Programming Technique

Kỹ Thuật lập trình
Programming technique

Vũ Đức Vượng
SE-FIT-HUT
vuongvd@yahoo.de

Programming Technique

Programming Languages – classifications and Program Development

Nội dung

Programming languages và
Phân loại NNLT

Develop Web pages : HTML, scripting languages, DHTML, XML, WML, và các phần mềm tạo trang web

procedural programming languages

Multimedia
authoring programs

visual programming languages

6 bước của chu trình phát triển phần mềm
Six steps in the program development cycle

object-oriented programming languages

Sự khác biệt giữa structured design và object-oriented design

nonprocedural languages and tools

Những cấu trúc cơ sở dùng thiết kế các giải pháp cho chương trình

Next

Chương trình máy tính và ngôn ngữ lập trình
Computer Programs and Programming Languages

Computer program?

- **Tập hợp các lệnh chỉ dẫn cho máy tính thực hiện nhiệm vụ**
  - **Programming language**—Dùng để viết các lệnh, chỉ thị

Next

programming language - NNLT

***Một NNLT** là 1 hệ thống các ký hiệu dùng để liên lạc , trao đổi 1 nhiệm vụ/ thuật toán với máy tính, làm cho nhiệm vụ được thực thi*. **Nhiệm vụ được thực thi gọi là một *computation*, nó tuân thủ một độ chính xác và những quy tắc nhất quán.**

Với mỗi ngôn ngữ lập trình, ta cần nắm bắt, thấu hiểu những gì ?: Có 3 thành phần căn bản của bất cứ 1 NNLT nào.

Mô hình ngôn ngữ-Language paradigm **là những nguyên tắc chung cơ bản, dùng bởi LTV để xd chương trình.**

Cú pháp - Syntax **của ngôn ngữ là cách để xác định cái gì là hợp lệ trong cấu trúc các câu của ngôn ngữ; Nắm được cú pháp là cách để đọc và tạo ra các câu trong các ngôn ngữ tự nhiên, như tiếng Việt, tiếng Anh. Tuy nhiên điều đó không có nghĩa là nó giúp chúng ta hiểu hết ý nghĩa của câu văn.**

Ngữ nghĩa – semantics **của 1 program trong ngôn ngữ ấy. Rõ ràng, nếu không có semantics, 1 NNLT sẽ chỉ là 1 mớ các câu văn vô nghĩa; như vậy semantics là 1 thành phần không thể thiếu của 1 ngôn ngữ.**

Có rất nhiều NNLT, khoảng 1000 ngôn ngữ (60's đã có hơn 700) – phần lớn là các ngôn ngữ hàn lâm, có mục đích riêng hay phát triển bởi 1 tổ chức để phục vụ cho bản thân họ.

Cont…

Về cơ bản, chỉ có 4 mô hình NNLT chính:

- ***Imperative (Procedural) Paradigm*** **(Fortran, Pascal, C, Ada, ...)**
- ***Object-Oriented Paradigm*** **(SmallTalk, Java, C++)**
- ***Logic Paradigm*** **(Prolog)**
- ***Functional Paradigm*** **(Lisp, ML, Haskell)**

Những tính chất cần có với các chương trình phần mềm là :

- **Tính mềm dẻo scalability / Khả năng chỉnh sửa modifiability**
- **Khả năng tích hợp integrability / Khả năng tái sử dụng reusability**
- **Tính chuyển đổi, linh hoạt, độc lập phần cứng -portability**
- **Hiệu năng cao -performance**
- **Độ tin cậy - reliability**
- **Dễ xây dựng**
- **Rõ ràng, dễ hiểu**
- **Ngắn gọn, xúc tích**

HOẠT ĐỘNG CỦA 1 CHƯƠNG TRÌNH

- **Computer program được nạp vào *primary memory* như là 1 tập các lệnh bằng ngôn ngữ máy, tức là một dãy tuần tự các số nhị phân - *binary digits*.**
- **Tại bất cứ một thời điểm nào, computer sẽ ở một trạng thái *state* nào đó.**
- **Đặc điểm cơ bản của trạng thái là con trỏ lệnh *instruction pointer* trỏ tới lệnh mã máy tiếp theo để thực hiện.**
- **Thứ tự thực hiện các nhóm lệnh mã máy được gọi là luồng điều khiển *flow of control*.**

MACHINE CODE

- Máy tính chỉ nhận các tín hiệu điện tử - có, không có -tương ứng với các dòng bits.
- 1 program ở dạng đó gọi là *machine code.*
- Ban đầu chúng ta phải dùng machine code để viết CT:
- Quá phức tạp, giải quyết các bài toán lớn là không tưởng

```
23fc 0000 0001 0000 0040
0cb9 0000 000a 0000 0040
6e0c
06b9 0000 0001 0000 0040
60e8
```

ASSEMBLY LANGUAGE

- **NN Assembly là bước đầu tiên của việc xây dựng cơ chế viết chương trình tiện lợi hơn – thông qua các ký hiệu, từ khóa và cả mã máy.**
- Tất nhiên, để chạy được các chương trình này thì phải dịch (assembled) thành machine code.
- Vẫn còn phức tạp, cải thiện không đáng kể

```
   movl    #0x1,n
compare:
   cmpl    #oxa,n
   cgt     end_of_loop
   acddl   #0x1,n
   bra     compare
end_of_loop:
```

HIGH LEVEL LANGUAGE

- **Thay vì dựa trên phần cứng (machine-oriented) cần tìm cơ chế dựa trên vấn đề (problem-oriented) để tạo chương trình.**
- **Chính vì thế *high(er) level* languages – là các ngôn ngữ lập trình gần với ngôn ngữ con người hơn – dùng các từ khóa giống tiếng anh – đã được xây dựng như : Algol, Fortran, Pascal, Basic, Ada, C, …**

PHÂN LOẠI THEO THỜI GIAN

- **1940s : Machine code**
- **1950s Khai thác sức mạnh của MT: Assembler code, Autocodes, first version of Fortran**
- **1960s Tăng khả năng tính toán: Cobol, Lisp, Algol 60, Basic, PL/1 --- nhưng vẫn dùng phong cách lập trình cơ bản của assembly language.**
- **1970s Bắt đầu cuộc khủng hoảng phần mềm "software crisis":**
  1. Giảm sự phụ thuộc vào máy – Tính chuyển đổi.
  2. Tăng sự đúng đắn của CT -Structured Programming, modular programming và information hiding.

  Ví dụ : Pascal, Algol 68 and C.

Continue …

- **1980s Giảm sự phức tạp – object orientation, functional programming.**
- **1990s Khai thác phần cứng song song và phân tán (parallel và distributed) làm cho chương trình chạy nhanh hơn, kết quả là hàng loạt ngôn ngữ mở rộng khả năng lập trình parallel cũng như các NNLT chuyên parallel như occam được xd.**
- **2000s Genetic programming languages, DNA computing, bio-computing?**
- **Trong tương lai : Ngôn ngữ lt lượng tử : Quantium ?**

SOFTWARE CRISIS

Khái niệm software crisis bao gồm hàng loạt vấn đề nảy sinh trong việc phát triển phần mềm trong hững năm 1960s khi muốn xd những hệ thống phần mềm lớn trên cơ sở các kỹ thuật phát triển thời đó.

Kết quả:

- **1.Thời gian và giá thành tăng vọt tới mức không thể chấp nhận nổi.**
- **2.Năng xuất của các LTV không đáp ứng yêu cầu.**
- **3.Chất lượng phần mềm bị giảm, thấp.**

Để giải quyết các vấn đề kể trên , chuyên ngành software engineering ra đời.

CÁC THẾ HỆ NNLT
LANGUAGE GENERATIONS

Generation	Classification
1st	Machine languages
2nd	Assembly languages
3rd	Procedural languages
4th	Application languages (4GLs)
5th	AI techniques, inference languages
6th	Neural networks (?), others….

Computer Programs and Programming Languages

Low-level languages và high-level languages?

Low-level language

Machine-dependent
Phụ thuộc phần cứng, chỉ chạy trên một loại máy tính

Machine và assembly languages là ngôn ngữ bậc thấp low-level

High-level language

Machine-independent
Thường không phụ thuộc phần cứng, có thể chạy trên nhiều loại máy tính khác nhau

Next

PHÂN LOẠI THEO MỨC ĐỘ TRỪU TƯỢNG

Level	Instructions	Memory handling
Low level languages	Dạng bits – giống các lệnh machine	Truy cập và cấp phát trực tiếp bộ nhớ
High level languages	Dùng các biểu thức và các dòng điều khiển xác định	Truy cập và cấp phát bộ nhớ qua các lệnh, toán tử -
Very high level languages	Hoàn toàn trừu tượng, độc lập phần cứng	Che dấu hoàn toàn việc truy cập và tự động cấp phát bộ nhớ

Next

DECLARATIVE và NON-DECLARATIVE PROGRAMMING

- **Các ngôn ngữ có thể chia thành 2 nhóm : "Cái gì cần lưu trữ" và "Lưu trữ như thế nào".**
- **Nhóm 1 gọi là** declarative **(tường thuật -chính là functional và logic languages).**
- **Nhóm 2 gọi là** non-declarative **hay** procedural **(tức là các ngôn ngữ mệnh lệnh).**

Next

Procedural Languages – Ngôn ngữ thủ tục

Procedural language?

Lập trình viên viết các chỉ thị hướng dẫn cho máy tính cai gì cần làm và làm như thế nào

Sử dụng hàng loạt các từ giống tiếng anh để viết các chỉ thị - instructions

Còn gọi là **third-generation language (3GL)**

Các ngôn ngữ thông dụng là BASIC, COBOL, PASCAL, C,C++ và JAVA

Click to view animation

Next

Procedural Languages

Trình dịch - Compiler

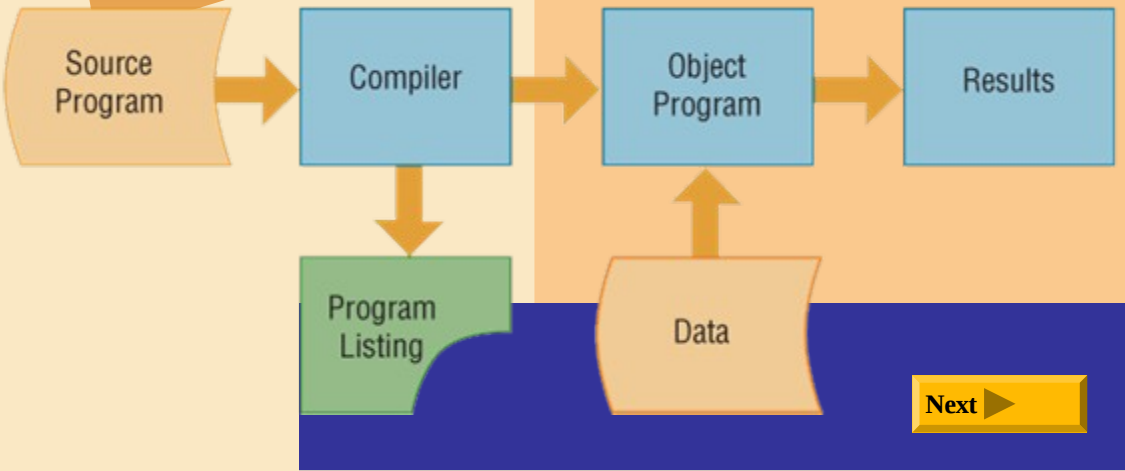
- Là chương trình thực hiện biên dịch toàn bộ chương trình nguồn thành mã máy trước khi thực hiện

```
*    COMPUTE REGULAR TIME PAY
     MULTIPLY REGULAR-TIME-HOURS BY HOURLY-PAY-RATE
          GIVING REGULAR-TIME-PAY.

*    COMPUTE OVERTIME PAY
     IF OVERTIME-HOURS > 0
          COMPUTE OVERTIME-PAY = OVERTIME-HOURS * 1.5 * HOURLY-PAY-RATE
     ELSE
          MOVE 0 TO OVERTIME-PAY.

*    COMPUTE GROSS PAY
     ADD REGULAR-TIME-PAY TO OVERTIME-PAY
          GIVING GROSS-PAY.

*    PRINT GROSS PAY
     MOVE GROSS-PAY TO GROSS-PAY-OUT.
     WRITE REPORT-LINE-OUT FROM DETAIL-LINE
          AFTER ADVANCING 2 LINES.
```

