Lecture Outline

- Introducing the Lecturer
- Introducing the Tutor
- Aims and Objectives of the unit
- Text Books
- Lecture structure
- Tutorial structure
- Assessment
- SGO–Creating Programs

Let’s begin

Introducing the Lecturer

Office: RmS4.02, Tower Building
Ph: 9903 1911
Mb: 0407 886 791
Email: angela.carbone@infotech.monash.edu.au
URL: http://www.sims.monash.edu.au/Staff/Angela

Consultation:
- Probably best straight after the lecture
- Generally NOT available Thursdays & Monday
- Please email or phone me to make an appointment

Introducing the tutors

- Shiu Ming Tsang (Eder)
  – Wed 1-3pm, Rm T201 (start in week 2)
  – Eder_tsang@hotmail.com

Aims and Objectives

What will you learn?
- To design programs for Windows that:
  – Manipulate data in files
  – Perform calculations and generate reports
  – Are useful
- General principles of programming
  – Skills transferable to other programming languages or settings
- Methodologies of problem solving

Resources

- Unit Materials
  – Unit Book with 12 study guides, reading
  – “Supporting Files” – Available on Unit Web Site
  – Assignments on the website
- Text Books
  – Zak: “Microsoft Visual Basic .NET Reloaded”
  – You should have your own copy of each book
- Aliased with other subject CGO1810
  – But there are differences
Lectures

• Lecture Time
  – Wed 10-12pm Rm A2.05 Caulfield
• Lecture Materials
  – Will be posted on the subjects Website for students to download
• Lecture Attendance and Etiquette
  – Attendance is highly recommended
  – Please turn mobile phones on silent vibration

Lecture Outline

1. L1: Introduction to IMS1901
2. L2: Object and Classes
3. L3: Application Development
4. L7: Variables and Data Types
5. L9: The Selection Structure I
7. L13: The Repetition Structure I
8. L15: Variables and Data Types
9. L17: File I/O
10. L18: Arrays I
11. L21: String Manipulation
12. L23: Structures
13. L25: Exam Revision

L2: Program Design
L4: Introduction to VB.NET IDE
L6: Runtime Assignment
L9: Input and Output
L10: The selection Structure II
L12: TEST 1 10%
L14: The Repetition Structure II
L16: Functions and Subroutines I
L18: List Boxes
L20: Arrays II
L22: TEST 2 10%
L24: Invited Guest
L26: Exam Revision

L3: Introduction to VB.NET/IDE
L5: Functions and Subroutines I
L7: Functions and Subroutines I
L9: Functions and Subroutines I
L11: File I/O
L13: Functions and Subroutines I
L15: Functions and Subroutines I
L17: Functions and Subroutines I
L19: Functions and Subroutines I
L21: Functions and Subroutines I
L23: Functions and Subroutines I
L25: Functions and Subroutines I

Mid - Semester Break ****

L8: Input and Output
L10: The selection Structure II
L12: TEST 1 10%
L14: The Repetition Structure II
L16: Functions and Subroutines I
L18: Arrays II
L20: Arrays II
L22: TEST 2 10%
L24: Invited Guest
L26: Exam Revision

Tutorials

• Start in week 2
• Please check your room
• Tutorial Allocations (via Allocate +)
  – http://allocate.cc.monash.edu.au
• Tutorial Exercises
  – http://www.sims.monash.edu.au
  – Then click on [courses, simunits, IMS1906]
  – OR

Tutorial Organisation

1. Exercise Sheet
2. 1. Introduction to program design
3. 2. Designing a copyright screen
4. 3. Creating a simple Calculator
5. 4. Using Variables and Data Types
6. 5. A custom made quote
7. 6. Calculating Income Tax
8. 7. Applying Repetition Structure
9. 8. Writing functions and Subroutines
10. 9. File I/O
11. **** Mid Semester Break ****
12. 10. Arrays and List Boxes
13. 11. Structures
14. 12. Random Access Files

Assessment

• Exam Result comprises:
  – End of semester exam 40%
  – Mid-semester Test 1 10%
  – Mid-semester Test 2 10%
• Assignments
  – Assignment 1 10%
  – Assignment 2 15%
  – Assignment 3 15%
• Tutorials:
  – Attendance will be recorded and participation will be factored into the final mark

Assignments

• Three assignments 40%
  – Assignment 1
  – Assignment 2
  – Assignment 3
Assessment Notes (See Unit Outline)

- **Plagiarism**
- Standards for presentation [s2.1]
- Extensions [s2.3]
- Student Problem Resolution Procedure [s3]
- Pass requirements [s4]

Learning Strategies

- **Active listening**
  - attentively listen, by focusing on what is said, so you can answer questions pertaining to the information
- **Reading**
  - Practicing & learning by mistakes
  - Ask for help or clarification during lectures and tutorials

SG0: Creating Programs

- **Part 1 – Computer Systems**
  - Hardware/Software
  - Programming languages
  - Compilers and Interpreters
  - Program Development

