31251 – Data Structures and Algorithms

Week 2, Autumn 2020

Xianzhi Wang

- Standard I/O
- Exceptions
- Pointer
- Class
- Source Files Organisation
- Data Structures
  - Queue
  - Stack
  - C++ Standard Template Library

Standard I/O

- I/O with Terminal (`#include<iostream>`)
  - Read a fragment: `cin >> [variable name]`
  - Read a whole line: `getline(cin, [variable name])`
  - While: `cout << [thing to write]`
- I/O with File (`#include<fstream>`)
  - Read a fragment: `cin >> [variable name]`
  - Read a whole line: `getline(cin, [variable name])`
  - While: `[file name] << [thing to write]`
- Practice 1

Exceptions

- C++ can `throw` exceptions, just like Java.
- C++ does define a set of exceptions, defined in `<exception>`: `http://www.cplusplus.com/doc/tutorial/exceptions/`

```
try{
      // do something
      throw (parameter name);
} catch([Exception Type 1] [parameter name]){
      // do something
} catch (Exception Type 2] [parameter name]){
      // do something
} catch (...){
      cout << "Default Exception!" << endl;
}
```

- Practice 2

- Declaration
  - `student * s_ptr = new student();` // declaring a pointer
  - `student s;` // declaring an object
- Dereferencing
  - `s_ptr->score;` // pointer->attribute
  - `s.score;` // object.attribute
- Memory deallocation
  - `delete s_ptr;` // explicitly deallocate space
  - Nothing needs to be done.
- Practice 3

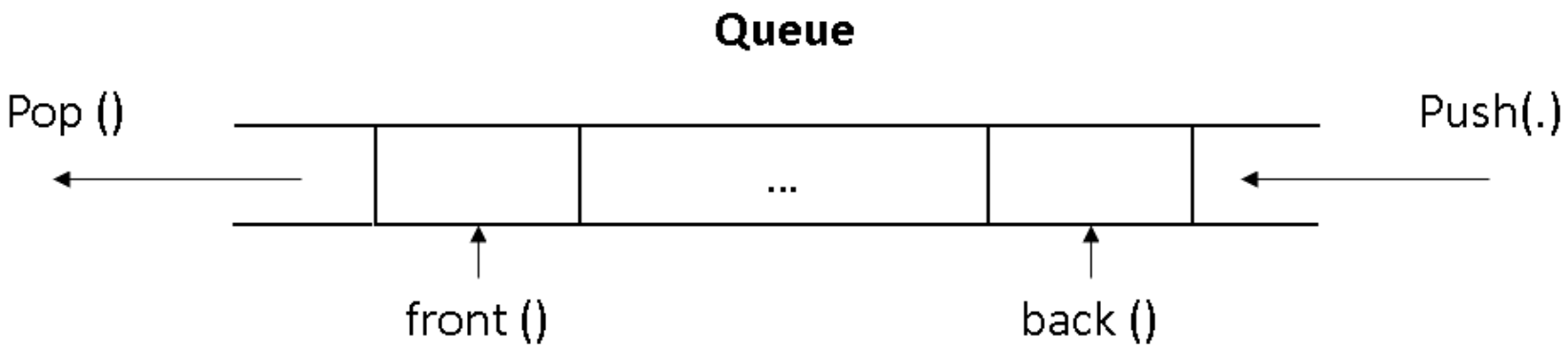
- Special member functions
  - Constructor is automatically called when object (instance of class) create, e.g., `LinkedList()`.
  - Destructor is automatically called before the object's memory is deallocated due to the use of `delete` (if the object has a destructor), e.g., `~LinkedList()`.
- Create class:
  - Static (declaring an object of class): `LinkedList list;`
  - Dynamic (declaring a pointer to an object of class): `LinkedList* list = new LinkedList();`
- Delete class (to avoid memory leak):
  - Delete a pointer: `delete [pointer to thing to delete]`.
  - Delete an array of pointers: `delete[] [array variable]`.

Source Files Organisation

- Source Package
  - Header Files
    - #inlcude & Namespace
    - Abstract Data Types
      - Classes
        - Attributes
        - Inner Classes
        - Constructors/Destructor (declaration)
        - Getters & Setters (declaration)
        - Other Functions (declaration)
  - Source Files
    - Class Functions (implementation)
    - Main Function

- Practice 4

- The (basic) Queue is the basic FIFO (first-in-first-out) data structure.
- It keeps things in order (like a list), but...
- Things can only be added to the back and taken from the front.



