A Set of Programming Projects for a Second Programming Course

Kailash Chandra

Computer Science - Information Systems
Pittsburg State University
Pittsburg, KS 66762

Abstract

A set of six related programming projects are presented. These projects are based on the assignments given to a class taking a second course in Pascal programming. The first project started with a source code of a working program assigned to the students giving them a jump start and then taking them step by step to a project where they developed a simple full-screen visual text editor. It has been a very successful experiment. It is hoped that the others can use similar programming assignments in their classes.

Introduction

The purpose of this paper is to describe six related programming projects for a second course in Pascal programming. These are based on a course titled "Principles of Software Design" which the author taught in the Spring semester of 1992 at Pittsburg State University in the Computer Science - Information Systems Department. A book written by Sahni[1] was used as a textbook for the course and several other books were recommended to supplement it [2-8]. The purpose of the first project was to give the students a jump start by asking them to type in a working simple text-viewing program to display text stored in a dense list and make it work. The entire set of projects consisted of six assignments. The sixth project resulted into a simple full-screen visual text editor. The projects were required to be written in Turbo Pascal but could easily be written in any other language such as C, C++, Modula-2, Ada, or QuickBASIC.

A Summary of Assignments

The first assignment for the students was to type in a complete Pascal program from the hard copy provided to the students. This was a working program which displayed text in a window on the screen supporting Home, End, PgUp, PgDn, UpArrow, DownArrow, and other edit keys. Some of the important concepts that the students learned from this assignment were: defining constants, detecting control keys, passing arrays as parameters, and passing value and variable parameters. The 400-line program with several procedures and functions was explained line by line in class.

In the second assignment, the students were asked to write at least ten functions or procedures out of a list of 28 to handle a singly-linked list of string values: from initializing a list to displaying and disposing it. Though, only ten of the procedures were required to be done, all the procedures and functions were explained in class. The purpose of this assignment was to give the students a good understanding of pointer variables, memory allocation, and memory deallocation. Some of the students tried to save the pointer information to the file also.

The third assignment was to create a doubly-linked list, display it, save it to a file, dispose it freeing the memory taken by it, and load it from the file. The purpose of this assignment was to give the students a good understanding of doubly-linked lists.

In the fourth assignment, the students were asked to do everything they did in the first assignment except that they had to use a doubly-linked list instead of a dense list to store the text. This resulted into a program that allowed one to view a text file with vertical scrolling capabilities while making use of the doubly-linked list structure to store text and all the available memory.

The fifth assignment was to write a single-line editor supporting LeftArrow, RightArrow, Home, End, and other edit keys. In this assignment, the user was supposed to input a string of characters of up to a maximum specified length and edit the information if necessary using several edit keys.

In the last assignment, the students were asked to combine the fourth and the fifth assignments so that not only could the text file be viewed but it could also be modified with additional features such as deleting a line, inserting a new line, dynamically allocating memory when a line is inserted, and releasing memory when a line is deleted. In this assignment, the following features were also added: splitting a line into two lines when the Enter key is pressed in the insert mode and combining two lines into one line when the BackSpace key is pressed at the beginning of a line. The result was a full-screen visual editor with vertical scrolling, line inserting, line deleting, and other simple text-editing features.

First Assignment

Type in the Pascal program from the hard copy (provided to the students). This is a working program to display text in a window on the screen supporting several edit keys for vertical scrolling. A brief description of all the functions and procedures is as follows:

```
function CopiesChar(Count: integer;
   c: char): string;
   {Returns a string with Count copies of a specified
   character}
function KeyCodeI: integer;
   {Returns a unique integer code for each key
   pressed}
procedure Beep;
   {Makes a beep for 50 ms
procedure DisplayList(r, c, wid,
   deep: integer; Border: string; fa:
   StringArray; fc: integer; var
   Selected: integer);
   {Displays a list of items on the screen and allows
   you to scan through it, where r is the row for the
   left upper corner of the window, c is the column for
   the left upper corner of the window, wid is the
   width of the items in the list, deep is the depth of
   the window, Border is the six-character string
   defining the border for the window, fa is the array
   of items to be displayed, fc is the number of items
   in the list, Selected is the item number of the
   focused item when the Enter key was pressed or
   zero if the escape key was pressed to exit}
procedure DrawBox(r1, c1, r2, c2:
   integer; Border: string);
   {Draws a box on the screen where Border is
   defined by a string of six characters '┌─┐│┘└'}
procedure Locate(Row, Col: byte);
   {Moves the cursor to Row and Col position on the
   screen}
procedure NormalVideo;
   {Changes to normal video mode, white on black}
procedure PrintPadR(s: string; w:
   byte);
   {Prints the string in s with trailing blanks if
   necessary, it also adds a leading blank and a
   trailing blank}
procedure ReverseVideo;
   {Changes to reverse video mode, black on white}
type StringArray = array [1..100] of
   string[80];
   {StringArray type definition}
```

