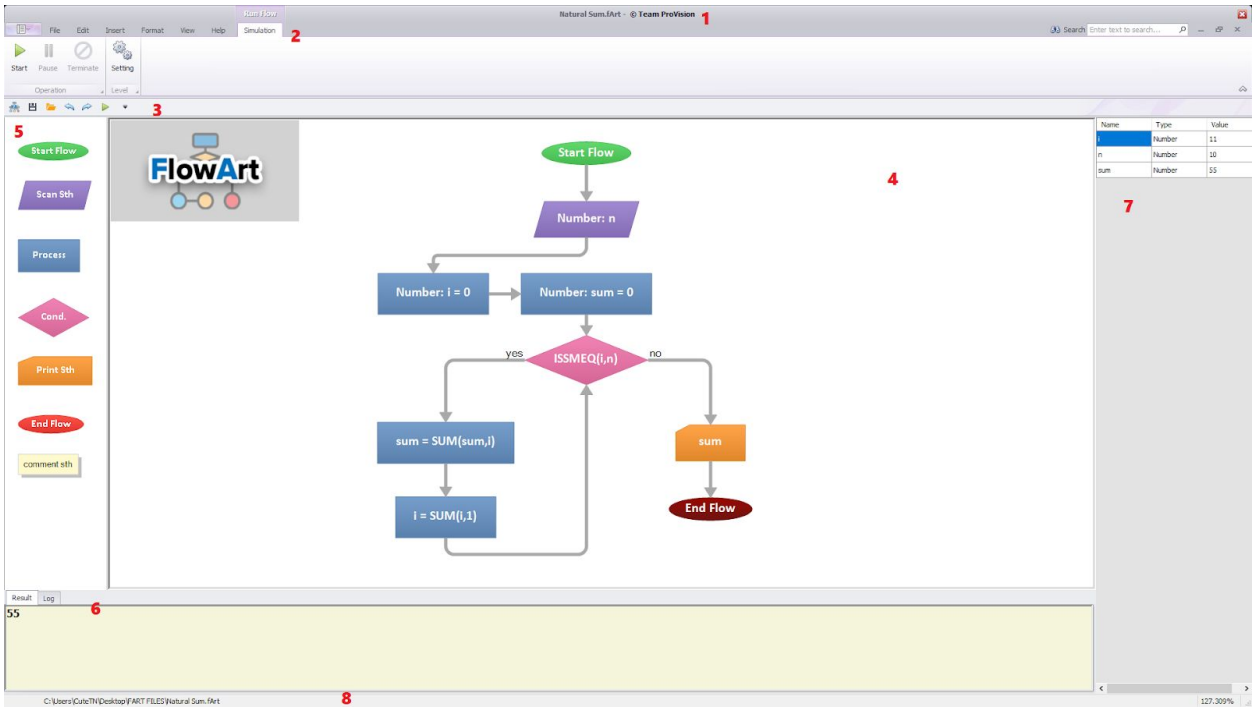
Xây dựng được một mô hình lưu đồ thuật toán có thể xuất ra được các định dạng ảnh khác nhau. Song song đó, ta có thể thực hiện giải quyết các bài toán tin học cơ bản trên chính mô hình đó, bằng một loại ngôn ngữ được thiết kế đặc biệt của riêng nhóm. Đề tài sẽ tiếp tục được phát triển với nhiều chức năng mở rộng sau môn học này.

5. Công nghệ sử dụng

Chương trình được xây dựng trên mô hình Winforms có sử dụng .NET Framework 4.6.1 và được code trên Visual Studio 2017 trở lên. Ngoài ra có sự hỗ trợ của hai framework là DevExpress 19.1 và GoDiagram v6.0

II. GIAO DIỆN CHƯƠNG TRÌNH (UI)

1. Màn hình giao diện chính (Main Interface)



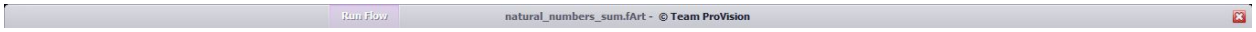
Được xây dựng và phát triển bằng Winfoms với sự hỗ trợ của DevExpress (chủ yếu để làm Ribbon Menu Bar). Giao diện thanh lịch có bố cục rõ ràng và màu sắc tươi sáng.

STT	Thành phần	Chức năng
1	Thanh tiêu đề (Title Bar)	Hiển thị tên bản vẽ, tên nhóm
2	Thanh công cụ chính (Ribbon Menu Bar)	Cung cấp tất cả chức năng để thao tác với các bản vẽ lưu đồ trong chương trình
3	Thanh công cụ truy cập nhanh (ToolBar)	Cung cấp các thao tác nhanh mà người dùng thường sử dụng (có thể tự chọn)
4	Trang vẽ (Design Map)	Làm việc trực tiếp với bản vẽ lưu đồ ở đây

5	Bảng chọn khối hình (Block Catalogue)	Cung cấp các khối hình để kéo thả qua bên trang vẽ
6	Khu vực thông báo (Message Area)	Hiển thị trang kết quả (result) và trang các lệnh theo trình tự được thực thi và lỗi (log) khi mô phỏng lưu đồ
7	Bảng theo dõi các biến (Watch Box)	Hiển thị danh sách tên, kiểu dữ liệu, dữ liệu của toàn bộ các biến khi chạy lưu đồ
8	Thanh trạng thái (Status bar)	Hiển thị các thông tin về trạng thái của bản vẽ lưu đồ (tỉ lệ kích thước, một số thao tác vừa thực hiện, tên đường dẫn file,...)

2. Thanh tiêu đề (Title Bar)

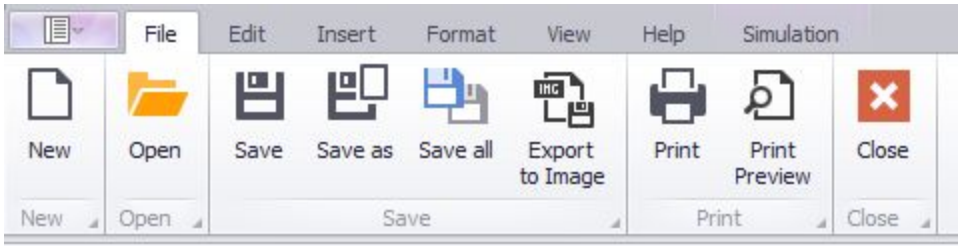
Nằm ngang trên cùng cửa sổ chương trình. Các thông tin về tên bản vẽ hiện tại (Design Name), tên nhóm (Team ProVision), nút Close button được đặt ở đây



3. Thanh công cụ (Ribbon Menu Bar)

Là nơi chứa tất cả chức năng phục vụ trong phần mềm. Gồm 7 tab menu:

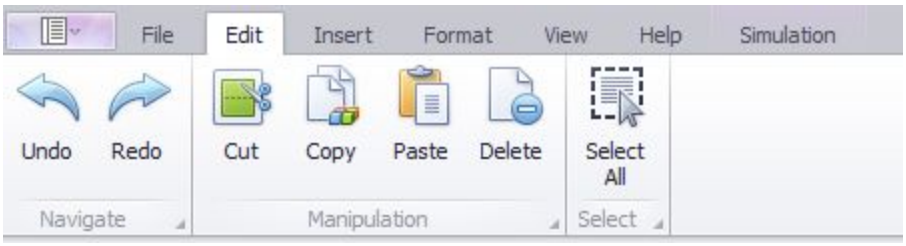
a) File Tab:



- New: Tạo một design mới (MDI)
- Open: Mở một design đã có theo đường dẫn người dùng mong muốn, có định dạng fArt
- Save: Lưu design hiện tại xuống máy tính theo đường dẫn người dùng mong muốn (Lưu ngầm định nếu design đã có đường dẫn trước đó), có định dạng fArt

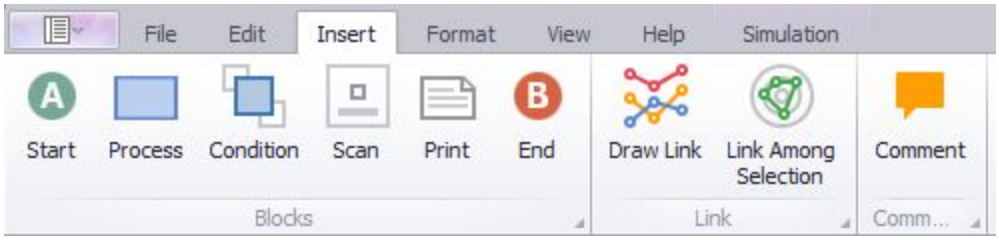
- Save as: Lưu design hiện tại xuống máy tính theo đường dẫn người dùng mong muốn (Không ngầm định)
- Save all: Lưu tất cả các design đang mở
- ***Export To Image:*** xuất design hiện tại thành file có định dạng ảnh (jpg, jpeg, jpe, jfif, png)
- Print: Thiết lập và in design hiện tại
- Print Preview: Xem tổng quan design trước khi in
- Close: Đóng design hiện tại.

b) Edit Tab



- Undo: Quay trở lại trạng thái trước đó của design
- Redo: Quay trở lại trạng thái trước hành động Undo
- Cut: Cắt các object được chọn vào clipboard
- Copy: Sao chép các object được chọn vào clipboard
- Paste: Dán các object đã Copy hoặc Cut từ clipboard
- Delete: Xóa các object được chọn
- Select all: Chọn tất cả object nằm trong bản vẽ

c) Insert Tab:

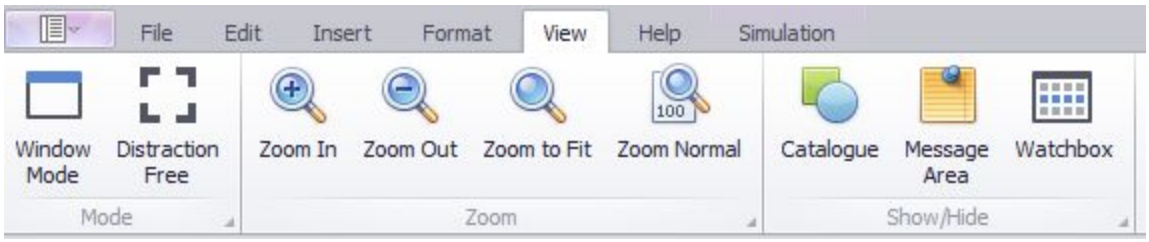


- Hộp Blocks gồm các button Start, Process, Condition, Scan, Print, End để thêm các block vào design một cách nhanh chóng
- ***Draw Link:*** thêm các link giữa các block khi được chọn
- Link Among Selection: chọn 2 block và nhấn nút này để tạo link giữa 2 block đó

d) Format Tab:

