Summary of Ada95 Code

Mark Allen Weiss
School of Computer Science
Florida International University
University Park
Miami, FL 33199

Abstract

This handout lists the source files for the Ada95 code that updates the Ada83 code in my text *Data Structures and Algorithm Analysis in Ada*, (Benjamin/Cummings, 1993). Included in this handout is a brief list of Ada95 changes.

1 Chapter 1

- `fig1_2.adb`: a simple recursive routine with a test program
- `fig1_3.adb`: an example of infinite recursion
- `fig1_4.adb`: recursive routine to print numbers, with a test program
- `fig1_5.adb`: program that counts the number of lines in a file
- `complex_numbers.ads`: package specification for complex numbers
- `complex_numbers.adb`: package implementation for complex numbers
- `complex_numbers_test.adb`: test program for complex numbers
- `error_pack.ads`: package specification for error routines
- `error_pack.adb`: package implementation for error routines

A few Ada95 features are introduced in the new code. They are:

1. Compilation unit `X` must be placed in a source file named `X.adb`. There is a limit of one compilation unit per source file.

2. `Text_IO` is now `Ada.Text_IO`. The same goes for the package `Integer_Text_IO`.
3. For packages, the specification goes in a file with suffix `.ads`; the body goes in `.adb`.
4. The trivial package `Error_Pack` is provided.

Read the supplement: *Using GNAT on a Unix system* provided as part of the courseware.

2 Chapter 2

- `fig2_5.adb`: $O(n^3)$ maximum subsequence sum algorithm
- `fig2_6.adb`: $O(n^2)$ maximum subsequence sum algorithm
- `fig2_7.adb`: $O(n \log n)$ maximum subsequence sum algorithm
- `fig2_8.adb`: $O(n)$ maximum subsequence sum algorithm
- `fig2_9.adb`: Test program for binary search
- `binary_search.ads`: Specification of generic function `Binary_Search`
- `binary_search.adb`: Implementation of generic function `Binary_Search`
- `fig2_10.adb`: Euclid's algorithm, with a test program
- `fig2_11.adb`: Recursive exponentiation algorithm, with a test program

Once again, not much is new.

1. Complete programs are provided.
2. `Binary_Search` is written as a generic.

3 Chapter 3

- `linked_lists.ads`: Package specification for linked list
- `linked_lists.adb`: Package implementation for linked list
- `linked_lists_test.adb`: Test program for linked list package
- `polynomials.ads`: Package specification for polynomial

- polynomials.adb: Package implementation for polynomial
- polynomials_test.adb: Test program for polynomials
- cursor_lists.ads: Package specification for cursor linked list
- cursor_lists.adb: Package implementation for cursor linked list
- stack_array.ads: Package specification for stack: array version
- stack_array.adb: Package implementation for stack: array version
- stack_list.ads: Package specification for stack: list version
- stack_list.adb: Package implementation for stack: list version
- stack_test.adb: Test program for stacks
- queue_array.ads: Package specification for queue: array version
- queue_array.adb: Package implementation for queue: array version
- queue_test.adb: Test program for queues

The most important change in this chapter is the use of the package Ada.Finalization to achieve the equivalent of the C++ constructor and destructor. There are some other minor changes.

1. Make_Null is changed to Make_Empty.
2. Lists and stacks that use dynamically allocated memory are now derived from the abstract tagged type Ada.Finalization.Limited_Controlled. This changes the implementation in mostly trivial ways. For example, L.Next is replaced by L.Header.Next in the list code. A procedure for Initialize and Finalize is also written for these packages.
3. "=" is now overloadable, and so it is used in place of Equals.

Read the supplement: *Using Finalization in Ada95* provided as part of the courseware.

Read the supplement: *Using Inheritance for Heterogeneity* provided as part of the courseware.

Also, as an exercise, make the changes to Cursor_Lists to allow automatic reclamation of resources.

4 Chapter 4

- `search_tree_package.ads`: Package specification for binary search tree
- `search_tree_package.adb`: Package implementation for binary search tree
- `search_tree_package_test.adb`: Test program for binary search trees
- `search_avl.ads`: Package specification for AVL tree
- `search_avl.adb`: Package implementation for AVL tree
- `search_avl_test.adb`: Test program for AVL trees

The main change here is the incorporation of `Ada.Finalization`. The changes are:

1. Type `Search_Tree` is no longer merely an access type. It is now a derived type from `Ada.Finalization.Limited_Controlled`.
2. Consequently, for `Tree_Node`, the `Left` and `Right` fields are of type `Tree_Ptr`.
3. Consequently, the recursive calls use a nesting technique: The dispatching routine, such as `Find` calls an internal recursive `Find`, passing the `Root` as a parameter.
4. Dispatching routines must be declared in the package specification. Thus, in the `Search_Tree` package, function `Height` is declared in the specification.