A Pure Virtual Class for a Queue of ints

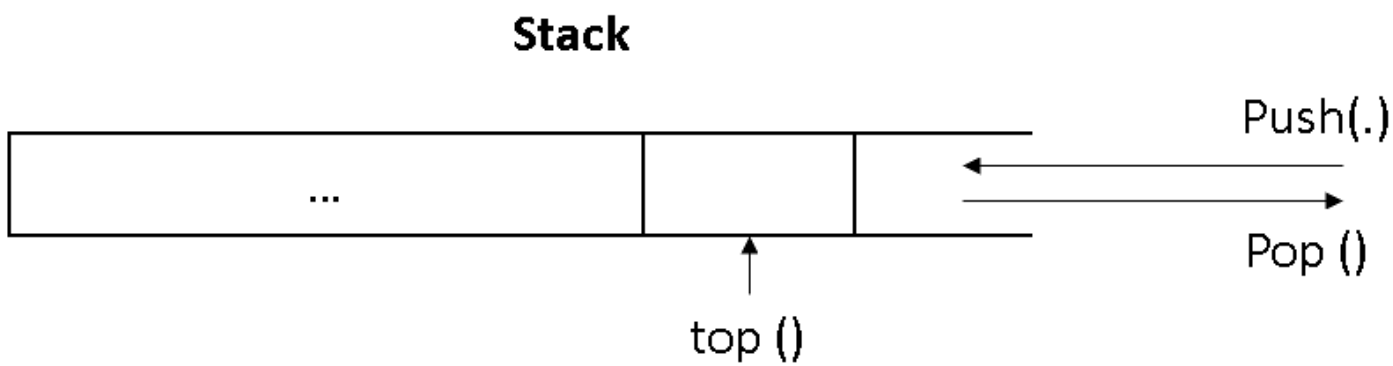
```
class intQueue {
  public:
    virtual ~intQueue() {};
    virtual bool empty() = 0;
    virtual size_type size() = 0;
    virtual int front() = 0; // get the front element
    virtual int back() = 0; // get the end element
    virtual void push(int x) = 0;
    virtual void pop() = 0;
};
```

Other types of Queue

- A `Deque` is a double-ended queue - you can add and remove at both ends!
  - This is really useful for implement other data structures.

- A `Priority Queue` is a queue, but elements are inserted with a priority, and come out in priority order.

- A Stack is like a queue, but it's a last-in-first-out (LIFO) data structure.
- You can add to the "top", and
- remove from the "top".

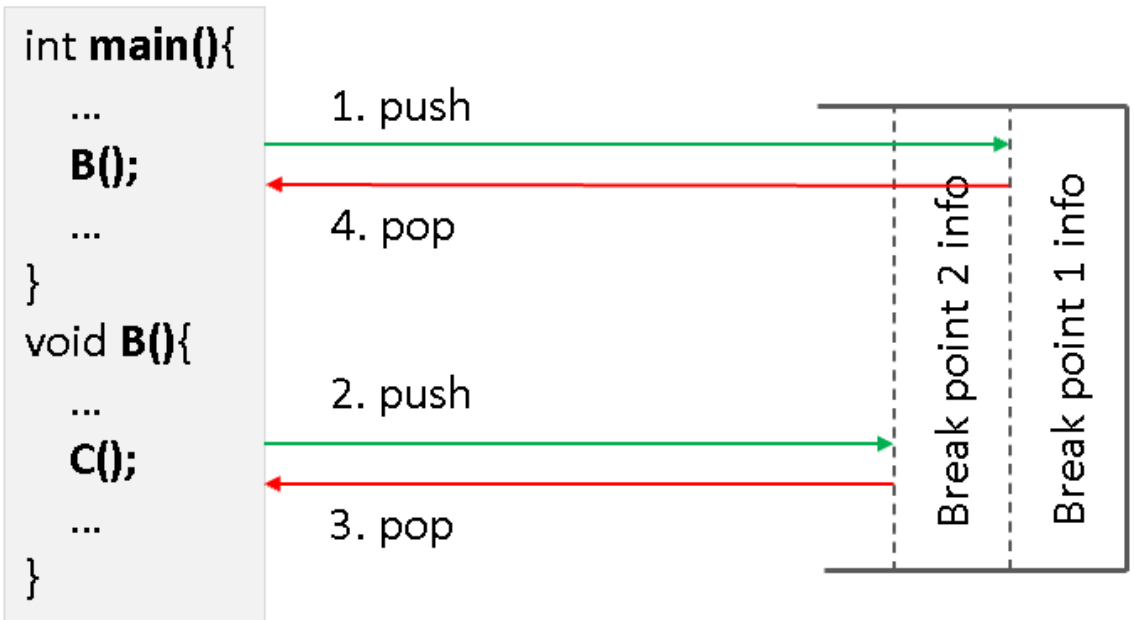


A Pure Virtual Class for a Stack of ints

```
class intStack {
  public:
    virtual ~intStack() {};
    virtual bool empty() = 0;
    virtual size_type size() = 0;
    virtual int top() = 0; // get the top element
    virtual void push(int x) = 0;
    virtual void pop() = 0;
};
```