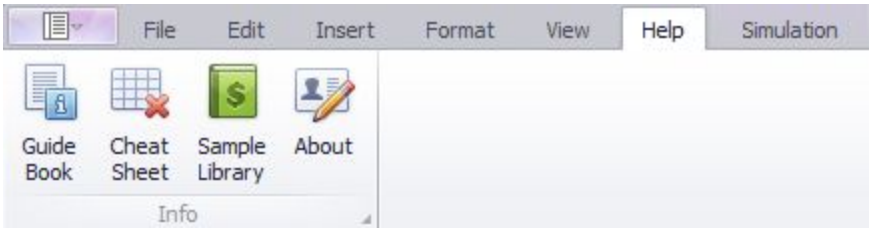
- ***Align Horizontally:*** canh thẳng hàng các object được chọn theo chiều ngang
- ***Align Vertically:*** canh thẳng hàng các object được chọn theo chiều dọc
- Design Properties: mở hộp thoại để đổi tên của design

e) View Tab:



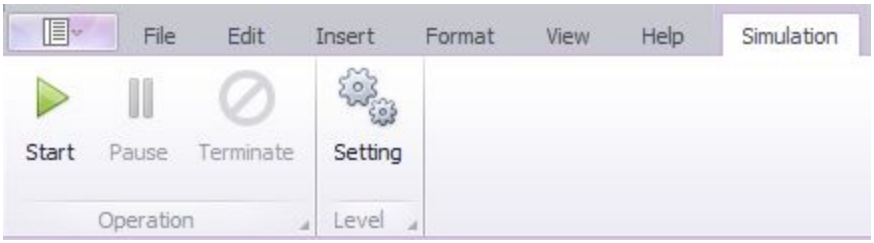
- Window Mode: chuyển màn hình sang chế độ cửa sổ
- Distraction Free: chuyển màn hình sang chế độ toàn màn hình
- Zoom In: phóng to design hiện tại
- Zoom Out: thu nhỏ design hiện tại
- ***Zoom To Fit:*** chỉnh tỉ lệ kích thước design cho cân đối với design map (phục vụ cho việc in, xuất file ảnh, quan sát)
- ***Zoom Normal:*** đặt giá trị tỉ lệ kích thước design về 100%
- Catalogue: hiển thị/che đi block catalogue
- Message Area: hiển thị/che đi message area
- Watch Box: hiển thị/che đi watch box

f) Help Tab:



- Guide book: mở trang tài liệu hướng dẫn sử dụng FlowArt
- ***Cheat Sheet:*** mở danh sách tổng hợp các hàm sử dụng trong FlowArt
- About: hiển thị thông tin về nhóm tác giả của phần mềm

g) Simulation Tab:



- ***Play:*** thực hiện chạy lưu đồ thuật toán được vẽ
- ***Pause:*** tạm dừng quá trình simulation
- ***Terminate:*** thoát khỏi quá trình simulation
  (sẽ giải thích ở phần III.2.e)
- ***Setting:*** mở hộp thoại để chỉnh sửa các chi tiết thuộc về quá trình simulation (Auto Run, Speed Level, Fading Effect, Sound Effect, Clear Result, Notify on end) (sẽ giải thích ở phần III.2.d)

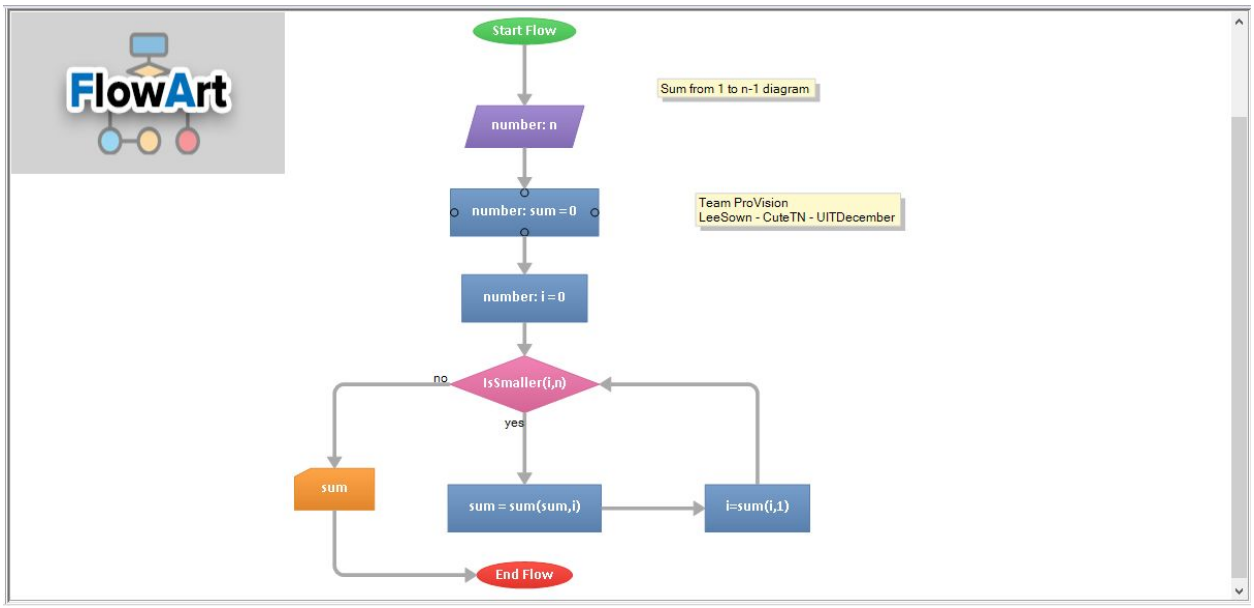
h) Thanh tìm kiếm (Search bar):

Nằm góc trên bên phải. Thực hiện việc tìm kiếm từ khóa xuất hiện trong bản vẽ, tăng UX trong những bản vẽ lớn

4. Thanh công cụ truy cập nhanh (ToolBar)

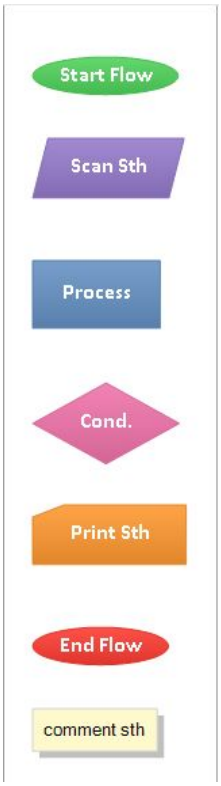
Có thể tùy chỉnh nằm ở trên hoặc ở dưới Menu bar. Người dùng tự chọn những nút chức năng thường sử dụng như Open, Zoom To Fit, Play để thêm vào, thao tác nhanh, tăng UX

5. Trang vẽ (Design Map)



Nằm ở giữa và chiếm diện tích lớn nhất màn hình. Là nơi người dùng trực tiếp vẽ, di chuyển, chỉnh sửa các object trên bản vẽ, có thể chọn bỏ hiển thị các thành phần như Block Catalogue, Message Area, Watch Box để có không gian làm việc tối đa.

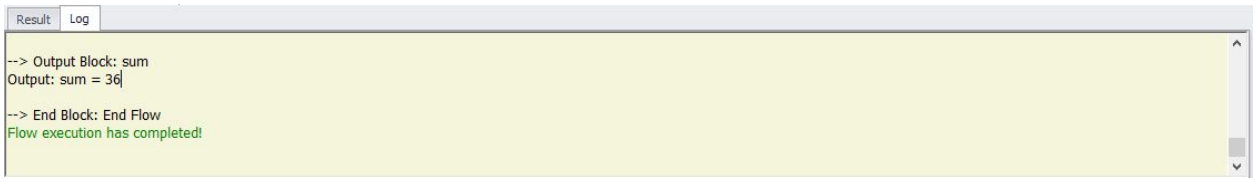
6. Bảng chọn khối hình (Block Catalogue)



Nằm ở phía bên trái màn hình (dock left), có thể chọn xuất hiện/che đi. Là nơi chứa các block để người dùng kéo thả qua design map, theo thứ tự gồm có:

- Start Flow: bắt đầu lưu đồ
- Scan: nhập
- Process: xử lí dữ liệu
- Condition: điều kiện
- Print: xuất
- End Flow: kết thúc lưu đồ

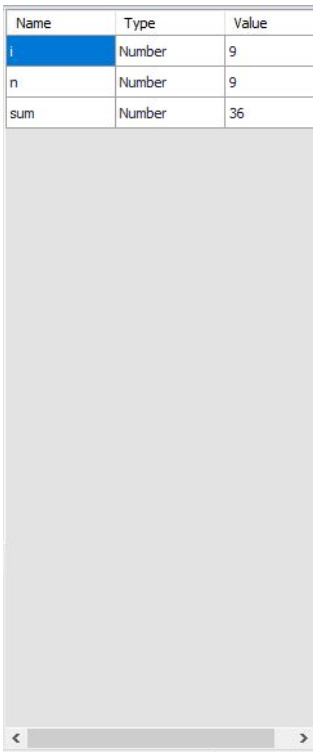
7. Khu vực thông báo (Message Area)



Nằm ở phía dưới Design Map, có thể chọn xuất hiện/che đi. Là nơi hiển thị:

+ Trang kết quả (result): những giá trị xuất ra từ khối Print
+ Trang tập lệnh và lỗi (log) khi mô phỏng lưu đồ: phục vụ việc theo dõi simulation, chạy từng bước để hiểu rõ thuật toán, phát hiện lỗi ngay trong lúc mô phỏng lưu đồ

8. Bảng theo dõi các biến (Watch Box)

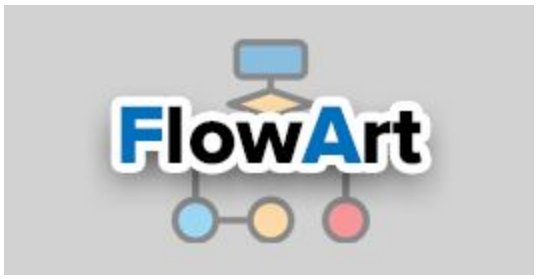


Nằm ở bên phải màn hình (dock right), có thể chọn xuất hiện/che đi. Là nơi hiển thị thông tin về tất cả biến (tên, kiểu dữ liệu) và sự thay đổi dữ liệu của chúng trong quá trình simulation, giúp người dùng dễ theo dõi thuật toán, tăng UX

9. Thanh trạng thái (Status bar)

Nằm phía dưới cùng cửa sổ chương trình. Các thông tin về tỉ lệ kích thước design, một số thao tác vừa thực hiện, tên đường dẫn file,... được thấy ở đây

