Homework Assignment 3

Due Date: 18:30, April 7, 2016

1 Problem 1

Use stream error states to check whether the input is a number or not. If yes, translate a decimal number to hexadecimal and octal. Your program should keep prompting for a number until `EOF` is reached.

1.1 Sample Input

```
10
d
```

1.2 Sample Output

```
a 12
not a number!
```

2 Problem 2

Design a program that use data type `unsigned short int`, 2 bytes, to store three variable **day**, **month**, and **year** by one variable **date**. Notely, four variables are the same data type. Then, extract the original data, **day**, **month**, and **year**, from **date**.

p2.cpp

```
#include"date.h"
#include<iostream>

int main(){
        short int yr,mon,day;

        //construct data
        Date date;
        std::cin>>yr>>mon>>day;
        date.setDate(yr,mon,day);

        std::cout<<date.getDate_data()<<std::endl;
        date.showDate();

        return 0;
}
```

date.h

```
#ifndef DATE
#define DATE
#include<iostream>

class Date{
         unsigned short int Date_data;
public:
        //constructors
        Date():Date_data(0){};

        //get_funcs
        unsigned short int getDate_data()const;
        unsigned short int getDay()const;
        unsigned short int getMon()const;
        unsigned short int getYr()const;

        //set_funcs
        void setDate(const short int&,const short int&,const short int&);

        void showDate();
};
#endif
```

2.1 Sample input

```
2011 5 4
```

2.2 sample output

```
5796
2011 5 4
```

Hint:

1. **Bitwise operator is a good method to implement but you still could use the method you design.**
2. **You do not need to modify `date.h` and `p2.cpp`. TA will use another test files (each function of class) for scores.**
3. **The year should be started from 2000.**
4. **It is OK that the result of date are not the same as sample. TA would focus on the result of extraction, day, month, and year.**

3 Problem 3

As we introduced in class, a stack is a last-in, first-out data structure—the last item pushed (inserted) on the stack is the first item popped (removed) from the stack. In this assignment, you will design your own class, named `MyStack`, to provides similar functionality by using a linked list, rather than an array, to store the stack contents. Given the interface of such a stack class in `MyStack.h`, you are required to implement the stack class by using a linked list and to implement all of the member functions listed in `MyStack.h`. Class `MyStack` is a template class that allows any type of elements to be pushed onto a `MyStack` object.
Notes: You do not need to modify `MyStack.h`.

MyStack.h

```
#ifndef MYSTACK_H_INCLUDED
#define MYSTACK_H_INCLUDED
#include <cstddef> // for std:size_t
using namespace std;

template <class T>
class MyStack{
public:
    MyStack();
    ~MyStack();
    void push(T&);
    void pop();
    T& top();
    size_t getSize() const;
    bool empty() const;
private:
    class Node {
        friend class MyStack<T>;
    public:
        Node(T&, Node*);
    private:
        T &value;   // the value of the node
        Node *next; // a pointer to the next node
    };
    Node *head;  // a pointer to the head of the linked list
                 // also a pointer to the top of the stack
    size_t size; // the number of elements in the stack
};
#endif // MYSTACK_H_INCLUDED
```

3.1 Class `MyStack`

3.1.1 Data Members of `MyStack`

Class `MyStack` has two private data members:

1. `head`: a pointer to the head node of the linked list—`head` is also a pointer to the top node of the stack.
2. `size`: the number of elements in the stack.

Each node in the linked list that implements the stack has the type of class `Node`.

3.1.2 Member Functions of `MyStack`

The member functions of class `MyStack` include:

1. `MyStack()`
   Constructor of class `MyStack`.
2. `~MyStack()`
   Destructor of class `MyStack`.
3. `void push(T&)`
   The function inserts an element of type T onto the top of the stack.
4. `void pop()`
   The function removes an element from the top of the stack.

5. `T& top()`
   The function returns an element from the top of the stack. The element is not removed from the stack.

6. `size_t getSize() const`
   The function returns the number of elements in the stack.

7. `bool empty() const`
   The function returns whether the stack is empty.

3.2 Class `MyStack::Node`

3.2.1 Data Members of `MyStack::Node`

Class `MyStack::Node` has two private data members:

1. `value`: the value of the node, which is of type `T`, and

2. `next`: a pointer to the next node in the linked link.