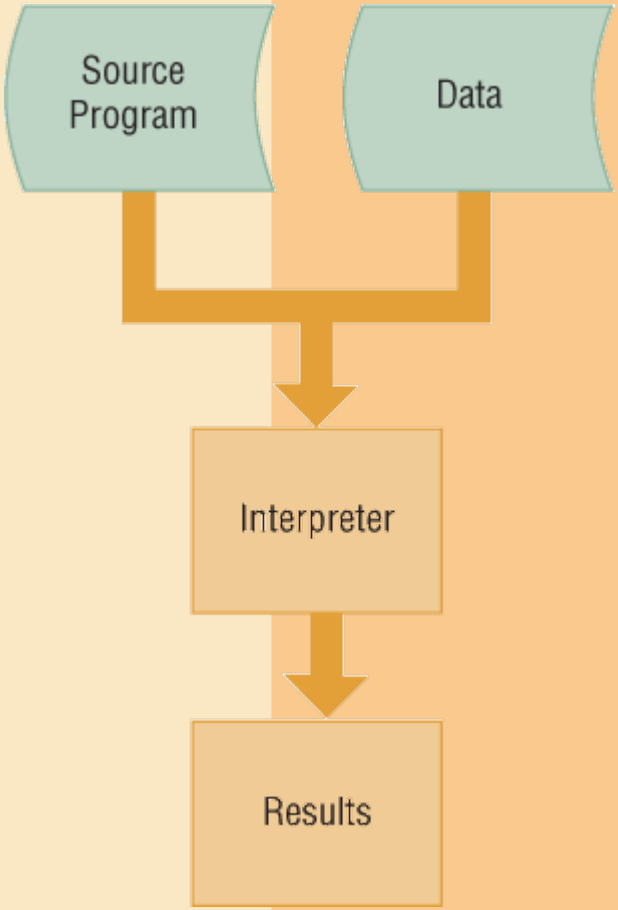


Next

Procedural Languages

Thông dịch - Interpreter?

- Là chương trình dịch và thực hiện từng dòng lệnh của chương trình cùng lúc
- Không tạo ra object program



p. 667 Fig. 13-5

Next

Procedural Languages

BASIC?

- Được thiết kế để cho những người mới học tiếp can một cách đơn giản NNLT
- Beginner's All-purpose Symbolic Instruction Code

```
REM COMPUTE REGULAR TIME PAY
Regular.Time.Pay = Regular.Time.Hours * Hourly.Pay.Rate

REM COMPUTE OVERTIME PAY
If Overtime.Hours > 0 THEN
    Overtime.Pay = Overtime.Hours * 1.5 * Hourly.Pay.Rate
ELSE
    Overtime.Pay = 0
END IF

REM COMPUTE GROSS PAY
Gross.Pay = Regular.Time.Pay + Overtime.Pay
REM PRINT GROSS PAY
PRINT USING "The gross pay is $##,###.##"; Gross.Pay
```


Next

Procedural Languages

COBOL?

- Dùng cho các ứng dụng trong kinh tế
- Các lệnh giống tiếng anh làm cho code dễ đọc, viết và chỉnh sửa
- COmmon Business-Oriented Language

```
*    COMPUTE REGULAR TIME PAY
     MULTIPLY REGULAR-TIME-HOURS BY HOURLY-PAY-RATE
          GIVING REGULAR-TIME-PAY.

*    COMPUTE OVERTIME PAY
     IF OVERTIME-HOURS > 0
          COMPUTE OVERTIME-PAY = OVERTIME-HOURS * 1.5 * HOURLY-PAY-RATE
     ELSE
          MOVE 0 TO OVERTIME-PAY.

*    COMPUTE GROSS PAY
     ADD REGULAR-TIME-PAY TO OVERTIME-PAY
          GIVING GROSS-PAY.

*    PRINT GROSS PAY
     MOVE GROSS-PAY TO GROSS-PAY-OUT.
     WRITE REPORT-LINE-OUT FROM DETAIL-LINE
          AFTER ADVANCING 2 LINES.
```

Next

Procedural Languages

C?

- **Là NNLT rất mạnh, ban đầu được thiết kế để lập trình hệ thống -write system software**
- **Yêu cầu những kỹ năng lập trình chuyên nghiệp**

```
/* Compute Regular Time Pay                                    */
rt_pay = rt_hrs * pay_rate;

/* Compute Overtime Pay                                        */
if (ot_hrs > 0)
     ot_pay = ot_hrs * 1.5 * pay_rate;
else
     ot_pay = 0;

/* Compute Gross Pay                                           */
gross = rt_pay + ot_pay;

/* Print Gross Pay                                             */
printf("The gross pay is %d\n", gross);
```

Next

Object-Oriented Programming Languages

Object-oriented programming (OOP) language?

Dùng để hỗ trợ thiết kế HĐT object-oriented design

Object là phần tử chứa đựng cả dữ liệu và các thủ tục xử lý dữliệu

Lợi ích cơ bản là khả năng tái sử dụng -reuse existing objects

Event-driven— Hướng sự kiện Kiểm tra để trả lời một tập cá[illegible]

Event là hành động mà chương trình cần đáp ứng

C++ và Java là các NN hoàn toàn HĐT

object-oriented languages

Next

Object-Oriented Programming Languages

C++?

- Chứa đựng các thành phần của C, loại bỏ những nhược điểm và thêm vào những tính năng mới để làm việc với object-oriented concepts
- Được dùng để phát triển các Database và các ứng dụng Web

```
// portion of a C++ program that allows users to create a new zip code from a
// string or a number and expand zip codes, as appropriate, to a 10-digit number

ZipC::ZipC( const unsigned long zipnum )
{
  ostringstream strInt;
  strInt << zipnum;
  code = strInt.str();
}

const string ZipC::getCode()
{
  return code;
}

void ZipC::setCode(const string newCode)
{
  code = newCode;
}

void ZipC::expand( const string suffix )
{

  if(code.length() == 5 &&       // small size?
     suffix.length() == 4)       // length ok?
  {
     code += "-";
     code.append(suffix);
  }
}
```

Next

Object-Oriented Programming Languages

Java?

- Phát triển bởi Sun Microsystems
- Giống C++ nhưng dùng trình dịch just-in-time (JIT) để chuyển source code thành machine code

```
public void actionPerformed(ActionEvent e)
   {
      foundKey = false;

      //Search for the key pressed
      for (int i = 0; i < keysArray.length && !foundKey; i++)
         if(e.getSource() == keysArray[i])    //key match found
         {
            foundKey = true;
            switch(i)
            {
               case 0: case 1: case 2: case 3: case 4:  //number buttons
               case 5: case 6: case 7: case 8: case 9:  //0 - 9
               case 15:                                 //decimal point button
                  if(clearText)
                  {
                     lcdField.setText("");
                     clearText = false;
                  }
                  lcdField.setText(lcdField.getText() + keysArray[i].getLabel());
                  break;
```

Next

Object-Oriented Programming Languages

Visual programming language?

Visual programming environment (VPE) Cho phép developers kéo và thả các objects để xd programs

Cung cấp giao diện trực quan hoặc đồ họa để tạo source code

Đôi khi được gọi là fifth-generation language

LTV viết và phát triển chương trình trong các segments

Thường được dùng trong môi trường RAD (rapid application development)

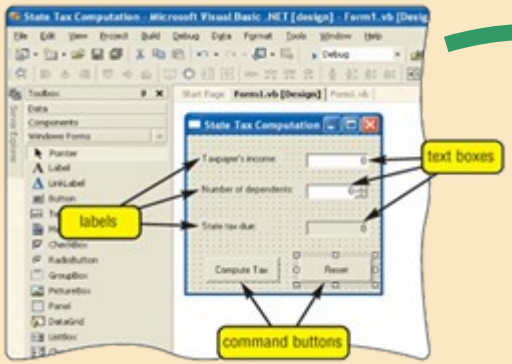
Next

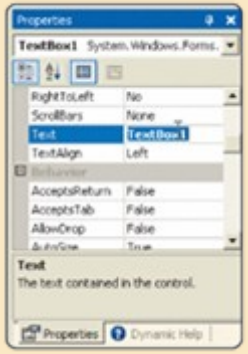
Object-Oriented Programming Languages

Visual Studio .NET 2003, 2005?

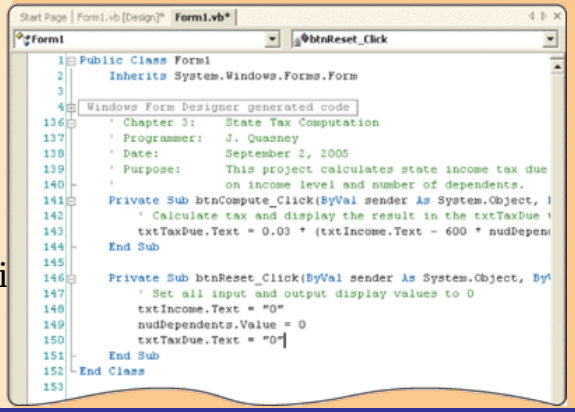
- **Bước phát triển của visual programming languages và RAD tools**
- **.NET là tập hợp các công nghệ cho phép program chạy trên Internet**
- **Visual Basic .NET 2003-5 dùng để xd các ct hướng đối tượng phức tạp**

Step 1. LTV thiết kế giao diện người dùng - user interface.





Step 2. LTV gán các thuộc tính cho mỗi object trên form.



Step 3. LTV viết code để xác định các action cần thực hiện đối với các sự kiện cần thiết.

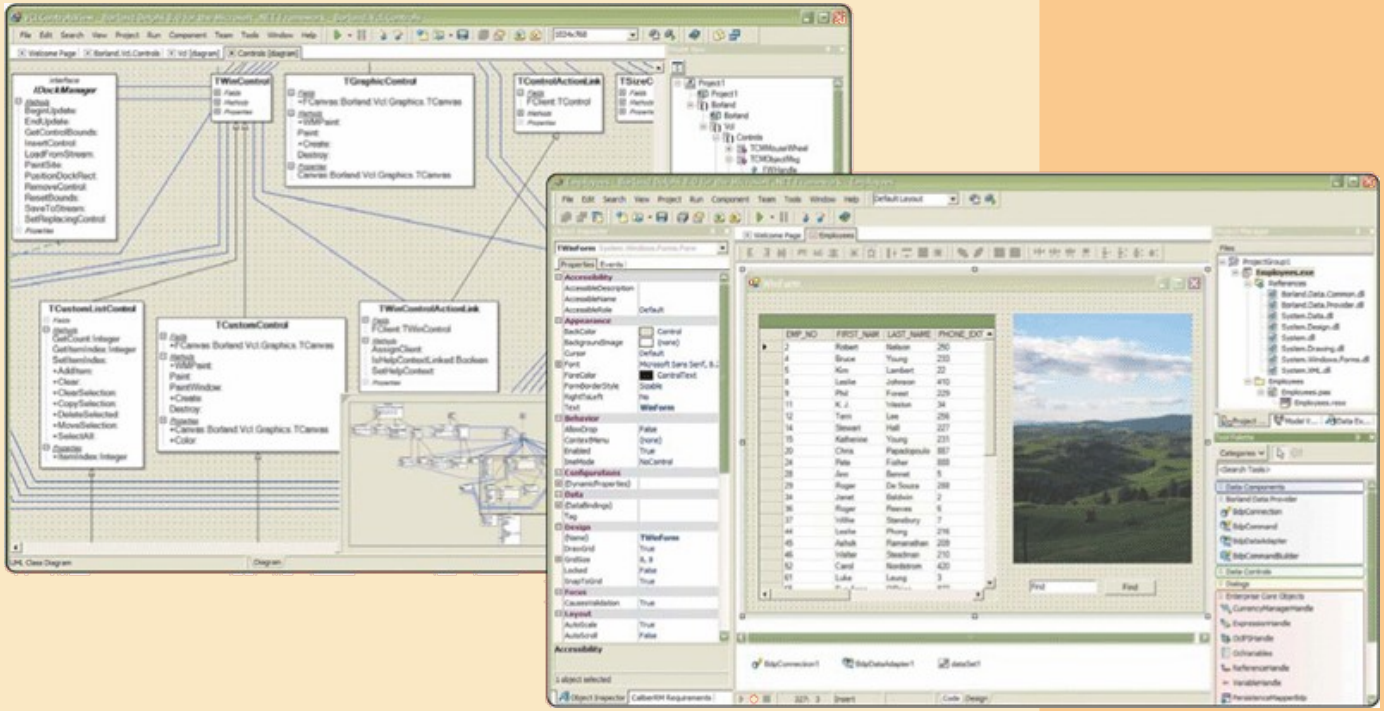
Step 4. LTV kiểm tra application.

