10/10

Printout for cs320-09-P-A03-VectorP.hpp

```
// File: VectorP/VectorP.hpp

#ifndef VECTORP_HPP_
#define VECTORP_HPP_

#include <iostream> // istream, ostream.
#include "ASeq.hpp"
using namespace std;

// ========= VectorP =========
// Pre: The parameter P is a pointer type.
template<class P>
class VectorP : public ASeq<P> {
private:
    P *_data;
    int _cap; // Invariant: 0 < _cap, and _cap is a power of 2.
    int _size; // Invariant: 0 <= _size <= _cap.

    void doubleCapacity();

    VectorP(VectorP const &rhs); // Disabled.
    VectorP &operator=(VectorP const &rhs); // Disabled.

public:
    VectorP();
    // Post: This vector is initialized with capacity of 1 and size of 0.

    virtual ~VectorP();

    void append(P const &e);
    // Post: Element e is appended to this vector, possibly increasing cap().

    int cap() const  override {return _cap;}
    // Post: The capacity of this vector is returned.

    void insert(int i, P const &e);
    // Pre: 0 <= i && i <= size().
    // Post: Items [i..size()-1] are shifted right and element e is inserted at position
i.
    // size() is increased by 1, possibly increasing cap().

    P &operator[](int i) override; // For read/write.
    P const &operator[](int i) const override; // For read-only.

    P remove(int i);
    // Pre: 0 <= i && i < size().
    // Post: Pointer element at index i is removed from position i and returned.
    // The caller is responsible for maintaining the pointer, including possible garbage
collection.
    // items [i+1..size()-1] are shifted left.
    // size() is decreased by 1 (and cap() is unchanged).

    int size() const {return _size;}
    // Post: The size of this vector is returned.

    void toStream(ostream &os) const;
    // Post: A string representation of this vector is returned to output stream os.

private:
};

// ========= Constructor =========
template<class P>
VectorP<P>::VectorP() {
    _data = new P[1];
    _cap = 1;
    _size = 0;
    _data[0] = nullptr;
}

// ========= Destructor =========
template<class P>
VectorP<P>::~VectorP() {
    for (int i = 0; i < _size; i++) {
        delete _data[i];
        _data[i] = nullptr;
    }
    delete [] _data;
    _data = nullptr;
}

// ========= append =========
template<class P>
void VectorP<P>::append(P const &e) {
    if (_size == _cap) {
        doubleCapacity();
    }
    _data[_size++] = e;
}

// ========= doubleCapacity =========
template<class P>
void VectorP<P>::doubleCapacity() {
    int i;
    P *newDat = new P[2 * _cap];
    for (i = 0; i < _size; i++) {
        newDat[i] = _data[i];
    }
    for (i = 0; i < _cap; i++) {
        _data[i] = nullptr;
    }
    delete[] _data;
    _data = newDat;
    _cap *= 2;
}

// ========= insert =========
template<class P>
void VectorP<P>::insert(int i, P const &e) {
    int k = i;
    if(_cap == _size){
        doubleCapacity();
    }
    for (int j = _size; j != k; j--) {
        _data[j] = _data[j-1];
    }
    _size++;
    _data[i] = e;
}


// ========= operator[] =========
template<class P>
P &VectorP<P>::operator[](int i) {
    if (i < 0 || _size <= i) {
        cerr << "VectorP index out of bounds: index == " << i << endl;
        throw -1;
    }
    return _data[i];
}
```

```
template<class P>
P const &VectorP<P>::operator[](int i) const {
    if (i < 0 || _size <= i) {
        cerr << "VectorP index out of bounds: index == " << i << endl;
        throw -1;
    }
    return _data[i];
}

// ========= operator<< =========
template<class P>
ostream &operator<<(ostream &os, VectorP<P> const &rhs) {
    rhs.toStream(os);
    return os;
}

// ========= remove =========
template<class P>
P VectorP<P>::remove(int i) {
    P temp = _data[i];
    for (int j = i; j < (_size-1); j++){
        _data[j] = _data[j+1];
    }
    _size--;
    return temp;
}

// ========= toStream =========
template<class P>
void VectorP<P>::toStream(ostream &os) const {
    os << "(";
    for (int i = 0; i < _size - 1; i++) {
        os << *_data[i] << ", ";
    }
    if (_size > 0) {
        os << *_data[_size-1];
    }
    os << ")";
}

#endif

// new page
```