10. Logo và icon của FlowArt:



III. TỔNG HỢP CÁC CHỨC NĂNG CHÍNH VÀ KỸ THUẬT SỬ DỤNG

Ngay sau khi chọn đề tài cho đồ án, nhóm đã xác định 2 mục tiêu quan trọng phải đạt được. Đó là:

+ Vẽ được lưu đồ thuật toán hoàn thiện

+ Mô phỏng từng bước chạy lưu đồ thuật toán, hiển thị kết quả, quá trình chạy và lỗi (nếu có) cho người sử dụng

Tương ứng với 2 nhiệm vụ đề ra cũng là những thách thức kỹ thuật lớn nhóm phải vượt qua: về mặt design view (front end), Data and Block processing (back end) và simulation (kết hợp front end và back end)

1. Mục tiêu 1:

Gồm các chức năng:

a) Tạo block (Creating)

Người dùng có thể tạo 1 block trong bản vẽ, có các đặc điểm:

- Một block được mặc định có 1 chức năng cụ thể, theo quy tắc vẽ sơ đồ khối (Function specification) (xem lại phần II.6)
- Có thể chỉnh sửa nội dung (Content Editing): Double click vào block để thay đổi nội dung
- Tự thay đổi kích thước theo chữ (Autosize): tăng UI và UX
- Có các điểm cổng nối (port) với các block khác: 4 cổng, khi rê chuột vào block sẽ hiện ra, khi không rê chuột vào sẽ tự động mất đi

Các hàm được cài đặt chủ yếu trong class FlowBlock

```
public enum BlockType
{
    Start = 1,
    End = 2,
    Process = 3,
    Input = 4,
    Output = 5,
    Condition = 6,
}
```

```
        public FlowBlock(BlockType k)
        {
            InitCommon();
            this.Kind = k;
        }
```

```
        private void InitCommon()
        {
            [...Set up the block with its general properties]
            // let it be BlockType.Process first
            InitShape("Process", new GoDrawing(GoFigure.Rectangle),
blockColor, blockColor, 22, 15, 22, 15);
        }
```

```
        public virtual void InitShape(string content, GoShape shape, Color
bColor, Color pColor, float topMarginX, float topMarginY, float
bottomMarginX, float bottomMarginY)
        {
            // set up the design of a block
            shape.FillSimpleGradient(bColor, Lighter(bColor), MiddleTop);
            shape.PenColor = Darker(bColor);

            this.TopLeftMargin = new SizeF(topMarginX, topMarginY);
            this.BottomRightMargin = new SizeF(bottomMarginX,
bottomMarginY);
            this.Background = shape;
            this.Text = content;
        }
```

```
        protected virtual void OnKindChanged(BlockType oldkind, BlockType
newkind)
        {
            Color blockColor = ColorOfType(newkind);
            // update the blocks, based on the Kind of block this now is
            switch (newkind)
            {
                case BlockType.Start:
```

```
                    {
                        InitShape("Start Flow", new
GoDrawing(GoFigure.Ellipse), blockColor, blockColor, 20, 5, 20, 3);
                        UpdatePorts("o", "o", "o", "o");
                        break;
                    }
                    // similarly we set up other types of blocks
                    }
                default: throw new InvalidEnumArgumentException("newkind",
(int)newkind, typeof(BlockType));
            }
        }
```

Ngoài ra còn có nhiều hàm khác phối hợp để hoàn thiện chi tiết 1 khối hình như: CreatePort, UpdatePorts, ChangeValue,... Lớp FlowBlock thừa kế lớp GoTextNode của GoDiagram API

b) Kéo thả block (Dragging dropping)

Có thể kéo 1 block từ Block Catalogue thả vào design, kéo thả 1 block tự do trong design Sử dụng tính chất FlowView thừa kế class GoView, và MyPalette thừa kế class GoPalette trong GoDiagram API và FlowBlock

c) Liên kết các block bằng đường nối (Connecting)

Có thể nối các block với nhau bằng link, có các đặc điểm:

- Đều có điểm bắt đầu và kết thúc (FromPort, ToPort)
- Một block có thể có nhiều link nối vào nhưng chỉ có thể có 1 link ra
- Khối Start chỉ có link ra, khối End chỉ có link vào
- Các đường nối tự động tìm vị trí port gần nhất
- Hình dạng và vị trí các link có thể thay đổi tùy người dùng

Các hàm về link được cài đặt xuyên suốt trong chương trình, nhưng chủ yếu trong class LinkHandle

```
        public FlowBlock Predecessor // get the block which starts link
        {
            get { return myPredecessor; }
            set { myPredecessor = value; }
        }
```

```
        public GoPort FindNearestPort(PointF pt, FlowBlock fb)
        {
            float maxdist = 10e20f;
            GoPort closest = null;
            GoPort p;
            p = fb.TopPort;
            if (p != null)
            {
                 float dist = (p.Left - pt.X) * (p.Left - pt.X) + (p.Top -
pt.Y) * (p.Top - pt.Y);
                if (dist < maxdist)
                {
                    maxdist = dist;
                    closest = p;
                }
            }
            // similar to TopPort we set up Left, Bottom, Right
            return closest;
        }
```

Xử lí 2 sự kiện DoMouseDown và DoMouseMove để thực hiện việc kéo thả link hợp lí.
Các đường link được tạo ra trên cơ sở class GoLink trong GoDiagram API. Hàm tạo link (CreateLink) được cài đặt trong các lớp quan trọng FlowView

```
        public override IGoLink CreateLink(IGoPort from, IGoPort to)
        {
            IGoLink il = base.CreateLink(from, to);
            if (il != null)
            {
                GoLabeledLink l = il.GoObject as GoLabeledLink;
                if (l != null)
                {
                    FlowBlock fromNode = from.Node.GoObject as FlowBlock;
                    if (fromNode != null && fromNode.Kind ==
BlockType.Condition) // condition block handling
                    {
                        GoText t = new GoText();
                        t.Text = "yes";
```

```
                    t.Selectable = false;
                    t.Editable = true;
                    l.FromLabel = t;
                }
            }
        }
        return il;
    }
```

d) Chọn các object (Selecting)

Có thể chọn một hoặc nhiều hình theo các quy tắc thông dụng:

- Nhấn chuột trái vào object để chọn
- Nhấn giữ Shift/Ctrl và chọn nhiều object
- Nhấn tổ hợp Ctrl + A hoặc nút Select All trong Edit Tab để chọn tất cả các hình
- Kéo khung chọn trong design quét tất cả hình cần chọn

e) Di chuyển các block cùng các đường nối (Moving respectively)

Khi các block có đường nối với nhau, việc di chuyển 1 block không làm phá hủy cấu trúc lưu đồ, thay vào đó các block và đường link vẫn giữ vị trí tương đối với nhau. Các block có thể đè lên nhau về mặt hình ảnh nhưng bản chất chúng vẫn là riêng biệt. Tương tự với các link nối.

f) Thao tác với các block trong quá trình vẽ (Manipulating)

Gồm tất cả thao tác cần thiết: cut, copy, paste, undo, redo của một phần mềm vẽ hình.

Các hàm làm việc trực tiếp với class FlowView.

Trong quá trình implement, nhóm đã bắt gặp khái niệm NonSerialization.

g) Canh thẳng hàng các block (Aligning)

Hỗ trợ về cả chiều ngang, chiều dọc, là chức năng giúp việc thiết kế một cách đẹp và cân đối nhất, tăng UI và UX. Các object đang chọn được canh thẳng hàng với nhau.

Hai hàm này được cài đặt trong lớp FlowView, sử dụng tâm của các block so theo trục Ox và trục Oy

```
    public virtual void AlignVerticalCenters()
    {
        GoObject obj = this.Selection.Primary;
        if (obj != null && !(obj is IGoLink))
        {
            float Y = obj.SelectionObject.Center.Y;
            foreach (GoObject temp in this.Selection)
```

```
                {
                    GoObject t = temp.SelectionObject;
                    if (!(t is IGoLink))
                        t.Center = new PointF(t.Center.X, Y);
                }
            }
            else
            {
                MessageBox.Show("Alignment failure: Primary Selection
is empty or a link instead of a block.");
            }
        }
    // similarly we set up AlignHorizontalCenters()
```

h) Điều chỉnh tỉ lệ của bản vẽ (Zooming)

Hỗ trợ 4 chế độ: Zoom In, Zoom Out, Zoom To Fit, Zoom Normal

Có thể theo dõi tỉ lệ bản vẽ ở thanh status. Nhỏ nhất 1%, lớn nhất 1000%.

Các hàm làm việc trực tiếp với class FlowView. Trong đó, DocScale là 1 thuộc tính của FlowView, trả về tỉ lệ của design hiện tại)

Hàm ZoomIn và ZoomOut đơn giản là dùng hai phép tính chia và nhân với tỉ lệ 0.9

```
        public virtual void ZoomIn()
        {
            float newscale = (float)(Math.Round(this.DocScale / 0.9f *
100) / 100);
             this.DocScale = newscale;
        }
```

Đối với hàm ZoomNormal ta cho DocScale bằng 1, còn hàm ZoomToFit khéo léo thay đổi giữa vị trí hiện tại của design với vị trí phù hợp cùng hàm RescaleToFit được hỗ trợ để design vừa vặn trong view tổng thể

i) Thêm chú thích cho bản vẽ (Comment adding)

Người dùng có thể thêm chú thích (comment) vào những nơi cần thiết, tăng UX

Comment thuộc lớp GoComment của GoDiagram

j) Lưu/mở file dưới định dạng riêng (Saving - Opening)

Các design có thể được lưu vào máy tính dưới định dạng riêng của FlowArt (.fArt). Sau đó có thể được người dùng mở ra để tiếp tục edit.

k) Xuất ra file ảnh của lưu đồ/In lưu đồ (Exporting to image - Printing)

Chương trình xuất ra một file ảnh của toàn bộ design và lưu vào máy tính, là một chức năng để người dùng có được những bức ảnh trực quan, phục vụ nhu cầu chia sẻ và học tập.

