Data structures and algorithms

Tutorial 4

Amr Keleg

Faculty of Engineering, Ain Shams University

April 16, 2019

Contact: `amr_mohamed@live.com`

Outline

1 Linked List methods
- Practice

2 The stack

3 Stack in STL

Outline

1 Linked List methods
- Practice

2 The stack
- Basic structure
- Implementation (1) - Static Array
- Sheet 2 - Question 4
- How to solve shallow copy of pointers?
- Sheet 3 - Question 4
- Strategy
- INFIX and POSTFIX
- Infix to postfix conversion
- Postfix Evaluation

3 Stack in STL
- Stack Example

Make sure you can implement the functions correctly.
`https://www.hackerrank.com/domains/data-structures/linked-lists`

Outline

1 Linked List methods

2 The stack
- Basic structure
- Implementation (1) - Static Array
- Sheet 2 - Question 4
- How to solve shallow copy of pointers?
- Sheet 3 - Question 4
- Strategy
- INFIX and POSTFIX
- Infix to postfix conversion
- Postfix Evaluation

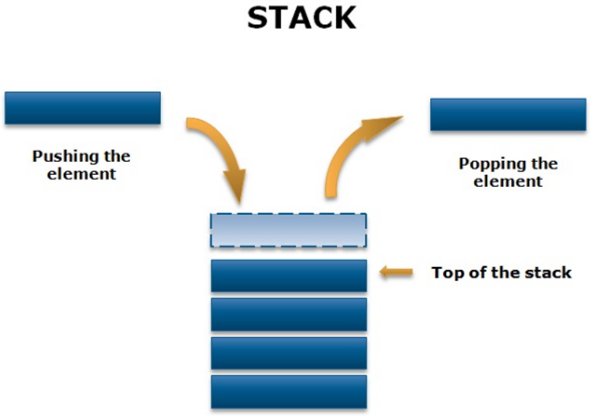
3 Stack in STL

Outline

1 Linked List methods
- Practice

2 The stack
- Basic structure
- Implementation (1) - Static Array
- Sheet 2 - Question 4
- How to solve shallow copy of pointers?
- Sheet 3 - Question 4
- Strategy
- INFIX and POSTFIX
- Infix to postfix conversion
- Postfix Evaluation

3 Stack in STL
- Stack Example


STACK
Pushing the element
Popping the element
Top of the stack

```
class Stack{
  public:
    Stack();
    int top();
    void push(int v);
    void pop();
};
```

Outline

1 Linked List methods
- Practice

2 The stack
- Basic structure
- Implementation (1) - Static Array
- Sheet 2 - Question 4
- How to solve shallow copy of pointers?
- Sheet 3 - Question 4
- Strategy
- INFIX and POSTFIX
- Infix to postfix conversion
- Postfix Evaluation

3 Stack in STL
- Stack Example

```
#include<cassert>
class Stack{
    int arr[1000];
    int size;
  public:
    Stack(){ size=0; }
    int top(){
      assert(size>0);
      return arr[1000-size];
    }
    void push(int v){
      assert(size<1000);
      arr[1000-size-1] = v;
    }
    void pop(){
      assert(size>0);
      size--;
    }
    bool empty(){ return size==0; }
};
```

Outline

1 Linked List methods
- Practice

2 The stack
- Basic structure
- Implementation (1) - Static Array
- Sheet 2 - Question 4
- How to solve shallow copy of pointers?
- Sheet 3 - Question 4
- Strategy
- INFIX and POSTFIX
- Infix to postfix conversion
- Postfix Evaluation

3 Stack in STL
- Stack Example

4. Create a member function for stack to compare between two stacks, the function should takes one stack as a parameter and compares it to the class stack it return either true or false.

```
class Stack{
  ....
  bool is_equal(Stack s);
};
```

```
bool is_equal(Stack s){
  while(!empty() && !s.empty()){
    if (top() != s.top())
      return false;
    pop();
    s.pop();
  }
  return empty() && s.empty();
}
```

What is the problem here?

```
bool is_equal(Stack s){
  Stack c = *this;
  while(!c.empty() && !s.empty()){
    if (c.top() != s.top())
      return false;
    c.pop();
    s.pop();
  }
  return c.empty() && s.empty();
}
```

What will happen if the array was dynamically allocated?

```
bool is_equal(Stack s){
  Stack c = *this;
  while(!c.empty() && !s.empty()){
    if (c.top() != s.top())
      return false;
    c.pop();
    s.pop();
  }
  return c.empty() && s.empty();
}
```

Outline

1 Linked List methods
- Practice

2 The stack
- Basic structure
- Implementation (1) - Static Array
- Sheet 2 - Question 4
- How to solve shallow copy of pointers?
- Sheet 3 - Question 4
- Strategy
- INFIX and POSTFIX
- Infix to postfix conversion
- Postfix Evaluation

3 Stack in STL
- Stack Example

Operator overloading: (Overload the assignment operator)

```
void operator=(const Stack &s) {
    this->size = s.size;
    for(int i=0; i<1000; i++){
        this->arr[i] = s.arr[i];
    }
}
```

```
bool is_equal(Stack s){
  Stack c = *this;
  while(!c.empty() && !s.empty()){
    if (c.top() != s.top())
      return false;
    c.pop();
    s.pop();
  }
  return c.empty() && s.empty();
}
```

Should s be passed by reference? Should s be passed as const Stack s?

