NHẬP MÔN
CÔNG NGHỆ PHẦN MỀM
(INTRODUCTION TO SOFTWARE ENGINEERING)

Chương 8: Xây dựng phần mềm

- 1. Khái niệm

1. Khái niệm

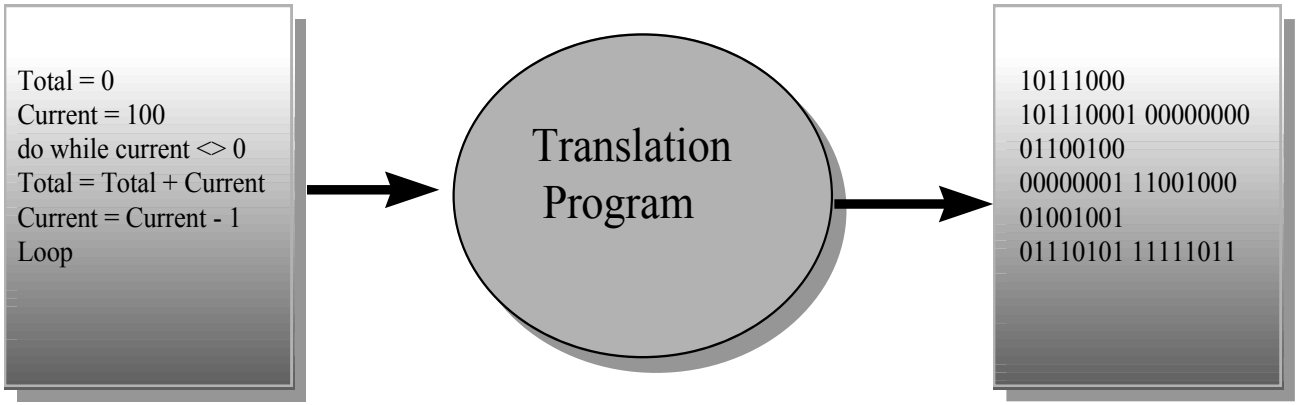
- Mã hóa là quá trình chuyển đổi thiết kế của một hệ thống sang một ngôn ngữ máy.
- Giai đoạn viết mã này liên quan đến việc chuyển đặc tả thiết kế thành mã nguồn.
- Việc biên soạn tài liệu đi kèm với mã nguồn là cần thiết để có thể dễ dàng xác minh sự phù hợp giữa mã với bản đặc tả của nó.
- Công việc mã hóa được thực hiện bởi lập trình viên là người độc lập với người thiết kế. Mục tiêu không phải là giảm nỗ lực và chi phí của giai đoạn mã hóa, mà là để cắt giảm chi phí của các giai đoạn sau.
- Chi phí kiểm thử và bảo trì có thể được giảm đáng kể với việc mã hóa hiệu quả.

Mục tiêu của lập trình

- 1. Để chuyển thiết kế của hệ thống sang ngôn ngữ máy, thực hiện các tác vụ theo chỉ định của thiết kế.
- 2. Để giảm chi phí của các giai đoạn sau: Chi phí kiểm tra và bảo trì có thể giảm đáng kể với việc mã hóa hiệu quả.
- 3. Làm cho chương trình dễ đọc hơn: Chương trình phải dễ đọc và dễ hiểu. Việc mã hóa cần đảm bảo mục tiêu làm tăng khả năng hiểu mã và đọc mã trong quá trình tạo ra phần mềm dễ bảo trì.

Để tiến hành việc cài đặt thiết kế, cần phải sử dụng ngôn ngữ lập trình bậc cao.

Translating from High-level Language to Binary



2. Lịch sử ngôn ngữ lập trình

- Các ngôn ngữ thế hệ thứ nhất:
  - Ngôn ngữ lập trình mã máy (machine code)
  - Ngôn ngữ lập trình assembly
- Các ngôn ngữ thế thế thứ hai:
  - FOTRAN, COBOL, ALGOL, BASIC
  - Phát triển 1950-1970
- Các ngôn ngữ thế hệ thứ ba
  - Ngôn ngữ lập trình cấp cao vạn năng (cấu trúc)
  - Lập trình hướng đối tượng
  - Lập trình hướng suy diễn – logic
- Các ngôn ngữ thế hệ thứ tư

Các loại ngôn ngữ lập trình

Procedural: Các chương trình nguyên khối chạy từ đầu đến cuối và không có sự can thiệp của người dùng ngoài đầu vào

- Basic, QBasic, QuickBasic
- COBOL
- FORTRAN
- C

Object Oriented/Event Driven (OOED): Các chương trình sử dụng các objects để đáp ứng các sự kiện (events); sử dụng các đoạn mã cho mỗi object

- JAVA
- Visual Basic
  - **Visual Basic for Applications (VBA)**
- Visual C++

Các đặc điểm của ngôn ngữ lập trình

3. Các công cụ lập trình

- Môi trường: DOS, WINDOWS, UNIX/LINUX
- Editors, Compilers, Linkers, Debuggers
- TURBO C, PASCAL
- MS C, Visual Basic, Visual C++, ASP
- UNIX/LINUX: C/C++, gcc (Gnu C Compiler)
- JAVA, CGI, perl
- C#, .NET

Các công cụ lập trình

- Công cụ lập trình C/C++:
  - Turbo C
  - Visual Studio
  - Eclipse
- Công cụ lập trình Java
  - Eclipse
  - Netbean
- Công cụ lập trình C#, .NET
  - Visual Studio.NET
  - MSDN Library
- Công cụ lập trình web
  - Visual Studio và SQL Server

4. Quy trình lập trình

- **Xác định bài toán**
  - **Đầu vào**
  - **Đầu ra**
- **Các bước xử lý để tạo kết quả**

Input	**Processing**	**Output**
Num-1	Read 3 numbers	Total
Num-2	Add numbers together	
Num-3	Print Total number	

Các bước trong lập trình

- Lập trình
- Kiểm tra thuật toán đã được cài đặt
- Thực hiện chương trình trên máy tính
  - Compile
  - Correct syntax errors
  - Run program with test data
  - Correct logic errors
- Viết tài liệu

