


COSC 1530



**Introduction to Computer Use II:
Programming**

Winter 2005 (Section M)

Introduction

Wednesday, January 4 2006

Bill Kapralos

COSC 1530, Winter 2006, Bill Kapralos

Overview (1):

- **Administrative Details**
 - Some course preliminaries
- **Introduction to Computer Programming**
 - What exactly is computer programming ?
 - What is the purpose of computer programming ?
 - Solving problems with computers
 - Current programming paradigms

Administrative Details

Course Preliminaries (1):

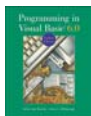
Instructor: Bill Kapralos

Email: billk@cs.yorku.ca

Office hours: Monday 2:30-3:30pm Room CSE 2015

Wednesday 2:30-3:30pm Room CSE 2015

Textbook: Visual Basic: Programming for Literacy by Peter Cribb
(web version: <http://www.cs.yorku.ca/~peterc/VBhandbook/>
or, hard copy can be purchased from bookstore)



Although not required, additional recommended textbook references are available from the course website → the book "Programming in Visual Basic 6.0 (2nd edition), by J.C. Bradley and A.C. Millspaugh" contains VB 6.0 CD

Course Preliminaries (2):

Official course website:

<http://www.cs.yorku.ca/course/1530>

Check course website regularly as all course related information will be conveyed via the course website, including assignments, announcements etc.

Non-official course website:

<http://www.cs.yorku.ca/~billk/cosc1530/home.html>

Website specific to Section M. Lecture notes will be made available here as will any announcements specific to Section M "Preliminary" lecture notes will be made available before the lecture and "final" lecture notes will be made available after the lecture

Course Preliminaries (3):

Tentative Grading Scheme:

Test 1	15%
Test 2	20%
Final Exam	48% (or 100% - see below)

Lab Exercises 17%
(10 labs throughout the semester each one worth 1%, 2% or 4% → see course outline via website for detailed schedule)

Grading Scheme Flexibility:

The only absolutely required measure of your learning achievement is the final exam. If you opt out of other assessment measures the final exam will then be worth 100%. The weighting of the final exam will be reduced by the weighting of whatever other assessment measures you choose to participate in.

Course Objectives (1):

▫ Three Main Course Objectives

- To build on the achievements of CSE1520.03 - to develop further understanding of computers and more **advanced computer skills**.
- To develop elementary programming skills in the **Visual Basic** programming language.
- To develop and strengthen general **problem solving skills** and logical thinking through the writing and understanding of well-structured computer programs

Course Objectives (2):

▫ Important Note

- This course will involve a considerable amount of "simple" programming
- You won't learn much in this course by only attending lectures. Effective problem solving using a programming language requires a lot of practice
 - Using Visual Basic either at home on your computer or in the Glade lab. It is unlikely that you will achieve even an average grade without doing the lab work and assignments conscientiously.

Course Objectives (3):

▫ Important Note (cont.)

- The "key" to learning how to program and to becoming a good programmer is actually very simple!

Practice, Practice, Practice,
Practice and more Practice
and then, more Practice!

Course Content ("Roadmap") (1):

• **Topic A:**

- Introduction to Problem Solving and Visual Basic
 - Computer based problem solving & algorithms
 - Introduction to Visual Basic and programming → the IDE (Integrated Development Environment) - forms and controls, basic concepts of classes and objects (object properties), events and programming responses to events, types of programming errors (syntax, runtime, logic)

Course Content ("Roadmap") (2):

• **Topic B:**

- Variables, data types and expressions
 - Using the computer's memory
 - Representing data and program instructions → machine language and assembly language; high-level programming languages and translation
 - Declaring variables and constants
 - Operators, expressions, and assignment
 - Use of Visual Basic functions (data type conversion functions, date functions)
 - Controlling focus

Course Content ("Roadmap") (3):

- **Topic C:**
 - Control structures - selection
 - Control structures → boolean values, conditions and comparison operators
 - if statements → Boolean operators, nested if statements
 - Using CheckBox and Option controls
 - Using a MessageBox
 - Validation of input using the Validation event

Course Content ("Roadmap") (4):

- **Topic D:**
 - Control structures - iteration
 - Repetition → initialization, repeated statements (body), termination
 - Counted and conditional loops
 - Manipulating string data using iteration
 - String functions
 - The List Box control

Course Content ("Roadmap") (5):

- **Topic E:**
 - Subprograms - functions and procedures
 - Information hiding and abstraction
 - Modular design → coupling and cohesion
 - Communicating data → global data versus arguments
 - Parameter passing → by value and by reference
 - Scope of variables
 - The ComboBox control.

