• Sample PMT online…
    • Browse
      http://www.cs.wmich.edu/gupta/teaching/cs1120/sumI05/PMT/2004_1/

Stacks and Queues

Checking for Balanced Braces

Balanced:  abc{dc}pa{cdd}p
Unbalanced:  {aadf(df@#)}dфа{df}
Pseudocode

```c
while (not end of line) {
    if (ch is '{')
        add ch to Stack
    else if (ch is '}') {
        if (Stack is empty)
            return unbalanced;
        remove from Stack
    }
    if (Stack is empty)
        return balanced;
    return unbalanced;
}
```

Definition of Stack

- **ADT level**
  - A stack is an ordered group of homogeneous items, in which the removal and addition of stack items can take place only at the top of the stack
  - A stack is a LIFO: last in, first out

Stack ADT

```c
using ADT stack for brace matching
stack.createStack();
while (not end of line) {
    if (ch is '{')
        stack.push(ch);
    else if (ch is '}') {
        if (!stack.isEmpty()){
            stack.pop();
        } else{
            stack.destroyStack();
            return unbalanced;
        }
    } //endelseif
} //endwhile
if (stack.isEmpty())
    return balanced;
stack.destroyStack();
return unbalanced;
```
A Stack -- Graphic

Array-Based Implementation

```cpp
class Stack {
    Public:
        Stack();
        ~Stack();
        bool isEmpty();
        void push(Item item);
        void pop();
        Item getTop();
    Private:
        Item items[MAX];
        int top;
};
```

Array-Based Implementation

```cpp
Stack::Stack(): top(-1) {}  
Stack::~Stack() {} 
bool Stack::isEmpty() {  
    return top == -1;  
}
void Stack::push(Item item){  
    if (top == MAX-1) exit(0);  
    items[++top] = item;  
}
void Stack::pop(){  
    if (isEmpty()) exit(0);  
    --top;  
}
Item Stack::getTop(){  
    if (isEmpty()) exit(0);  
    return items[top];  
}
```
class Stack {
    public:
    private:
        List aList;
};

Stack::Stack() {} 
Stack::~Stack() {} 

bool Stack::isEmpty() {
    return aList.isEmpty(); 
}

void Stack::push(Item item) {
    aList.insert(item, 1); 
}

void Stack::pop() {
    aList.remove(1); 
}

Item Stack::getTop() {
    return aList.retrieve(1); 
}
Compare Three Implementations

• Array v.s. LinkedList
  - Fixed size v.s. dynamic size
• LinkedList v.s. ADT List
  - Efficiency v.s. Simplicity

Application: Algebraic Expressions

• Infix expressions
  - Every binary operator appears between its operands: \(a+b\cdot c\), \((a+b)\cdot c\)
• Prefix expressions
  - Operator appears before its operands: \(+a\cdot bc\), \(*+abc\)
• Postfix expressions
  - Operator appears after its operands: \(abc*+\), \(ab+c*\)

Evaluating Postfix Expressions
Pseudocode

Stack aStack;
for (each token t in the input) {
  if (t is an operand)
    aStack.push(t);
  else {
    operand2 = aStack.getTop();
    aStack.pop();
    operand1 = aStack.getTop();
    aStack.pop();
    result = operand1 t operand2;
    aStack.push(result);
  }
}