Phương pháp lập trình có cấu trúc

- Outline Solution
  - Nhiệm vụ chính
  - Các bước xử lý chính
  - Các cấu trúc điều khiển chính (vòng lặp, rẽ nhánh, v.v.)
  - Các biến chính

Phương pháp lập trình có cấu trúc

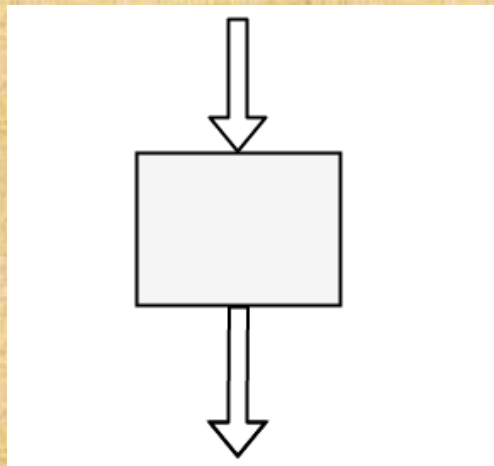
- Chia toàn bộ chương trình thành các mô-đun nhỏ để chương trình trở nên dễ hiểu. Mục đích của lập trình cấu trúc là tuyến tính hóa luồng điều khiển thông qua một chương trình máy tính sao cho trình tự thực thi tuân theo trình tự mà mã được viết.
- Cấu trúc động của chương trình giống với cấu trúc tĩnh của chương trình. Điều này nâng cao khả năng đọc, khả năng kiểm tra và khả năng sửa đổi của chương trình. Luồng kiểm soát tuyến tính này có thể được quản lý bằng cách hạn chế tập hợp các cấu trúc ứng dụng được phép thành một mục nhập duy nhất, các định dạng lối ra duy nhất.
- Chúng tôi sử dụng lập trình có cấu trúc vì nó cho phép người lập trình hiểu chương trình một cách dễ dàng. Nếu một chương trình bao gồm hàng nghìn lệnh và một lỗi xảy ra thì rất phức tạp để tìm ra lỗi đó trong toàn bộ chương trình, nhưng trong lập trình có cấu trúc, chúng ta có thể dễ dàng phát hiện lỗi và sau đó đến vị trí đó và sửa nó. Điều này giúp tiết kiệm rất nhiều thời gian.

Lập trình có cấu trúc

- Cho phép người lập trình hiểu chương trình một cách dễ dàng.
- Trong lập trình có cấu trúc, chúng ta có thể dễ dàng phát hiện lỗi và tìm đến vị trí để sửa nó.
- Điều này giúp tiết kiệm rất nhiều thời gian.

Rule 1: Code Block

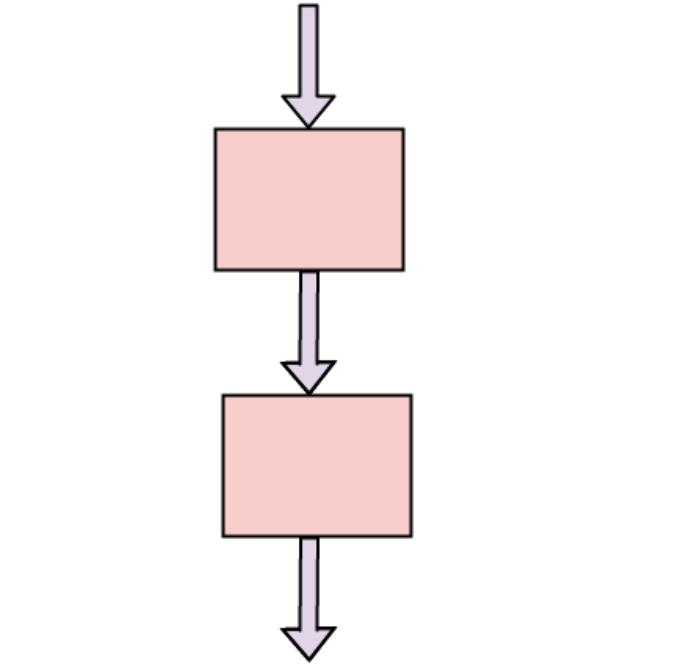
- Nếu điều kiện đầu vào đúng, nhưng điều kiện thoát sai thì lỗi đó phải nằm trong khối. Điều này không đúng nếu việc thực thi được phép nhảy vào một khối (do đó lỗi có thể ở bất kỳ đâu trong chương trình, gỡ lỗi trong những trường hợp này khó hơn nhiều).
- **Rule 1 of Structured Programming:** Một khối mã được cấu trúc, như thể hiện trong hình. Trong điều kiện biểu đồ luồng, một hộp có một điểm vào và một điểm thoát được cấu trúc. Lập trình có cấu trúc là một phương pháp làm rõ ràng rằng chương trình là đúng.



