Stacks

February 6, 2019

Overview

- What are stacks?
- Where are stacks used?
- Stack behavior
- Example (with code)

What are Stacks?

- A LIFO[1] structure.
- A stack is a pile of *things*:
  - A stack of dishes in a cafeteria/restaurant.
  - A stack of papers in an office.

[1] Last In, First Out

Where are stacks used?

- A stack of activation records are maintained when a program is executed. [An activation record contains the data needed for each execution of a function.]
- Compilers frequently use a stack of symbols when parsing source code.
- Simulate recursion.
- Graphics — Transformation matrices.

Where are stacks used?

- Calculators (RPN)
- Programming languages
  - Argument passing
  - PostScript
  - Forth
  - JVM (Java Virtual Machine)
  - more...[2]

[2] *Stack-oriented programming language*
`https://en.wikipedia.org/wiki/Stack-oriented_programming_language`

Stack Behavior

Standard operations:

- Push
- Pop
- Peek
- (Print / Show)

Stack Behavior

Standard operations with a stack, `s`:

- `s.push( x )` adds an item to the top of a stack.
- `x = s.pop()` removes the item on the top of a stack.
- `x = s.peek()` looks at the item on the top of a stack.

"Well formed" Parentheses

The general problem can be stated as[3]: Determine whether a set of paired symbols is used appropriately.

The specific problem is: Given a set of different types of paired symbols, determine whether the opening and closing versions of each type are paired correctly.

[3] Nell Dale, C++ Plus Data Structures, 2003

“Well formed” Parentheses

For our problem, we consider the parenthesis pairs `()`, `[]`, and `{}`. Any number of other characters may appear in the input, but a closing parenthesis symbol must match the last unmatched opening parenthesis symbol and all parenthesis symbols must be matched when the input is finished.

“Well formed” Parentheses — Valid

Well-Formed Expressions

```
( xx ( xx ( ) ) xx )

[ ] ( ) { }

(  [  ]  { xxx } xxx (  ) )

( [ { [ ( ( [ { x } ] ) x ) ] } x ] )
```

“Well formed” Parentheses — Invalid

Ill-Formed Expressions

```
] [

( aa ( bb ( ) ) xzx ) xwx)

xxx ) (  xx  [  xxx  )  xx  ]

( [ { [ ( ( [ { x } ] ) x ) ] } x )
```

"Well formed" Parentheses — Solution Methods

Solution Methods

- Count
- Stacks

Counting can fail! For example: `)  a  b  b  a  (`

Stack will work, but could overflow for large input.

Stack Class Interface–public


```
/* stack.h
 */

class Stack
{
public:
    Stack()
    {
        top = NULL;
    }

    void Push( char c );
    char Pop();

    bool IsEmpty();

    void Print();

private:
    ... next page
};
```

Stack Class Interface–private


```
private:
    struct stackNode
    {
        char info;
        struct stackNode* next;
    };

    typedef struct stackNode StackNode;
    typedef StackNode *StackNodePtr;

    StackNodePtr top;
};
```

Parentheses Balance: Header


```
/* parenBalance.cpp

   Determine if "parentheses" are balanced.
   Parentheses:  (), [], and {}

   Bruce M. Bolden                September 15, 2010
 */

#include <iostream>

using namespace std;

#include "stack.h"

bool IsOpen( char symbol );
bool IsClose( char symbol );
bool Matches( char symbol, char openSymbol );
```

Parentheses Balance: main() – Overview


```
int main()
{
    (Variable declarations)

    Prompt/Initial input

    Processing/Additional input

    Output

    return 0;
}
```

Parentheses Balance: main() – Input


```
int main()
{
    char symbol;      //  current input character
    char openSymbol;  //  top symbol on stack

    Stack s;
    bool balanced = true;

       //  Prompt, get first symbol
    cout << "Enter an expression and press return." << endl;
    cin.get( symbol );

    ....
}
```

Parentheses Balance: main() – processing loop


```
....

while( symbol != '\n' && balanced )
{
    if( IsOpen(symbol) ) {
        s.Push(symbol);
    }
    else if( IsClose(symbol) )
    {
        if( s.IsEmpty() )
            balanced = false;
        else  {
            openSymbol = s.Pop();
            balanced = Matches(symbol, openSymbol);
        }
    }
        //  Get next symbol
    cin.get(symbol);
}

....
```

Parentheses Balance: main() – Output


```
    ....

    if( balanced ) // stack empty?
        cout << "Expression is well formed." << endl;
    else
        cout << "Expression is not well formed."  << endl;

    return 0;
}
```

Parentheses Balance: Auxiliary Functions


```
/* IsOpen:  Open "parenthesis" symbol?  */
bool IsOpen( char c )
{
    if( (c == '(') || (c == '{') || (c == '[') )
        return true;
    else
        return false;
}

/* IsClose:  Close "parenthesis" symbol?  */
bool IsClose( char c )
{
    if( (c == ')') || (c == '}') || (c == ']') )
        return true;
    else
        return false;
}
```

Parentheses Balance: Auxiliary Functions


```
/*  Matches:  Same left and right paren "type"?  */
bool Matches( char symbol, char openSymbol )
{
    return( ((openSymbol == '(') && symbol == ')') ||
            ((openSymbol == '{') && symbol == '}') ||
            ((openSymbol == '[') && symbol == ']')   );
}
```

Stack Class—Testing


```
% g++ parenBalance.cpp stack.cpp
% ./a.out
Enter an expression and press return.
)(
Expression is not well formed.

% ./a.out
Enter an expression and press return.
( a b b a )
Expression is well formed.

% ./a.out
Enter an expression and press return.
[ a { b } ( b ) a ]
Expression is well formed.
```

Stack: Push()


```
void Stack::Push( char c )
{
   StackNodePtr newNode = new StackNode;
   //StackNodePtr newNode = (StackNodePtr)malloc(sizeof(Stack));

   newNode->info = c;
   newNode->next = top;

   top = newNode;
}
```

Stack: Pop()


```
char Stack::Pop()
{
   char c = 'Z';        //  value of top
   StackNodePtr del;    //  deleted node

   if( top == NULL ) {
      cout << "Error::Pop: stack empty" << endl;
      //printf( "Error::Pop: stack empty\n" );
   }
   else {
      c = top->info;
      del = top;
      top = top->next;

      del->next = NULL;

      delete del;       //free( (void *)del );
   }

   return c;
}
```

Stack: IsEmpty()


```
bool Stack::IsEmpty()
{
   return top == NULL;
}
```

Stack: Print()


```
void Stack::Print()
{
   StackNodePtr p = top;

   cout << "{ ";                    //printf( "{ " );
   while( p != NULL )
   {
       cout << p->info << ", ";    //printf( "%3c, ", p->info );
       p = p->next;
   }
   cout << "{ ";                    //printf( " }\n" );
}
```