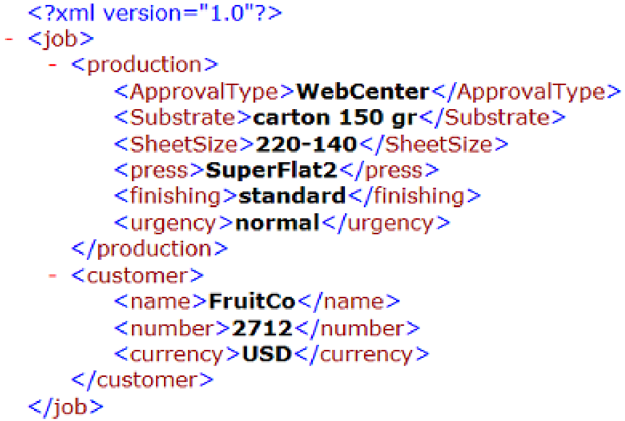
Outline

1 Linked List methods
- Practice

2 The stack
- Basic structure
- Implementation (1) - Static Array
- Sheet 2 - Question 4
- How to solve shallow copy of pointers?
- Sheet 3 - Question 4
- Strategy
- INFIX and POSTFIX
- Infix to postfix conversion
- Postfix Evaluation

3 Stack in STL
- Stack Example

Use stack to check the consistency of an XML file, the XML is based on having opening tag and closing tag, between the open and closing tags there exist the information for the element. a consistent XML file has balanced number of open and closing tags. For example the opposite figure shows a balanced XML file



```
  <?xml version="1.0"?>
- <job>
  - <production>
      <ApprovalType>WebCenter</ApprovalType>
      <Substrate>carton 150 gr</Substrate>
      <SheetSize>220-140</SheetSize>
      <press>SuperFlat2</press>
      <finishing>standard</finishing>
      <urgency>normal</urgency>
    </production>
  - <customer>
      <name>FruitCo</name>
      <number>2712</number>
      <currency>USD</currency>
    </customer>
  </job>
```

Outline

1 Linked List methods
- Practice

2 The stack
- Basic structure
- Implementation (1) - Static Array
- Sheet 2 - Question 4
- How to solve shallow copy of pointers?
- Sheet 3 - Question 4
- Strategy
- INFIX and POSTFIX
- Infix to postfix conversion
- Postfix Evaluation

3 Stack in STL
- Stack Example

```
Assumption: < and > for the same tag will appear
in the same line.
```

- Read the file line by line
- Ignore comments and preprocessor tags
- While the line contains tags
- Extract them
- If openning tag - Push to stack
- If closing tag - Check the top of the stack
- If stack isn't empty in the end - ERROR

Outline

1 Linked List methods
- Practice

2 The stack
- Basic structure
- Implementation (1) - Static Array
- Sheet 2 - Question 4
- How to solve shallow copy of pointers?
- Sheet 3 - Question 4
- Strategy
- **INFIX and POSTFIX**
- Infix to postfix conversion
- Postfix Evaluation

3 Stack in STL
- Stack Example

Infix: 2 + 3 * 5
Postfix: 2 3 5 * +

Outline

1 Linked List methods
- Practice

2 The stack
- Basic structure
- Implementation (1) - Static Array
- Sheet 2 - Question 4
- How to solve shallow copy of pointers?
- Sheet 3 - Question 4
- Strategy
- INFIX and POSTFIX
- Infix to postfix conversion
- Postfix Evaluation

3 Stack in STL
- Stack Example

- 4 3 + 5 2 * -
- x t y * + 2 /
- b ~ b b * 4 a * c * + 2 a * /

Outline

1 Linked List methods
- Practice

2 The stack
- Basic structure
- Implementation (1) - Static Array
- Sheet 2 - Question 4
- How to solve shallow copy of pointers?
- Sheet 3 - Question 4
- Strategy
- INFIX and POSTFIX
- Infix to postfix conversion
- Postfix Evaluation

3 Stack in STL
- Stack Example

- (X*y) + t
- (A*B) / (X-Y)

Outline

1 Linked List methods

2 The stack

3 Stack in STL
- Stack Example

Outline

1 Linked List methods
- Practice

2 The stack
- Basic structure
- Implementation (1) - Static Array
- Sheet 2 - Question 4
- How to solve shallow copy of pointers?
- Sheet 3 - Question 4
- Strategy
- INFIX and POSTFIX
- Infix to postfix conversion
- Postfix Evaluation

3 Stack in STL
- Stack Example

```
#include<stack>

stack<string> st;
st.push("str1");
st.push("str2");
while(!st.empty()){
  cout<<st.top()<<endl;
  st.pop();
}
```

- Stack: `http://www.cplusplus.com/reference/stack/stack/`

Feedback form: `https://goo.gl/forms/VgArSE8HzbaQgHli2`