Next

Object-Oriented Programming Languages

Delphi?

- Là 1 công cụ lập trình trực qua mạnh
- Hợp với những ứng dụng chuyên nghiệp và Web lớn

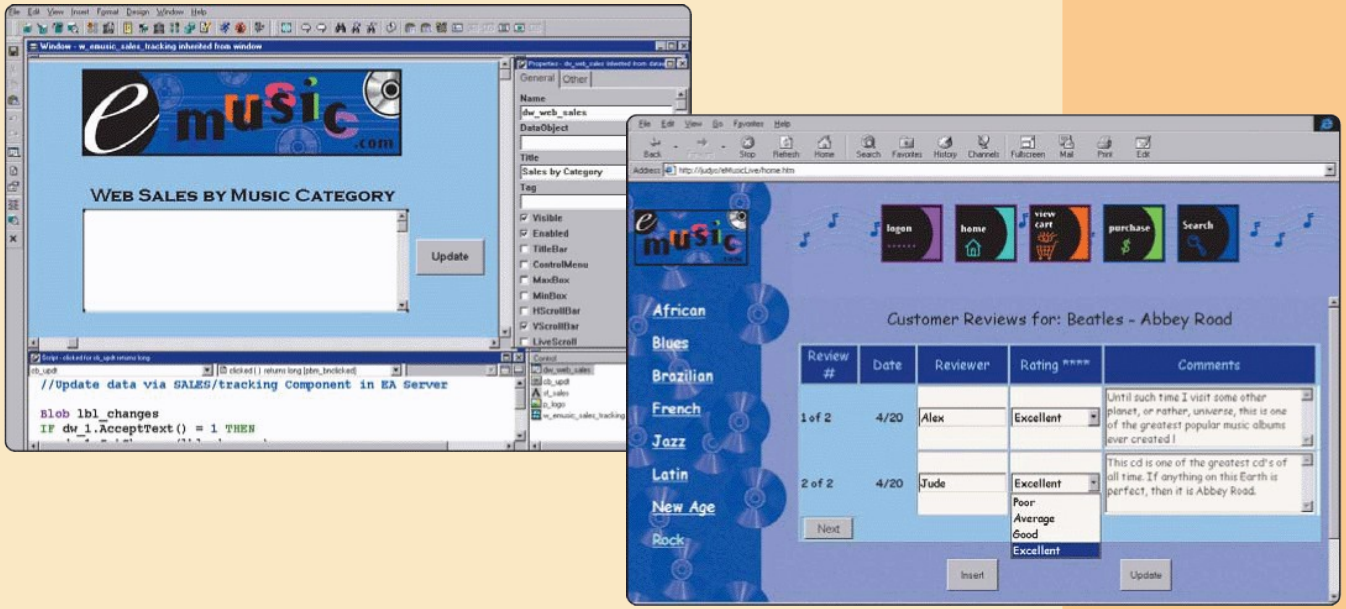


Next

Object-Oriented Programming Languages

PowerBuilder?

- Một công cụ lập trình trực quan mạnh khác
- Phù hợp với các ứng dụng Web-based hay các ứng dụng lớn HĐT - object-oriented applications



Next

Nonprocedural Languages and Program Development Tools

- **nonprocedural languages và program development tools?**

Nonprocedural Language

LTV viết các lệnh giống tiếng anh hoac tương tác với môi trường trực quan để nhận được các dữ liệu từ files hay database

Program Development Tools

Các chương trình thân thiện với người sử dụng được thiết kế để trợ giúp cả LTV lẫn người sử dụng trong việc tạo chương trình

Next

RPG (Report Program Generator)?

- Các ngôn ngữ LT phi thủ tục được dùng để tạo các báo cáo, thiết lập các thao tác tính toán và cập nhật files

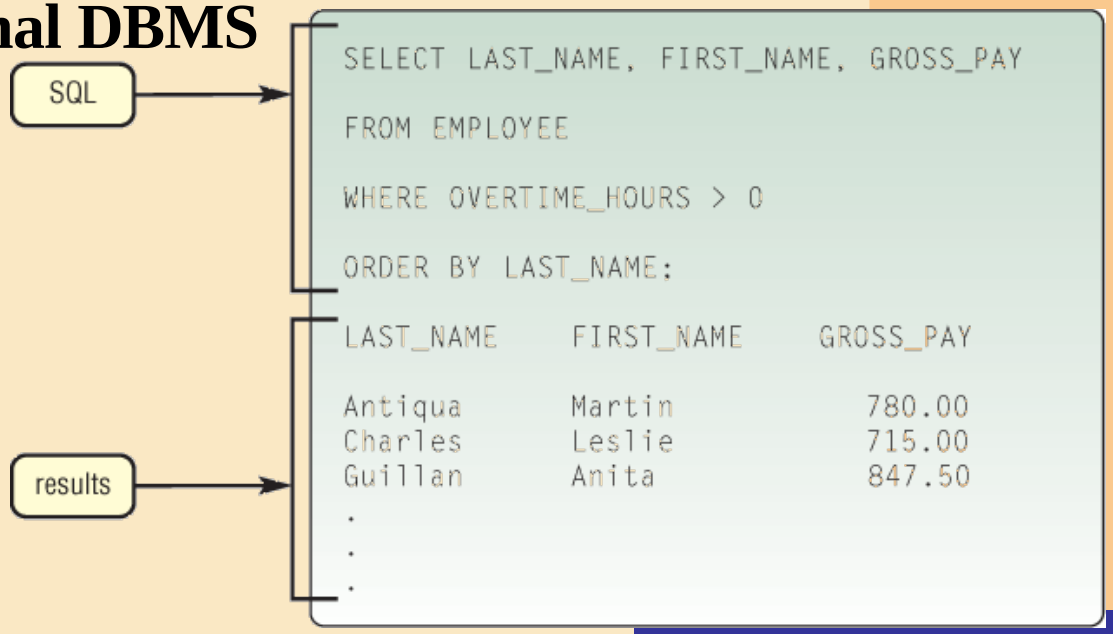
```
C* COMPUTE REGULAR TIME PAY
C           RTHRS     MULT RATE            RTPAY     72
C*
C* COMPUTE OVERTIME PAY
C           OTHRS     IFGT 0
C           RATE      MULT 1.5             OTRATE    72
C           OTRATE    MULT OTHRS           OTPAY     72
                      ELSE
C                     INZ                  OTPAY     72
C
C* COMPUTE GROSS PAY
C           RTPAY     ADD  OTPAY           GRPAY     72
C
C* PRINT GROSS PAY
C                     EXCPTDETAIL
C
C*
O* OUTPUT SPECIFICATIONS
OQPRINT  E            DETAIL
O                                    23 'THE GROSS PAY IS $'
O                     GRPAY  J       34
```


Next

Nonprocedural Languages and Program Development Tools

NN thế hệ IV fourth-generation language (4GL)?

- Là các ngôn ngữ phi thủ tục cho phép truy cập dữ liệu trong csdl
- NNLT 4GL thông dụng là SQL ,Access, là các ngôn ngữ truy vấn . Cho phép users quản trị dữ liệu trong csdl quan hệ relational DBMS

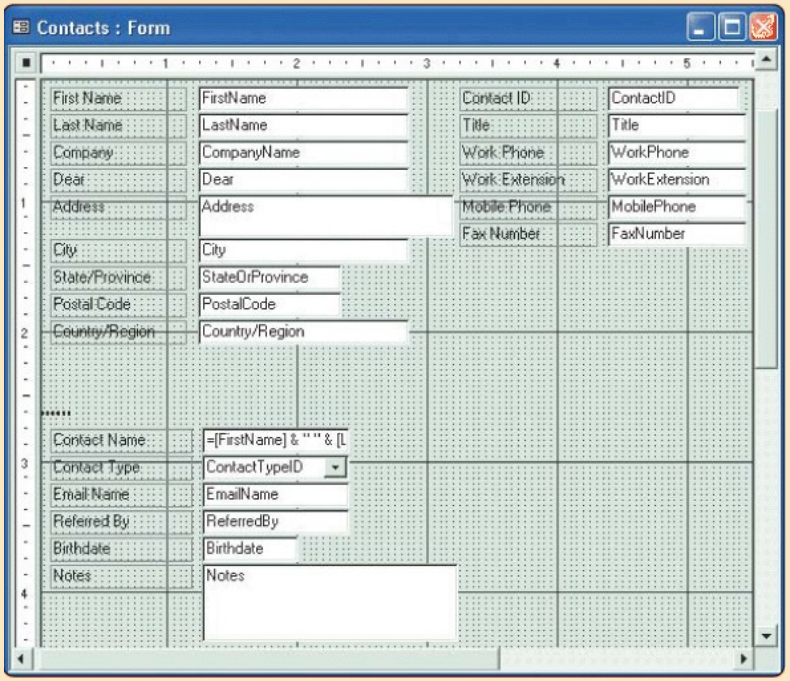


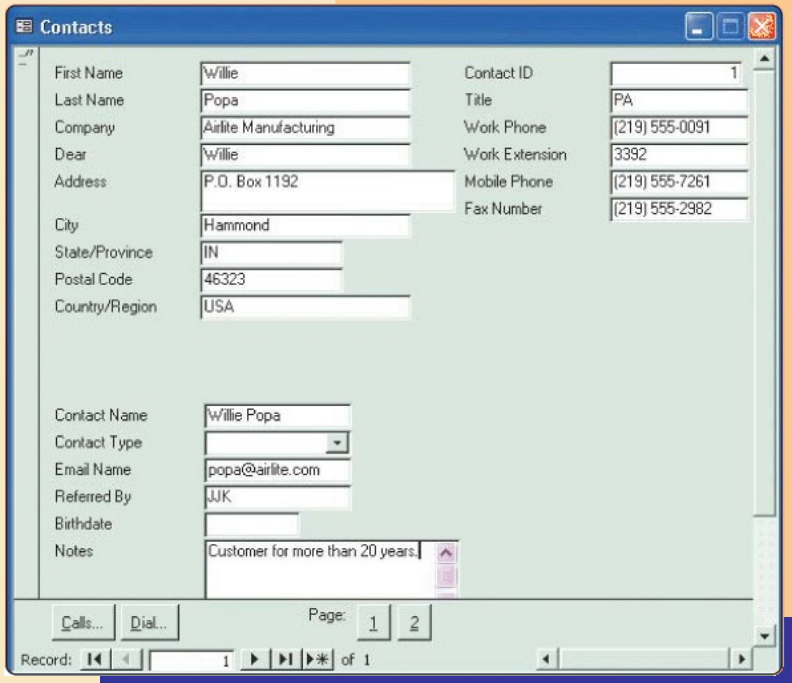
p. 675 Fig. 13-15

Next

Application generator?