Stacks and Queues are two of the most used data structures:

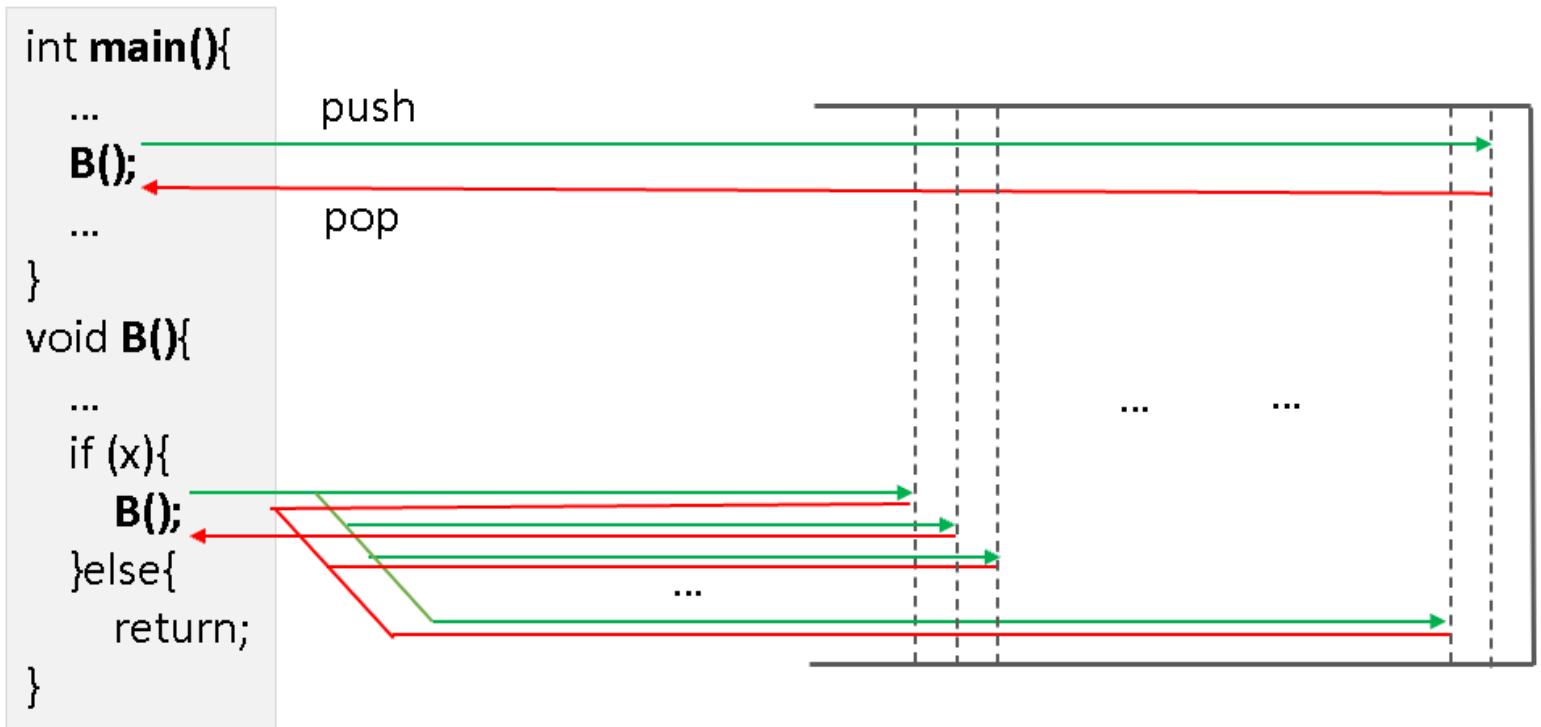
- OS: Multi-Level Priority Queues for process scheduling.
- Networking (e.g., routers): buffers of all kinds are Queues.
- Compiler: stacks are built into all programming languages.



- Practice 5

A first Understanding of Recursion

- Recursion: a function repeatedly calling itself.
- Consequence: often excessive stack usage and poor efficiency.
- We aim to avoid it in algorithm design.



More data structures are available in C++:

- Week 3
  - Vector
  - List
- Week 8
  - Map, MultiMap
  - Set, MultiSet
  - Unordered Set/Map/MultiSet/MultiMap

Reference: `https://www.geeksforgeeks.org/the-c-standard-template-library-stl/`

Let's do more coding (if time permits).