Universität Innsbruck - Institut für Informatik
Prof. Clemens Ballarin, Robert Binna, Friedrich Neurauter, Francois Scharffe and Sarah Winkler

15. April 2008

Proseminar Algorithmen und Datenstrukturen

Exercise sheet 4

Exercise 1 (Structs)

This exercise deals with records (structs) in C. Information how to use them can be found in chapter 6 from "Brian W. Kernighan, Dennis M. Ritchie: The C Programming Language." For each function write a small test program to verify the correct behaviour. Document your code and use meaningful identifiers.

a) Define a record that can be used to represent points. Create a function that given two points calculates the distance between those points.

b) Use the record defined in exercise 1.a) to define the record of a rectangle. Create a function that calculates the area of a rectangle.

   Hint: 2 points are sufficient to represent a rectangle whose edges are parallel to the axes of the coordinate system.

c) Create a function to calculate the smallest surrounding rectangle of two rectangles. The smallest surrounding rectangle of two rectangles is defined as the smallest rectangle that contains both rectangles. Use the structures defined in the exercises above.

   To find the smallest surrounding rectangle the solution finds the extreme coordinates of this rectangle. They are defined as the minimum x-,y-coordinate and the maximum x-,y-coordinate.

Algorithm 1 The Longest Upsequence

Input: $X[1 \ldots n]$ (Array of Integers, $n \geq 1$)
Output: Length k of the longest upsequence
$Y[1 \ldots k]$ (Array of Integers, representing the longest upsequence)

```
1:  begin
2:    k := 1;
3:    M[1][1] := X[1];
4:    for i := 2 to n do
5:      if X[i] ≥ M[k][k] then
6:        k := k + 1;
7:        M[k][k] := X[i];
8:        for j := 1 to k do
9:          M[k][j] := M[k − 1][j];
10:     else if X[i] < M[1][1] then
11:       M[1][1] := X[i];
12:     else
13:       bestimme j mit M[j−1][j−1] ≤ X[i] < M[j][j];
14:       M[j][j] := X[i];
15:       for l := 1 to j − 1 do
16:         M[j][l] := M[j − 1][l];
17:   Y := M[k];
18: end
```

Exercise 2 (The Longest Upsequence)

Consider the algorithm for the longest upsequence. Extend the algorithm that calculates the length of the longest upsequence to calculate one longest upsequence itself. Provide the algorithm in pseudocode and explain the used datastructures.

The datastructures used in the algorithm above are as follows. X represents the sequence of numbers that are searched for the longest upsequence. k is the length of the longest upsequence. Y is the upsequence itself. The temporary array M is a 2-dimensional array that stores the upsequences of length $1..k$, that contain the smallest largest element.

Exercise 3 (FIFO - Circular Buffer)

In the lecture the data structure circular buffer was explained.

a) Consider the pseudocode given in the lecture and implement the following functions in C. The ringbuffer is represented by a global array and two global indices (one representing the read and one the write index). Write tests that cover each function.

- **init** initializes a circular buffer (if it is already initialized reset it and clear all entries)
- **empty** calculates if the circular buffer is empty
- **full** calculates if the circular buffer is full
- **enqueue** adds a new value to the circular buffer
- **dequeue** removes a value from the end

b) In the code from the lecture there are conditional statements that check the size of the buffer. Modulo operations can be used to check the array boundaries instead. Modify your functions in a way that modulo-arithmetics are used instead of the if-statements.

c) If n is the size of the array that is used as a buffer for the FIFO queue, the capacity of the queue is only n - 1. Modify your program so that the whole capacity of the buffer can be used.

Hint: Check if you need to change the global data structure.

The approach for the solution is to introduce an additional flag. This flag is used to distinguish between full and empty FIFO. The reason for this is that in a FIFO that utilizes the underlying buffer in both states, full and empty, both pointers read and write pointer point at the same position within the array. It is important to see that there can be stored one more element in the FIFO by the cost of an additional flag. Memory can only be saved if the FIFO stores elements of a data type that is greater than the data type of the flag.

d) In the previous exercise the data structure is represented with several global variables. Introduce a new record type that combines all these global states within one record.

The solution creates a record that has the following fields: the backbuffer of the FIFO, a pointer to the the read position in the backbuffer, a pointer to the write position in the backbuffer and a flag if the FIFO is full or not.

e) In the previous exercises there exists only one global FIFO. This prevents the program from dealing with multiple circular buffers. Modify your program to support multiple circular buffers and modify your base operations to take an arbitrary FIFO as an argument.