- **Là chương trình tạo source code hoặc machine code từ các specification**
- **Bao gồm các chương trình tạo Report , form, và tạo menu**
  - **Form** cung cấp các vùng để vào dữ liệu

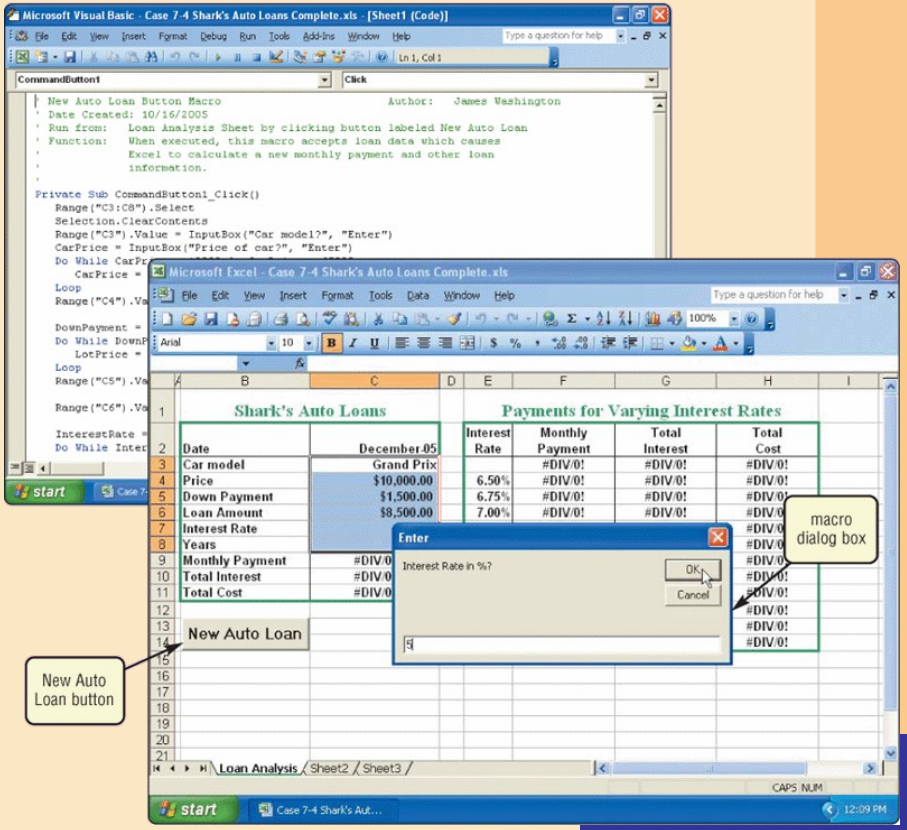




Next

Visual Basic for Applications (VBA)?

- **Macro programming language**
  - **Macro**—Dãy các lệnh dùng để tự động hóa các công việc

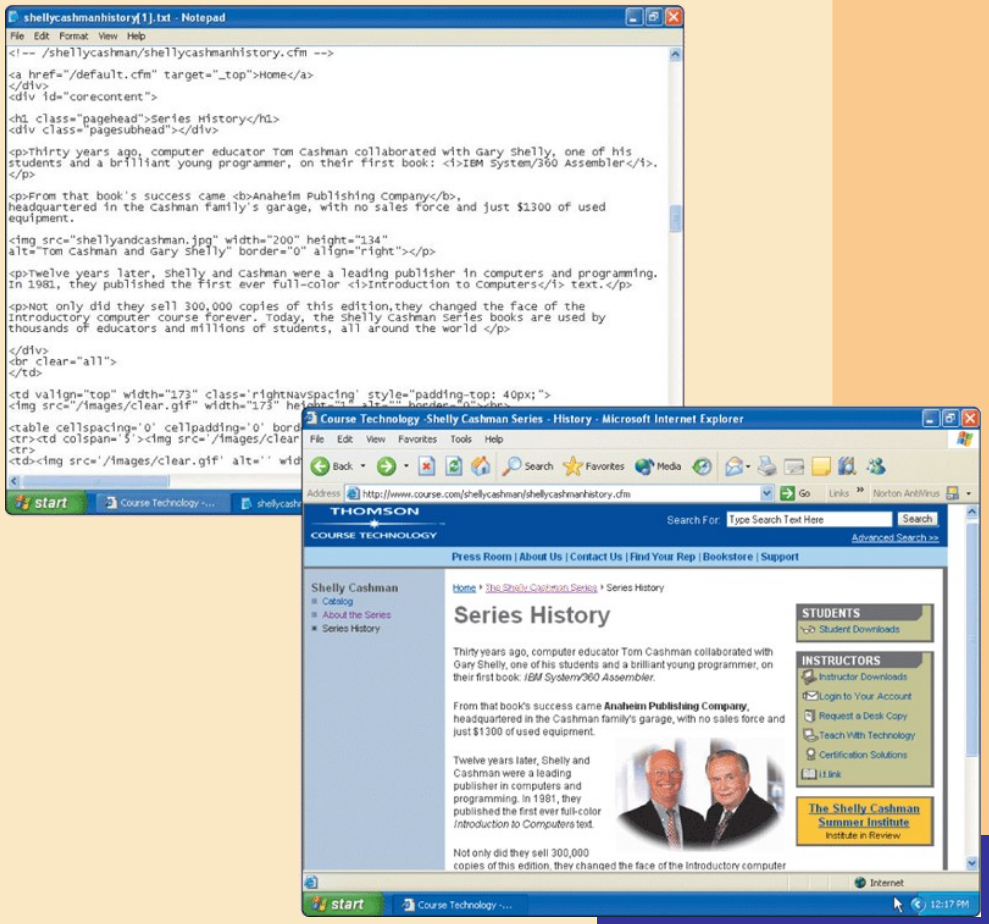


Next

Web Page Development

HTML (Hypertext Markup Language)?

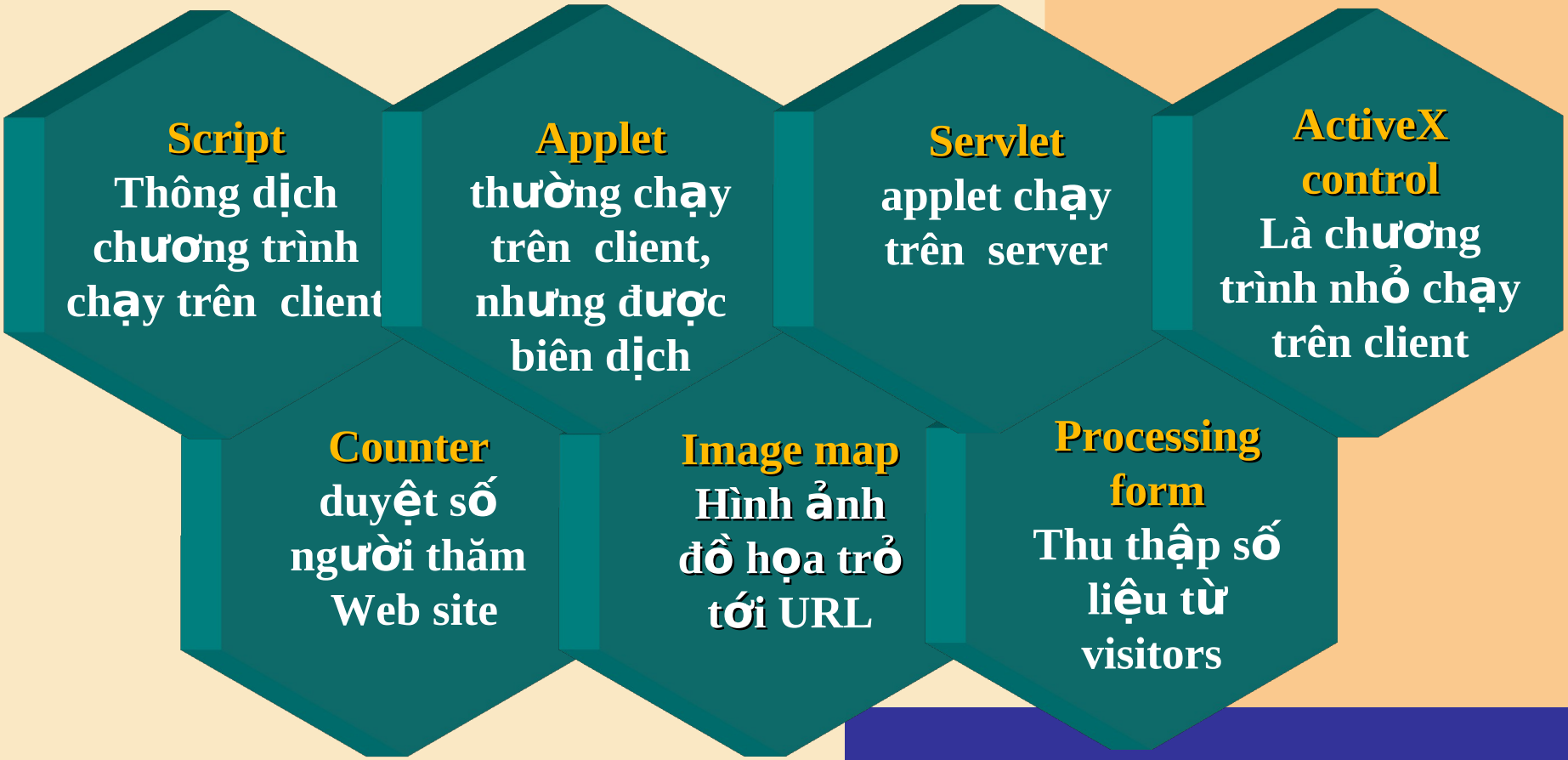
- Dùng để tạo các trang Web



Next

Web Page Development

Các hiệu ứng đặc biệt và các phần tử tương tác được thêm vào trang Web như thế nào ?

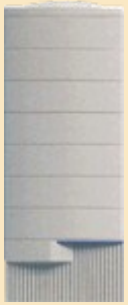


Next

Web Page Development

Common gateway interface (CGI)?

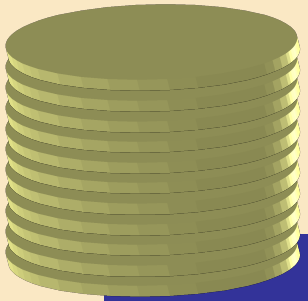
- **Chuẩn giao tiếp xác định cách thức Web server giao tiếp với các nguồn tài nguyên bên ngoài**
  - **CGI script**—program quản trị việc gửi và nhân dữ liệu qua CGI



Step 1. LTV lưu các CGI program trong 1 thư mục đặc biệt trên Web server ví dụ /cgi-bin.

Step 2. Webmaster tạo 1 liên kết giữa CGI program và Web page. Khi 1 user hiện trang Web , CGI program sẽ automatically starts.

Step 3. Khi user khẳng định submits 1 yêu cầu, nó sẽ được gửi cho CGI program. CGI program kết nối với database và lấy các ttin cho user. Ví dụ user yêu cầu xem phim *The Wizard of Oz*.



Step 4. CGI program nhận thông tin từ database, két hợp chúng dưới dạng HTML , và gửi cho trình duyệt Web của User.