The program allows the following keys with corresponding movement/action:

Command	**Movement/Action**
Ctrl-PgDn	Last item of the list
Ctrl-PgUp	Beginning of the list
DownArrow	Next item in the list
End	Last item in the window
Enter	Select the highlighted item
Esc	Exit without selecting
Home	First item in the window
PgDn	Next page/window
PgUp	Previous page/window
UpArrow	Previous item in the list

Second Assignment

Implement any ten of the following linked list manipulation procedures and/or functions. Use meaningful variable names, document your code, submit the source code, and a sample run with enough cases to show that your program works most of the time.

```
type ptrType = ^NodeType;
  {Type declaration for the pointers}
type NodeType = record
  begin
  info: string;
  next: ptrType;
  end;
  {Type declaration for the list items}
function sllAt(head:ptrType;
  position: integer): string;
  {Returns the item at a position in the list, position
  of the first item is 0}
function sllCount(head: ptrType):
  integer;
  {Returns the number of items in a list}
function sllIndexOf(head: ptrType;
  value: string);
  {Returns the position of a given value}
procedure sllCheck(head: ptrType);
  {Checks the integrity of a list for count, circular
  references, and other possible errors}
procedure sllCombine(head1, head2:
  ptrType; var head3: ptrType);
  {Creates a list that has all the items in the two lists
  including the duplicates}
procedure sllDifference(head1, head2:
  ptrType; var head3: ptrType);
  {Creates a list of items from the first list that are
  not in the second list}
procedure sllDisplayAll(head:
  ptrType);
  {Displays all the items in a list}
procedure sllDisplayAllAcross(head:
  ptrType; across: integer);
  {Displays all the items in a specified columns}
procedure sllDisplayAt(head: ptrType;
  position: integer);
  {Displays the item at a given position}
procedure sllDisplayDistinct(head:
  ptrType);
  {Displays all the distinct values in a list}
procedure sllDisposeAll(var head:
  ptrType);
  {Disposes all the items from a list and releases the
  memory taken by them}
procedure sllDisposeAt(var head:
  ptrType; position: integer);
  {Disposes an item from a list at a given position}
procedure sllDisposeAtFromTo(var
  head: ptrType; position1,
  position2: integer);
  {Disposes all the items between the two given
  positions including the items at the positions}
procedure sllDisposeDups(var head:
  ptrType);
  {Disposes all the duplicate items}
procedure sllDisposeValue(var head:
  ptrType; value: string);
  {Searches and dispose the item that matches a
  given value from a list}
procedure sllDisposeValueAll(var
  head: ptrType; value: string);
  {Searches and dispose all the items that match a
  given value from a list}
procedure sllFillRandom(var head:
  ptrType; var count: integer);
  {Inserts a random number of random items in a
  list}
procedure sllInit(var head: ptrType);
  {Initializes a singly-linked list}
procedure sllInsert(var head:
  ptrType; value: string);
  {Inserts an item in the list at the tail of the list}
procedure sllInsertAt(var head:
  ptrType; value: string; position:
  integer);
  {Inserts an item at a specified position in the list}
procedure sllIntersect(head1, head2:
  ptrType; var head3: ptrType);
  {Creates a list of common items in two lists}
procedure sllLoad(var head: ptrType;
  fileName: string);
  {Creates a list of values from a text file}
procedure sllReplaceAll(head:
  ptrType; valueFor, valueWith:
  string);
  {Searches for all occurrences of a given value and
  replace them with another given value}
procedure sllReplaceAt(head: ptrType;
  position: integer; value: string);
  {Replaces the value of an item at a given position}
procedure sllSortBubble(var head:
  ptrType);
  {Sorts a list in ascending order using Bubble sort}
procedure sllSortOther(var head:
  ptrType);
  {Sorts a list in ascending order using any other
  sort method}
procedure sllStore(head: ptrType;
  fileName: string);
  {Saves the contents of a list to a text file}
procedure sllUnion(head1, head2:
  ptrType; var head3: ptrType);
  {Creates a list that has all the items from the two
  lists excluding the duplicates}
```