Sử dụng hàm GetBitmap để lấy được giá trị bitmap đã render của lưu đồ

```
        public virtual void ExportToImage()
        {
            SaveFileDialog saveDialog = new SaveFileDialog();
                saveDialog.Filter = "Image files (*.jpg, *.jpeg, *.jpe,
*.jfif, *.png) | *.jpg; *.jpeg; *.jpe; *.jfif; *.png";
            saveDialog.DefaultExt = "png";
            saveDialog.AddExtension = true;
            saveDialog.FileName = this.Doc.Name;

            if (saveDialog.ShowDialog() == DialogResult.OK)
            {
                Bitmap bmp = this.GetBitmap(); // get the bitmap rendered
from FlowArt
                bmp.Save(saveDialog.FileName, ImageFormat.Jpeg);
            }
        }
```

★ Kỹ thuật: Các chức năng từ a đến k đều được thực hiện qua quá trình tìm hiểu framework GoDiagram (user guide, forum, …). Các hàm được tham khảo và chỉnh sửa cho phù hợp với hướng đi của phần mềm. Từ đó, cả nhóm đã rèn luyện được khả năng tiếp cận và đọc hiểu tốt một framework, làm việc framework một cách tối ưu nhất, tận dụng các hỗ trợ của nó để hiện thực hóa từng chức năng

l) Phím tắt cho toàn bộ chương trình (Hotkey implementing)

Hầu hết các chức năng trong chương trình đều gán với một tổ hợp phím tắt, có chú thích khi rê chuột vào từng nút trong Menu bar. Tất cả đều nhằm phục vụ user một cách tối đa, và hotkey là cần thiết đối với một phần mềm thế này.

Tra bảng phụ lục để biết danh sách các hotkey được hỗ trợ

m) Quản lý các trang vẽ dưới dạng MDI (MDI structure deploying)

Nhóm quyết định xây dựng các design của chương trình dưới dạng MDI để người dùng có thể thực hiện nhiều bản vẽ khi mở ứng dụng. Tính chất này phổ biến đối với loại app này.

Các cửa sổ MDI thực chất là 1 design thuộc lớp FlowMap, được cài đặt linh hoạt để chứa FlowView và FlowDocument

2. Mục tiêu 2:

Gồm các chức năng:

a) Xử lý dữ liệu trong lưu đồ (Data processing)

- Khai báo biến (Variable declaration): Để sử dụng một biến trong FlowArt, người dùng cần phải thực hiện thao tác khai báo nó. Hiện FlowArt đã cung cấp 3 kiểu dữ liệu để tương tác bao gồm: Number (số), String (xâu) và Boolean (True/False)
- Gán biến (variable assignment): Người dùng có thể thay đổi giá trị của các biến đã khai báo thông qua câu lệnh gán
- Khởi tạo biến (variable initialization): Sự kết hợp giữa việc khai báo và việc gán biến, người dùng có thể gán cho biến 1 giá trị ngay khi khai báo chúng
- Xóa biến (variable deletion): Người dùng có thể xoá biến tạm không còn dùng nữa. Sau đó có thể sử dụng lại tên biến này với mục đích khác, kiểu dữ liệu khác
- Thực hiện các hàm khi chạy lưu đồ (functions handling): FlowArt cung cấp rộng rãi các hàm Built-in hỗ trợ người dùng tương tác với giá trị của các biến

★ Kỹ thuật

Mặc dù không có những kỹ thuật quản lý bộ nhớ hiệu quả, nhóm đặt trọng tâm vào **tính đúng**, **tính nhanh** và **tính linh hoạt** trong cách xử lý dữ liệu người dùng (user variables).

Các biến người dùng được lưu thông qua class *VarInfo*, dữ liệu của biến gồm 3 trường chính:

- **type:** mang thông tin kiểu dữ liệu của biến. Thông tin này mang kiểu dữ liệu DataType được khai báo như sau:

```
public enum DataType
{
    FA_Number,
    FA_String,
    FA_Boolean,
}
```

- **Value:** là một biến string chứa thông tin giá trị của biến. Nếu biến thuộc kiểu dữ liệu String, Value lưu trực tiếp giá trị đó. Nếu biến thuộc kiểu dữ liệu Number hoặc Boolean, biến sẽ được ép kiểu về string và lưu vào value
- **isNull:** là một biến boolean. isNull mang giá trị TRUE khi biến chưa được gán hoặc chưa được nhập giá trị, giá trị gán vào nó không hợp lệ

Ngoài các thuộc tính, VarInfo còn chứa các hàm đặc biệt để làm việc với dữ liệu:

- Các hàm để kiểm tra tính hợp lệ của dữ liệu:

```
        static private bool ValidNumberValue(string value)
        {
            try
            {
                double temp = double.Parse(value);

                // Infinity and Nan should be treated as not valid
                if( double.IsInfinity(temp) )
                    return false;
                if( double.IsNaN(temp) )
                    return false;

                return true;
            }
            catch
            {
                // take advantage of C# try-catch to check if this is a
number
                // double.Parse will throw an exception if value does not
represent a number
                return false;
            }
        }

        static private bool ValidBooleanValue(string value)
        {
            // make sure boolean value is not case sensitive
            value = value.ToUpper();

            // boolean value can only be TRUE, FALSE, YES or NO
            return ( value == "TRUE" || value == "FALSE" || value == "YES"
|| value == "NO");
        }

        static private bool ValidStringValue(string value)
        {
            // string is always a string :v
            return true;
        }
```

- Hàm để chuẩn hoá dữ liệu: đối với kiểu Boolean, chỉ cần chuyển thành chữ hoa để đảm bảo giá trị của boolean không phân biệt hoa/thường. Đối với kiểu Number, người dùng có

thể nhập những thứ không mong muốn như số 0 thừa, số bắt đầu bằng dấu "." (được C# hiểu là 0.),... hàm này giúp chuyển con số đó thành cách viết hợp lý hơn

```
        static public string StandardizeValue(string value, DataType
dataType)
        {
            // stop standardizing if the value itself is not valid
            if( !ValidValue(value, dataType) )
                return value;

            string result = value;
            if( dataType == DataType.FA_Boolean )
            {
                // uppercase all character of value to make sure this is
not case sensitive
                result = value.ToUpper();
            }
            else
            if( dataType == DataType.FA_Number )
            {
                // user may input some weird stuff...
                // the solution is just parse it to a C# double and change
back to string :)
                double doubleVal = double.Parse(value);
                result = doubleVal.ToString("0." + new string('#', 339));
            }

            return result;
        }
```

- Các hàm dùng để chuyển đổi từ kiểu dữ liệu của FlowArt thành kiểu dữ liệu tương đương của C# và ngược lại. Kiểu Number, String, Boolean của FlowArt lần lượt tương đương với các kiểu dữ liệu Double, String, Boolean của FlowArt
- Hàm để kiểm tra 2 giá trị có bằng nhau hay không: thực hiện đơn giản bằng cách chuyển đổi thành các dữ liệu của C# và so sánh. Riêng kiểu dữ liệu Number được so sánh bằng cách đặc biệt bằng phương pháp kiểm tra sai số: $x = y \Leftrightarrow |x-y| \leq \varepsilon$. Với ε được chọn 10^{-9}.

Các biến của người dùng được quản lý thông qua một cấu trúc dữ liệu Sorted Dictionary có sẵn trong C#. Danh sách này được quản lý bởi class FlowDataManager.

```
SortedDictionary< string, VarInfo > UserVariables
```

Các chức năng để người dùng tương tác với biến chủ yếu là xử lý khéo léo chuỗi, với hướng tiếp cận Naive, chắp vá dần các edge cases. Ngôn ngữ được quy định trong FlowArt bao gồm một số ký tự đặc biệt để đánh dấu các thành phần trong câu lệnh của người dùng

- Khai báo biến:
    - Cú pháp: <Tên kiểu dữ liệu>: <Tên biến>. Dấu ":" được sử dụng đặc biệt cho việc khai báo biến. Hàm xử lý nhận biết câu lệnh gán bằng cách kiểm tra trong chuỗi chỉ có ký tự đặc biệt là ":".
    - Các vấn đề có thể thấy ngay khi thực hiện câu lệnh khai báo: kiểm tra tính hợp lệ của tên biến, kiểm tra biến đã được khai báo và đang được sử dụng hay không, kiểm tra kiểu dữ liệu có phải là một trong các kiểu dữ liệu cho phép hay không
    - Quy định đặt tên biến khá giống với đa số ngôn ngữ lập trình: chỉ được dùng các ký tự latin, các ký tự số và ký tự "_"; Tên biến không được bắt đầu bằng số.
    - Cách xử lý phép khai báo biến được tóm tắt thông qua các bước sau:

```
        public void HandleDeclaration(string declaration, out string
varName, out string ErrorReport)
        {
            // remove unwanted spaces
            string temp = RemoveSpacing(declaration);
            ErrorReport = "";

            // if the declaration itself is empty, throw an error
immediately
            if( temp == "" )
            {
                ErrorReport = "Illegal expression";
                varName = "";
                return;
            }

            string typeName = "";
            varName = "";

            // slpit 2 sides
            bool leftSide = true;
            for(int i = 0 ; i < temp.Length; i++)
            {
                // typeName is the substring before ":"
                // varName is the substring after ":"
                ...
            }
```

```
            // if ":" character is not found...
            if( leftSide )
            {
                varName = declaration;
                return;
            }
            typeName = typeName.ToUpper();
            DataType dataType;

            // get FA data type that corresponds to the typeName of user
            switch (typeName)
            {
                // dataType = < new var dataType >
                ...
            }

            // Add this new variable to UserVariables
            DeclareVariable(varName, dataType, out ErrorReport);
            if( ErrorReport != "" )
            {
                return;
            }
        }
```

- Trong đó, hàm DeclareVariable được viết như sau:

```
        public void DeclareVariable(string varName, DataType dataType, out
string ErrorReport)
        {
            if( ! ValidName(varName) )
            {
                ErrorReport = "Error: Invalid variable name: " + varName;
                return;
            }

            if( Exist(varName) )
            {
                ErrorReport = "Error: Duplicate variable name: " +
varName;
                return;
            }

            ErrorReport = "";
            VarInfo temp = new VarInfo(dataType, "", true);

            // add new variable
            UserVariables.Add(varName, temp);
        }
```

- Gán biến:

- Cú pháp: <Tên biến> = <Giá trị / Biểu thức>. Dấu "=" được sử dụng đặc biệt cho việc gán biến. Hàm xử lý nhận biết câu lệnh gán bằng cách kiểm tra trong chuỗi chỉ có ký tự đặc biệt là "=".