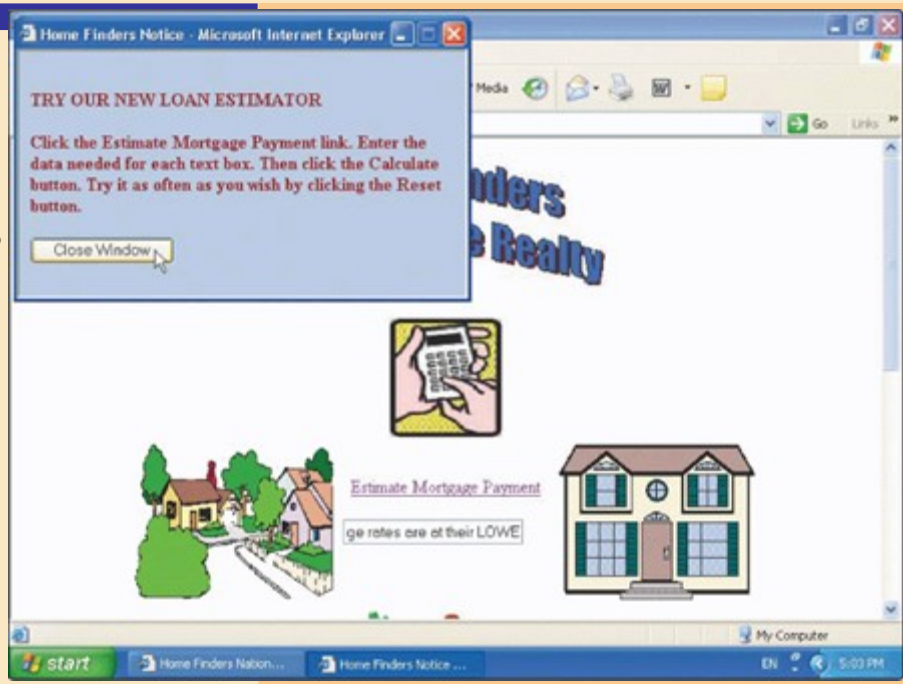
p. 680 Fig. 3-19

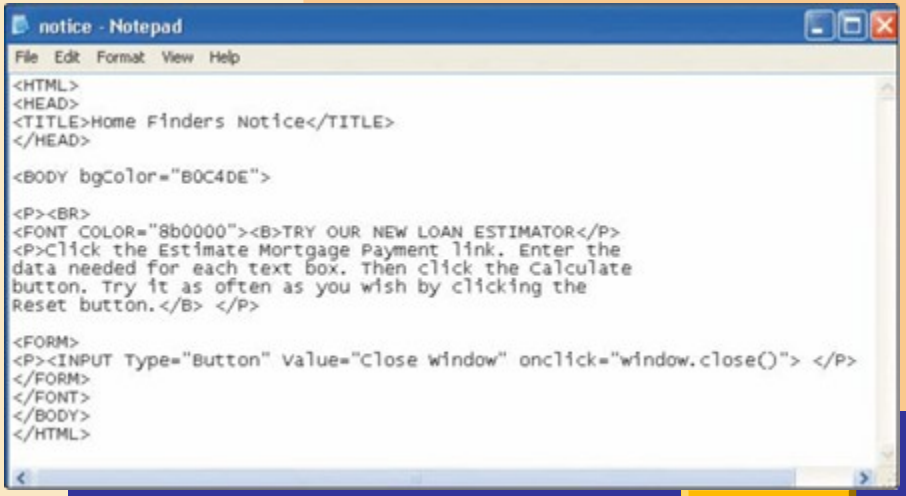
Next

Web Page Development

Scripting language?

- **Rất dễ học và dễ sử dụng**
  - **JavaScript**—thêm các nội dung động và các phần tử tương tác vào Web page
  - **VBScript** (Visual Basic, Scripting Edition)—Thêm tính thông minh và tương tác vào Web page
  - **Perl** (Practical Extraction and Report Language)—Có khả năng xử lý văn bản rất mạnh



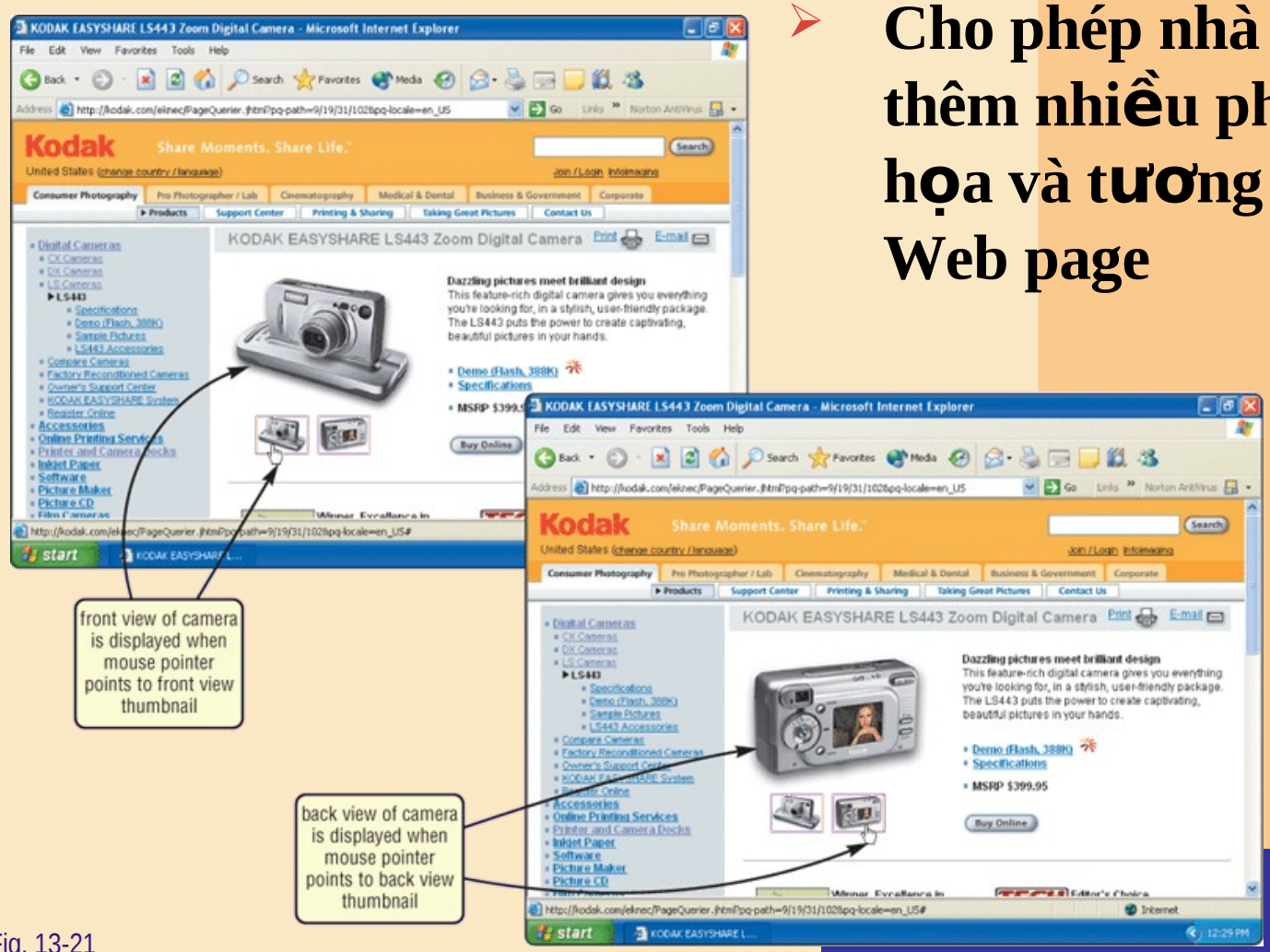


Next

Web Page Development

Dynamic HTML (DHTML)?

- **Cho phép nhà phát triển thêm nhiều phần tử đồ họa và tương tác vào Web page**



p. 682 Fig. 13-21

Next

Web Page Development

XHTML, XML, và WML?

XHTML
(Extensible HTML)
tạo khả năng Web sites có thể hiện dễ ràng hơn trên các trình duyệt

Chứa các tính năng của HTML và XML

XML
(Extensible Markup Language)
Cho phép developers có thể tạo các thẻ - tags – riêng của mình

Server gửi toàn bộ bản ghi cho client, tạo khả năng cho client có thể thực hiện việc xử lý mà không phải quay lại server

WML
(Wireless Markup Language)
Cho phép developers có thể thiết kế những trang cho các trình duyệt chuyên dụng – mobil, …

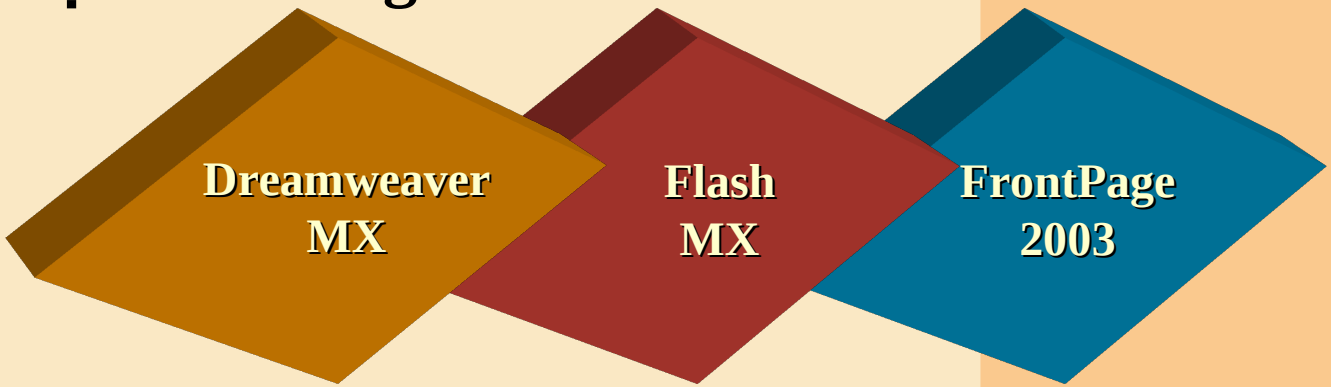
Sử dụng chuẩn wireless application protocol (WAP), để xác định cách thức các thiết bị không dây liên lạc với Web

Next

Web Page Development

Web page authoring software?

- Tạo các trang Web hoàn hảo mà không cần dùng HTML
- Tự tạo các trang HTML

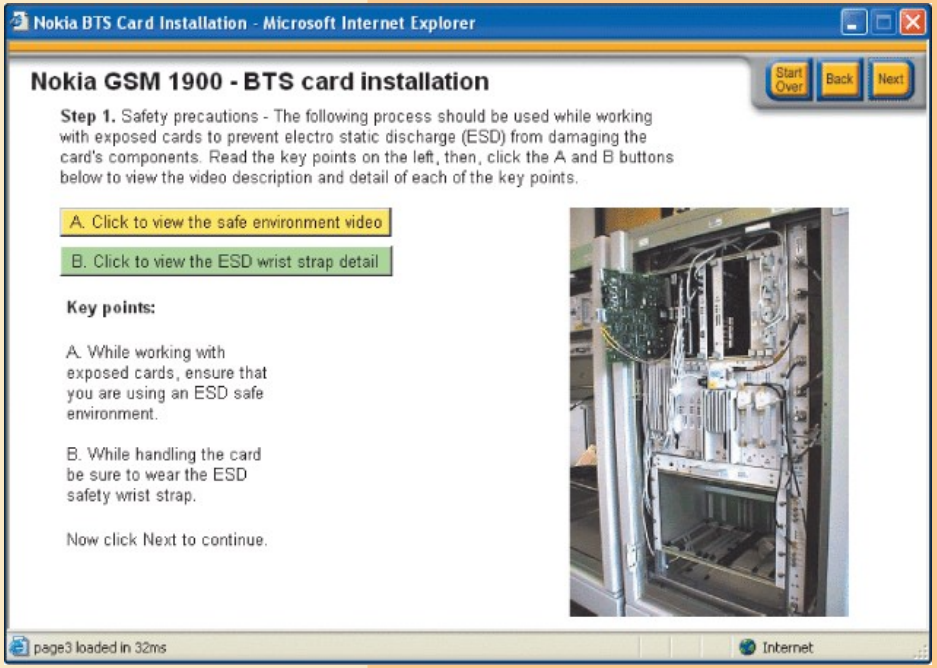


Next

Multimedia Program Development

Multimedia authoring software?