Rule1: Code block is structured

Rule 2: Sequence

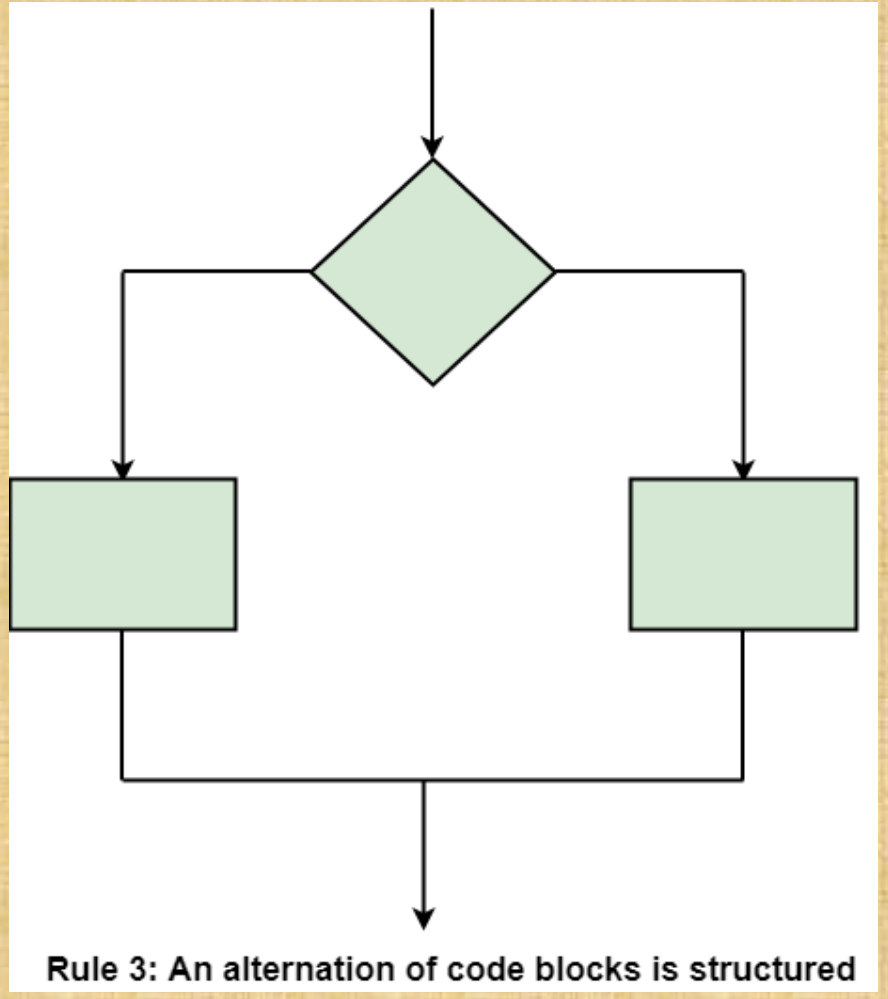
- Một chuỗi khối là đúng nếu điều kiện thoát của mỗi khối khớp với điều kiện vào của khối sau.
- Toàn bộ chuỗi có thể được coi là một khối duy nhất, có điểm vào và điểm ra.
- **Rule 2 of Structured Programming:** Hai hoặc nhiều khối được kết nối liền nhau



Rule2: A sequence of code blocks is structured

Rule Three: Alternation

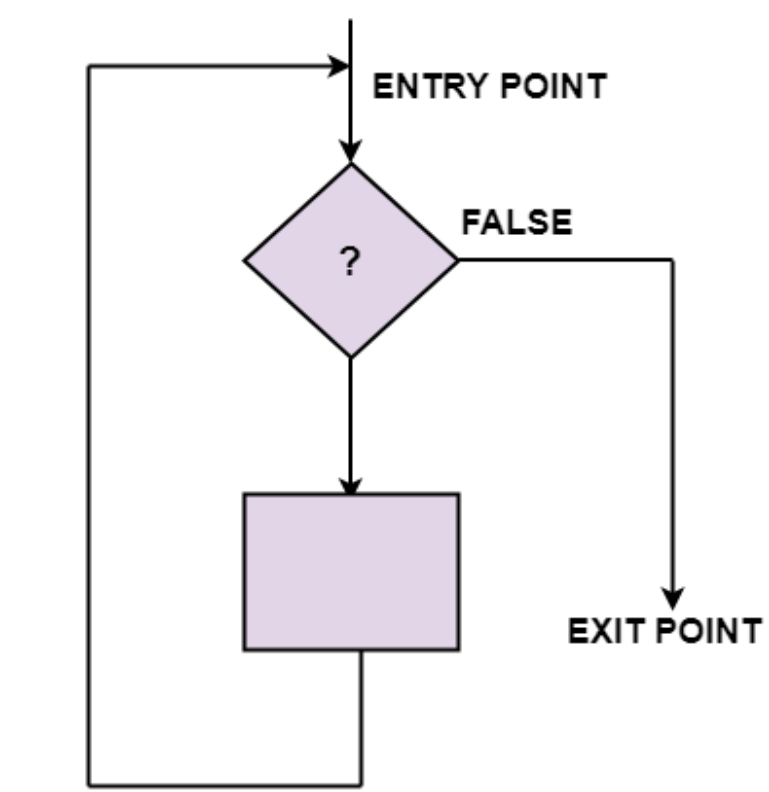
- **Rule 3 of Structured Programming**: Tùy chọn If-then-else, mỗi lựa chọn là một khối mã. Cấu trúc rẽ nhánh được sử dụng để đáp ứng điều kiện thoát.