Course Content ("Roadmap") (6):

- **Topic F:**
 - External files and databases - using classes & objects
 - Files, records and fields
 - Reading and writing files → Common Dialog control, the FileSystemObject and TextStream classes
 - Simple error handling
 - Using multiple forms
 - Creating a database from a text file → creating our own class to act as a DataSource class; the RecordSet class, the BindingCollection, Binding and DataBinding classes, creating objects of a class

Course Lab Work (1):

- **The Glade Lab**
 - A laboratory of workstation computers located in two rooms (CB160 & CB161) in the Chemistry Building (CB)
 - Open from about 8am to 11pm weekdays / weekends → visit the lab for the exact scheduling of hours
 - ~50 machines in lab, which is also used by students in COSC1520 → first come first served basis
 - No method for you to reserve use of a machine
 - It is your responsibility to make sure that you can complete the course work!
 - Congestion in the lab is NOT an acceptable excuse for handing in lab work late.

Course Lab Work (2):

- **Lab Exercises**
 - Your textbook contains many exercises for you to work on → is strongly suggested you do actually work on them even though you can choose not to submit them!
 - Your textbook is more like a "workbook" and is not intended to be followed as a typical textbook but rather, go through the examples it provides
 - The online version contains many animations etc. illustrating the concepts it is describing

Obtaining Help (2):

- **Various ways to obtain help**
 - The TA will hold office hours in the Glade laboratory
 - Instructor office hours → please take advantage of the opportunity to approach me!
 - Instructor will also respond to e-mail questions at times other than his/her office hours, although the reply will necessarily come after a short delay → make use of e-mail!
 - Instructor may also occasionally drop by the lab → I will try and announce this before I do

Introduction to Programming

Intro. to Computer Programming (1):

- **What is an Algorithm ?**
 - A set of step-by-step instructions for accomplishing a particular task
 - The task may be anything → a common example of an algorithm is a recipe



Intro. to Computer Programming (2):

- **Algorithm Kitchen Analogy**
 - Think of the hardware comprising a computer as a restaurant kitchen → it is equipped to produce anything the customer (user) requests but sits idle until an order (command) is placed
 - The chef in our kitchen serves as the CPU, waiting for requests to come in → when someone does provide an order, (lets say for French toast) the chef responds by following the instructions of the appropriate recipe (algorithm)

Intro. to Computer Programming (3):

- **Algorithm Kitchen Analogy (cont.)**
 - The recipe itself is the software → provides instructions telling the hardware what to do to produce the desired output
 - If recipe is correct, the desired dish is produced (e.g., French toast)

Intro. to Computer Programming (4):

- **Algorithm Kitchen Analogy (cont.)**

- If the recipe (software) is unclear or incorrect or contains **bugs** (errors) then output may not be what user wanted!
- For example, recipe to the right is vague → do we include egg shells - unclear!



Intro. to Computer Programming (5):

Algorithm Kitchen

Analogy (cont.)

- A more detailed recipe (algorithm) for making French toast
 - More specific and less prone to errors
 - Less ambiguous

SUZANNE'S FRENCH TOAST FANTASTIQUE: THE ALGORITHM

1. Prepare the batter by following these instructions.
 - 1a. Crack 2 eggs so whites and yolks drop in bowl; discard shells.
 - 1b. Beat eggs 30 seconds with wire whisk, fork, or mixer.
 - 1c. Mix in 1 teaspoon vanilla extract, 1/2 teaspoon cinnamon, and 1/4 cup milk.
2. Place 1 tablespoon butter in frying pan and place on 350° heat.
3. For each of six pieces of bread, follow these steps:
 - 3a. Dip slice of bread in mixture.
 - 3b. For each of the two sides of the bread do the following steps:
 - 3b1. Place the slice of bread in the frying pan with this (uncooked) side down.
 - 3b2. Wait 1 minute and then peek at underside of bread; if lighter than golden brown, repeat this step.
 - 3c. Remove bread from frying pan and place on plate.
4. Serve bread with maple syrup, sugar, or tart jelly.

Intro. to Computer Programming (6):

What is Computer Programming ?

- Process of creating a computer program
 - A sequence of instructions to enable the computer to do something e.g., to perform a particular task, whatever the task may be
 - Computer programs generally start as algorithms → e.g., step-by-step human readable instructions for accomplishing a particular task
 - Programmer's job is to convert the algorithm into a program by adding details, testing, hammering out "rough spots" etc.

Intro. to Computer Programming (7):

Why Program Computers ?

- Although computers are extremely fast and seem smart at performing certain operations it is up to us to tell the computer what to do
 - Computers are actually not smart at all and very limited → capable of performing only the most basic arithmetic operations (e.g., $7 + 13 = 20$) and some logical comparisons (e.g., is x less than y)
 - Computers *seem* smart because they can perform these arithmetic operations and logical comparisons *quickly* and very *accurately*

Intro. to Computer Programming (8):

▪ **Why Program Computers ? (cont.)**

- Amazingly, everything we have seen a computer do is the result of a sequence of extremely simple arithmetic and logic operations done very quickly
- The challenge of computer programmers is to come up with the instructions to put those simple instructions together in ways that are useful and appropriate
