# Overview

In this exercise you will modify the **Orders.dbv** program that you created in [Exercise – 1](http://jobfunc2.cu.net/Job%20Functions/Programmer/Programmer%20Handbook/Synergy%20Training%20-%20Synergy%20Language%20Essentials/Exercise%20-%201.docx). You will now begin to process order detail lines for new orders. First you will create a window in which you will process the detail fields. Once this basic structure is complete you will introduce dynamic memory into the application. The requirement is to allow the user to enter as many detail lines onto an order as they want, you will start off supporting a maximum of three lines, but if the requirements exceed this, you will resize the dynamic memory to allow more to be entered.

# Resources

* [Synergy Best Practices - Coding Standards](http://jobfunc2.cu.net/Job%20Functions/Programmer/Programmer%20Handbook/Tims%20Best%20Practices%20-%20Standards/Synergy%20Best%20Practices%20-%20Coding%20Standards.docx)

# Exercise

1. Using Visual Studio, open the previously created “Orders” project.
2. Using Visual Studio, open “Orders.dbv”.
3. Create a new internal subroutine called pAddOrderDetail.
4. Amend the pAddOrders routine to call the pAddOrderDetail routine, once the order header details have been entered.
5. Compile, link, and run the program.
6. Change the order detail include file from a RECORD to a STRUCTURE.
7. In the startup routine, allocate dynamic memory sufficient to store three order detail lines. You will need to create local work variables for the memory handle and to keep track of how many order detail lines can be stored in the dynamic memory as well as how many are currently stored.
8. In the pShutDown routine, de-allocate the dynamic memory.
9. Define parametric macros that allow a whole detail record, or an individual field in the detail record to be referenced.
10. In the pAddOrderDetail routine, change the calls to DataInput to pass the dynamic memory references (using the parametric macros) instead of the old field names. Don’t forget to store away the current order number and detail line number in each detail record as it is entered.
11. When an order line has been entered, go around for another. If the user enters a blank stock code then exit from the loop. Keep track of the number of detail lines you have entered and make sure you don’t exceed the amount of memory currently allocated!
12. Alter the program to re-size the dynamic memory if the number of line items entered exceeds the amount of memory currently allocated. Each time you re-size, allocate additional memory for another three detail lines.
13. Compile, link, and run the program.

# Discussion

Again, this is a complex exercise, which should be approached in small parts.