Third Assignment

Implement the following procedures. Use meaningful variable names, use proper indentations, document your code, and submit the source code and at least five sample runs to show that your procedures work most of the time.

```
type ptrType = ^NodeType;
   {Type declaration for the pointers}
type NodeType = record
  begin
  info: string;
  next: ptrType;
  prev: ptrType;
  end;
  {Type declaration for the list items}
procedure dllDisplay(head: ptrType);
  {Displays the contents of a doubly-linked list}
procedure dllDisplayReverse(tail:
  ptrType);
  {Displays the contents of a doubly-linked list in
  reverse order, from tail to head}
procedure dllDispose(var head, tail:
  ptrType);
  {Disposes the memory taken by the elements of a
  doubly-linked list}
procedure dllLoad(var head, tail:
  ptrType; fileName: string);
  {Loads the values from a text file into a doubly-
  linked list}
procedure dllFillRandom(var head,
  tail: ptrType; var count:
  integer);
  {Creates a doubly-linked list of random values}
procedure dllSave(head: ptrType
  fileName: string);
  {Saves the elements of a doubly-linked list to a text
  file}
procedure dllSort(head: ptrType);
  {Sorts the elements of a doubly-linked list}
```

Write a driver program that calls different procedures in the following order and displays the memory available after every procedure call:

FillRandom,
Display,
Sort,
Display,
Save,
Dispose,
Load,
Display,
DisplayReverse,
Dispose

Fourth Assignment

Write a text file viewer program in Pascal with the following features:

1. Use a doubly-linked list to load the text file contents.
2. Display the focused line in reverse video.
3. Display the memory available before the file is loaded, after the file is loaded, and after disposing the list.

Support the following keys:

Command	**Movement**
Ctrl-End	Bottom of window
Ctrl-Home	Top of window
Ctrl-PgDn	End of file
Ctrl-PgUp	Beginning of file
Ctrl-W	Scroll up one line
Ctrl-Z	Scroll down one line
DownArrow or Ctrl-X	Line down
Esc	Free memory and quit the program
PgDn or Ctrl-C	Page down
PgUp or Ctrl-R	Page up
UpArrow or Ctrl-E	Line up

Fifth Assignment

Write a procedure named EditLine in Pascal and a driver program to test it until the string 'Quit' is entered. Also write the necessary procedures and functions to support it.

Prototype of the EditLine procedure is:

```
procedure EditLine(maxLength: byte;
  oldLine: string; var newLine:
  string; autoRepeat: boolean; var
  exitCode: integer);
  {Edits a line where maxLength is the maximum
  number of characters allowed, oldLine is the line
  to be edited, newLine is the returned line,
  autoRepeat is a switch to allows auto repeat, and
  exitCode returns the code for the key which caused
  exit from the procedure}
```

Support the following keys:

Command	Function
BackSpace	Destructive backspace
Ctrl-LeftArrow or Ctrl-A	Move cursor left one word
Ctrl-Q Del	Delete all characters from the cursor position to the beginning of the line
Ctrl-QY	Delete all characters from cursor position to the right end of the line
Ctrl-RightArrow or Ctrl-F	Move cursor right one word
Ctrl-T	Delete word from cursor position to the right
Ctrl-Y	Wipe entire line
Del or Ctrl-G	Delete character at cursor position
DownArrow	Exit with new line and a code in exitCode
End or Ctrl-QD	Move cursor to right end of current line
Enter	Exit with new line and a code in exitCode
Esc	Exit with new line and a code in exitCode
Home or Ctrl-QS	Move cursor to beginning of current line
Ins or Ctrl-V	Turn insertion on or off
LeftArrow or Ctrl-H	Move cursor left one character
PgDn or Ctrl-C	Exit with new line and a code in exitCode
PgUp or Ctrl-R	Exit with new line and a code in exitCode
RightArrow or Ctrl-D	Move cursor right one character
Shift-Tab	Move cursor left one word
Tab	If in insert mode insert 8 blanks else move cursor right one word
UpArrow or Ctrl-E	Exit with new line and a code in exitCode