- Kết hợp văn bản, đồ họa, hoạt hình, âm thanh và video trong 1 bài trình diễn có tương tác
- Sử dụng cho computer-based training (CBT) và Web-based training (WBT)
- Software includes Toolbook, Authorware, và Director



p. 684 Fig. 13-22

Next

Các Programming Languages khác

ADA

ALGOL

APL

FORTH

FORTRAN

HYPERTALK

LISP

LOGO

MODULA-2

PASCAL

PILOT

PL/I

PROLOG

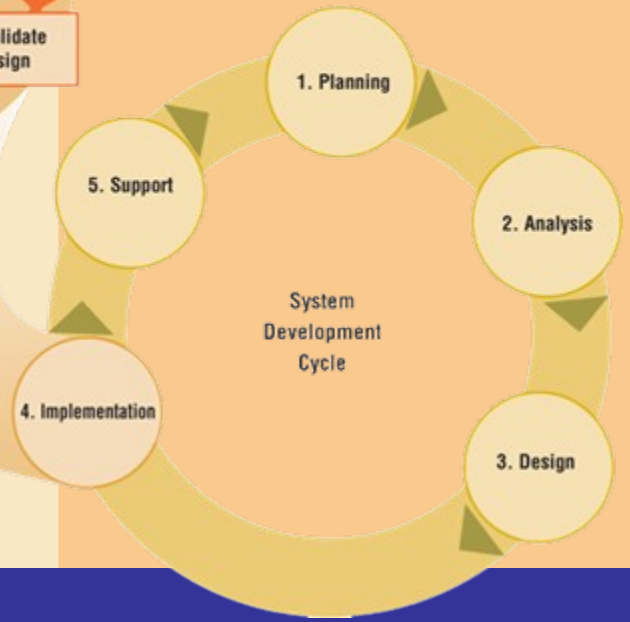
SMALLTALK

Next

Chu trình phát triển Program

Program development cycle?

- **Là các bước mà LTV dùng để xây dựng programs**
  - **Programming team**—Nhóm LTV cùng xd chương trình



p. 686 Fig. 13-24

Next

Step 1 — Analyze Requirements

Các việc cần làm khi phân tích yêu cầu?

1. **Thiết lập các requirements**
2. **Gặp các nhà phân tích hệ thống và users**
3. **Xác định input, output, processing, và các thành phần dữ liệu**
   - **IPO chart**—Xác định đầu vào, đầu ra và các bước xử lý

IPO CHART

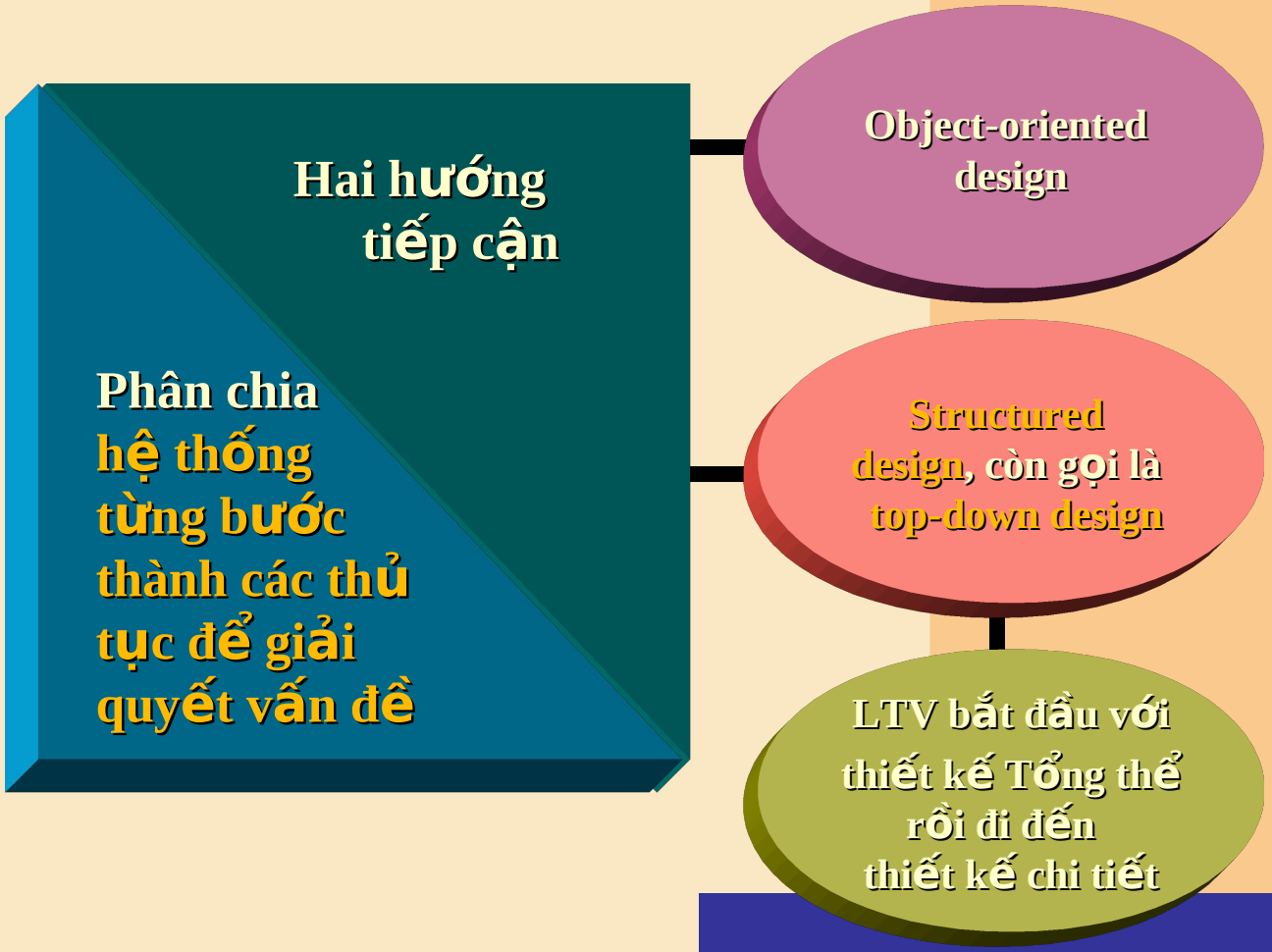
Input	Processing	Output
Regular Time Hours Worked	Read regular time hours worked, overtime hours worked, hourly pay rate.	Gross Pay
Overtime Hours Worked	Calculate regular time pay.	
Hourly Pay Rate	If employee worked overtime, calculate overtime pay.	
	Calculate gross pay.	
	Print gross pay.	

p. 687 Fig. 13-25

Next

Step 2 — Design Solution

Những việc cần làm trong bước thiết kế giải pháp?

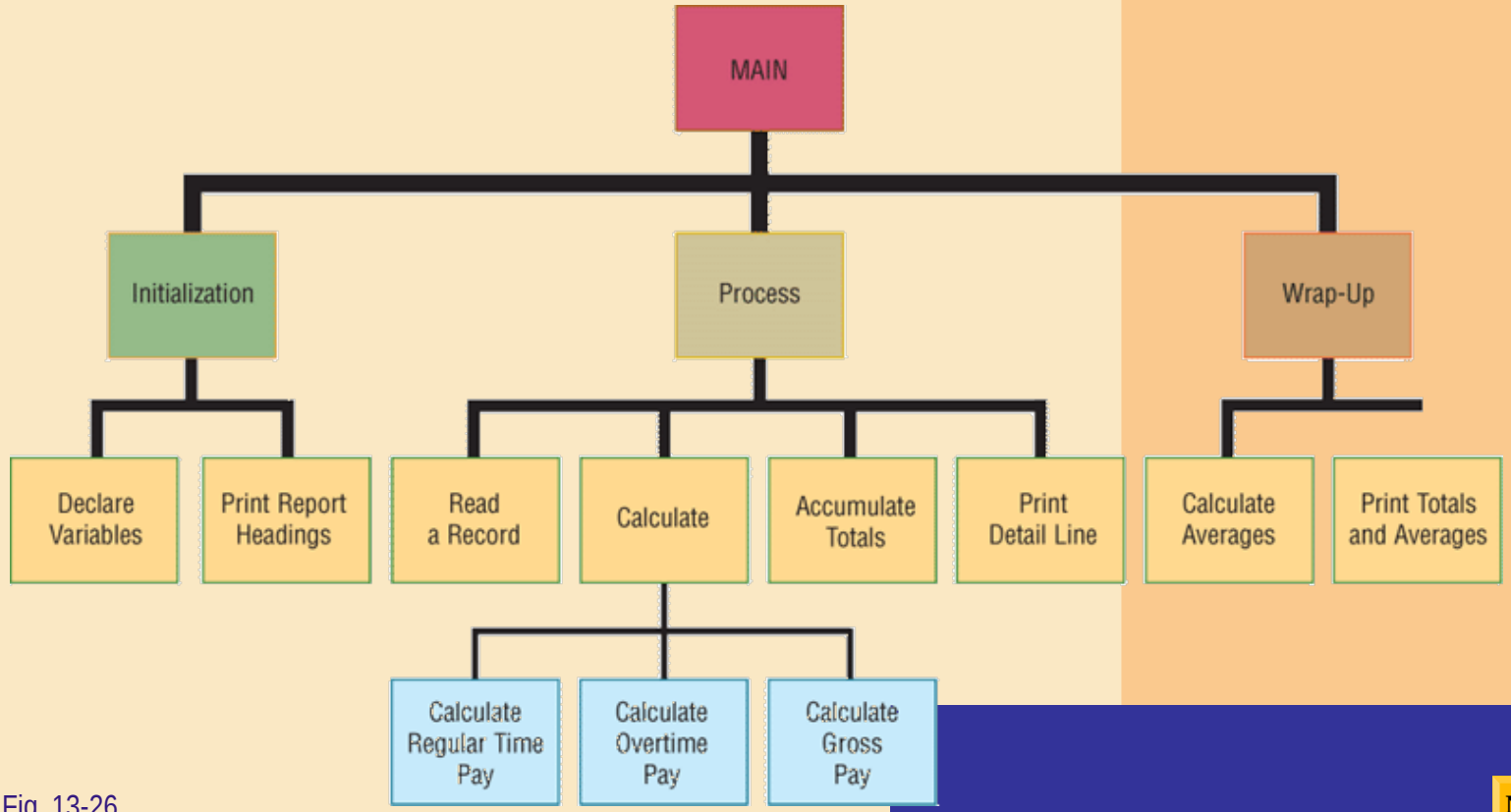


Next

Step 2 — Design Solution

Sơ đồ phân cấp chức năng- hierarchy chart?