- Khởi tạo biến:
  - Cú pháp: <Tên kiểu dữ liệu> : <Tên biến> = < Giá trị / Biểu thức >. Phép khởi tạo thực chất là sự kết hợp giữa phép khai báo và phép gán biến. Hàm xử lý nhận ra câu lệnh khởi tạo bằng cách kiểm tra trong chuỗi có 2 ký tự đặc biệt lần lượt là ":" và "=".
  - Tính hợp lệ của phép khởi tạo biến tương tự như việc xét tính hợp lệ của việc việc khai báo biến và gán biến
  - Phép khởi tạo và phép gán được xử lý chung trong hàm sau:

```
        public VarInfo HandleAssignment(string assignment, out string
ErrorReport)
        {
            string temp = RemoveSpacing(assignment);
            string varName = "";
            string NameAndType = "";
            string expression = "";
            ErrorReport = "";

            if( temp == "" )
            {
                ErrorReport = "Illegal expression";
                return VarInfo.NullVal;
            }
            // splitting 2 sides
            bool leftSide = true;
            for(int i = 0 ; i < temp.Length ; i++)
            {
                // NameAndType is the substring before "="
                // expression is the substring after "="
                ...
            }

            // handle declaration
            // also
            // split varName from NameAndType
            HandleDeclaration(NameAndType, out varName, out ErrorReport);
            // if there is error
            if(ErrorReport != "")
                return VarInfo.NullVal;
```

```
            // if there is no equal sign, then its value is NULL
            if ( leftSide )
            {
                if( Exist(varName) )
                {
                    return UserVariables[varName];
                }
                else
                {
                    AssignVariable(varName, VarInfo.NullVal, out
ErrorReport);
                    return VarInfo.NullVal;
                }
            }

            VarInfo rightSideResult = HandleExpression(expression, out
ErrorReport);
            // if there is error
            if(ErrorReport != "")
                return VarInfo.NullVal;

            AssignVariable(varName, rightSideResult, out ErrorReport);
            // if there is error
            if(ErrorReport != "")
                return VarInfo.NullVal;

            return UserVariables[varName];
        }
```

- Trong đó hàm AssignVariable được viết như sau:

```
        public void AssignVariable(string varName, VarInfo varInfo, out
string ErrorReport)
        {
            if (! Exist(varName) )
            {
                ErrorReport = "Error: Not declared variable name: " +
varName;
                return;
            }

            VarInfo temp = UserVariables[varName];

            if( varInfo.type != temp.type )
            {
```

```
                ErrorReport = "Error: Wrong expected data type: " +
varName;
                return;
            }

            ErrorReport = "";
            temp = new VarInfo(varInfo);
            UserVariables[varName] = temp;
        }
```

- Xoá biến:
    - Cú pháp: ~ <tên biến>. Dùng để xoá 1 biến nếu biến này đã từng tồn tại trước đó
    - Không giống như 2 lệnh kia, lệnh xoá biến có thể được thực hiện một cách đơn giản bằng cách kiểm tra ký tự đầu tiên (“~”), và phần sau đó có phải là tên một biến hay không. Cụ thể:

```
        public void HandleDeletion(string expression, out string
ErrorReport)
        {
            ErrorReport = "";
            expression = RemoveSpacing( expression );

            if( expression == "" )
                return;
            if( expression[0] != '~' )
                return;

            string varName = RemoveSpacing( expression.Remove(0,1) );
            DeleteVariable(varName, out ErrorReport);
            if(ErrorReport != "")
                return;
        }
```

    - Hàm DeleteVariable được định nghĩa như sau:

```
        private void DeleteVariable(string varName, out string
ErrorReport)
        {
```

```
            ErrorReport = "";

            if(! ValidName(varName) )
                ErrorReport = "Invalid variable name: " + varName;

            UserVariables.Remove(varName);
        }
```

b) Hằng số (Constants)

★ Giới thiệu:

- Ở FlowArt, hằng số được chia làm 2 loại: hằng số thông thường và hằng số đặc biệt.
  - Hằng số thông thường (Regular constants): Với các giá trị hằng số mang kiểu Number, người dùng có thể nhập trực tiếp cách biểu diễn của số đó, sử dụng dấu "." cho dấu phẩy thập phân và dấu "-" để biểu diễn số âm. Với các giả trị hằng số mang kiểu String, người dùng có thể nhập giá trị của chuỗi trong cặp dấu """, riêng ký tự """ được biểu diễn bằng """".
  - Hằng số đặc biệt (Special constant): Các hằng số built-in với các giá trị đặc biệt VD: True, False, Yes, No, Pi, E,... Các hằng số được biệt phải được nhập bắt đầu bằng ký tự đặc biệt "$" (VD: $True)

★ Kỹ thuật: để việc định nghĩa thêm hằng số Built-in cho FlowArt một cách thuật tiện, nhóm đã tận dụng khả năng của class Object và tính đa hình trong C#:

```
        static private void AddConst(string constName, object value)
        {
            constName = constName.ToUpper();

            // handle if value has data type double
            if( value is double )
            {
                double temp = (double)value;
                // VarInfo has a constructor to create a Number variable
from a double
                BuiltInConst.Add( constName, new VarInfo(temp) );
                return;
            }
```

```
            ... // Do the same with int, bool, string, char

            throw new Exception("unhandled datatype");
        }
```

Sau đó việc thêm 1 hằng số có thể thực hiện ngay chỉ với 1 dòng code!

```
AddConst("Yes",                 true);
AddConst("No",                  false);
AddConst("Pi",                  Math.PI);
AddConst("E",                   Math.E);
```

c) Xây dựng hệ thống các hàm hỗ trợ trong lưu đồ (Various functions supported)

★ Giới thiệu:

- Các thao tác trên biến hầu hết là thông qua các hàm.
- Ở FlowArt, các phép toán không được sử dụng (trừ dấu âm cho hằng số âm), mà được chuyển thành các hàm (VD: thay vì viết 1+2 thì phải viết là Sum(1,2)). Cú pháp này được lấy cảm hứng từ các ngôn ngữ lập trình hàm (functional programming language).
- FlowArt hỗ trợ một số lượng hàm dựng sẵn (Built-in functions) phong phú: hiện đã có hơn 70 functions (và các overload)

★ Kỹ thuật:

- Đã dự kiến trước số lượng built-in functions có thể rất nhiều, nên nhóm dành rất nhiều thời gian cho việc thiết kế code trước khi thực sự bắt tay vào lập trình.
- Thiết kế được lựa chọn là cho tất cả các hàm Built-in được thiết kế cùng 1 delegate trong C#. Cụ thể, delegate này đại diện cho tất cả các hàm nhận vào một danh sách các tham số (parameters) có kiểu dữ liệu VarInfo, và luôn trả về một giá trị VarInfo

```
Func< List<VarInfo>, VarInfo>>
```

- Từ đó mỗi hàm thuộc delegate có thể được định nghĩa một cách thuận tiện nhất:

```
        static public VarInfo Sum( List<VarInfo> parameters )
        {
            double temp = 0;
            foreach (var parameter in parameters)
            {
                if( parameter.type == DataType.FA_Number &&
!parameter.isNull )
                    temp += parameter.toCS_Double();
```

```
            else
                return VarInfo.NullVal;
        }

        VarInfo result = new VarInfo( temp );
        return result;
    }

    static public VarInfo Sub( List<VarInfo> parameters )
    {
        // check if the parameters are 2 numbers
        if( CheckParameterTypes(parameters, "NN") )
        {
            // get 2 number
            double x = parameters[0].toCS_Double();
            double y = parameters[1].toCS_Double();
            double temp;

            try
            {
                temp = x - y;
            }
            catch
            {
                // return error
                return VarInfo.NullVal;
            }

            VarInfo result = new VarInfo(temp);
            return result;
        }

        return VarInfo.NullVal;
    }
```

- Với việc coi hàm là 1 kiểu dữ liệu (delegate), các hàm có thể dễ dàng thêm vào danh sách hàm cho người sử dụng:

```
// Built-In functions container
static private SortedDictionary< string, Func< List<VarInfo>, VarInfo > >
                BuiltInFunctions;
...
AddFunction( "Sum",     FlowArtLib.Sum );
AddFunction( "Mul",     FlowArtLib.Mul );
AddFunction( "Min",     FlowArtLib.Min );
AddFunction( "Max",     FlowArtLib.Max );
...
```

- Phương pháp thiết kế code trên đã chứng minh rõ hiệu quả làm việc của nó khi nhóm đã có thể cung cấp người dùng hơn 70 hàm dựng sẵn mà không làm ảnh hưởng đến thời gian làm các kế hoạch khác của nhóm.

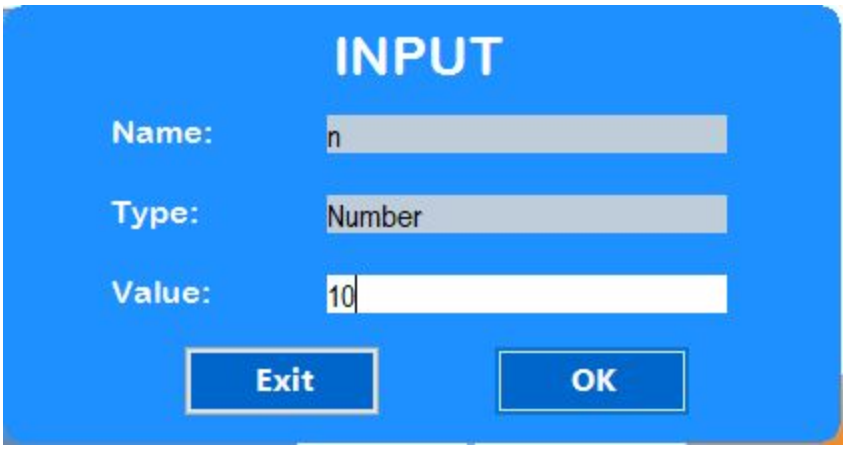
d) Xử lí khối hình trong lưu đồ (Block processing)

❑ **Khối Start:**

- đánh dấu vị trí bắt đầu chạy flow. Người dùng chỉ được vẽ đúng 1 khối Start, nếu không, sẽ nhận thông báo lỗi.
  - ★ Kỹ thuật: Do người dùng có thể tạo thêm và xoá Start block một cách tuỳ ý, việc truy vết khối Start Block nhanh là rất khó khăn. Thay vào đó nhóm đã tiếp cận việc tìm khối Start Block bằng phương pháp Brute force: duyệt tất cả các khối block để tìm vị trí của Start Block (cũng như đếm số lượng khối Start Block trên lưu đồ)

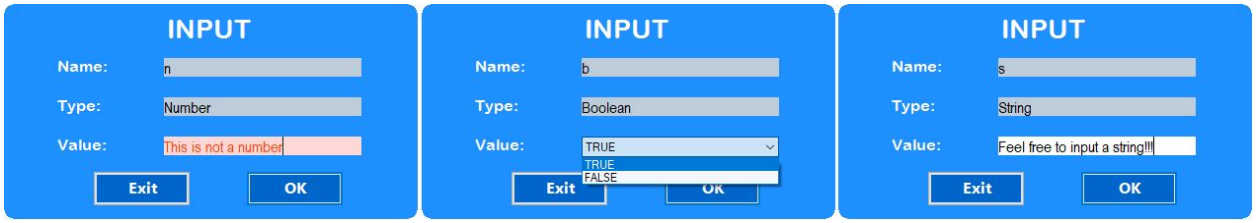
❑ **Khối Scan:**

- Người dùng có thể tuỳ ý nhập giá trị của một biến ngay trong lúc chạy lưu đồ. Khi quá trình chạy đến khối Scan, một cửa sổ nhập sẽ được hiện ra:



- Tuỳ vào kiểu dữ liệu của người nhập, khối Scan sẽ cung cấp những cách nhập liệu khác nhau. Khi nhập giá trị Number, TextBox "Value" sẽ chuyển sang màu đỏ và không cho

phép người dùng nhập nếu xâu input không là biểu diễn của một số. Khi nhập giá trị Boolean, Người dùng sẽ nhận được ComboBox thay cho TextBox để nhập giá trị True/False



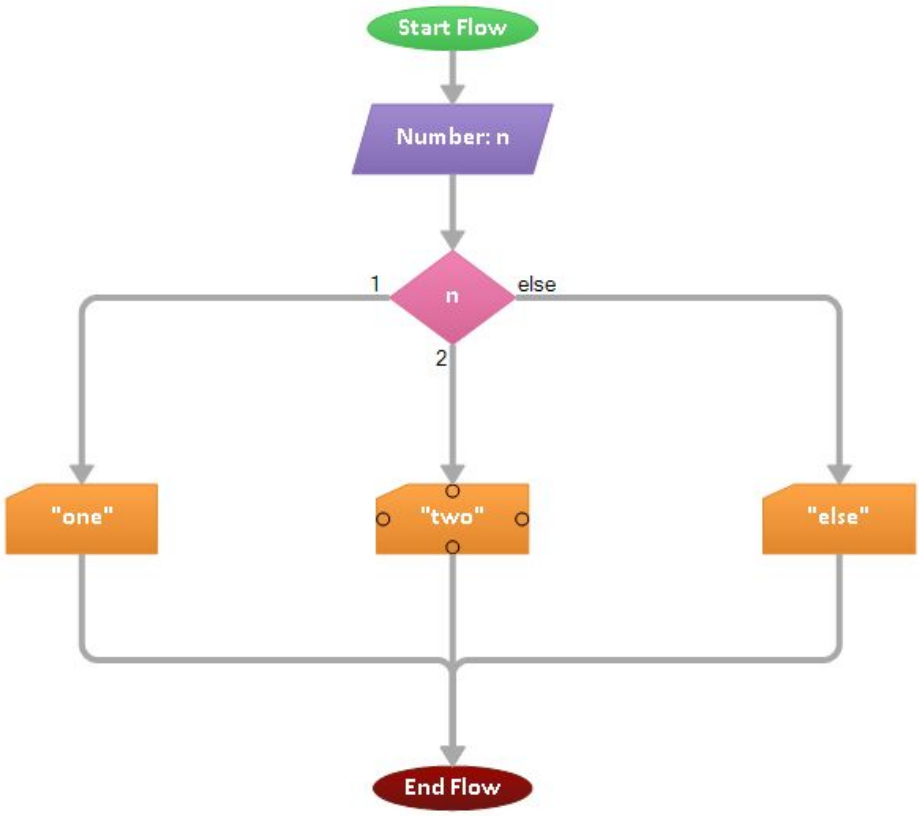
- Người dùng có thể nhập một biến đã được khai báo trước đây bằng cách đặt tên biến cần nhập vào khối Scan. Hơn nữa, người dùng cũng có thể khai báo biến ngay tại khối này.

❑ Khối Process:

- Khối Process là nơi tập trung các chức năng xử lý dữ liệu của người dùng: phép khai báo, khởi tạo, gán và xoá biến đều được thực hiện bởi khối này với đúng cú pháp cần thiết
  - ★ Kỹ thuật: Khối Process là chính là khối có nhiều hành vi nhất trong số tất cả các khối. Để dễ dàng giải quyết vấn đề nhận dạng loại lệnh, FlowArt đưa ra các quy định về ký tự đặc biệt. Dựa vào các ký tự đặc biệt đó, khối Process có thể nhận biết loại hành vi cần thực hiện.

❑ Khối Condition

- Khối Condition thực hiện chức năng rẽ nhánh của các cấu trúc thuật toán.
- Khác với khối Condition được thể hiện trên các sơ đồ khối được dùng trong giảng dạy và các phần mềm vẽ lưu đồ trước đây, FlowArt giới thiệu khối Condition với cơ chế rẽ nhánh dựa trên *giá trị* của một biến/biểu thức mà nó chứa, chứ không dựa trên *điều kiện* → tiện hơn trong việc rẽ nhiều nhánh, với nhiều điều kiện lồng nhau. Hành vi này tương tự với cấu trúc lệnh *switch case* của C#. Giá trị đầu ra của khối Condition còn có thể có một nhãn đặc biệt là "**else**", chương trình sẽ chọn đi vào nhánh này nếu tất cả các nhánh còn lại không thoả khớp.



★ Kỹ thuật: Đây là khối duy nhất mà việc quyết định đường đi phải dựa trên giá trị của giá trị chứa trong nó. Do đó phá huỷ tính chất chung của hàm tìm Node liền kề (vốn không quan tâm đến giá trị của Block). Giải pháp cho việc này là chấp nhận và giữ nguyên một cấu trúc chung cho hàm:

```
public FlowBlock GetNextBlock(FlowBlock CurrentBlock, VarInfo blockResult,
out string ErrorReport)
```

Ngoài ra còn phải kiểm tra kỹ các điều kiện giá trị đầu ra: giá trị phải có nghĩa và không bị trùng nhau, không được có 2 nhãn else,...

❏ Khối Print:

- Khối Print xử lý nhiệm vụ in một giá trị biểu thức đến khu vực Result (thuộc khu vực thông báo).

❏ Khối End

- Khi gặp khối End, quá trình chạy flow sẽ kết thúc.

★ Kỹ thuật chung:

- Các hàm xử lý logic cho các khối Block đều được truyền kèm theo một tham số đặc biệt: ErrorReport. Biến này có nhiệm vụ thông báo lỗi về lại cho hàm trước đó. Tham số này trả về 1 xâu rỗng khi và chỉ khi không có lỗi xảy ra. Việc thông

báo lỗi bằng 1 tham biến truyền ngược về 1 hàm xử lý chung đem đến nhiều thuận tiện rõ rệt trong việc thông báo lỗi

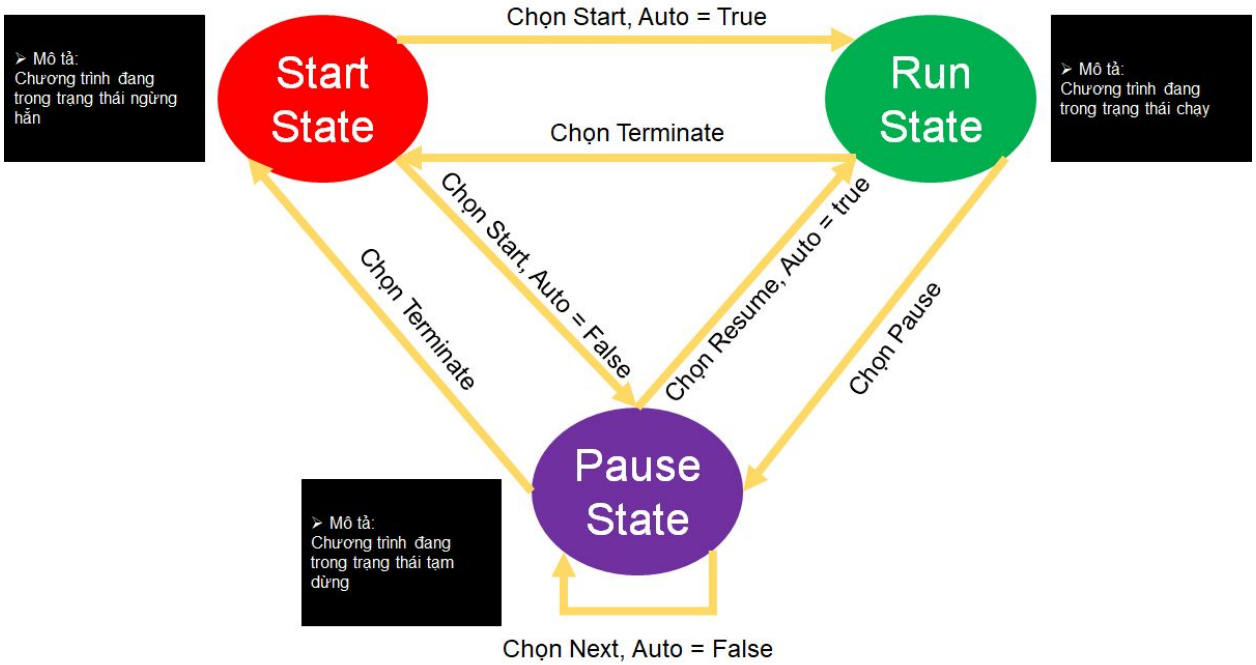
- Event handler đã được sử dụng để bắt các sự kiện kết thúc flow bởi khối end, xảy ra lỗi, kết thúc flow bởi người dùng chọn "**Terminate**", khi Flow vừa chạy tới khối tiếp theo, và khi thay đổi *trạng thái chạy*,...

e) Chạy mô phỏng lưu đồ thuật toán: (Simulation Implementing)

- **Điều khiển (Control)**
  - Chạy (Play): Nút play để bắt đầu chạy mô phỏng chương trình, hoặc tiếp tục khi chương trình vẫn chưa hoàn thành việc chạy. Tuỳ vào trạng thái hiện tại của flow, nút play sẽ được thay đổi thành nút "Resume" hoặc "Next" cho hợp lý.
  - Tạm dừng (Pause): Tạm dừng thực thi sơ đồ. Người dùng có thể tiếp tục chạy, hoặc dừng hẳn
  - Thoát (Terminate): dừng hẳn việc mô phỏng lưu đồ

★ Kỹ thuật:

- Để đảm bảo quản lý hết mọi trường hợp có thể xảy ra nhóm đã sử thiết kế một finite state machine (FSM) như bên dưới. Hiệu quả của việc sử dụng FSM có thể thấy: việc hiện thực hoá các chức năng điều khiển diễn ra nhanh chóng, các điều kiện được quản lý cân đối, không rườm rà khó hiểu nhưng vẫn đảm bảo đầy đủ.



Sơ đồ xử lí trạng thái + sự kiện trong simulation

- Ở State Run (đang chạy), để việc chạy lưu đồ không ảnh hưởng đến việc thực thi phần còn lại của chương trình. Nhóm đề ra 2 giải pháp là Timer (bắt sự kiện Tick) hoặc Multithread, trong đó phương pháp Timer được chọn. Trong quá trình thiết lập chức năng chạy, nhóm đã gặp phải lỗi bất đồng bộ (Asynchronization) và đã sửa lỗi bằng **phương pháp Blocking**

- **Hiển thị thông điệp**
  - Kết quả khi chạy thành công lưu đồ (Result handling).