Study Guide 0: Objectives

- **Understand terms:**
  - Computer Software/Hardware
  - Computer Program
- **Understand what is meant by:**
  - Programming Language
  - Compiler, Interpreter
  - Structured Programming
  - Algorithm
- **Describe steps involved in design and development of computer programs**
- **List the 7 stages of the Program Development Process**

Components of a Computer System

Five elements:
- hardware
- software
- data (raw data → useful information)
- people (e.g. computer programmers, users)
- procedures (software development methodology)

[Unit Guide, SG 0, p2]

Basic Computer Hardware

- Input device
- Output device
- Central processing unit (CPU)
- Main memory
- Secondary storage device
Software

- System software – Operating System
  - programs to manage the computer’s hardware and devices
  - provide a user interface – typically textual or graphical
- Application software
  - programs to perform specific tasks (e.g. word processor, database management system, web browser, your programs)
  - “Commercial Off The Shelf” (COTS)
- Set of instructions which “tell” computer what to do
- Convert data into information
- Are particularly useful for
  - Large amounts of data
  - Complicated/repetitive processing

People write Computer Programs

- Human-readable highly-structured text
- Tells computer what to do and how
- Example Languages:
  - Machine Language
  - Assembly Language
  - Procedure-oriented languages (Fortran, C, Cobol)
  - Object-oriented languages (Visual Basic .NET, Java,C++/C#)

Compilers and Interpreters

- Allows the programmer to use instructions that more closely resemble the English language
- Require either an interpreter or a compiler to convert English-like instructions to machine code
- Interpreter
  - translates high level instructions into machine code line-by-line
- Compiler
  - translates the entire program into machine code before running the programming
  - Compilers are always much faster than interpreters

Program Development process

Seven Steps:
- Define problem
- Outline solution
- Develop algorithm
- Test the algorithm
- Code the algorithm into a programming language.
- Run program
- Document and maintain

[Robertson, p5]

Algorithm

- A set of instructions
  - Lists the steps involved in accomplishing a task.
- Examples from real-life
  - “The steps for how to bake chocolate cake”
  - “How to drive from Melbourne Airport to Monash Gippsland”
- In programming terms an algorithm must:
  - Be lucid/clear, precise and unambiguous
  - Give the correct solution in all cases; and
  - Eventually end

[Robertson, p. 5]

SG 0 Summary

- Terms
  - Computer Software/Hardware
  - Computer Program
  - Programming Language
  - Compiler
  - Interpreter
- List the 7 stages of the Software Development Process.
- Explain what is meant by an algorithm.
Reading/Questions

Today…
• Unit Book, Study Guide 0
• Robertson, Chapter 1

Next lecture…
• Unit Book, Study Guide 1
• Robertson, Chapters 2.3
• Szymanski article (in Appendix A of Unit Book)
• Is there anything you’d like to ask?

Lecture Outline

• Program Design
  – Modular programming
  – Top down design
  – Algorithms
  – The Structure Theorem
  – IPO charts
  – Algorithm development
  – Desk Checking

Why Program?

• To Learn
  – The power of writing programs
  – The challenge of writing programs
  – The cost of writing programs
  – The joy of programming

• Remember
  – If you write specifications for programmers (which is what analysts do), you have to have a deep understanding of what you’re writing about.

SG 1 - Objectives

• List the qualities of a good program.
• Describe the features of structured program design.
• Use top-down development techniques to refine a problem to its lowest level.
• Describe the 3 control structures in the structure theorem.
• Use pseudo code to design algorithmic solutions to simple computational problems.
• Design algorithms which use the three control structures of the structure theorem.

Qualities of good programs

• Correct
• Readable
• Easy to maintain and modify
• Reliable and robust
• Efficient (time/space)
• Flexible
  (Szymanski, pp. 288–289)
Structured Programming

- Structured Programming:
  - Encompasses:
    > The Structure Theorem
    > Modular design
    > Top-down development
  - Read Robertson section 1.2
- Why is it important?
  - Read Szymanski p.287-289 (Unit Book, Appendix)

Modular Program Design

- Programming problems often very large and complex
- Large complex problems handled as sets of smaller problems called modules
- Each module responsible for one task
- Makes the problem psychologically easier to manage
  - people can generally only process or retain 4 to 7 ideas at a time
  - programmers can usually only work efficiently with blocks of 20 - 50 lines of code
- Allows several programmers to work on one program simultaneously

Top-down Development

- Decompose the problem into its major stages, starting with the overview level
- At each stage, identify the sub-problems which make up that stage
- Continue this way until the problems can’t be broken down further

Algorithm

- Sequence of precise steps to solve the problem
  - no ambiguity
- Finite number of steps.
  - An algorithm’s logic must terminate
- Expressed in a precise language or graphical form
  - Pseudo code (structured English)
  - Flowcharts
  - Other formats

Algorithm Notations

- Pseudo code
  - Structured English
  - Uses key-words for control and structure elements
  - Suits “verbal” thinkers
  - Written in a text editor or word processor
- Flowcharts
  - Diagrammatic, with minimal pseudo code
  - Shapes represent control and structure elements
  - Arrows between shapes show progression
  - See Robertson, Appendix 1, for one flowchart notation