Sixth Assignment

Write a full-screen text editor program in Pascal with the following features:

1. Use a doubly-linked list to load the text file contents.
2. Display the focused line in reverse video.
3. Display the memory available before the file is loaded, after the file is loaded, after the list is saved, and after the list is disposed.
4. Describe the purpose of each procedure and function, give enough comments, use meaningful variable names where possible, and use proper indentations. Submit the printed source code, source code on a diskette and the compiled program.

Support the following keys:

Command	Function
BackSpace	If in the middle of a line then delete the character left to the cursor and move cursor to the left, otherwise move the current line at the end of the line before
Ctrl-End	Move cursor to the bottom of the window
Ctrl-Home or Ctrl-QE	Move cursor to the top of the window
Ctrl-KS	Save the doubly-linked list and continue editing
Ctrl-LeftArrow or Ctrl-A	Move cursor to the left one word
Ctrl-N	Insert a return, moving text to next line and leaving cursor where it was when Ctrl-N was pressed
Ctrl-PgDn or Ctrl-QC	Move cursor to the end of the file
Ctrl-PgUp or Ctrl-QR	Move cursor to the beginning of the file
Ctrl-Q Del	Delete all characters from the cursor position to the beginning of the line
Ctrl-QY	Delete all characters from cursor to the right
Ctrl-RightArrow or Ctrl-F	Move cursor right one word
Ctrl-T	Delete word from cursor position to the right
Ctrl-W	Scroll up one line
Ctrl-Y	Delete focused line
Ctrl-Z	Scroll down one line
Del or Ctrl-G	Delete character at cursor position

Command	Function
DownArrow or Ctrl-X	Move cursor one line down
End or Ctrl-QD	Move cursor to right end of current line
Enter	If in insert mode then insert a line carrying all the characters starting at the cursor to the new line else move cursor to the beginning of the next line
Esc or Ctrl-KD	Save the linked list, free the memory, and quit program
Home or Ctrl-QS	Move cursor to beginning of the line
Ins or Ctrl-V	Turn insertion on or off
LeftArrow or Ctrl-S	Move cursor left
PgDn or Ctrl-C	Page down
PgUp or Ctrl-R	Page up
RightArrow or Ctrl-D	Move cursor right one character
Shift-Tab	Move cursor left one word
Tab	If in insert mode insert 8 blanks else move cursor right one word
UpArrow or Ctrl-E	Move cursor one line up

Conclusion

It is felt that the projects in the given sequence are valuable. The students are exposed to a complete working program and are systematically allowed to add useful features until it ends up to be something useful. The projects also provide an opportunity to plan through the process of decomposition or modularization, a feature that clearly distinguishes the novice from the expert programmer. The projects also emphasize practice in building programs from previously existing modules by adding new modules.

References

[1] Sahni, Sartaj, Software Development in Pascal, The Camelot Publishing Company, Second Edition, 1989.

[2] Horowitz, Ellis and Sartaj Sahni, Fundamentals of Data Structures, Computer Science Press, 1976.

[3] Horowitz, Ellis and Sartaj Sahni, Fundamentals of Data Structures in Pascal, Computer Science Press, Second Edition, 1987.

[4] Knuth, Donald E., The Art of Computer Programming - Fundamental Algorithms, Addison-Wesley Publishing Company, Second Edition, 1973.

[5] Nance, Douglas W. and Thomas L. Naps, Introduction to Computer Science: Programming, Problem Solving, and Data Structures, West Publishing, Second Edition, 1992.

[6] Naps, Thomas L. and Bhagat Singh, Introduction to Data Structures with Pascal, West Publishing, 1986.

[7] Salmon, William I., Structures and Abstractions, Irwin, 1992.

[8] Santi, Barbara L., Lydia Mann, and Fred Zlotnick, Algorithms, Programming, Pascal, Wadsworth Publishing, 1987.