- **Trực quan hóa các modules ct**
- **Còn gọi là sơ đồ cấu trúc**



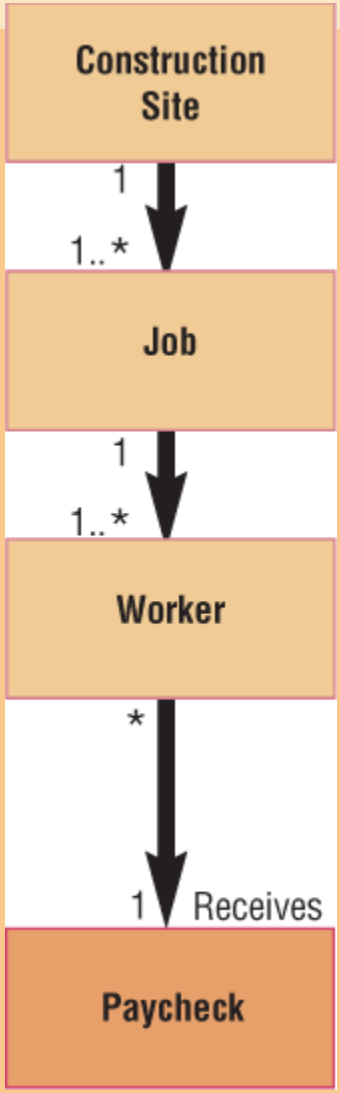
p. 688 Fig. 13-26

Next

Step 2 — Design Solution

Object-oriented (OO) design là gì?

- **LTV đóng gói dữ liệu và các thủ tục xử lý dữ liệu trong 1 object**
  - Các objects được nhóm lại thành các classes
  - Biểu đồ lớp thể hiện trực quan các quan hệ phân cấp quan hệ của các classes

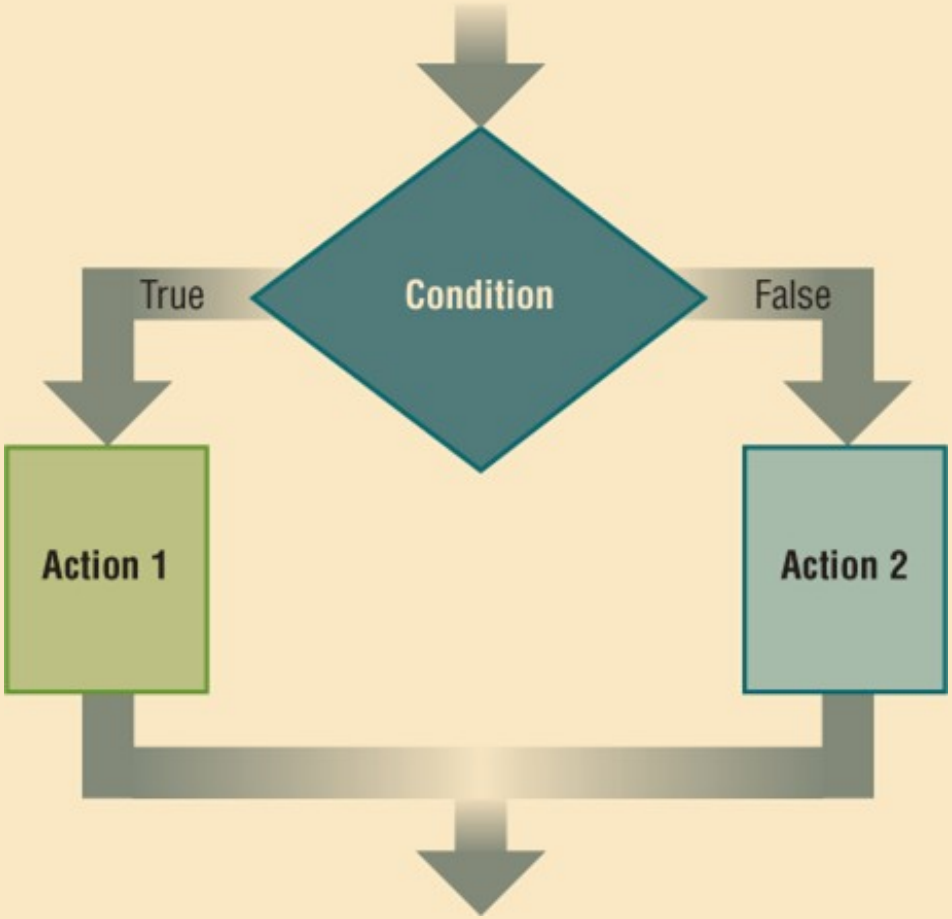


p. 689 Fig. 13-27

Next

Step 2 — Design Solution

Cấu trúc tuyển chọn



- **Chỉ ra action tương ứng điều kiện**
- **2 kiểu**
  - Case control structure
  - **If-then-else control structure**—dựa theo 2 khả năng: true or false

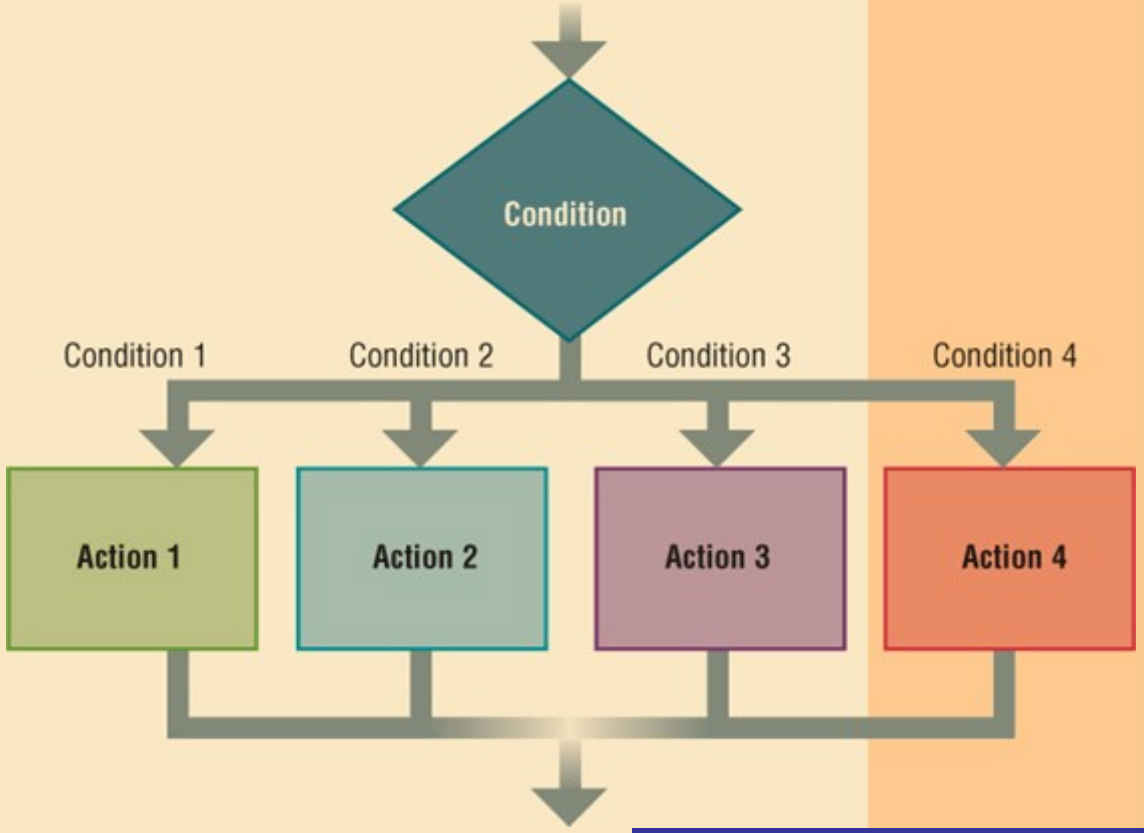
p. 690 Fig. 13-29

Next

Step 2 — Design Solution

Case control structure

- Dựa theo 3 hoặc nhiều hơn các khả năng

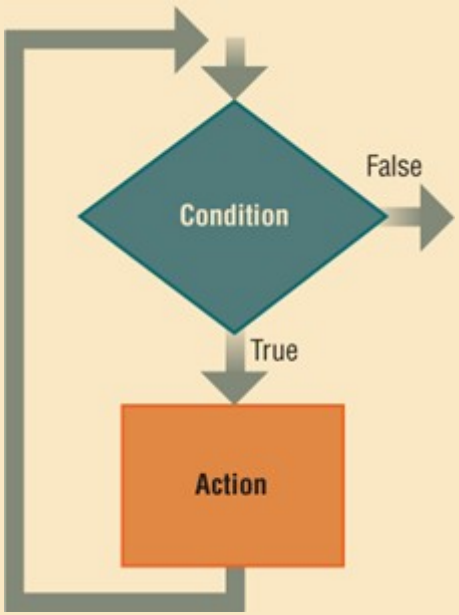


p. 690 Fig. 13-30

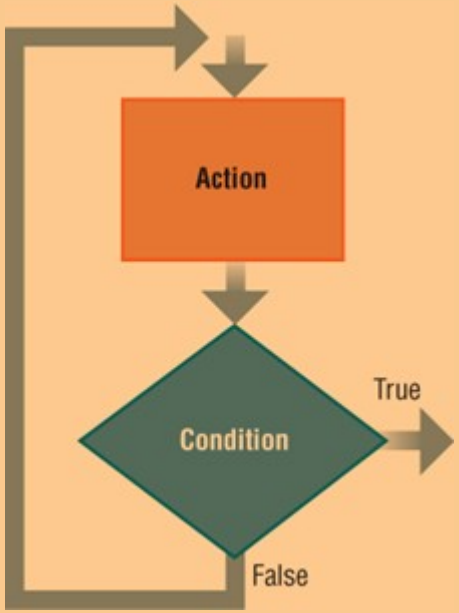
Step 2 — Design Solution

Cấu trúc lặp

- **Cho phép CT thực hiện 1 hay nhiều actions lặp đi lặp lại**
  - **Do-while control structure**—lặp khi điều kiện còn đúng
  - **Do-until control structure**—Lặp cho đến khi điều kiện đúng



Do-While Control Structure

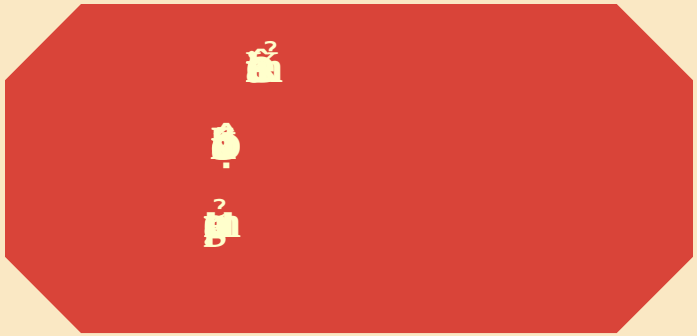


Do-Until Control Structure

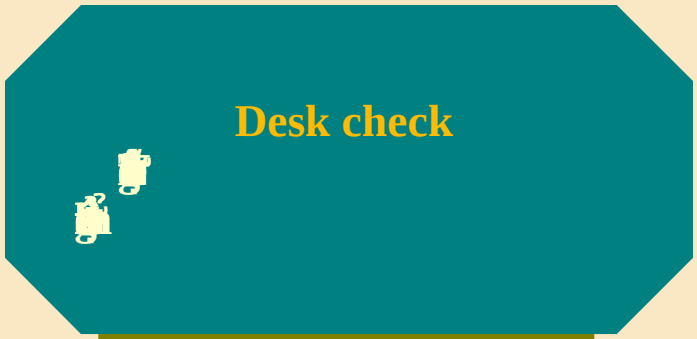
Next

Step 3 — Validate Design

Những điều cần làm trong giai đoạn này?