Pseudo code

- Basic computer operations:
  1. Receive Input
     read or get
  2. Output
     Print, Write, Put Out, Output or Display
  3. Arithmetic:
     + (add), - (subtract), * (multiply), / (divide)
  4. Assign a value to a variable (storage location)
  5. Compare two values, then select a course of action
  6. Repeat a group of actions

[Robertson, p12-14]
Control Structures

- The Structure Theorem: any computer program can be built from just three control structures:
  - Sequence
  - Selection
  - Repetition
- Used to determine (control) the order in which the steps in an algorithm are to be executed

All problem solutions can be expressed as combination of:
- Sequences
- Selection or decision structures
- Loops or repetition structures

SG 1: Summary

- Qualities of a good program
- Features of structured program design.
- Top-down development.
- Pseudo code and flow charts for expressing algorithms
- The structure theorem:
  - Sequence
  - Selection
  - Repetition

Steps involved in writing a program

1. Define the Problem – Use English
2. Outline the Solution – Write Algorithms
3. Desk Check the Algorithms
4. Plan and Design the User Interface
5. Translate the Algorithms into Code
6. Document and Maintain the program
Defining the Problem

• Identify:
  – inputs
  – outputs
  – processing
• Record these in a defining diagram or IPO Chart

Example Problem 1

Problem
You need to read in a tax rate as a percentage and the prices of five items. The program is to calculate and display the total price and the amount of this total which is the tax payable on those items.

How do we turn this into a computer program?
1. Identify the nouns
2. Identify inputs and outputs
3. Identify processes (verbs)
4. Write an algorithm

Step 1: Identify the nouns

Problem
You need to read in a tax rate as a percentage and the prices of five items. The program is to calculate and display the total price and the amount of this total which is the tax payable on those items.

Step 2: Identify inputs and outputs

<table>
<thead>
<tr>
<th>Input</th>
<th>Processing</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>tax rate</td>
<td>read tax rate</td>
<td>total price</td>
</tr>
<tr>
<td>item price 1</td>
<td>read item price 1</td>
<td>tax payable</td>
</tr>
<tr>
<td>item price 2</td>
<td>read item price 2</td>
<td></td>
</tr>
<tr>
<td>item price 3</td>
<td>read item price 3</td>
<td></td>
</tr>
<tr>
<td>item price 4</td>
<td>read item price 4</td>
<td></td>
</tr>
<tr>
<td>item price 5</td>
<td>read item price 5</td>
<td></td>
</tr>
<tr>
<td>item price 6</td>
<td>calculate item price 1 to 6</td>
<td></td>
</tr>
<tr>
<td>item price 7</td>
<td>calculate item price total</td>
<td></td>
</tr>
<tr>
<td>item price 8</td>
<td>calculate tax amount</td>
<td></td>
</tr>
<tr>
<td>item price 9</td>
<td>display item price total</td>
<td></td>
</tr>
<tr>
<td>item price 10</td>
<td>display tax amount</td>
<td></td>
</tr>
</tbody>
</table>

Step 3: Identify processes (verbs)

Problem
You need to read in a tax rate as a percentage and the prices of five items. The program is to calculate and display the total price and the amount of this total which is the tax payable on those items.

Steps involved in writing a program

1. Define the Problem – Use English
2. Outline the Solution – Write Algorithms
3. Desk Check the Algorithms
4. Plan and Design the User Interface
5. Translate the Algorithms into Code
6. Document and Maintain the program
Write an Algorithm

CalculateTotalPriceAndTaxPayable
Read taxRate
Read item1, item2, item3, item4 and item5
totalPrice = item1 + item2 + item3 + item4 + item5
taxPayable = totalPrice * taxRate
Print totalPrice, taxPayable
END

Variables

Identifying Inputs and Outputs

1. Underline the nouns (objects):
   A program is required to read the length and width of a rectangular house block, and the length and width of the rectangular house on the block. The program should compute and display the time required to cut the grass around the house at a rate of two square metres per minute.

Steps involved in writing a program

1. Define the Problem – Use English
2. Outline the Solution – Write Algorithms
3. Desk Check the Algorithms
4. Plan and Design the User Interface
5. Translate the Algorithms into Code
6. Document and Maintain the program
Desk Checking

- **Necessary to ensure algorithm is correct**
  - Can save time when you come to coding and testing
- **Draw up a table of expected results**
  - Specifically look for boundary conditions
  - e.g. -1, 0, 1
- **Draw up a second table:**
  - Variable names on horizontal axis
  - Key steps/points in the algorithm on the vertical axis.
- **Step through the algorithm, writing down changes to variables’ value**
- **Compare the two tables** [Robertson, p28]

Algorithm Design - Summary

- **Write down Inputs/Processing/Outputs (IPO)**
- **Write algorithms**
- **Desk Check the Algorithms**
- **Then move on to implementation**
  - VB.NET Program Design
  - VB.NET Coding
- **Reading**
  - Unit Book, SG 1
  - Robertson Chapters 2, 3
  - Szymanski article (Unit Book, Appendix A)