Rule 3: An alternation of code blocks is structured

Rule 4: Iteration

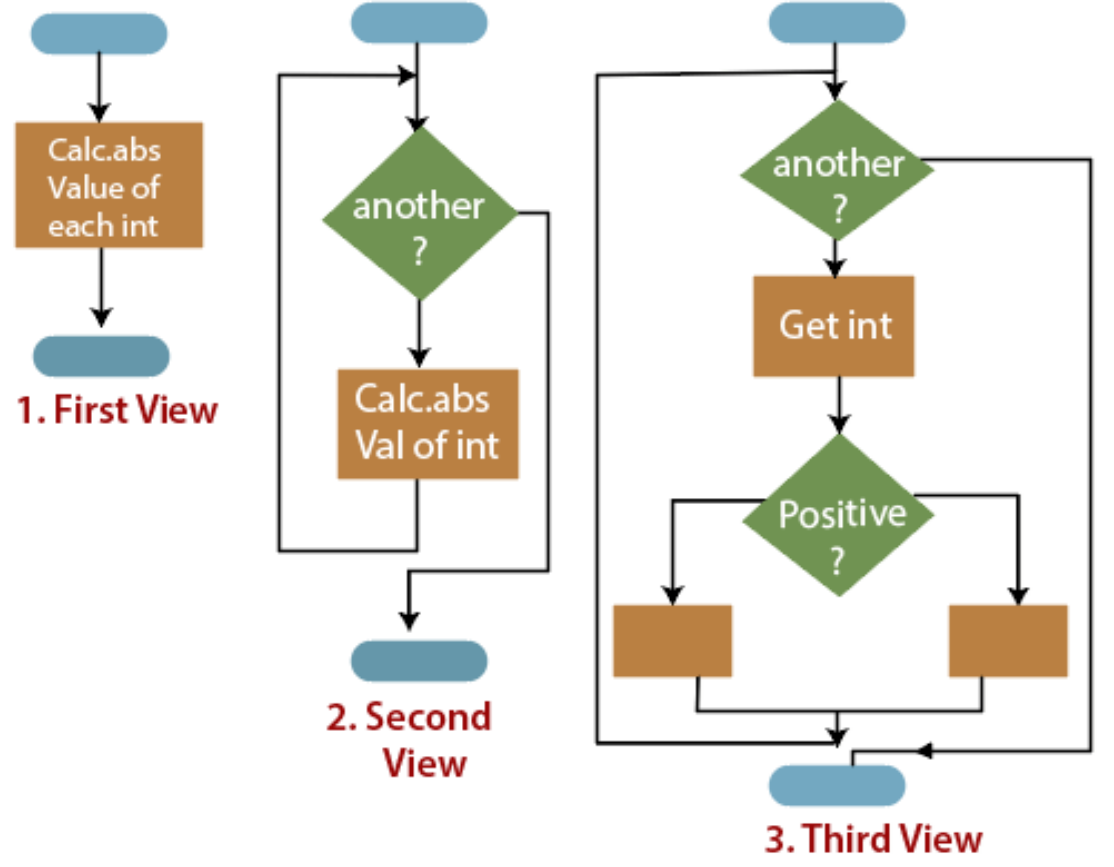
- **Rule 4 of Structured Programming**: Vòng lặp (trong khi) gồm một điểm vào và một điểm ra. Điểm vào có điều kiện phải được thỏa mãn và điểm ra có các yêu cầu sẽ được đáp ứng. Không có bước nhảy vào biểu mẫu từ các điểm bên ngoài của mã.



Rule 4: Iteration of code blocks is structured

Rule 5: Nested Structures

- **Rule 5 of Structured Programming**: Một cấu trúc có một điểm vào và ra duy nhất



Object-Oriented Event-driven Programming (OOED)

OOED sử dụng các đối tượng **objects**, hoặc các modules độc lập kết hợp dữ liệu và mã chương trình để chuyển các message cho nhau.

OOED dễ làm việc hơn vì nó trực quan hơn các phương pháp lập trình truyền thống.

Ví dụ: Visual Basic

Người dùng có thể kết hợp các đối tượng một cách tương đối dễ dàng để tạo ra các hệ thống mới hoặc mở rộng các hệ thống hiện có.

Thuộc tính của đối tượng là thuộc tính liên kết với một đối tượng.

Phương thức của đối tượng là những hoạt động mà đối tượng có thể thực hiện.

Đối tượng phản hồi các sự kiện.

OOED Programming Process

Quy trình sáu bước để viết một chương trình máy tính OOED:

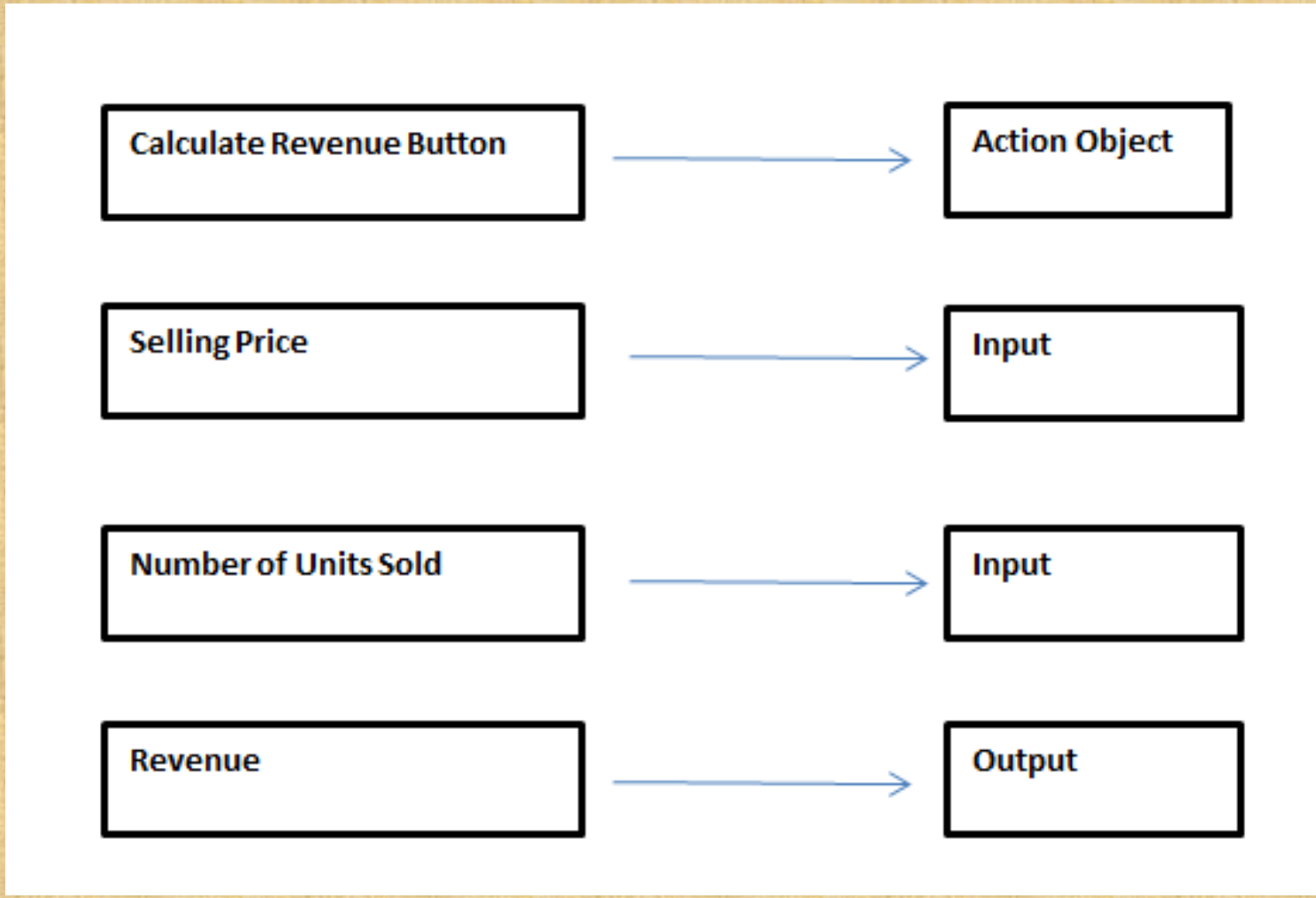
1. Xác định vấn đề.
2. Tạo giao diện
3. Phát triển logic cho các đối tượng hành động
4. Viết và kiểm tra mã cho các đối tượng hành động
5. Kiểm tra dự án tổng thể
6. Làm tài liệu

