LẬP TRÌNH GIAO DIỆN TRÊN WINDOWS

Nội dung

1	Lịch sử phát triển lập trình GUI trên Windows
2	Ứng dụng WinForms “Hello World” (demo)
3	Tổng quan về ứng dụng WinForms
4	Xử lý sự kiện trong ứng dụng WinForms

Lịch sử phát triển lập trình GUI

1. WinForms (Windows Forms) – Đầu thập niên 2000

- Ra mắt cùng với .NET Framework (2002).
- Là công nghệ GUI dựa trên **wrapper của Win32 API**, giúp lập trình viên tạo giao diện bằng kéo-thả trong Visual Studio.
- Ưu điểm: đơn giản, dễ học, tích hợp tốt với Visual Studio.
- Hạn chế: giao diện ít hiện đại, khó tùy biến UI phức tạp, hiệu suất không cao cho đồ họa nặng

Lịch sử phát triển lập trình GUI

2. WPF (Windows Presentation Foundation) – 2006

- Xuất hiện từ .NET Framework 3.0.
- Dựa trên **DirectX**, hỗ trợ đồ họa vector, 3D, animation, data binding mạnh mẽ.
- Sử dụng **XAML** để tách phần giao diện (UI) và logic (code-behind).
- Cho phép thiết kế UI hiện đại, hiệu ứng mượt hơn nhiều so với WinForms.
- Tuy nhiên: phức tạp hơn, học khó hơn, và ít được dùng trong ứng dụng doanh nghiệp nhỏ

Lịch sử phát triển lập trình GUI

3. UWP (Universal Windows Platform) – 2015

- Giới thiệu cùng Windows 10.
- Hướng đến việc viết một ứng dụng chạy trên nhiều thiết bị: PC, tablet, Xbox, HoloLens.
- Cũng dùng **XAML**, có khả năng sandbox, bảo mật tốt, dễ phân phối qua Microsoft Store.
- Nhưng hạn chế: bị ràng buộc vào Windows 10+, nhiều API khác biệt so với WPF/WinForms, hệ sinh thái chưa thật sự bùng nổ.

Lịch sử phát triển lập trình GUI

4. WinUI (Windows UI Library) – 2018 đến nay

- Ban đầu là **WinUI 2** (dựa trên UWP).
- **WinUI 3** tách biệt hẳn khỏi UWP, trở thành nền tảng UI hiện đại cho Windows.
- Dùng XAML hiện đại, hỗ trợ fluent design, tối ưu cho Windows 11.
- Là bước tiến để thay thế UWP, hướng đến **"native UI framework"** cho Windows trong tương lai.

Lịch sử phát triển lập trình GUI

5. .NET MAUI (Multi-platform App UI) – 2021

- Kế thừa Xamarin.Forms.
- Cho phép viết **một codebase C#/XAML** nhưng build được cho nhiều nền tảng: Windows, Android, iOS, macOS.
- Trên Windows, MAUI dùng **WinUI 3** làm backend giao diện.
- Hướng tới phát triển **ứng dụng đa nền tảng hiện đại** trong hệ sinh thái .NET 6+.

Lịch sử phát triển lập trình GUI

2002	**WinForms**: dễ dùng, ứng dụng doanh nghiệp nhỏ
2006	**WPF**: hiện đại hơn, mạnh mẽ cho desktop app
2015	**UWP**: universal apps cho Windows 10, nhưng không thành công như mong đợi
2018	**WinUI 2/3**: bước chuyển sang UI hiện đại cho Windows 11
2021	**.NET MAUI**: đa nền tảng (Windows + mobile + macOS), tương lai của cross-platform UI