3.2.2 Member Functions of `MyStack::Node`

Class `MyStack::Node` includes only a member function:

- `MyStack::Node(T&, Node*)`
  Constructor of class `MyStack::Node`. It takes two arguments to initialize a node. The first argument is the value of the node, and the second argument is a pointer to the next node.

Note that class `MyStack` is declared a friend class of class `MyStack::Node`, so class `MyStack` can access any members of class `MyStack::Node`.

Notes: Your implementation of the first part of the assignment should be written in `MyStack.cpp`.

Notes: You should insert the following declaration at the end of `MyStack.cpp`. Check out `http://www.parashift.com/c++-faq-lite/separate-template-fn-defn-from-decl.html` for detail explanation.

MyStack.cpp

```
...
template class MyStack<char>;
```

3.3 Client codes of `MyStack`

You will use the following client codes to demonstrate the functionality of your stack class.

client1.cpp

```
#include <iostream>
#include "MyStack.h"
using namespace std;

int main(void) {
    MyStack<char> stack;
    char input;
    char *value[5];

    for(int i=0; i<5; i++) {
        cin >> input;
        value[i] = new char(input);
        stack.push(*value[i]);
    }
    cout << endl;
    while( !stack.empty() ) {
        cout << stack.top() << endl;
        stack.pop();
    }
    for(int i=0; i<5; i++)
        delete value[i];

    return 0;
}
```

3.3.1 Sample Input

```
a
b
c
d
e
```

3.3.2 Sample Output

```
e
d
c
b
a
```

3.4 Class `Coordinate`

Recall that class `MyStack` is a template class that allows any type of elements to be pushed onto a `MyStack` object. You are given a class, named `Coordinate`, which defines two data members, `x` and `y`, to represent a position on the X-axis and Y-axis, so as to test the functionality of your `MyStack` class.

Coordinate.h

```
#ifndef COORDINATE_H_INCLUDED
#define COORDINATE_H_INCLUDED
class Coordinate {
public:
    Coordinate(int, int);
    int getX() const;
    int getY() const;
private:
    const int x;
    const int y;
};
#endif // COORDINATE_H_INCLUDED
```

Coordinate.cpp

```
#include "Coordinate.h"

Coordinate::Coordinate(int x, int y): x(x), y(y) {}

int Coordinate::getX() const {
    return x;
}

int Coordinate::getY() const {
    return y;
}
```

Notes: You do not need to modify `Coordinate.h` and `Coordinate.cpp`.

3.5 Client codes of `MyStack` and `Coordinate`

You will use the following client codes to demonstrate the functionality of your stack class with class `Coordinate`.

client2.cpp

```
#include <iostream>
#include "MyStack.h"
#include "Coordinate.h"
using namespace std;

int main(void) {
    MyStack<Coordinate> stack;
    int input1, input2;
    Coordinate *cord[5];

    for(int i=0; i<5; i++) {
        cin >> input1 >> input2;
        cord[i] = new Coordinate(input1, input2);
        stack.push(*cord[i]);
    }
    cout << endl;
    while( !stack.empty() ) {
        cout << stack.top().getX() << ' ' << stack.top().getY() << endl;
        stack.pop();
    }
    for(int i=0; i<5; i++)
        delete cord[i];

    return 0;
}
```

Notes: You should insert the following declaration at the end of `MyStack.cpp`.

MyStack.cpp

```
...
template class MyStack<Coordinate>;
```

3.5.1 Sample Input

```
1 2
3 4
5 6
7 8
9 10
```

3.5.2 Sample Output

```
9 10
7 8
5 6
3 4
1 2
```

4 How to submit the assignment?

1. Name the source code of each problem as following:
   - Prblem 1: p1.cpp
   - Prblem 2: date.cpp
   - Prblem 3: MyStack.cpp
2. Do not rename the files or put them into any directory. Upload them directly to the **e-Campus (E3)** system. You will get no credit if you don't follow the rule. Note that the penalty for late homework is **15% per day**, and late homework will not be accepted after 3 days past the due date. In addition, homework assignments must be individual work. If I detect what I consider to be intentional plagiarism in any assignment, the assignment will receive **zero credit.**