Logic error
các sai sót khi thiết kế gây ra những kết quả không chính xác



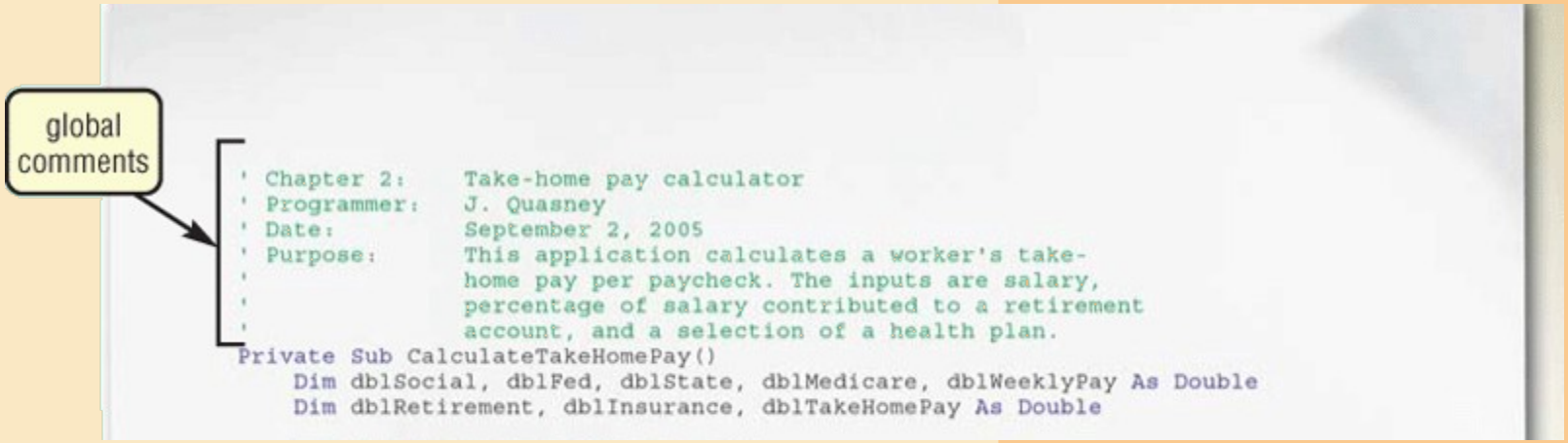
Test data
các dữ liệu thử nghiệm giống như số liệu thực mà CT sẽ thực hiện

Next

Step 4 — Implement Design

implementation?

- **Viết code : dịch từ thiết kế thành program**
  - **Syntax**—Quy tắc xác định cách viết các lệnh
  - **Comments**—program documentation
- **Extreme programming (XP)—coding và testing ngay sau khi các yêu cầu được xác định**



p. 695 Fig. 13-37

Next

Step 5 — Test Solution

Những việc cần làm ?

Đảm bảo CT chạy thông và cho kq chính xác

Debugging—Tìm và sửa các lỗi syntax và logic errors

Kiểm tra phiên bản beta, giao cho Users dùng thử và thu thập phản hồi

Next

Step 6 — Document Solution

Là bước không kém quan trọng

- **2 hoạt động**

Rà soát lại program code—loại bỏ các **dead code**, tức các lệnh mà ct không bao giờ gọi đến

Rà soát, hoàn thiện documentation

Next

Tóm lại

Có hàng loạt các NNLT dùng để viết computer programs

Chu trình phát triển chương trình và các công cụ được dùng để làm cho quá trình này hiệu quả hơn

4 mô hình lập trình cơ bản

Nói thêm về các Mô hình lập trình

- Programming paradigm
  - · Là 1 khuôn mẫu - pattern dùng như một Mô hình lập trình máy tính
  - . Là 1 mô hình cho 1 lớp các NNLT có cùng những đặc trưng cơ bản
- Programming technique
  - · Liên quan đến các ý tưởng thuật toán để giải quyết một lớp vấn đề tương ứng
  - · Ví dụ: 'Divide and conquer' và 'program development by stepwise refinement'
- Programming style
  - . Là cách chúng ta trình bày trong 1 computer program
  - · Phong cách tốt giúp cho chương trình dễ hiểu, dễ đọc, dễ kiểm tra -> dễ bảo trì, cập nhật, gỡ rối, tránh bị lỗi
- Programming culture
  - · Tổng hợp các hành vi lập trình, thường liên qua đến các dòng ngôn ngữ lập trình
  - · Là tổng thể của Mô hình chính, phong cách và kỹ thuật lập trình
  - . Là nhân cách đạo đức trong lập trình cũng như khai thác các CT

Bốn Mô hình lập trình cơ bản

Bốn Mô hình lập trình cơ bản là :

- Imperative paradigm
- Functional paradigm
- Logical paradigm
- Object-oriented paradigm

Bên cạnh đó có thể kể đến :

- Visual paradigm
- Parallel paradigms

Một vài mô hình mới khác :

Concurrent programming

Distributed programming

Extreme programming

…

Tuy nhiên chúng ta chỉ tập trung vào mô hình 1 và sơ qua về 3 mô hình cơ bản còn lại

Bài tập lớn

7 nhóm, Tìm hiểu trên mạng, làm tiểu luận, báo cáo trên lớp

- Functional paradigm
- Logical paradigm
- Visual paradigm
- Parallel paradigms
- Concurrent programming
- Distributed programming
- Extreme programming

Tuần 3 : Phân nhóm – Tìm tài liệu trên mạng

Tuần 4 : Nhắc nhở - Tóm lược nội dung

Tuần 5 : Nộp phác thảo nội dung báo cáo

Tuần 10 : Nộp báo cáo : word : 10-20 trang

Tuần 11 : Làm slides : 10-15 slides – đủ trình bày trong 45’

Tuần 12+13 : Báo cáo trên lớp

Imperative paradigm

- Với Mô hình này ý tưởng cơ bản là các lệnh gây ảnh hưởng đáng kể đến trạng thái chương trình.
- Mỗi imperative program bao gồm
  - *Declarative statements* – các lệnh khai báo, chúng cung cấp các tên cho biến . Các biến này có thể thay đổi giá trị trong quá trình thực hiện Ct .
  - *Assigment statements – Lệnh gán :* gán giá trị mới cho biến
  - *Program flow control statements – Các lệnh điều khiển cấu trúc chương trình* : Xác định trình tự thực hiện các lệnh trong chương trình.
  - Module : chia ct thành các ct con : Functions & Procedures

Imperative paradigm

Các đặc trưng chính của Mô hình này là·

- Về mặt nguyên lý và ý tưởng : Công nghệ phần cứng digital và ý tưởng của Von Neumann
- Các bước tính toán, thực hiện với mục đích kiểm soát cấu trúc điều khiển,· Chúng ta gọi các bước là các mệnh lệnh - *commands*
- Tương ứng với cách mô tả các công việc hàng ngày như là trình tự nấu ăn hay sửa chữa xe cộ
- Những lệnh đặc trưng của imperative languages là : Assignment, IO, procedure calls
- Các ngôn ngữ đại diện : Fortran, Algol, Pascal, Basic, C
- Các thủ tục và hàm chính là hình ảnh về sự trừu tượng : che dấu các lệnh trong CT con, có thể coi CT con là 1 lệnh
- Còn gọi là "Procedural programming"

Functional paradigm

- Functional programming trên nhiều khía cạnh là đơn giản và rõ ràng hơn imperative . Vì nguồn gốc của nó là toán học thuần túy: Lý thuyết hàm. Trong khi imperative paradigm bắt nguồn từ ý tưởng công nghệ cơ bản là digital computer, phức tạp hơn, kém rõ ràng hơn lý thuyết toán học về hàm.
- Functional programming dựa trên nền tảng khái niệm toán học về hàm và 1 NNLT hàm bao gồm ít nhất những thành phần sau :
  - **Tập hợp các cấu trúc dữ liệu và các hàm liên quan**
  - **Tập hợp các hàm cơ sở - Primitive Functions.**
  - **Tập hợp các toán tử .**

Functional paradigm

Các đặc trưng cơ bản :

- Về mặt nguyên lý và ý tưởng : Toán học và lý thuyết hàm
- Các giá trị tạo được là không thể biến đổi *non-mutable*
- Không thể thay đổi các yếu tố của giá trị hợp thành
- Giống như phương thuốc, có thể tạo một phiên bản của các giá trị hợp thành : một giá trị trung gian
- Trừu tượng 1 biểu thức đơn thành 1 hàm và hàm có thể tính toán như là 1 biểu thức
- Các hàm là những giá trị đầu tiên
- Hàm là dữ liệu hoàn chỉnh, giống như số, danh sách, ...
- Thích hợp với xu hướng tính toán theo yêu cầu
- Mở ra những khả năng mới

Ví dụ về Functional programming

```
function GT(n: longint) : longint;
var  x  : longint;
Begin
  x:=1;
  while (n > 0) do begin
        x := x * n;
        n := n - 1;
  end;
  GT := x;
End;
```

```
Function GT(n: longint) : Longint;
Begin
  if n=1 then GT :=1
  else  GT := n* GT(n-1);
End;
```

Với functional paradigm, ta có thể viết

```
GT n =
if n = 1 then 1
else n * GT(n - 1);
```

logic paradigm

- Mô hình lập trình logic hoàn toàn khác với các mô hình còn lại.
- Mô hình này đặc biệt phù hợp với những lĩnh vực liên quan đến việc rút ra những kiến thức từ những sự kiện và quan hệ cơ bản – lĩnh vực trí tuệ nhân tạo. Có vẻ như mô hình này không gắn với những lĩnh vực tính toán nói chung.
- Trả lời 1 câu hỏi thông qua việc tìm các giải pháp
- Các đặc trưng:
  - Về nguyên tắc và ý tưởng : Tự động kiểm chứng trong trí tuệ nhân tạo
  - Dựa trên các chân lý- tiên đề axioms,các quy luật suy diễn - inference rules, và các truy vấn queries.
  - Chương trình thực hiện từ việc tìm kiếm có hệ thống trong 1 tập các sự kiện, sử dụng 1 tập các luật để đưa ra kết luận

object-oriented paradigm

Mô hình hướng đối tượng thu hút được sự quan tâm và nổi tiếng từ khoảng 20 năm nay. Lý do là khả năng hỗ trợ mạnh của tính bao gói và gộp nhóm logic của các khía cạnh lập trình. Những thuộc tính này rất quan trọng khi mà kích cỡ các chương trình ngày càng lớn.

Nguyên nhân cơ bản và sâu sắc dẫn đến thành công của mô hình này là :

- Cơ sở lý thuyết đỉnh cao của mô hình. 1 CT HĐT được xây dựng với những khái niệm, tư tướng làm cơ sở, điều đó rất quan trọng và theo cách đó tất cả các kỹ thuật cần thiết cho lập trình trở thành thứ yếu.

Gủi thông điệp giữa các objects để mô phỏng sự tiến triển theo thời gian của hàng loạt các hiện tượng trong thế giới thực

object-oriented paradigm …

Các đặc trưng

- Nguyên lý và ý tưởng : Lý thuyết về concepts, và các mô hình tương tác trong thế giới thực
- Dữ liệu cũng như các thao tác trên dữ liệu được đóng gói trong objects
- Cơ chế che dấu thông tin được sử dụng để tránh những tác động từ bên ngoài object
- Các Objects tương tác với nhau qua việc truyền thông điệp, đó là phép ẩn dụ cho việc thực hiện các thao tác trên 1 object
- Trong phần lớn các NNLT HĐT, objects được nhóm lại trong classes
  - **Objects trong classes có chung các thuộc tính, cho phép lập trình trên lớp, thay vì lập trình trên từng đối tượng riêng lẻ**
  - **Classes đại diện cho concepts còn objects đại diện cho hiện tượng**
  - **Classes được tổ chức trong cây phả hệ có kế thừa**
  - **Tính kế thừa cho phép mở rộng hay chuyên biệt hóa lớp**