Lịch sử phát triển lập trình GUI

Đặc điểm	WinForms (2002)	WPF (2006)	UWP (2015)	WinUI (2018–nay)	.NET MAUI (2021–nay)
Công nghệ nền	Wrapper của Win32 API, GDI+	DirectX, XAML	XAML, sandbox, Universal API	XAML hiện đại, Fluent Design	XAML + C#, đa nền tảng
Ngôn ngữ chính	C#, VB.NET	C#, XAML	C#, XAML	C#, XAML	C#, XAML
Khả năng UI	Cơ bản, ít hiệu ứng	Vector graphics, 3D, animation, data binding mạnh	UI hiện đại, tối ưu cho touch, sandboxed	Fluent design, UI native Windows 10/11	UI native cho Windows (WinUI) + Android/iOS/macOS
Khai triển	Desktop app Windows	Desktop app Windows	Microsoft Store (Windows 10+)	Desktop app Windows 10/11 (không cần Store)	App đa nền tảng (1 code chạy nhiều OS)
Ưu điểm	Dễ học, kéo-thả trực quan, nhanh	UI hiện đại, tách biệt UI/logic, mạnh cho enterprise app	Universal app, bảo mật, phân phối qua Store	Giao diện hiện đại nhất, thay thế UWP, native Win11	Đa nền tảng, một codebase, tương lai của .NET
Nhược điểm	Giao diện cũ, khó tùy biến	Học khó hơn, phức tạp	Bị giới hạn Windows 10+, API riêng biệt	Chưa phổ biến rộng rãi, còn mới	Còn đang hoàn thiện, có thể lỗi khi cross-platform
Tình trạng hiện tại	Cũ, vẫn dùng trong ứng dụng doanh nghiệp cũ	Vẫn mạnh cho desktop, còn được dùng nhiều	Xu hướng giảm, nhường chỗ cho WinUI	Được Microsoft ưu tiên, tương lai của Windows UI	Đang phát triển mạnh, thay Xamarin.Forms

Tổng quan về ứng dụng WinForms

Một ứng dụng WinForms về mặt giao diện thường có:

- **Form** (cửa sổ chính và các cửa sổ khác)
- **Control** (thành phần UI người dùng tương tác)
- **Component** (thành phần phi giao diện, hỗ trợ logic)
- **Menu/ToolStrip/StatusStrip** để điều hướng và hiển thị trạng thái
- **Layout container** để sắp xếp UI gọn gàng

Tổng quan về ứng dụng WinForms

1. Form

- **Form** là **đơn vị cơ bản nhất** trong giao diện WinForms.
- Mỗi form tương ứng với một cửa sổ (window).
- Có thể có **Main Form** (cửa sổ chính khi chạy app) và các **Dialog Form** (hộp thoại phụ).
- Trong Visual Studio, khi tạo Form1.cs, sẽ có:
  - Form1.cs → chứa code logic)
  - Form1.Designer.cs → chứa code sinh tự động từ Designer (các control bạn kéo-thả).
  - Form1.resx → file resource (lưu chuỗi, hình ảnh...).

Tổng quan về ứng dụng WinForms

2. Control (các thành phần UI)

- Các **Control** được đặt trên Form để tạo giao diện người dùng.
- Có thể chia thành nhóm:
  - **Control nhập/xuất dữ liệu**: TextBox, Label, RichTextBox, PictureBox.
  - **Control điều khiển**: Button, CheckBox, RadioButton, ComboBox, ListBox.
  - **Control bố cục (container)**: Panel, GroupBox, TabControl, SplitContainer.
  - **Control dữ liệu nâng cao**: DataGridView, TreeView, ListView.
- Người dùng sẽ tương tác trực tiếp với các control này

Tổng quan về ứng dụng WinForms

3. Component (phi giao diện)

- Các thành phần hỗ trợ nhưng **không hiển thị trực tiếp** trên UI.
- Ví dụ: Timer, ImageList, ErrorProvider, ToolTip.
- Khi kéo vào form trong Designer, chúng xuất hiện ở **component tray** bên dưới, chứ không nằm trên giao diện.

Tổng quan về ứng dụng WinForms

4. Menu và thanh công cụ

- **MenuStrip**: tạo thanh menu (File, Edit, Help...).
- **ToolStrip**: tạo thanh công cụ (toolbar) với icon, nút nhanh.
- **StatusStrip**: thanh trạng thái ở cuối cửa sổ.