Bất kể một chương trình được viết tốt như thế nào, mục tiêu sẽ không đạt được nếu chương trình không giải quyết được vấn đề (sai).

Step One: Define Problem

- Trước khi tạo ứng dụng máy tính để giải quyết một vấn đề, trước tiên cần phải xác định rõ nó.
- Cần xác định đầu vào và đầu ra.
- Phải xác định dữ liệu được đưa vào chương trình và kết quả được xuất ra từ nó.
- Phác thảo giao diện (giao tiếp) ứng dụng.
- Xác định các hành động của các đối tượng cần mã hóa.

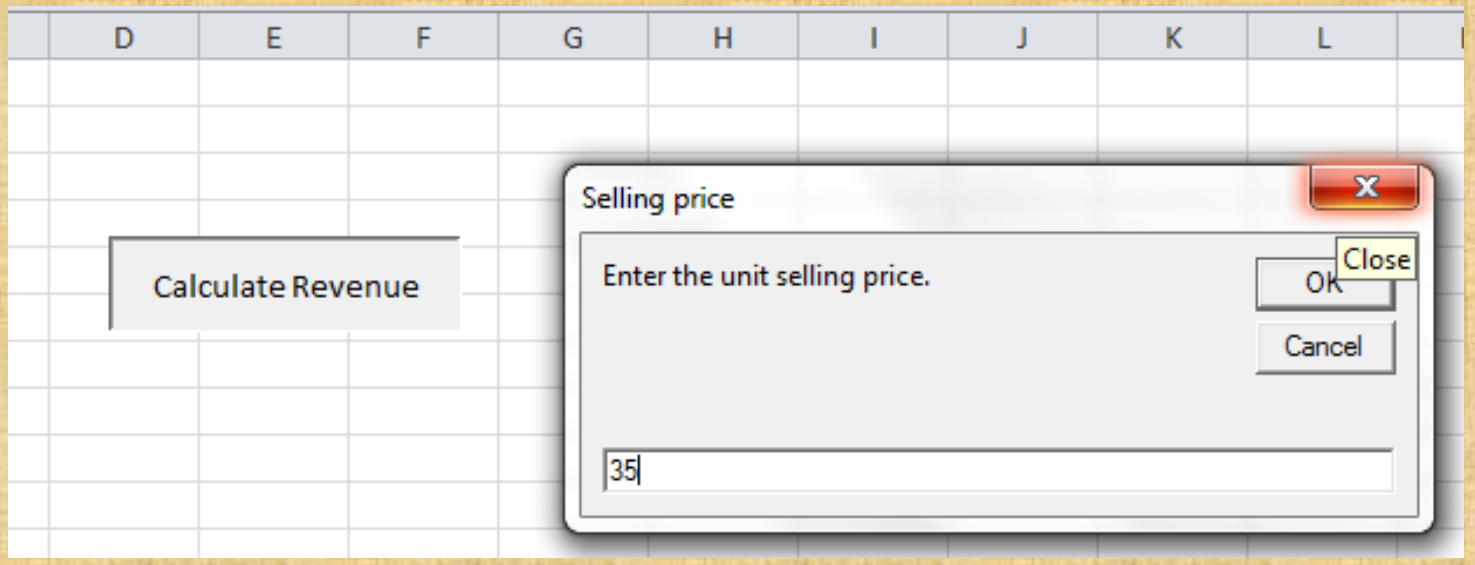
Ví dụ. Giao diện tính toán doanh thu

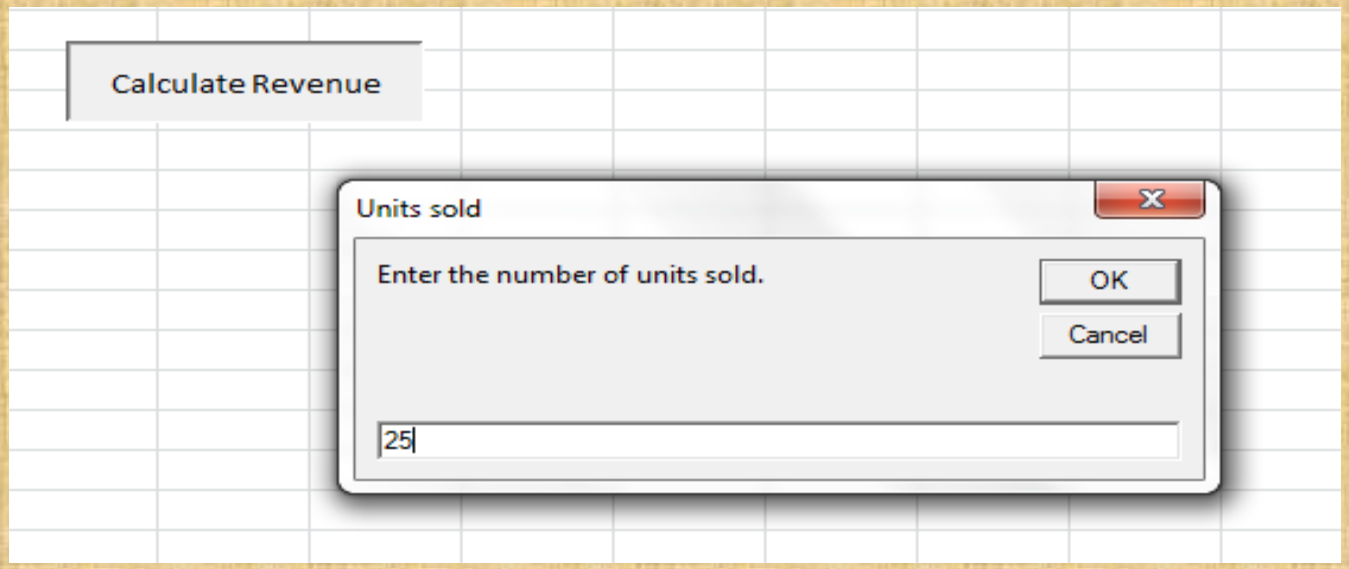


Step Two: Create Interface

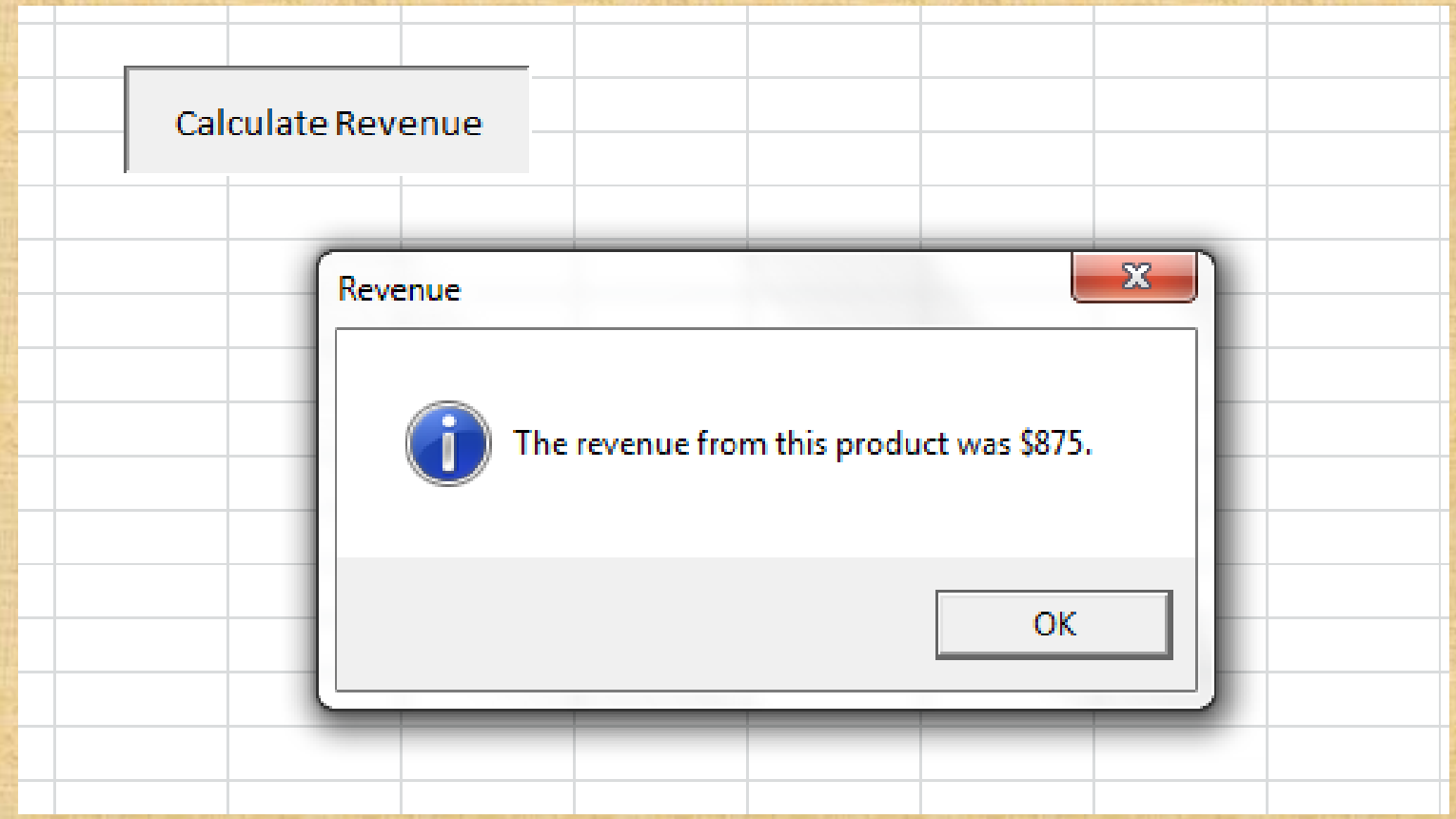
- Ví dụ: tạo giao diện tính toán doanh thu.
  - button for action
  - inputBox for input (Selling Price)
  - inputBox for input (Units Sold)
  - MessageBox for output

Calculate Revenue Interface - Input





Calculate Revenue Interface - Output



Step Three: Develop Logic for Action Objects

- Xác định mỗi đối tượng hành động làm gì để phản ứng với các sự kiện. Đây là logic cho từng đối tượng.
- Sử dụng Bảng Nhập / Xử lý / Đầu ra (IPO) và Mã giả để phát triển logic
- Bảng IPO hiển thị các đầu vào, đầu ra và quá trình xử lý để chuyển đầu vào thành đầu ra

IPO Table for Calculate Revnue Button

Input	Processing	Output
Unit Price Quantity Sold	revenue = unitPrice X quantitySold	Revenue

Pseudocode for Count High Values Button

```
Begin procedure
    Input Selling Price
   Input Quantity Sold
    Revenue = Selling Price x Quantity Sold
    Output Revenue

End procedure
```

Step Four: Write and Test Code of Action Objects

- Chuyển đổi logic được thể hiện trong mã giả sang ngôn ngữ lập trình (sử dụng bộ từ vựng và cú pháp của một ngôn ngữ).
- Kiểm tra và sửa mã đối tượng được viết, giai đoạn này được gọi là gỡ lỗi.