  - Tập lệnh theo trình tự được thực thi khi chạy và xuất lỗi (nếu có) (Log and error handling).

- **Bảng theo dõi biến (Watch box):** bảng biến sẽ được cập nhật liên tục trong quá trình chạy lưu đồ. Bảng theo dõi biến đề cập rõ tên, kiểu dữ liệu và giá trị (dưới dạng code) của từng biến hiện có. Biến chưa có giá trị được gán cho một giá trị đặc biệt là $NULL. Biến đã được xoá đi sẽ tự động biến mất khỏi bảng theo dõi.

- **Live coding:** lấy cảm hứng từ một số ngôn ngữ lập trình cho phép chức năng live coding, FlowArt cho phép người dùng chạy thay đổi cấu trúc Flow ngay trong khi chạy Flow để thay đổi hành vi Flow.

3. Các chức năng bổ sung

Song song thực hiện hai mục tiêu trên, nhóm cũng tạo thêm các chức năng đặc biệt, phục vụ người dùng một cách tốt nhất

a) Auto Complete

★ Giới thiệu:

- Việc nhớ và nhập chính xác tên hàm, tên biến, tên hằng là một việc rất gây khó khăn cho người dùng. Nắm bắt vấn đề trên, nhóm đã tiến hành thực hiện chức năng nhắc code - Auto Complete
- Chức năng Auto Complete có khả năng không chỉ gợi ý tên hàm, tên hằng (Built-in) mà còn nhận biết được các tên biến do người sử dụng vừa khai báo ở bất cứ đâu trong lưu đồ.
- Chức năng Auto Complete không chỉ đưa ra những gợi ý từ tiền tố của từ đang nhập, mà gợi ý từ khớp với từ đang nhập nhất, thông qua các ký tự xuất hiện, vị trí và thứ tự xuất hiện của chúng

★ Kỹ thuật:

- Cơ bản:
  - Sử dụng 1 ListBox gắn liền với TextBox của người dùng. ListBox này sẽ được cung cấp danh sách hàm, hằng và biến ngay lúc Form được khởi tạo.
  - Form bắt sự kiện TextChanged của Textbox để có thể cập nhật liên tục từ sẽ được gợi ý.
  - Form sẽ tự **động tắt gợi ý** khi người dùng đang nhập một **hằng số**, một **hằng chuỗi**, hoặc khi người dùng đang **khai báo một biến** mới
- Nâng cao:
  - Việc so sánh độ ưu tiên giữa các từ được gợi ý dựa trên một số hành vi thường thấy của người dùng: người dùng thường gõ đúng chữ cái đầu tiên; chữ cái cuối cùng có thể sai hoặc sai thứ tự; có thể nhập thiếu chữ cái; những chữ cái khớp càng gần nhau càng tốt;... Từ đó xây dựng một hàm Heuristic phù hợp nhất.
  - Thuật toán được sử dụng để sắp xếp là **Quicksort**

b) Tùy chỉnh quá trình chạy lưu đồ (Simulation User setting)

- Auto Run: Người chỉnh có thể bật chế độ chạy lưu đồ tự động, hoặc chạy từng bước bằng cách tự bấm Next
- Speed Level: Tuỳ chỉnh tốc độ chạy của lưu đồ
- Fading Effect (class DarkenBlockEffectPlayer): tạo hiệu ứng sáng dần sau khi in đậm block trong lúc chạy lưu đồ. Hiệu ứng này không chỉ trông đẹp trong lúc chạy mà còn giúp dễ theo dõi luồng chạy của lưu đồ hơn. Tuy nhiên hiệu ứng này có thể gây ra độ trễ (lag) cho chương trình.

- ❏ Sound Effect (class AudioManager): Tạo hiệu ứng âm thanh trong lúc chạy, có các âm thanh đặc biệt khi chạy gặp lỗi hoặc chạy thành công. Người cùng có thể tuỳ chỉnh việc bật/tắt âm thanh

- ❏ Clear Result: Người dùng có thể tuỳ chỉnh việc xoá kết quả (result, log) cũ trước khi chạy lại lưu đồ.

- ❏ Notify on end: Người dùng có thể cho phép chương trình hiện Message Box thông báo khi chương trình chạy gặp lỗi hoặc khi chương trình chạy thành công

c) Searching

★ Giới thiệu:

- Có thể sử dụng tổ hợp Ctrl + F để thực hiện việc tìm kiếm một từ khoá thuộc các khối trên lưu đồ.
- Nếu người dùng đang chọn ít nhất 2 khối, quá trình tìm kiếm sẽ chỉ xét trên các khối được chọn. Ngược lại, chương trình sẽ tự động chọn tất cả khối và bắt đầu quá trình tìm kiếm (trên tất cả khối)
- Người dùng có thể tiếp tục gõ Enter để hiện thị kết quả tìm kiếm tiếp theo

★ Kỹ thuật:

- Do việc duyệt các giá trị khối diễn ra không liên tục trong 1 vòng lặp mà tuỳ thuộc vào hành vi người dùng. Danh sách khối được chứa trong 1 container đặc biệt của framework Northwood.Go, nên việc tìm kiếm và sử dụng con trỏ để duyệt qua container này đã đem lại một số trở ngại nhất định. Tuy nhiên nhóm đã tìm hiểu và sử dụng thành công Enumerator:

```
SearchEnumerator = ViewToSearch.Selection.GetEnumerator();
...
if(SearchEnumerator.Current is FlowBlock)
{
    FlowBlock Block = SearchEnumerator.Current as FlowBlock;
    if ( MatchedBlock(Block, KeyWord) )
        return Block;
}
```

IV. THIẾT KẾ CODE (CODE DESIGN)

1. Kiến trúc chương trình:

Chương trình được xây dựng trên cơ sở front end - back end và sự kết hợp giữa 2 phần đó trong lập trình phần mềm.
Phần code được chia thành các lớp với nhiệm vụ rõ ràng, dễ cho việc phát triển, bảo trì và sửa chữa mã nguồn:

MainBase	Là class chính của chương trình, tổng hợp tất cả tương tác giữa người dùng và chương trình, giữa các dữ liệu trong chương trình với nhau, và hàm Main để khởi động
MainInterface	Là class chính của chương trình, form tổng hợp toàn bộ giao diện và các thành phần được cài đặt
FlowArtLib, FlowArtLib_001, FlowArtLib_002	Là thư viện các hàm được xây dựng (built-in functions) của FlowArt, phục vụ cho việc simulation
FlowBlock	Chứa thông tin của 1 khối hình cụ thể
FlowDocument	Chứa các thông tin của 1 design về mặt nội dung và tập object thuộc nó
FlowView	Chứa các thông tin của 1 design về mặt hình thức và tương tác của người dùng với nó
FlowMap	Là một form MDI chứa 1 design map, bao gồm FlowDocument và FlowView kết hợp
FlowDataManager	Quản lý việc xử lí dữ liệu của lưu đồ
FlowRunManager	Quản lý quá trình chạy lưu đồ
FunctionHandler	Quản lý việc xử lí các hàm trong FlowArtLib
LinkHandle	Chứa các thông tin về 1 đường link và cách xử lí nó
SmartAutoCompletionSorter	Quản lý việc tổ chức Auto Complete một cách tối ưu
VarInfo	Chứa thông tin của 1 biến cụ thể

Ngoài ra có những lớp hỗ trợ khác:

AudioManager	Quản lý mặt âm thanh khi mô phỏng lưu đồ
DarkenBlockEffectPlayer	Quản lý việc hightlight các block khi mô phỏng
frmBlockEdit	Chứa thông tin của hộp thoại thay đổi nội dung các khối hình trong design
frmScan	Chứa thông tin của hộp thoại nhập dữ liệu từ khối Scan
frmInfo	Chứa thông tin của hộp thoại thay đổi tên design
frmRunSetting	Chứa thông tin của hộp thoại tùy chỉnh chi tiết liên quan đến việc simulation
SpecialConstHandler	Quản lý các giá trị hằng số
ViewBlockSearcher	Quản lý việc tìm kiếm dữ liệu của Search bar

2. Các phương pháp xuất hiện:

a) Tái cấu trúc sơ khai (basic refactoring)

- Delegate: Sử dụng trong xử lý sự kiện. Ngoài ra delegate còn được sử dụng trong việc định nghĩa các hàm dựng sẵn một cách thuật tiện
- Event Handler: Áp dụng việc phát các sự kiện quan trọng trong quá trình chạy flow (có thể tìm thấy trong class FlowRunManager), qua đó áp dụng Observer pattern
- Lazy initialization: được áp dụng để việc ghép chức năng vào 1 class đã có sẵn nhanh và thuận tiện hơn mà không cần phải tìm hoặc chỉnh sửa các hàm constructor (hàm khởi tạo). Ngoài ra, kỹ thuật này còn được tìm thấy ở Singleton pattern
- Finite state machine: nhóm đã áp dụng được FSM trong việc quản lý các trạng thái chạy lưu đồ. Tuy nhiên nhóm chưa thực sự áp dụng được State pattern.

b) Singleton pattern:

★ Mục đích: quản lý toàn bộ project một cách dễ dàng hơn bằng cách sử dụng chỉ 1 đối tượng cụ thể, điều phối mọi hoạt động (áp dụng trong các class MainBase, MainInterface)

V. HOẠT ĐỘNG NHÓM TRONG QUÁ TRÌNH THỰC HIỆN ĐỀ TÀI

1. Phân công nhiệm vụ:

Thành viên	Nhiệm vụ
Lê Hoàng Minh Sơn	Nhóm trưởng, thực hiện quản lí đề tài và kết nối giữa front end và back end, vẽ lưu đồ
Quản Tiến Nghĩa	Nắm chính phần xử lí back end, chạy lưu đồ
Võ Thành Trung	Thực hiện phần UI design

2. Giai đoạn thực hiện:

Thời gian	Hoạt động
Tháng 9/2019 (giai đoạn bắt đầu)	- Lập nhóm, khảo sát, xác định đề tài - Đặt mục tiêu, tạo bản kế hoạch ban đầu về mặt hình thức và chức năng của phần mềm
Tháng 10/2019 (giai đoạn tạo nền móng)	- Tiếp tục tra cứu các tài liệu có nội dung liên quan - Xác định code trên cơ sở chia ra front end (vẽ, tương tác) và back end (xử lí data, mô phỏng) - Vẽ ra mô hình UML để hiện thực hóa phần mềm - Bước đầu bắt tay vào code trên nền tảng vững chắc đã xây dựng
Tháng 11/2019 (giai đoạn làm việc độc lập)	- Tìm ra hướng đi tốt hơn cho đồ án: tận dụng framework hỗ trợ để xây dựng chương trình nhanh hơn, tối ưu về mặt thời gian - Hai nhánh Front và Back đang làm việc độc lập - Thành công sơ khai trong mục tiêu đầu tiên: vẽ được 1 lưu đồ và người dùng có thể thao tác được trong đó
Tháng 12/2019 (giai đoạn cao điểm tập trung và nước rút)	- Hệ thống backend được xây dựng rất ổn định - Thiết kế UI nâng cấp hơn → đúng như kế hoạch - Các feature chính lần lượt được implement - Triển khai ghép front và back để hình thành nên kết quả ban đầu - Hoàn thành tốt đẹp mục tiêu thứ hai: mô phỏng thành công lưu đồ thuật toán → cả 2 mục tiêu đặt ra đều đạt được - Test với nhiều người dùng, fix bug, thêm các chức năng khác để tăng UI và UX

