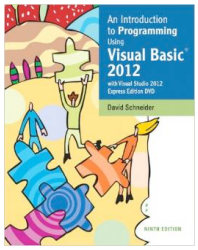
Elementary Programming in Visual Basic
CS 142 - Fall 2014
Credits: 3 hours

Instructor: David Furcy **Office:** Halsey 221
Email: furcyd@uwosh.edu **Phone:** 424-1182
Office Hours: MWF 1:30-2:30PM, T 10:00-11:00AM, R 10:00-noon, or by appointment

Class Meetings: Lectures: HS 367 9:10-10:10AM on Wednesdays and Fridays
Labs: HS 101C 9:10-10:10AM on Mondays

Prerequisites: Math-103 with grade C or better, or qualifying for either Math-104 or Math-171 via math placement exam

Required Textbook:



An Introduction to Programming Using Visual Basic 2012
Ninth Edition
David I. Schneider
Prentice Hall, 2014

Exams: All exams are closed book and may include a lab component.
Exam #1: Week of October 6
Exam #2: Week of November 3
Exam #3: Week of December 8

Note: If you have special needs, please come and talk to me at the end of the first class.

Course Overview: In this course, students will learn how to write programs that the computer can execute to solve given problems. The solution to a problem is formulated as an *algorithm*, which is an unambiguous sequence of instructions that, given an appropriate input, outputs the correct answer in a finite amount of time. A *computer program* is the concrete implementation of an algorithm. Therefore, computer programming involves several steps, from clearly specifying the problem at hand, to decomposing it into sub-problems, solving each sub-problem and putting the component solutions together into an algorithm, and finally, converting the algorithm into a working, well-documented program. This complex process is called the *program development cycle*.

In this course, we will describe and apply the program development cycle to concrete examples. We will also identify the fundamental building blocks of algorithms, such as sequencing, repetitions, decisions, modules, input, and output. Finally, we will learn the fundamentals of a programming language called Visual Basic 2012 with which we will develop our programs.

Course Goals: As a service course in computer programming for non-computer science majors, this course has two main sets of goals. First, students will:

- Acquire technical knowledge pertaining to the program development cycle and its stages, namely problem analysis, algorithm design, interface design, coding, testing and debugging, and documenting.
- Develop the technical skills of designing and implementing computer programs.

Second, by learning how to program a computer, students will develop various skills that are indispensable in the context of the liberal arts education promoted by the College of Letters and Science. Programming is an essentially human-centered practice. Programs are typically written by people for people.

- From the perspective of the user, the program is reduced to its interface, which must therefore take into account the strengths and limitations of human cognitive processes. A successful programmer must know the psychology of the users.
- From the perspective of the programmer's colleagues, the program is an artifact whose complexity can only be mitigated with clear, complete yet concise documentation. The successful programmer must have excellent writing skills.
- From the perspective of the programmer him- or herself, the program development cycle is a complex, multi-stage task that cannot be successfully completed without good time-management skills. Additionally, a successful programmer develops strong and diverse problem solving skills needed for each stage in the program development cycle:
    - Problem analysis stage: analytical and hierarchical thinking, as well as dealing with abstraction.
    - Algorithm design stage: rational, goal-directed thinking.
    - Coding stage: attention to details.
    - Testing and debugging stage: critical and systematic thinking.

Finally, program development is a cyclic process (hence its name) that often requires several iterations. In short, computer programming is a challenging but rewarding activity that not only fosters rational thinking but also teaches the essential virtue of perseverance.

We will be achieving all of these goals in a friendly, nurturing environment, through a variety of concrete, relevant and, hopefully, entertaining assignments.

Course Objectives: The main objectives are, for a student, upon completion of this course, to be able to:

- Apply the program development cycle to solve computational problems.
- Represent algorithms using flowcharts, pseudocode, and hierarchy charts.
- Design and implement a graphical user interface using VB 2012 forms and controls.

- Write algorithms that use loops and make decisions.
- Manipulate arrays in VB 2012 programs.
- Design modular code with Sub Procedures and Functions.
- Write event handlers that react to user input.
- Create, open, read from, and write to sequential files.
- Handle exceptional cases using "Try…Catch" blocks.

