CSCI 262
Data Structures

6 – Stacks

“Last in, first out”

Stacks are a LIFO (Last in, first out) structure. Think of pancakes:

This pancake was put on top last.
Which one would you eat first?
Which would you eat second?

Three Operations

- **top**: Look at the top item on the stack.
- **push**: Add an item to the top of the stack.
- **pop**: Remove the top item from the stack.

A Simple Stack Class

```cpp
class stack {
    public:
        char top();
        void push(char c);
        void pop();
        bool isEmpty();
    private:
        // private stuff
};
```

These operations are sometimes combined, e.g., pop() may return the top value on the stack as well as removing it from the stack.

Using Stacks

What does this code do?

```cpp
stack letters;
string text = "Data structures";
for (int j = 0; j < text.length(); j++) {
    letters.push(text[j]);
}
while (!letters.empty()) {
    cout << letters.top();
    letters.pop();
}
```

Applications

- **Syntax analysis**
  - Are parentheses, brackets, etc. balanced?
- **Traversing/searching branching structures**
  - Trees
  - Mazes
- **Programming languages/processors**
  - Forth, Postscript
  - Stack machines (e.g., Java virtual machine)
“The Stack”

When we talk about “the stack”, we usually mean a very specific stack; the memory stack of a running program:

<table>
<thead>
<tr>
<th>“Bottom” of stack</th>
<th>“Top” of stack</th>
</tr>
</thead>
<tbody>
<tr>
<td>main()</td>
<td>Local variables declared in main, return address, other stuff.</td>
</tr>
<tr>
<td>function1</td>
<td>Local variables declared in function1; arguments passed by value into function1, return address, other stuff.</td>
</tr>
<tr>
<td>function2</td>
<td>Etc.</td>
</tr>
<tr>
<td>function3</td>
<td>Each “frame” is created when the function is called, and destroyed when the function exits.</td>
</tr>
<tr>
<td>function4</td>
<td></td>
</tr>
</tbody>
</table>

StanfordCPPLib Stack

```cpp
#include "stack.h"

class Stack<ValueType> {

    // Operations:
    push(ValueType v)  // push value onto top of stack
    pop()              // return and remove top value
    peek()             // return top value
    size()             // return number of elements
    isEmpty()          // true if no elements
    clear()            // remove all elements
    toString()         // return a string representation of stack

```