# Option Strict

**Author:** Keith Alcock 10/07/2008 02:11:00 AM MDT



## VITAL INFORMATION

**Subject(s):** Careers, Computer Fundamentals 1-2

**Topic or Unit of Study:** Software Development

**Grade/Level:** 9-12

**Objective:** At the conclusion of this lesson students will be able to:

1. Write a program without relying on or resorting to implicit type conversions.

2. Explicitly convert between Integers, Doubles, and Strings.

3. Avoid unnecessary double conversions resulting from use of Val.

**Summary:** Students update the programs for one of their recent activities, likely to be the Regular Paycheck, to avoid all use of implicit type conversions. They achieve this by disabling support for implicit conversions (turning option strict on), waiting for Visual Basic to complain that the conversions it had been performing are no longer allowed, and then going through the resulting error messages adding explicit conversions CInt, CDbl, and CStr. The compiler continues type checking and when it no longer complains, the work is complete.

## IMPLEMENTATION

**Learning Context:** Students have just completed a number of calculators and paycheck programs. In assessing them, I noticed many gratuitous uses of Val and also reliance on implicit type conversion where Val had been neglected. Programs are working by accident rather than by design, and students need to know more about type. In this activity the simple calculator and have previously watched the Shift Happens video. The calculator program gets shifted in this exercise. Based on fictitious feedback from users, additional functionality has been specified. We're putting some of the soft in software and experiencing how small existing modules can be combined in different ways to produce new results.

**Procedure:** 1. The activity is written up in detail on a web page which is printed and attached to this lesson plan. Although students can read the instructions on their computers, they probably have questions and comments that interest the entire group. We therefore generally conduct such activities at the front of the room near the screen where everyone can see and hear. Such is the plan for this activity.

2. Work with the students through the background information, instructions, and their questions, and then set them to work on

updating one of their programs.

3. This activity is for once one that the computer can check. If the compiler finds no implicit conversions, then there aren't any. The teacher can therefore simply verify that the strict option is turned on and that the list of error messages is empty. Such is the assessment. Furthermore, students can continue on and complete their work in progress, since this activity will not render their program incompatible with what it was being used for previously.

**Differentiated Instruction:** There is little differentiation in this assignment. Students may work at their own pace as soon as group discussion is complete. Since they will be updating their own work, each student's project will start and end differently than their peers' projects.

**Sample Student Products:** The web page gives a before and after example for the teacher's Homework Calculator which should illustrate for the students what needs to be done. Their work should resemble it highly.

**Collaboration:** Students will work individually.

**Time Allotment:** 1 class period. 25 Min. per class.

**Author's Comments & Reflections:** Reflections will follow in a diary entry.

Re. Time Allotment: These 25 minutes may be recovered when future assignments are produced more quickly because of the techniques students learn here.

## MATERIALS AND RESOURCES

**Instructional Materials:** The web page for this activity is attached.

**Attachments**

1. **Option Strict**

**Resources:**

- Technology resources:
  Internet Explorer, Visual Basic

## STANDARDS & ASSESSMENT

**Standards:**

**AZ- Career and Technical Education Programs**

- **Level :** Career Preparation (Grades 10 - 12)
  - **Program :** Information Technology CIP No. 15.1200
    - **Option :** Software Development - Option C
      - **Competency :** 27.C DEMONSTRATE PROGRAM ANALYSIS AND DESIGN
        - **Indicator :** 27.5c Choose appropriate data structures
        - **Indicator :** 27.6c Use stepwise refinement to improve design
      - **Competency :** 28.C USE SOFTWARE TO CREATE PROGRAMS
        - **Indicator :** 28.1c Enter and modify code using a program editor
        - **Indicator :** 28.2c Compile and execute programs

- **Indicator :** 28.3c Correct syntax errors
- **Indicator :** 28.6c Employ debugging strategies to eliminate errors

• **Competency :** 31.C EMPLOY MODULARITY IN WRITING PROGRAMS

- **Indicator :** 31.1c Call standard library functions
- **Indicator :** 31.4c Write and use modules that return values

• **Competency :** 34.C USE SIMPLE DATA TYPES AND STRINGS

- **Indicator :** 34.1c Declare numeric, Boolean, character and string variables
- **Indicator :** 34.4c Write assignment statements for initializing and modifying variables

**Assessment/Rubrics:** The compiler assesses the resulting program and the teacher needs only to verify that the option is turned on and that the error list is empty. It is not planned to use scores from this activity on the report card, so no formal accounting of all the points is necessary.