VBA Code for Calculate Revenue Button

```
Sub CalculateRevenue()
    Dim unitPrice As Currency
    Dim quantitySold As Integer
    Dim revenue As Currency

    ' Get the user's inputs, then calculate revenue.
    unitPrice = InputBox("Enter the unit selling price.", "Selling price")
    quantitySold = InputBox("Enter the number of units sold.", "Units sold")
    revenue = unitPrice * quantitySold

    ' Report the results.
    MsgBox "The revenue from this product was " & Format(revenue, "$#,##0") _
        & ".", vbInformation, "Revenue"
End Sub
```

Step Five: Test Overall Project

- Kiểm tra toàn bộ project cần đảm bảo mỗi câu lệnh phải hoàn toàn chính xác nếu không thì chương trình có thể sẽ có lỗi.
- Cần đảm bảo chương trình được kiểm tra trên môi trường và dữ liệu thực thi thực tế mà chương trình sẽ được sử dụng.

Step Six: Document Project in Writing

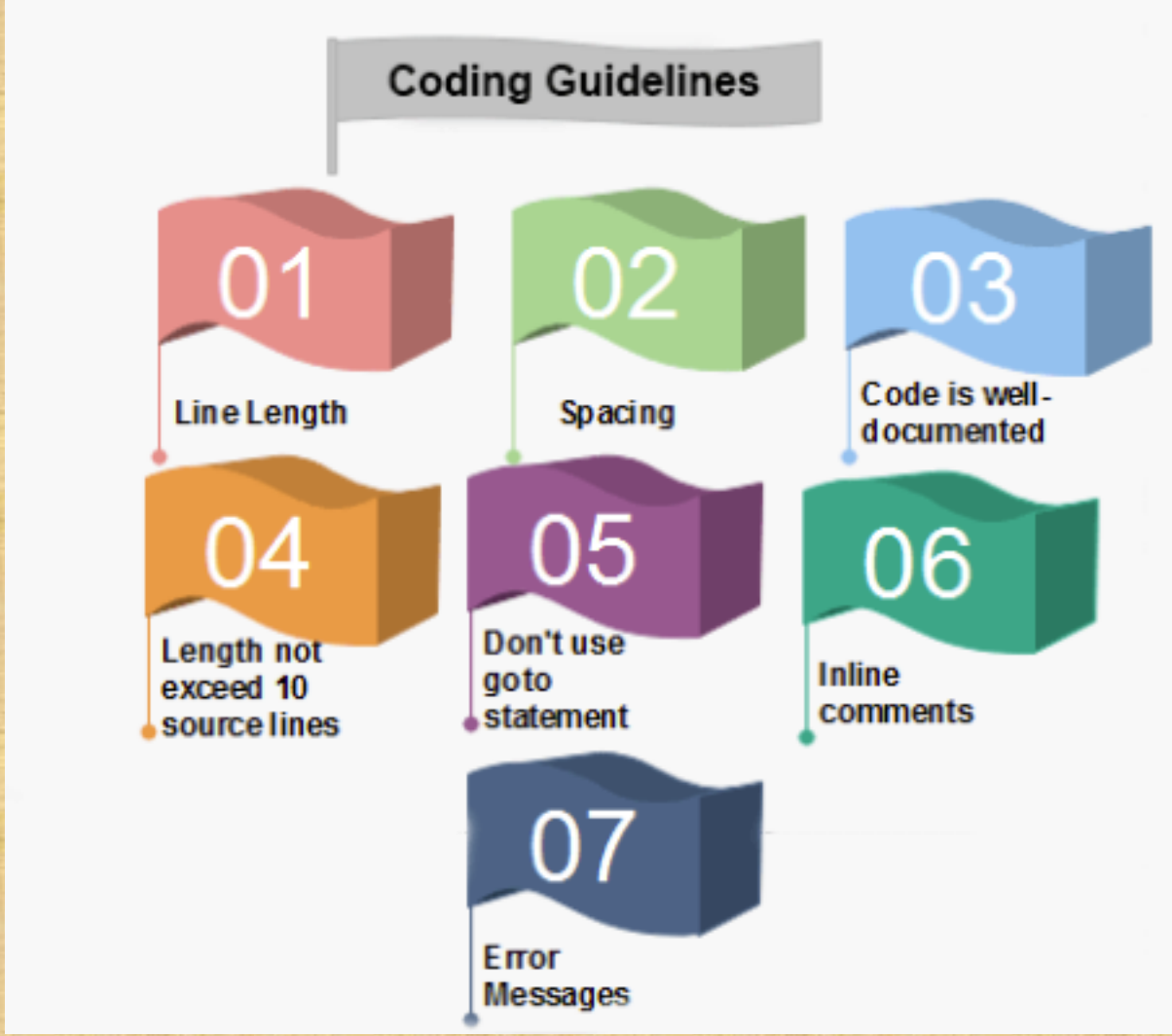
- Tài liệu bao gồm các mô tả bằng hình ảnh và văn bản của phần mềm trong đó chứa các mô tả phần lập trình và các mô tả hoặc hướng dẫn bên ngoài.
- Tài liệu là cần thiết vì dù sớm hay muộn, chương trình sẽ cần được bảo trì (sửa lỗi, thêm tính năng mới, xử lý các vấn đề chưa từng nghĩ tới, v.v.) Điều này KHÔNG thể thực hiện được nếu không có tài liệu.
- Tài liệu có thể bao gồm sách, hướng dẫn sử dụng và các mục tiêu của phần mềm.

5. Quy ước viết mã

Hướng dẫn viết mã

- Nguyên tắc viết mã cung cấp cho lập trình viên một tập hợp các phương pháp
- làm cho các chương trình để đọc và bảo trì.

Hướng dẫn viết mã



Hướng dẫn viết mã

1. Line Length: It is considered a good practice to keep the length of source code lines at or below 80 characters. Lines longer than this may not be visible properly on some terminals and tools. Some printers will truncate lines longer than 80 columns.

2. Spacing: The appropriate use of spaces within a line of code can improve readability. Example:

```
Bad: cost=price+(price*sales_tax)
fprintf(stdout ,"The total cost is %5.2f\n",cost);
Better: cost = price + ( price * sales_tax )
fprintf (stdout,"The total cost is %5.2f\n",cost);
```

3. The code should be well-documented: As a rule of thumb, there must be at least one comment line on the average for every three-source line.