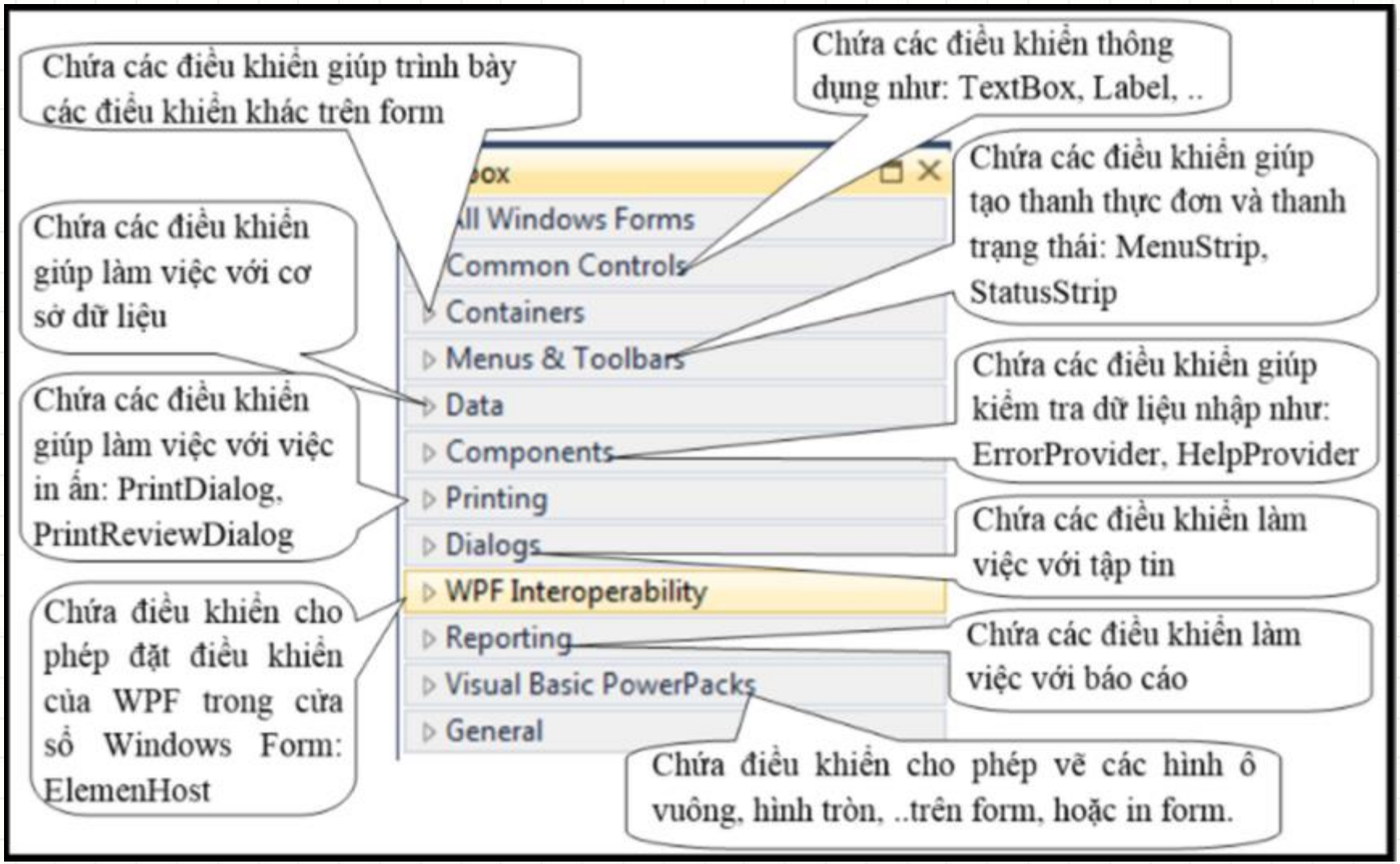
Tổng quan về ứng dụng WinForms

5. Layout và container

- **Panel**: vùng chứa control.
- **FlowLayoutPanel**: sắp xếp control tự động theo dòng.
- **TableLayoutPanel**: bố trí control dạng lưới
- **GroupBox**
- **SplitContainer**
- **TabControl**