5 Chapter 5

- `open_hash_table.ads`: Package specification for separate chaining
- `open_hash_table.adb`: Package implementation for separate chaining
- `open_hash_table_test.adb`: Test program for separate chaining
- `closed_hash_table.ads`: Package specification for quadratic probing hash table
- `closed_hash_table.adb`: Package implementation for quadratic probing hash table
- `closed_hash_table_test.adb`: Test program for quadratic probing hash tables
- `expanding_closed_hash_table.ads`: Package specification for quadratic probing hash table with rehashing

- `expanding_closed_hash_table.adb`: Package implementation for quadratic probing hash table with rehashing

More of the same. Finalization and the use of `"="` are the major changes. The code for quadratic probing fixes a bug in the textbook.

6 Chapter 6

- `binary_heap.ads`: Package specification for binary heap
- `binary_heap.adb`: Package implementation for binary heap
- `leftist_heap.ads`: Package specification for leftist heap
- `leftist_heap.adb`: Package implementation for leftist heap
- `priority_queue_test.adb`: Test program for priority queues

More of the same. The major change is finalization, and the recursion makes the changes more complex for the leftist heap. Most importantly, `Merge` is now handled more carefully. We need to guarantee that every node is in exactly one tree, or else finalization will double-dispose.

7 Chapter 7

- `sorters.ads`: Package specification for a collection of sorting and selection routines
- `sorters.adb`: Package implementation for a collection of sorting and selection routines
- `sort.adb`: A test program for the `Sorters` package

No changes.

8 Chapter 8

- `disjoint_sets.ads`: Package specification for disjoint sets
- `disjoint_sets.adb`: Package implementation for disjoint sets
- `disjoint_sets_test.adb`: A test program for disjoint sets

No changes.

9 Chapter 9

Graphs are not implemented because only pseudocode is in the text.

10 Chapter 10

- `fig10_38.adb`: Simple matrix multiplication algorithm with a test program
- `fig10_40.adb`: Algorithms to compute Fibonacci numbers
- `fig10_43.adb`: Inefficient recursive algorithm (see text)
- `fig10_45.adb`: Better algorithm to replace `fig10_43.adb` (see text)
- `fig10_46.adb`: Dynamic programming algorithm for optimal chain matrix multiplication, with a test program
- `fig10_53.adb`: All-pairs algorithm, with a test program
- `random_numbers.ads`: Package specification for uniform random numbers
- `random_numbers.adb`: Package implementation for uniform random numbers
- `random_numbers_test.adb`: Test program for random number package
- `fig10_62.adb`: Randomized primality testing algorithm, with a test program
- `tic_tac_alpha.adb`: Tic-tac-toe with alpha-beta pruning, with a marginal test program

Changes are minor.

1. The constants for the random number generator are changed slightly because of new research results.
2. A complete tic-tac-toe program with a skeleton testing driver is provided.

11 Chapter 11

No material is here.

12 Chapter 12

- `splay_tree_package.ads`: Package specification for top-down splay tree
- `splay_tree_package.adb`: Package implementation for top-down splay tree
- `splay_test.adb`: Test program for splay trees
- `red_black_tree_package.ads`: Package specification for top-down red black tree
- `red_black_tree_package.adb`: Package implementation for top-down red black tree
- `red_black_test.adb`: Test program for red black trees
- `aa_tree_package.ads`: Package specification for AA-tree
- `aa_tree_package.adb`: Package implementation for AA-tree
- `aa_test.adb`: Test program for AA-trees
- `pairing_heap.ads`: Package specification for pairing heap
- `pairing_heap.adb`: Package implementation for pairing heap
- `pair_test.adb`: Test program for pairing heaps

Chapter 12 is present in the second edition of the Pascal version. For compatibility, four generic packages are written that implement the following data structures:

1. Top down splay trees
2. Top down red black trees
3. AA-trees
4. Pairing heaps