
# Data Structures and Program Design in C

**Topic 7: Stacks and Queues**

Dr Manjusri Wickramasinghe

# Outline

- Stacks
- Queues

# Classification of Data Structures …(1)

- Data structures can be mainly classified into two (2) categories:
  - Linear
  - Non-Linear

- Linear data structures have their elements arranged sequentially so that a given element is attached to the previous and the next adjacent elements.
  - E.g. Arrays, Linked Lists, Stacks, and Queues.

# Classification of Data Structures …(2)

- Non-linear data structures have elements that are arranged in a non-sequential manner, which is called such.
  - E.g. Trees, Graphs

# Abstract Data Types

- Abstract Data Types (ADT) is a class of objects whose behavior is defined by a set of values and operations as seen by a user.

- Represents a high-level use of concept independent of the underlying implementation.

# Data Structures

- Data structures, on the other hand, provide how the behaviour mentioned by an ADT is implemented.

- With any data structure, we must learn three (3) fundamental operations. That is
  - How to insert data into the structure
  - How to delete data from the structure
  - How to search for data

- In addition, there would be structure-specific operations.

# Stacks ...(1)

- Stack is a linear data structure that follows a last in, first out (LIFO) policy.

- Stacks are data structures that implement many functionalities such as backtracking, functional calls, memory management, syntax checking, etc.

- Stacks are implemented using **array** or **linked list** data structures.

# Stacks ...(2)

- There is only a single entry point into the stack data structure.

- Since there is a single entry point, elements are inserted and deleted using the same point. The pointer representing the single entry point is the **top of the stack (TOS)**.

- There are three primary operations defined in the stack. These are:
  - `push(element)` → Insert an element with a specified value into the stack
  - `pop()` → deletes the element at the top of the stack and returns the value
  - `peek()` → retrieves an element at the top of the stack without deleting the element.

# Writing C Code in Multiple Files …(1)

- A C program can be in a single code file or multiple code files.

- Multiple code files can be in two forms as:
  - User defined headers (.h) files
  - User code (.c)

- The header files (.h files) are used to define aspects such as data types (primitive or user defined), function prototypes that would be used by the user code (.c files).

# Writing C Code in Multiple Files …(2)

- When declaring headers it is important to make sure that same types are not defined multiple times. Therefore it is important to have conditional compilation of macros.
  - Done by the preprocessor

- Usage

```
#ifndef <HEADER_H>
#define <HEADER_H>

//definition

#endif
```

# Writing C Code in Multiple Files …(3)

- To compile the multiple files, all .c files are compiled using gcc
  - E.g. `gcc <file1>.c <file2>.c`

- `extern` keyword
  - Used to increase the visibility of C variables and functions.
  - When used no memory is allocated
  - Usage

```
extern <data_type> <variable_name>;
```

# Memory Layout of a Program …(1)

- Generally a C program's memory representation consists of the following five (5) sections as:
  - Text segment → contains executable instructions. The text segment is shareable so only a single copy is required.
  - Data segment
    - Initialized → contains the global and static variables that are initialized.
    - Uninitialized (bss) → also called "block started by symbol" for uninitialized global and static variables or ones set to zero.
  - Heap memory → dynamic memory allocation takes place in heap memory.
    - Used by `malloc(…), free(…), realloc(…)` and `calloc(…).`
  - Stack memory → contains the program stack
    - The top of stack is tracked by the $SP special purpose register.
    - The set of value pushed for one function is called a 'stack frame'

# Memory Layout of  a Program …(2)

- The `size(…)` command reports the sizes in bytes of the text, data and the bss segment of memory.

# Recursion

- Recursion is called as “defining a problem in terms of itself” or in programming terms “a function calling another instance of itself”.
  - E.g. Factorial, Fibonacci number

- Every recursion has two properties as:
  - Base case (anchor point)
  - Recursive step

- Implemented using the stack memory.

# Queues …(1)

- Queues are linear data structures that follows a first in first out policy.
- Queues are implemented using **array** or **linked list** data structures.
- Queues has applicability is various forms for example the process schedule is a priority queue, keyboard input buffer is a queue.

# Queues …(2)

- Since it is a FIFO structure, two pointers are required for the first element and the last element.

- These pointers are called **first** and **last** respectively.

- There are three primary operations defined in the queue. These are:
  - `enqueue(element)` → Insert an element with a specified value to the end of the queue using the last pointer.
  - `dequeue()` → deletes the element at the first pointer and returns the value.

# Types of Memory Allocations

- There are two (2) types of memory allocations as
  - Static
  - Dynamic

- Static memory allocation is done at compile time, and the allocated memory is not changed during the program's execution.

# Questions?