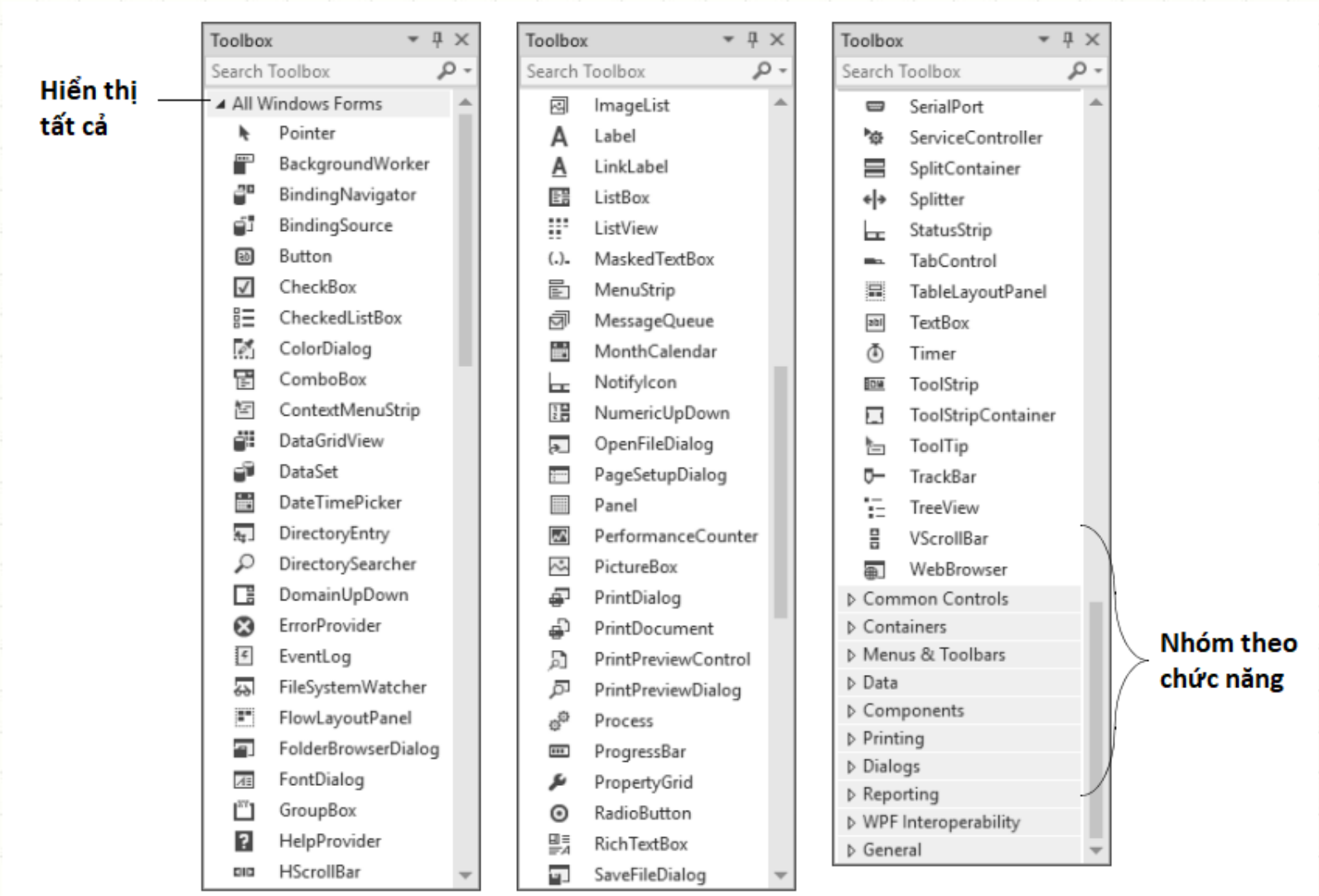
Tổng quan về ứng dụng WinForms

ToolBox trong Visual Studio



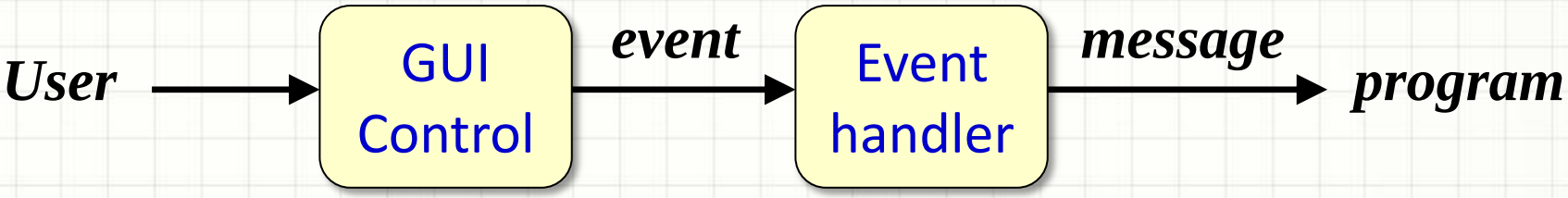
Tổng quan về ứng dụng WinForms

ToolBox trong Visual Studio

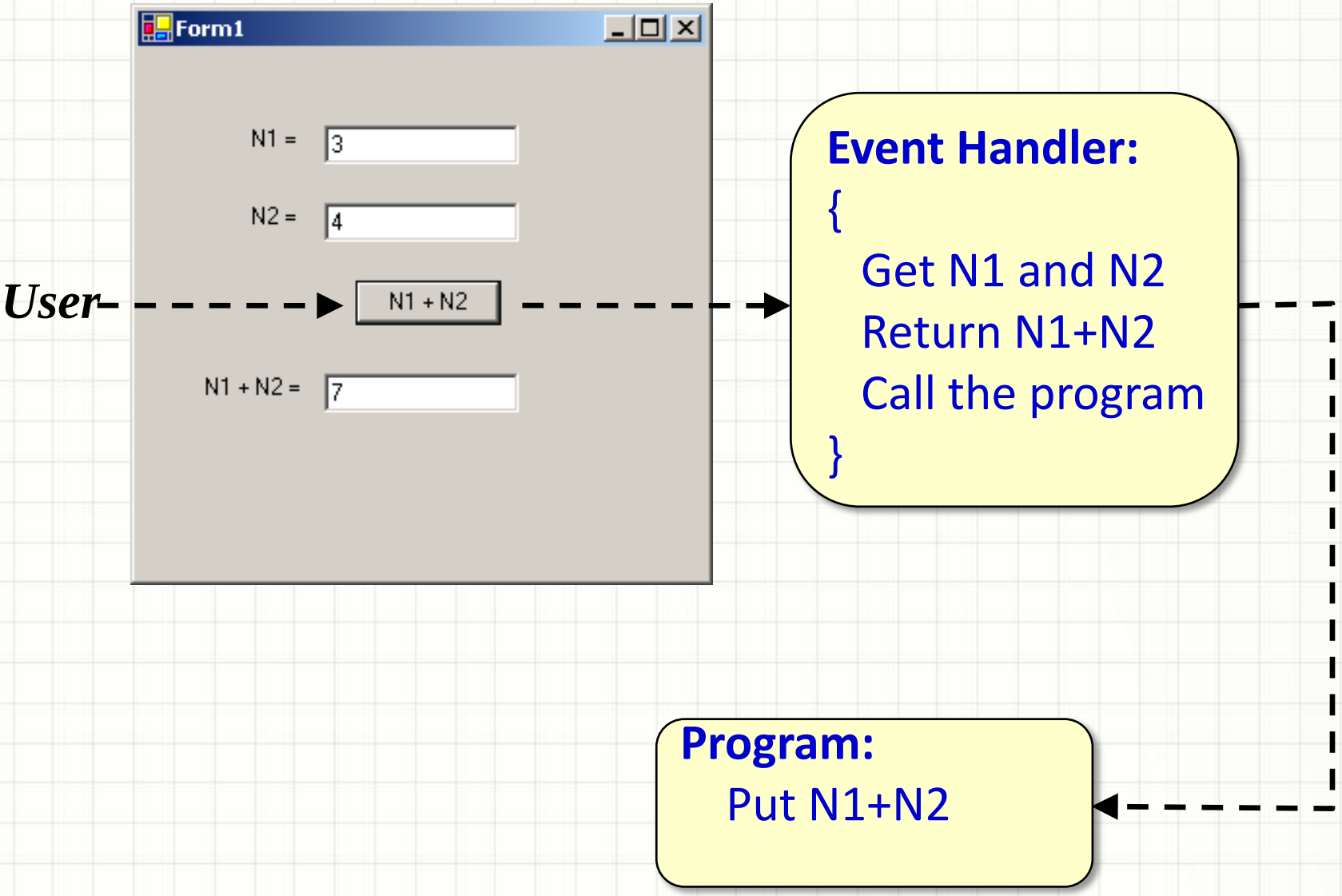


Xử lý sự kiện trong ứng dụng WinForms