Hướng dẫn viết mã

4. The length of any function should not exceed 10 source lines: A very lengthy function is generally very difficult to understand as it possibly carries out many various functions. For the same reason, lengthy functions are possible to have a disproportionately larger number of bugs.

5. Do not use goto statements: Use of goto statements makes a program unstructured and very tough to understand.

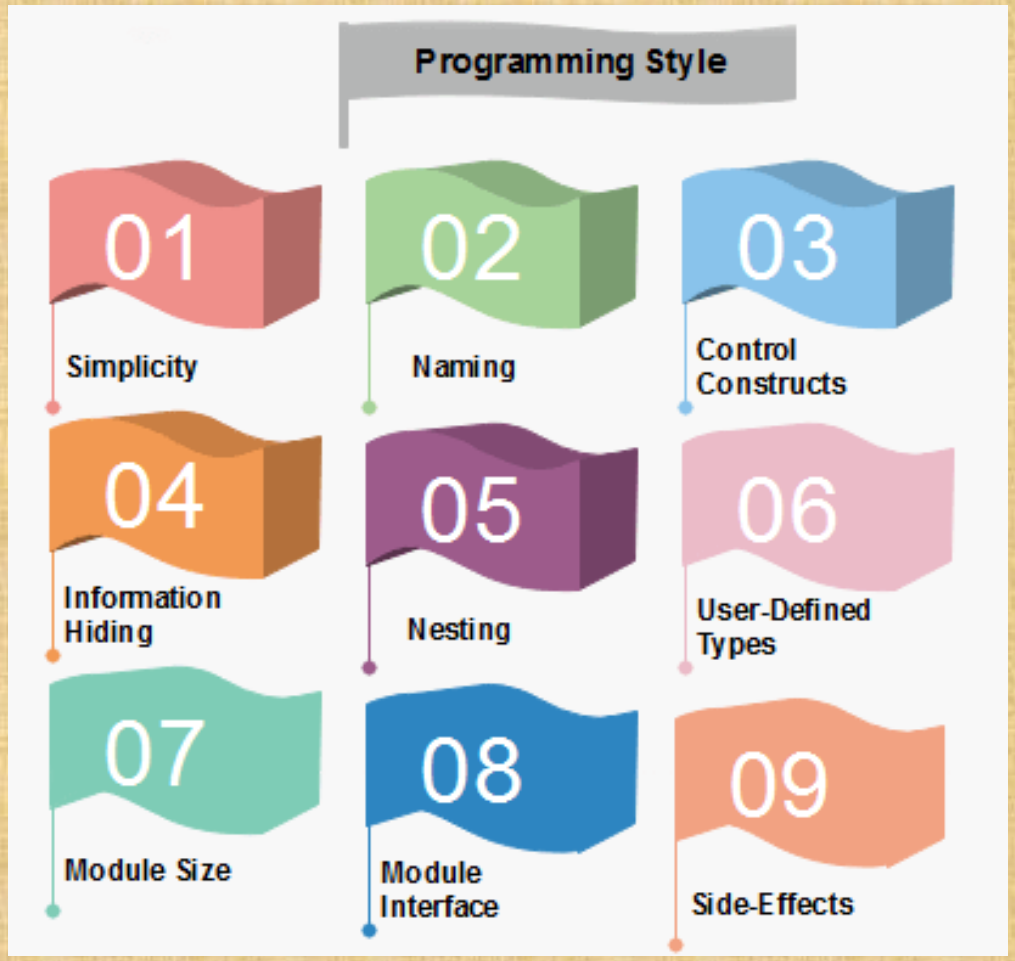
6. Inline Comments: Inline comments promote readability.

7. Error Messages: Error handling is an essential aspect of computer programming. This does not only include adding the necessary logic to test for and handle errors but also involves making error messages meaningful.

Phong cách lập trình

- Các kỹ thuật được sử dụng để viết mã nguồn cho một chương trình. Hầu hết các phong cách lập trình được thiết kế để giúp lập trình viên nhanh chóng đọc và hiểu chương trình cũng như tránh mắc lỗi.
- Một phong cách viết mã tốt có thể khắc phục nhiều khiếm khuyết của ngôn ngữ lập trình.
- Mục tiêu của phong cách lập trình tốt là cung cấp mã đơn giản và dễ hiểu.
- Phong cách lập trình được sử dụng trong một chương trình khác nhau có thể bắt nguồn từ các tiêu chuẩn mã hóa hoặc quy ước mã của một công ty hoặc tổ chức máy tính khác, cũng như sở thích của lập trình viên thực tế.

Phong cách lập trình



Phong cách lập trình

- **1. Clarity and simplicity of Expression:** The programs should be designed in such a manner so that the objectives of the program is clear.
- **2. Naming:** In a program, you are required to name the module, processes, and variable, and so on. Care should be taken that the naming style should not be cryptic and non-representative.
- **For Example:** a = 3.14 * r * r
  area of circle = 3.14 * radius * radius;
- **3. Control Constructs:** It is desirable that as much as a possible single entry and single exit constructs used.
- **4. Information hiding:** The information secure in the data structures should be hidden from the rest of the system where possible. Information hiding can decrease the coupling between modules and make the system more maintainable.

Phong cách lập trình

- **5. Nesting:** Deep nesting of loops and conditions greatly harm the static and dynamic behavior of a program. It also becomes difficult to understand the program logic, so it is desirable to avoid deep nesting.
- **6. User-defined types:** Make heavy use of user-defined data types like enum, class, structure, and union. These data types make your program code easy to write and easy to understand.
- **7. Module size:** The module size should be uniform. The size of the module should not be too big or too small. If the module size is too large, it is not generally functionally cohesive. If the module size is too small, it leads to unnecessary overheads.
- **8. Module Interface:** A module with a complex interface should be carefully examined.
- **9. Side-effects:** When a module is invoked, it sometimes has a side effect of modifying the program state. Such side-effect should be avoided where as possible.