Tháng 1/2020 (giai đoạn về đích)	- Tinh chỉnh, hoàn thiện sản phẩm và báo cáo, chuẩn bị tinh thần thuyết trình về đồ án

3. Thuận lợi, khó khăn:

a) Thuận lợi:

Trong quá trình làm việc, nhóm đã áp dụng được rất nhiều nguyên tắc trong việc phát triển phần mềm chuyên nghiệp, nhờ những đặc điểm sau:

- Các thành viên có tinh thần làm việc trách nhiệm, thực hiện đầy đủ các nhiệm vụ.
- Có kế hoạch rõ ràng ngay từ đầu về cả mặt nội dung và hình thức: các sơ đồ UML (class diagram, state machine, work process,...). Phân chia công việc giữa front end và back end → tạo nền móng vững chắc cho chương trình
- Tính sáng tạo đúng lúc của từng thành viên
- Quản lý nhóm và project hiệu quả thông quả thông qua các dịch vụ: Github, Google Drive, Trello, Facebook, Gmail,...

b) Khó khăn và cách vượt qua:

Tuy nhiên bất cứ dự án nào cũng gặp phải những bất lợi nhất định:

- Ban đầu thời gian làm việc của mỗi thành viên xung đột với nhau, khó gặp mặt trao đổi trực tiếp → làm việc thông qua mọi phương tiện và tính tự chủ của từng thành viên
- Có những lúc xuất hiện sự bất đồng quan điểm của các thành viên → nhóm trưởng đưa ra quyết định cuối cùng sau khi có sự cân nhắc
- Project được đánh giá là khó ngay từ đầu, và đôi lúc có nguy cơ không thể hoàn thành → Tiến hành research mọi khía cạnh, tìm ra hướng đi đúng (vận dụng framework trong việc tối ưu thời gian thực hiện)

VI. TỔNG KẾT ĐỒ ÁN

1. Kết quả đạt được

Nhóm tự đánh giá là đề tài cho môn học này thành công, với việc không những đạt được mục tiêu đặt ra mà còn thực hiện được sản phẩm có nhiều chức năng hơn nữa. Cụ thể:

- Hai mục tiêu của đề tài là vẽ được và chạy được hoàn thành vượt mong đợi
- Giao diện dễ hiểu, không rườm rà như các ứng dụng đã khảo sát
- Biểu diễn trực quan, sinh động các lưu đồ, hơn nữa có thể xuất ra file ảnh của lưu đồ để chia sẻ, in lưu đồ
- Tập trung vào các khối block cần thiết, và cố gắng đầu tư thật tốt cho trải nghiệm người dùng (UX) khi vẽ: align, zoom to fit,...
- Có hỗ trợ Auto complete một cách tối ưu, giống với Visual Studio Code
- Có hệ thống ngôn ngữ tự định nghĩa riêng, khai báo biến, input và output đều được làm rõ ràng, không mập mờ và gây khó trong quá trình sử dụng
- Có hệ thống built-in function đảm bảo đáp ứng hầu hết các loại lưu đồ
- Chức năng watch box và log rất hiệu quả trong việc kiểm tra quá trình chạy thuật toán
- Hệ thống báo lỗi: xuất hiện một dòng thông báo cho biết lỗi ở Log, kèm theo highlight ngay trên lưu đồ
- Có chức năng pause để lưu đồ dừng chạy ở một khối, và speed level để tùy chỉnh quá trình chạy cho các khối. Các chức năng này giúp người dùng tiện trong việc theo dõi luồng chương trình
- Có thanh tìm kiếm từ khóa
- Có mục Help menu, cung cấp documentation về chương trình
- Hỗ trợ âm thanh
- Phần mềm mang tính học thuật và nghiên cứu

2. Hướng phát triển

Dự án sẽ tiếp tục được phát triển sau báo cáo này theo kế hoạch nhóm đặt ra. Nhiều feature khác sẽ được thêm vào để ngày càng hoàn thiện FlowArt:

- Thêm kiểu dữ liệu mảng
- Có kèm hướng dẫn tour guide cho người mới sử dụng phần mềm
- Custom block: có khả năng lưu lại một lưu đồ dưới dạng một hàm và gọi lại khi cần
- Mở rộng thư viện các hàm
- Sample Library: tích hợp thư viện các lưu đồ mẫu được vẽ sẵn
- Tối ưu bộ nhớ
- Highlight syntax khi nhập thông tin khối block

So với mục đích hoàn thành một đồ án môn học, nhóm đặt ra mục tiêu xa hơn là ứng dụng đề tài này trong môi trường giáo dục, hỗ trợ các bạn học sinh, sinh viên mới tìm hiểu về lập trình, tư

duy thuật toán. Phần mềm hoàn toàn có thể sử dụng trong các môn học Nhập môn lập trình, Giải thuật, phục vụ cộng đồng các bạn đam mê tin học.

VII. KHÁC

1. Tài liệu tham khảo (reference) và các trang web hỗ trợ:

-
- Northwoods.go documentation
- GoDiagram Win for .NET & .NET Core (User Guide)
- Github
- DevExpress documentation
- Stackoverflow
- Northwood.Go forum
- Freesound.org

2. Bảng tra các từ khóa sử dụng trong báo cáo (word appendix)

Asynchronize: Bất đồng bộ (- Synchronize: đồng bộ hoá)
Block: khối hình
Block catalogue: bảng chọn khối hình
Block processing: xử lí thông tin của một khối hình trong lưu đồ
Built-in: định nghĩa, xây dựng sẵn
Clipboard: nơi lưu giữ tạm thời các object vừa cut hoặc copy
Constant: hằng số
Data processing: xử lí dữ liệu trong lưu đồ
Design: bản vẽ lưu đồ, là một thể hiện lưu đồ mà người dùng vẽ
Design Map: trang vẽ, là khu vực để vẽ và di chuyển các khối hình trên đó
Edit: chỉnh sửa thiết kế
Expression: Dòng lệnh / Biểu thức
Finite State Machine: Máy trạng thái hữu hạn
FlowArt: tên phần mềm
Function: Hàm
Functional Programming: lập trình hàm, một mô hình lập trình.

Link: đường nối giữa các khối hình
Log: tập hợp các dòng lệnh được thực hiện theo trình tự thời gian khi simulation
MDI: Multiple Document Interface
Message area: khu vực thông báo (nơi có trang result và trang log)
Object: vật thể, bao gồm các khối hình và đường nối
Observer pattern: mẫu thiết kế Observer

Overload functions: nạp chồng hàm, các hàm khác nhau về giá trị input có thể có các
Regular constants: hằng thông thường, thể hiện giá trị của nó thông qua 1 quy tắc nhất định
Simulation: mô phỏng quá trình chạy lưu đồ
Singleton pattern: mẫu thiết kế Singleton
Special constants: hằng đặc biệt, các hằng số dựng sẵn, mang các giá trị đặc biệt
State pattern: mẫu thiết kế State
User variables: biến do người dùng định nghĩa và tương tác
Variable assignment: gán biến
Variable declaration: khai báo biến
Variable deletion: xoá biến / huỷ biến
Variable initialization: khởi tạo biến
Watch Box: bảng theo dõi biến

3. Bảng phụ lục phím tắt (hotkey appendix)

STT	Tổ hợp phím	Chức năng
1	Ctrl + N	Bằng với việc bấm New trong File tab
2	Ctrl + O	Bằng với việc bấm Open trong File tab
3	Ctrl + S	Bằng với việc bấm Save trong File tab
4	Ctrl + Shift + S	Bằng với việc bấm Save All trong File tab
5	Ctrl + P	Bằng với việc bấm Print trong File tab
6	Ctrl + Shift + P	Bằng với việc bấm Print Preview trong File tab

7	Ctrl + F4	Bằng với việc bấm Close trong File tab
8	Ctrl + F	Tìm kiếm từ khóa trong lưu đồ
9	Ctrl + C	Bằng với việc bấm Copy trong Edit tab
10	Ctrl + X	Bằng với việc bấm Cut trong Edit tab
11	Ctrl + V	Bằng với việc bấm Paste trong Edit tab
12	Ctrl + Z	Bằng với việc bấm Undo trong Edit tab
13	Ctrl + Y	Bằng với việc bấm Redo trong Edit tab
14	Ctrl + A	Bằng với việc bấm Select All trong Edit tab
15	Del	Bằng với việc bấm Delete trong Edit tab
16	Ctrl + L	Bằng với việc bấm Link among selection trong Insert tab
17	Ctrl + Shift + H	Bằng với việc bấm Align Horizontally trong Format tab
18	Ctrl + Shift + V	Bằng với việc bấm Align Vertically trong Format tab
19	Alt + Left	Bằng với việc bấm Window Mode trong View tab
20	Alt + Right	Bằng với việc bấm Distraction Free trong View tab
21	Ctrl + ScrollUp	Bằng với việc bấm Zoom In trong View tab

22	Ctrl + ScrollDown	Bằng với việc bấm Zoom Out trong View tab
23	Shift + F6	Bằng với việc bấm Zoom to Fit trong View tab
24	Ctrl + F6	Bằng với việc bấm Zoom Normal trong View tab
25	Ctrl + 1	Bằng với việc bấm Catalogue trong View tab
26	Ctrl + 2	Bằng với việc bấm Message Area trong View tab
27	Ctrl + 3	Bằng với việc bấm Watchbox trong View tab
27	Ctrl + G	Bằng với việc bấm Guide Book trong Help tab
28	Ctrl + H	Bằng với việc bấm Cheat Sheet trong Help tab
29	Ctrl + I	Bằng với việc bấm More Info trong Help tab
30	F9	Bằng với việc bấm Play trong Simulation tab
31	Shift + F9	Bằng với việc bấm Pause trong Simulation tab
32	Ctrl + F9	Bằng với việc bấm Terminate trong Simulation tab
33	Ctrl + Q	Bằng với việc bấm Setting trong Simulation tab