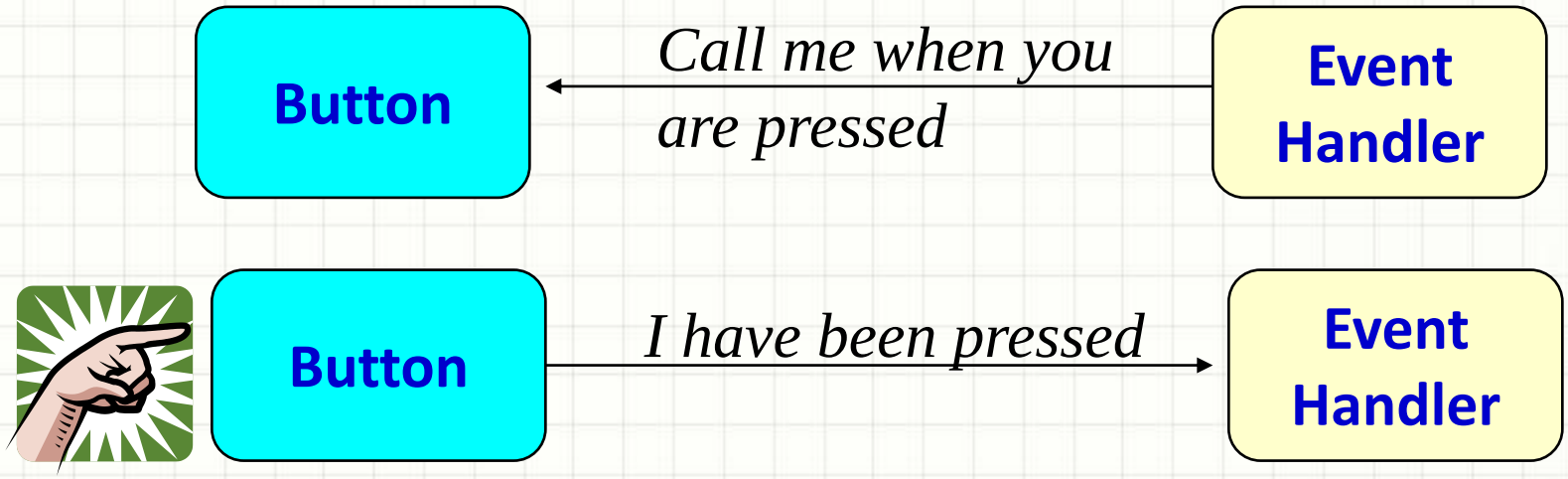
- Một *event* là một đối tượng biểu diễn một hành động
- Ví dụ:
  - The mouse is moved or button clicked
  - The mouse is dragged
  - A graphical button is clicked
  - A keyboard key is pressed
  - A timer expires
- Sự kiện thường tương ứng với thao tác của người dùng
- Có thể viết các hàm xử lý đối với sự kiện



Xử lý sự kiện trong ứng dụng WinForms



Xử lý sự kiện trong ứng dụng WinForms



Xử lý sự kiện trong ứng dụng WinForms

Event được xây dựng trên **delegate**, thực chất là một con trỏ hàm