Topic Coverage: We will cover the following topics:

- Introduction to computers and the Windows interface.
- Problem solving
- Fundamentals of Visual Basic 2012
- Procedures: Subprograms and Functions
- Decisions: "If" and Select Case" blocks
- Repetition: "Do" and "For…Next" loops
- Arrays
- Sequential files
- Additional controls (if time permits)
- Object-oriented programming (if time permits)

Course Grading Policy: Your final grade for this course will be based on several components, namely weekly labs, assignments, exams, and quizzes. Your overall numerical grade for the course will be computed as the weighted sum of the component grades using the following weights:

Component	Weight
Unannounced quizzes	10%
Weekly labs	15%
Assignments	30%
Exam #1	15%
Exam #2	15%
Exam #3	15%

While this overall grading scheme is fixed, I will be happy to discuss any issue you may have with individual grades. If you notice a mistake or have a question regarding a specific grade, please come and talk to me *as soon as possible*. Do not wait until the end of the semester to bring up grading issues. Also, I will *not* be available to discuss grades after the end of the final week. Finally, your letter grade for the course will be computed using the following mapping:

Numerical Score	Course Grade	Numerical Score	Course Grade
≥ 92	A	≥ 72	C
≥ 90	A-	≥ 70	C-
≥ 88	B+	≥ 68	D+
≥ 82	B	≥ 62	D
≥ 80	B-	≥ 60	D-
≥ 78	C+	< 60	F

Attendance and Participation: You are expected to not only attend **every** class meeting but also to come **prepared** for and **participate** actively in it.

Necessary preparation requires you to have studied and assimilated the material covered in previous sessions, to have met with me or the tutors outside of class to discuss any questions you may have, to have done the assigned reading (when applicable), and to have completed the programming assignments on time. I will use pop quizzes to test your level of preparation.

Actively participating includes asking and answering questions in class, doing well on the pop quizzes, and demonstrating your interest for the course by discussing it with me outside of class.

It is hard to imagine how a student could do well in this course while missing classes, attending them unprepared, or not participating.

On the positive side, I have high expectations for my students and will always support and encourage you. I **strongly encourage** you to **ask any question** or raise any issue you have with the course either during or at the end of class, or during my office hours. I will also gladly meet with you by appointment. Send me email to make an appointment. While I will meet with you as soon as my schedule permits, do not expect me to be widely available before an exam or when an assignment is due.

Late Submissions: I will describe the submission procedure for your labs and assignments when the time comes. However, let me point out right away that each one of them will come with a deadline (day and time) after which any submission will be considered late. The late-submission policy works as follows:

Turned in	Penalty
On the due date but after the deadline	10%
One day after the due date	20%
Two days after the due date	40%
Three days after the due date	60%
Four days after the due date	80%
More than four days after the due date	100%

Note that labs and assignments that are more than four days late receive no points. Weekend days and holidays DO count as "regular days" when computing late penalties. Each (late) day starts precisely at midnight. Extensions on labs and assignments may be granted at the discretion of the instructor if you provide a valid justification (in the form of a written excuse from a medical doctor or the Dean of Students Office) before the due date. Late submissions can easily be avoided by starting to work on the assignment right away and asking for help early if you get stuck.

If you miss a scheduled exam, you **may** be able to take a make-up exam provided you give me a valid justification (see above) ahead of time if possible. Only one make-up exam will be given. It will be a comprehensive exam scheduled at the end of the semester.

Collaborating versus Cheating: Unless otherwise stated in the assignment or lab, all submissions must be the work of a single student (the one whose name appears on the submission, that is). While it is acceptable and encouraged to discuss the assignments with others, you must submit your own work. You may not "borrow" any piece of code of any length from someone else, unless you can live with a zero and the other potential academic sanctions of cheating (see the UWO Student Discipline Code, Chapter UWS 14).

Final Note: Computer science classes require significant work. It takes considerable practice to develop the programming and analytical skills outlined in this syllabus. I expect that you will need to spend **at least three hours of effort outside of class for each in-class hour**.

I expect every committed and hardworking student to do well in this course. I am looking forward to a fun and rewarding semester together. Given my high standards, I could not be more satisfied than if everyone earned an A in this course.